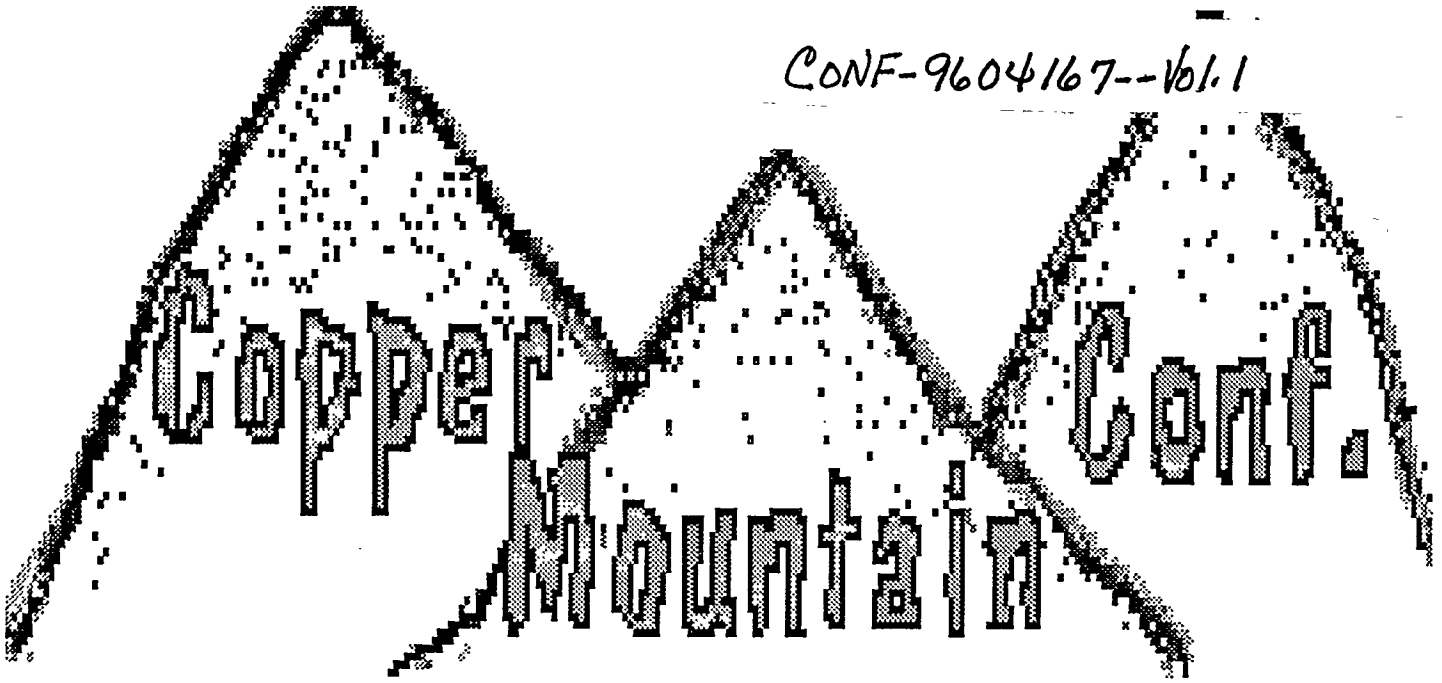


on Iterative Methods

April 9-13, 1996

Copper Mountain, Colorado

Proceedings supported by
Cray Research, Inc.

Volume I

RB

Organized by

Front Range Scientific Computations, Inc.
The University of Colorado

Sponsored by
Department of Energy
National Science Foundation

In cooperation with
SIAM Special Interest Group on
Numerical Linear Algebra

CONFERENCE CHAIRMEN

Tom Manteuffel
Steve McCormick

PROGRAM COMMITTEE

Loyce Adams
Steve Ashby
Howard Elman
Roland Freund
Anne Greenbaum
Seymour Parter
Paul Saylor
Nick Trefethen
Hank van der Vorst
Homer Walker
Olof Widlund

DISCLAIMER

TUESDAY, APRIL 9TH

SESSION I

Room A

Topic: *Nonlinear*	*Session Chair:* *Homer Walker*	
8:00 - 8:30	M.D. Tocci	Method of Lines Solution of Richards' Equation
8:30 - 9:00	C.T. Kelley	A Multilevel Method for Conductive-Radiative Heat Transfer
9:00 - 9:30	H. Walker	An Adaption of Krylov Subspace Methods to Path Following
9:30 - 10:00	J. Neuberger	A Numerical Method for Finding Sign-Changing Solutions of Superlinear Dirichlet Problems

Room B

Topic: *Parallel*	*Session Chair:* *Loyce Adams*	
8:00 - 8:30	A. Basermann	Parallel Preconditioning Techniques for Sparse CG Solvers
8:30 - 9:00	M. Field	Optimising a Parallel Conjugate Gradient Solver
9:00 - 9:30	A. Grama	Parallel Iterative Solvers and Preconditioners Using Approximate Hierarchical Methods
9:30 - 10:00	G. Li	A Block Variant of the GMRES Method on Massively Parallel Processors

Room C

Topic: *Preconditioning*	*Session Chair:* *Seymour Parter*	
8:00 - 8:30	C. Brooking	Using Sparse LU Factorisation to Precondition GMRES for a Family of Similarly Structured Matrices Arising from Process Modelling
8:30 - 9:00	C.H. Guo	Incomplete Block Factorization Preconditioning for Indefinite Elliptic Problems
9:00 - 9:30	S.D. Kim	Preconditioning Cubic Spline Collocation Method by FEM and FDM for Elliptic Equations
9:30 - 10:00	S. Parter	Preconditioning Chebyshev Spectral Methods by Finite-Element and Finite-Difference Methods

SESSION II

Room A

Topic: *Nonlinear*	*Session Chair:* *Homer Walker*	
10:30 - 11:00	D. Knoll	Enhanced Nonlinear Iterative Techniques Applied to a Non-Equilibrium Plasma Flow
11:00 - 11:30	V. Pan	Newton's Iteration for Inversion of Cauchy-like and Other Structured Matrices
11:30 - 12:00	M. Drexler	Fractal Aspects and Convergence of Newton's Method

Room B

Topic: *Parallel*	*Session Chair:* *Loyce Adams*	
10:30 - 11:00	G.C. Lo	Iterative Solution of General Sparse Linear Systems on Clusters of Workstations
11:00 - 11:30	Q. Yao	New Concurrent Iterative Methods with Monotonic Convergence
11:30 - 12:00	R. McLay	Improving Matrix-Vector Product Performance and Multi-Level Preconditioning for the Parallel PCG Package

Topic: Preconditioning	*Session Chair: Seymour Parter*		*Room C*
10:30 - 11:00	M. Tuma	Approximate Inverse Preconditioning of Iterative Methods for Nonsymmetric Linear Systems	
11:00 - 11:30	I. Mishev	Comparison of Different Preconditioners for Nonsymmetric Finite Volume Element Methods	
11:30 - 12:00	C.W. Oosterlee	An Evaluation of Parallel Multigrid as a Solver and a Preconditioner for Singular Perturbed Problems	

SESSION III

Topic: Nonlinear	*Session Chair: Homer Walker*		*Room A*
4:45 - 5:15	M. Pernice	NITSOL: A Newton Iterative Solver for Nonlinear Systems	
5:15 - 5:45	M. Trummer	Nonsymmetric Systems Arising in the Computation of Invariant Tori	
5:45 - 6:15	R. Renaut	Multisplitting for Linear, Least Squares and Nonlinear Problems	

Topic: Parallel	*Session Chair: Loyce Adams*		*Room B*
4:45 - 5:15	V. Menkov	Solving Block Linear Systems with Low-Rank Off-Diagonal Blocks is Easily Parallelizable	
5:15 - 5:45	Y. Shapira	Parallelizable Approximate Solvers for Recursions Arising in Preconditioning	
5:45 - 6:15	D. Xie	New Parallel SOR Method by Domain Partitioning	

Topic: Preconditioning	*Session Chair: Seymour Parter*		*Room C*
4:45 - 5:15	J.D. Moulton	Approximate Schur Complement Preconditioning of the Lowest Order Nodal Discretizations	
5:15 - 5:45	K. Chen	On Preconditioning Techniques for Dense Linear Systems Arising from Singular Boundary Integral Equations	
5:45 - 6:15	W.L. Wan	Fast Wavelet Based Sparse Approximate Inverse Preconditioner	
6:15 - 6:45	E. Meese	Combined Incomplete LU and Strongly Implicit Procedure Preconditioning	

Room TBA	*Workshop Chair*	
7:30 p.m.	*I. Duff*	Sparse Matrix Test Collections

WEDNESDAY, APRIL 10TH

SESSION I

Topic:	*Session Chair:*		*Room A*
FOSLS	*Steve McCormick*		
8:00 - 8:30	T. Manteuffel	First-Order System Least Squares (FOSLS) an Overview	
8:30 - 9:00	S. McCormick	First-Order System Least Squares (FOSLS) for the Pure Traction Problem in Linear Elasticity	
9:00 - 9:30	P. Bochev	Iterative Least-Squares Solvers for the Navier-Stokes Equations	
9:30 - 10:00	G. Starke	Least-Squares Finite Element Discretizations of Neutron Transport Equations in 3 Dimensions	
Topic:	*Session Chair:*		*Room B*
Parallel	*Paul Saylor*		
8:00 - 8:30	K. Maschhoff	A Portable Implementation of ARPACK for Distributed Memory Parallel Archetectures	
8:30 - 9:00	P. Gray	Iteration Schemes for Parallelizing Models of Superconductivity	
9:00 - 9:30	S. Veroneses	Scalable Implicit Methods for Reaction-Diffusion Equations in Two & Three Space Dimensions	
9:30 - 10:00	W. Joubert	Evaluation of Parallel Linear Solvers for Oil Reservoir Simulation Problems	
Topic:	*Session Chair:*		*Room C*
Preconditioning	*TBA*		
8:00 - 8:30	X. Wang	A Necessary and Sufficient Symbolic Condition for the Existence of Incomplete Cholesky Factorization	
8:30 - 9:00	V. Il'in	Advanced Incomplete Factorization Algorithms for Stiltijes Matrices	
9:00 - 9:30	S. Holmgren	Convergence Acceleration for Time-Independent First-Order PDE Using Optimal PNB-Approximations	
9:30 - 10:00	T. Huckle	Iterative Methods for Symmetric Ill-Conditioned Toeplitz Matrices	

SESSION II

Topic:	*Session Chair:*		*Room A*
FOSLS	*Steve McCormick*		
10:30 - 11:00	G. Fix	Application of Least-Squares Principles in Optimal Shape Design Problems	
11:00 - 11:30	B. Bloechle	First-Order Least-Squares for Second-Order Elliptic Problems with Discontinuous Coeeficients: Further Results	
11:30 - 12:00	J. Pasciak	Least-Squares Methods Involving the H^{-1} Inner Product	
Topic:	*Session Chair:*		*Room B*
Parallel	*Paul Saylor*		
10:30 - 11:00	S. Hutchinson	AZTEC: A Parallel Iterative Package for the Solving Linear Systems	
11:00 - 11:30			
11:30 - 12:00	D. Young	On the Implementation of Parallel Implicit USSOR Methods for Solving Discrete Periodic Problems	

Topic: *Arnoldi's Method*	*Session Chair:* *TBA*	*Room C*
10:30 - 11:00	F. Raeven	A New Arnoldi Approach for Polynomial Eigenproblems
11:00 - 11:30	R.B. Lehoucq	Re-Starting an Arnoldi Iteration
11:30 - 12:00	I. Elfadel	Stable Reduced-Order Models of Generalized Dynamical Systems Using Coordinate-Transformed Arnoldi Algorithms

SESSION III

Topic: *FOSLS*	*Session Chair:* *Steve McCormick*	*Room A*
4:45 - 5:15	B. Lee	First-Order System Least-Squares for the Hemholtz Equation
5:15 - 5:45	C-Y Lai	Multilevel Solvers of First-Order System Least Squares for Stokes Equations
5:45 - 6:15	W.T. Mahavier	A Numerical Method for Solving Singular DE's
6:15 - 6:45	M.A. Noor	Generalized Quasi Variational Inequalities

Topic: *Integral Equations*	*Session Chair:* *Van Henson*	*Room B*
4:45 - 5:15		
5:15 - 5:45	J. Tausch	Preconditioning First & Second Kind Integral Formulations of the Capacitance Problem
5:45 - 6:15	B. Singer	Green's Function of Maxwell's Equations and Corresponding Implications for Iterative Methods
6:15 - 6:45	B. Spotz	Iterative Solution of High Order Compact Systems

Topic: *Software*	*Session Chair:* *Iain Duff*	*Room C*
4:45 - 5:15	R. Pozo	Library Designs for Generic C++ Sparse Matrix Computations of Iterative Methods
5:15 - 5:45	F. Saied	MGLab3D: An Interactive Environment for Iterative Solvers for Elliptic PDEs in Two and Three Dimensions
5:45 - 6:15	S. Salvini	New Iterative Solvers for the NAG Libraries

Room TBA	*Workshop Chair*	
7:30 p.m.	*P. Forsyth*	Navier-Stokes & Euler Equations

THURSDAY, APRIL 11TH

SESSION I

Topic: *Navier-Stokes*	*Session Chair:* *Howard Elman*	*Room A*
8:00 - 8:30	V. Sarin	An Efficient Iterative Method for the Generalized Stokes Problem
8:30 - 9:00	P. Fischer	A Deflation based Parallel Algorithm for Spectral Element Solution of the Incompressible Navier-Stokes Equations
9:00 - 9:30	A. Wathen	An Iteration for Indefinite and Non-Symmetric Systems and its Application to the Navier-Stokes Equations
9:30 - 10:00	H. Elman	Perturbation of Eigenvalues of Preconditioned Navier-Stokes Operators
Topic: *Krylov Methods*	*Session Chair:* *M. Gutknecht*	*Room B*
8:00 - 8:30	T. DeLillo	Numerical Conformal Mapping Methods for Exterior and Doubly Connected Regions
8:30 - 9:00	M. Sosonkina	A New Adaptive GMRES Algorithm for Achieving High Accuracy
9:00 - 9:30	E. de Sturler	Truncation Strategies for (Nested) Krylov Methods
9:30 - 10:00	K. Ressel	Hybrid Lanczos-Type Product Methods
Topic: *Eigenvalues*	*Session Chair:* *Homer Walker*	*Room C*
8:00 - 8:30	K. Wu	Preconditioned Krylov Subspace Methods for Eigenvalue Problems
8:30 - 9:00	J. Baglama	A Numerical Method for Eigenvalue Problems in Modeling Liquid Crystals
9:00 - 9:30	A. Knyazev	A Subspace Preconditioning Algorithm for Eigenvector/Eigenvalue Computation
9:30 - 10:00	Z. Drmac	Stable Computation of Generalized Singular Values

SESSION II

Topic: *Navier-Stokes*	*Session Chair:* *Howard Elman*	*Room A*
10:30 - 11:00	I. Yavneh	Fast Multigrid Solution of the Advection Problem with Closed Characteristics
11:00 - 11:30	A.A. Lorber	Accelerated Solution of Non-Linear Flow Problems Using Chebyshev Iteration Polynomial Based Runge-Kutta Recursions
11:30 - 12:00	M. Murphy	Towards an Ideal Preconditioner for Linearized Navier-Stokes Problems
Topic: *Krylov Methods*	*Session Chair:* *M. Gutknecht*	*Room B*
10:30 - 11:00	M. Gutknecht	Look-Ahead Procedures for Lanczos-Type Product Methods Based on Three-Term Recurrences
11:00 - 11:30	E. Gallopoulos	Solving Modified Systems with Multiple Right-Hand Sides
11:30 - 12:00		

Topic: *Eigenvalues*	*Session Chair:* *Homer Walker*	*Room C*
10:30 - 11:00	A. Stathopoulos	Thick Restarting of the Davidson Method: an Extension to Implicit Restarting
11:00 - 11:30	X. Zou	Splitting the Determinants of Upper Hessenberg Matrices & the Hyman Method
11:30 - 12:00	M. Fernandes	A Combined Modification of the Newton's Method for Systems of Nonlinear Equations

Topic: *Student Papers Winners*	*Session Chair:* *T. Manteuffel & S. McCormick*	*Room A*
4:45 - 5:15	S. Knapek	Matrix-Dependent Multigrid-Homogenization for Diffusion Problems
5:15 - 5:45	M. Horn	A Superlinear Convergence Estimate for an Iterative Method for the Biharmonic Equation
5:45 - 6:15	A. Klawonn	Triangular Preconditioners for Saddle Point Problems with a Penalty Term

Cash Bar *Room TBA*
6:45 - 7:30 p.m.

Banquet *Room TBA*
7:30 - 9:30 p.m.

FRIDAY, APRIL 12TH

SESSION I

Topic:	*Session Chair:*		*Room A*
Domain Decomp.	*Olof Widlund*		
8:00 - 8:30	X.C. Cai	Newton-Krylov-Schwarz Algorithms for the 2D Full Potential Equations	
8:30 - 9:00	D. Keyes	Newton-Krylov-Schwarz Methods in Unstructured Grid Euler Flow	
9:00 - 9:30	U. Trottenberg	Adaptive Parallel Multigrid for Euler & Incompressible Navier-Stokes Equations	
9:30 - 10:00	O. Widlund	Domain Decomposition Methods for Mortar Finite Elements	
Topic:	*Session Chair:*		*Room B*
Krylov Methods	*Anne Greenbaum*		
8:00 - 8:30	V. Druskin	Extended Krylov Subspaces Approximations of Matrix Functions; Application to Computational Electromagnetics	
8:30 - 9:00	D. Sorensen	Krylov Subspace Methods for Computation of Matrix Functions	
9:00 - 9:30	T. Tamarchenko	Application of Spectral Lanczos Decomposition Method to Large Scale Problems Arising Geophyics	
9:30 - 10:00	A. Greenbaum	Some Uses of the Symmetric Lanczos Algorithm and Why it Works!	
Topic:	*Session Chair:*		*Room C*
CFD	*TBA*		
8:00 - 8:30	X. Zheng	Multigrid Solution of Incompressible Turbulent Flows by Using Two-Equation Turbulence Models	
8:30 - 9:00	S. Kumar	Nonlinear Krylov Acceleration of Reacting Flow Codes	
9:00 - 9:30	M.D. Tidriri	Schwarz-based Algorithms for Compressible Flows	
9:30 - 10:00	C. Liao	Multilevel Local Refinement and Multigrid Methods for 3-D Turbulent Flow	
Topic:	*Session Chair:*		*Room A*
Doman Decomp.	*Olof Widlund*		
10:30 - 11:00	X. Feng	A Mixed Finite Element Domain Decomposition Method for Solving Nearly Elastic Wave Equations in the Frequency Domain	
11:00 - 11:30	A. Jemcov	Representation of Discrete Steklov-Poincare Operator Arising in Domain Decomposition Methods in Wavelet Basis	
11:30 - 12:00	J. Xu	Simplified Approach to Some Nonoverlapping Domain Decomp. Methods	
Topic:	*Session Chair:*		*Room B*
Krylov Methods	*Anne Greenbaum*		
10:30 - 11:00	F. Campos	The Adaptive CCCG(n) Method for Efficient Solution of Time Dependent Partial Differential Equations	
11:00 - 11:30	T. Barth	Conjugate Gradient Algorithms Using Multiple Recursions	
11:30 - 12:00	J. Cullum	Iterative Methods for Solving Ax=b, GMRES/FOM versus QMR/BiCG	

SESSION III

Topic: *Domain Decomp.*	*Session Chair:* *Olof Widlund*	*Room A*
4:45 - 5:15	S. Maliassov	Domain Decomposition Method for Nonconforming Finite Element Approximations of Anisotropic Elliptic Problems on Nonmatching Grids
5:15 - 5:45	H. Zhao	Analysis of Generalized Schwarz Alternating Procedure for Domain Decomposition
5:45 - 6:15	R. Tezaur	Substructuring by Lagrange Multipliers for Solids and Plates
6:15 - 6:45	X.C. Tai	Some Nonlinear Space Decomposition Algorithms

Topic: *Krylov Methods*	*Session Chair:* *Anne Greenbaum*	*Room B*
4:45 - 5:15	E. Bobrovnikova	Iterative Methods for Weighted Least-Squares
5:15 - 5:45	A. Lumsdaine	Krylov Subspace Acceleration of Waveform Relaxation
5:45 - 6:15	C. Wagner	Tangential Frequency Filtering Decompositions
6:15 - 6:45	J. Zhang	Multigrid Solution of Convection-Diffusion Equation with High-Reynolds Number

Topic: *Markov Chains*	*Session Chair:* *Daniel Szyld*	*Room C*
4:45 - 5:15	T. Dayar	State Space Orderings for Gauss-Seidel in Markov Chains Revisited
5:15 - 5:45	G. Horton	On the Multi-Level Solution Algorithm for Markov Chains
5:45 - 6:15	D. Szyld	Threshold Partitioning of Sparse Matrices and Applications to Markov Chains
6:15 - 6:45		

Room TBA *7:30 p.m.*	*Workshop Chair* *Mike Heroux*	Sparse and Parallel BLAS

SATURDAY, APRIL 12TH

SESSION I

Topic: *Multigrid*	*Session Chair:* *Joel Dendy*	*Room A*
8:00 - 8:30	R. Alchalabi	Multigrid Method Applied to the Solution of an Elliptic, Generalized Eigenvalue Problem
8:30 - 9:00	J. Dendy	Some Multigrid Algorithms for SIMD Machines
9:00 - 9:30	C. Douglas	Multigrid on Unstructured Grids Using an Auxiliary Set of Structured Grids
9:30 - 10:00	M. Griebel	Multiscale Iterative Methods, Coarse Level Operator Construction and Discrete Homogenization Techniques

Topic: *Applications*	*Session Chair:* *TBA*	*Room B*
8:00 - 8:30	H.C. Chen	Embedding SAS Approach Into Conjugate Gradient Algorithms for Asymmetric 3D Elasticity Problems
8:30 - 9:00	M. Clemens	Iterative Methods for the Solution of Very Large Complex-Symmetric Linear Systems of Equations in Electrodynamics
9:00 - 9:30	T. Cwik	Matrix Equation Decomposition and Parallel Solution of Systems Resulting from Unstructured Finite Element Problems in Electromagnetics
9:30 - 10:00	S.W. Bova	Iterative Solution of the Semiconductor Device Equations

Topic: *Multiple RHS*	*Session Chair:* *Roland Freund*	*Room C*
8:00 - 8:30	W. Boyse	Multiple Solutions to Dense Systems in Radar Scattering using a Preconditioned Block GMRES Solver
8:30 - 9:00	R. Freund	The BL-QMR Algorithm for Non-Hermitian Linear Systems with Multiple Right-Hand Sides
9:00 - 9:30	M. Malhotra	Iterative Solution of Multiple Radiation and Scattering Problems in Structural Acoustics Using the BL-QMR Algorithm
9:30 - 10:00	T. Chan	Galerkin Projection Methods for Solving Multiple Related Linear Systems

SESSION II

Topic: *Multigrid*	*Session Chair:* *Joel Dendy*	*Room A*
10:30 - 11:00	R. Hornung	Adaptive Mesh Refinement and Multilevel Iteration for Multiphase, Multicomponenet Flow in Porous Media
11:00 - 11:30	J. Jones	Semi-Coarsening Multigrid Methods for Parallel Computing
11:30 - 12:00	P. Vanek	An Algebraic Multigrid Algorithm for Symmetric Positive Definite Linear Systems

Topic:	*Session Chair:*		*Room B*
Applications	*TBA*		
10:30 - 11:00	A. Frommer	Lattice QCD Computations: Recent Progress with Modern Krylov Subspace Methods	
11:00 - 11:30	R. Karamikhova	Numerical Solution of High-Kappa Model of Superconductivity	
11:30 - 12:00	M. Heroux	The Impact of Improved Sparse Linear Solvers on Industrial Engineering Applications	
Topic:	*Session Chair:*		*Room C*
Projection Methods	*Roland Freund*		
10:30 - 11:00	A. Popov	Projection Preconditioning for Lanczos-Type Methods	
11:00 - 11:30	R. Bramley	Partial Row Projection Methods	
11:30 - 12:00	P. Kolm	Generalized Subspace Correction Methods	

SESSION III

Topic:	*Session Chair:*		*Room A*
Multigrid	*Joel Dendy*		
4:45 - 5:15	S. Oliveira	A Multigrid Method for Variational Inequalities	
5:15 - 5:45	W. Schmid	A Multigrid Solution Method for Mixed Hybrid Finite Elements	
5:45 - 6:15	H.J. Bungartz	A Unidirectional Approach for d-Dimensional Finite Element Methods of Higher Order on Sparse Grids	
6:15 - 6:45	M. Brezina	Two-Level Method with Coarse Space Size Independent Convergence	
Topic:	*Session Chair:*		*Room B*
Applications	*Tom Russell*		
4:45 - 5:15	C. Yang	Numerical Computation of the Linear Stability of the Diffusion Model for Crystal Growth Simulation	
5:15 - 5:45	L. Borges	Highly Indefinite Multigrid for Eigenvalue Problems	
5:45 - 6:15	G.S. Lett	An Adaptive Nonlinear Solution Scheme for Reservoir Simulation	
6:15 - 6:45	A. Cardona	An Iterative Method to Invert the LTSn Matrix	
Topic:	*Session Chair:*		*Room C*
Helmholtz	*Roland Freund*		
4:45 - 5:15	E. Larsson	Iterative Solution of the Holmholtz Equation	
5:15 - 5:45	S. Kim	Iterative Procedures for Wave Propagation in the Frequency Domain	
5:45 - 6:15	J. Yoo	Multigrid for the Galerkin Least Squares Method in Linear Elasticity: The Pure Displacement Problem	
6:15 - 6:45			

TUESDAY, APRIL 9TH

Topic: *Nonlinear*	*Session Chair:* *Homer Walker*	*Room A*
8:00 - 8:30	M.D. Tocci	Method of Lines Solution of Richards' Equation
8:30 - 9:00	C.T. Kelley	A Multilevel Method for Conductive-Radiative Heat Transfer
9:00 - 9:30	H. Walker	An Adaption of Krylov Subspace Methods to Path Following
9:30 - 10:00	J. Neuberger	A Numerical Method for Finding Sign-Changing Solutions of Superlinear Dirichlet Problems

Method of Lines Solution of Richards' Equation

C. T. Kelley, C. T. Miller, and M. D. Tocci
Speaker: M. D. Tocci

We consider the method of lines solution of Richards' equation, which models flow through porous media, as an example of a situation in which the method can give incorrect results because of premature termination of the nonlinear corrector iteration. This premature termination arises when the solution has a sharp moving front and the Jacobian is ill-conditioned. While this problem can be solved by tightening the tolerances provided to the ODE or DAE solver used for the temporal integration, it is more efficient to modify the termination criteria of the nonlinear solver and/or recompute the Jacobian more frequently. In this paper we continue previous work on this topic by analyzing the modifications in more detail and giving a strategy on how the modifications can be turned on and off in response to changes in the character of the solution.

A MULTILEVEL METHOD FOR CONDUCTIVE-RADIATIVE HEAT TRANSFER *

J. M. BANOCZI† AND C. T. KELLEY†

We present a fast multilevel algorithm for the solution of a system of nonlinear integro-differential equations that model steady-state combined radiative-conductive heat transfer. The equations can be formulated as a compact fixed point problem with a fixed point map that requires both a solution of the linear transport equation and the linear heat equation for its evaluation. We use fast transport solvers developed by the second author [12], [10], to construct an efficient evaluation of the fixed point map and then apply the Atkinson-Brakhage [2], [3], method, with Newton-GMRES as the coarse mesh solver, to the full nonlinear system.

1. Introduction. We consider the normalized dimensionless form of the equations [17], [26], [27]. The radiative transport equation is

$$\mu\frac{\partial\psi}{\partial x}(x,\mu)+\psi(x,\mu)=\frac{c(x)}{2}\int_{-1}^{1}\psi(x,\mu')\,d\mu'+(1-c(x))\Theta^4(x), \tag{1}$$

for $x\in(0,\tau)$ with boundary conditions

$$\psi(0,\mu)=\varepsilon_l\Theta_l^4+\rho_l^s\psi(0,-\mu)+2\rho_l^d\int_0^1\psi(0,-\mu')\mu'\,d\mu',\mu>0 \tag{2}$$

and

$$\psi(\tau,\mu)=\varepsilon_r\Theta_r^4+\rho_r^s\psi(\tau,-\mu)+2\rho_r^d\int_0^1\psi(\tau,\mu')\mu'\,d\mu',\mu<0. \tag{3}$$

In the boundary conditions (2) and (3) $\varepsilon_i\geq 0$ for $i=l,r$. The coefficients for specular (ρ_l^s,ρ_r^s) and diffuse (ρ_l^d,ρ_r^d) reflection satisfy

$$\rho_i^d,\rho_i^s\geq 0 \text{ and } \varepsilon_i+\rho_i^s+\rho_i^d=1 \text{ for } i=l,r. \tag{4}$$

The local albedo $c(x)$ satisfies

$$0\leq c(x)\leq 1 \text{ for all } x\in[0,\tau].$$

Even though we consider the dimensionless form of the equations, we will still refer to Θ as temperature, ψ as intensity, and

$$f(x)=\frac{1}{2}\int_0^\tau\psi(x,\mu')\,d\mu' \tag{5}$$

as the scalar flux.

---

* C. T. Kelley will present the paper.

† North Carolina State University, Center for Research in Scientific Computation and Department of Mathematics, Box 8205, Raleigh, N. C. 27695-8205, USA. This research was supported by National Science Foundation grant #DMS-9321938 and a Cray Research Corporation Fellowship. Computing activity was partially supported by an allocation of time from the North Carolina Supercomputing Center.

The temperature Θ satisfies the diffusion equation.

$$\frac{\partial^2 \Theta}{\partial x^2} = Q(x), \tag{6}$$

for $x \in (0, \tau)$ with boundary conditions

$$\Theta(0) = \Theta_l, \Theta(\tau) = \Theta_r, \tag{7}$$

and coupling to the radiative transport equation by

$$Q(x) = \frac{1}{2N_c} \frac{d}{dx} \int_{-1}^{1} \mu' \psi(x, \mu') \, d\mu'. \tag{8}$$

In (8) N_c is the conduction to radiation parameter [17].

As in [9] the order of integration and differentiation in (8) can be changed and we obtain

$$\frac{d}{dx} \int_{-1}^{1} \mu' \psi(x, \mu') \, d\mu' = \int_{-1}^{1} \mu' \frac{\partial}{\partial x} \psi(x, \mu') \, d\mu'. \tag{9}$$

Hence, for $0 < x < \tau$,

$$Q(x) = \alpha(x)(\Theta^4(x) - f(x)), \tag{10}$$

where f is the scalar flux given (5) and

$$\alpha(x) = (1 - c(x))/N_c. \tag{11}$$

2. Formulation as a Fixed Point Problem. Following [9], we express the nonlinear system as a compact fixed point problem in the function Θ alone. The fixed point map $\Theta \to \mathcal{T}(\Theta)$ is computed as follows.

1. Solve the transport problem (1), (2), (3) and obtain f. This problem has a unique solution for any Θ.
2. Use Θ and f to form the right hand side of (6) using (10).
3. Solve $T_{xx} = Q$ subject to the boundary conditions (7) and set $\mathcal{T}(\Theta) = T$.

The map $\mathcal{T}$ is a completely continuous map on $C[0, \tau]$, [9], and it is natural to apply fast algorithms for compact fixed point problems, such as those from [2], [3], [7], [12], or [10]. The most efficient of these, when applicable, is the algorithm described in [10] and [12], which requires only one fine mesh fixed point map evaluation at each level. In order to apply that algorithm the discretization and fine-to-coarse intergrid transfers must be managed carefully. That can be done at least for the transport solve (step 1) in the problem under consideration here. We hope to report on the application of the method of [10] to the complete problem in the full paper.

Currently we solve the full problem with the method of [2]. In order to apply this method, we construct a strongly convergent and collectively compact [1] sequence of approximations to $\mathcal{T}$, $\{\mathcal{T}_l\}$. This is done with a finite difference discretization of Θ in space and a diamond-difference discrete ordinate discretization of the transport equation in space-angle. The transport solver for step 1 uses the method of [12] for efficiency with GMRES, [25], as the coarse mesh solver. Having formed the right hand side of (6), step 3 is carried out using a standard direct tridiagonal solver.

Using interpolation and restriction operators where necessary, we can regard $\mathcal{T}_l$ as a map on $C[0,\tau]$, [24], [12]. Nonlinear fixed point problems for the approximate maps

$$u - \mathcal{T}_l(u) = 0$$

are solved by first solving finite dimensional nonlinear equations and then using a variant of Nyström interpolation [2], [16], [22], to recover the values of the solution at arbitrary points in $[0,\tau]$. As will become clear from the description of the solver in § 3 we must only perform this solve at the coarsest mesh. Similarly the linearized problems

$$w - \mathcal{T}_l'(u)w = g,$$

when solved by a Krylov method such as GMRES, require only finite dimensional matrix-vector products to compute the operator-vector products.

3. Solver. For the full nonlinear problem we use the Atkinson-Brakhage method with a Newton-GMRES method as the coarse mesh solver. We give a short description of this approach. Letting $l = 0$ denote the coarsest mesh, the Atkinson-Brakhage iteration for the solution at level $L > 0$ can be written as

$$u_+ = u_c - [I + (I - \mathcal{T}_0')^{-1}\mathcal{T}_L'](u_c - \mathcal{T}_L u_c) \tag{12}$$

which is a modified Newton or Chord method [6], [23], [11] with

$$I + (I - \mathcal{T}_0')^{-1}\mathcal{T}_L'$$

used as an approximation to $(I - \mathcal{T}_L')^{-1}$. Here, $\mathcal{T}_l'$ is the Fréchet derivative of $\mathcal{T}_l$, the approximation at level l. The action of $(I - \mathcal{T}_0')^{-1}$ on a vector, which is required in the right side of (12), is approximated by a GMRES iteration which terminates on a sufficiently small relative residual. Collective compactness and strong convergence of the sequence $\{\mathcal{T}_l\}$ imply that if the coarse mesh is sufficiently fine, this iteration will converge [12], [10], [2], [3]. Moreover, the convergence can be made sufficiently rapid so that if a nested iteration, [7], or grid sequencing approach is used, where the fine level is increased as the iteration progresses, only a single iteration need be taken at each level (*i. e.* the update of u_l in step 2(b) in Algorithm `nestab` is only done once).

Denoting $l = 0$ as the coarse level and $l = L$ as the level of approximation at which a solution is required, the nested iteration form of the algorithm can be expressed as

ALGORITHM 3.1. `nestab`$(\mathcal{T}, L)$

1. *Solve* $u_0 = \mathcal{T}_0(u_0)$.
2. *For* $l = 1, \ldots, L$
   (a) *Initialize* $u_l = u_{l-1}$
   (b) *While* $\|u_l - \mathcal{T}_l(u_l)\| \geq \epsilon_l$

$$u_l = u_l - [I + (I - \mathcal{T}_0')^{-1}\mathcal{T}_l'](u_l - \mathcal{T}_l(u_l))$$

4. Numerical Results. In our implementation, level l corresponds to a uniform mesh of $N_l = 2^{l+l_0} + 1$ points, where $l_0 = 4$ for the computations reported here. The difference scheme is second order accurate [16] for a fixed angular mesh. The cost of an evaluation of $\mathcal{T}_l$ is $O(N_l)$ and therefore the cost of a complete solve using Algorithm `nestab` is $O(N_L)$.

The transport solve uses the method from [12]. We use an angular mesh of 40 points, corresponding to a double 20 point Gaussian quadrature on $[-1,1]$. This solver is more efficient than an Atkinson-Brakhage solver or a second-kind multigrid [8] approach. Transport solvers of this type require only the source iteration map and do not need a diffusion solver as a preconditioner as do some other multilevel approaches [18], [19], which are based on diffusion-synthetic acceleration, [14], [15]. However these other methods extend directly to more than one space dimension and Atkinson-Brakhage solvers have not, as yet, been applied to other than one dimensional transport problems. Using an existing fast 2D or 3D transport solver would extend the algorithm in this paper immediately to higher space dimension.

The coarse mesh solve in Step 1 was done with a Newton-GMRES code which terminated when the nonlinear residual was below 10^{-6}. This was implemented in a straightforward manner and converged without any need for globalization of the nonlinear iteration.

Step 2(b) could also be replaced with a Newton-GMRES solve of the full equation at level l, requiring that the nonlinear residual be reduced by a fixed factor $< 1/4$. Because of the compactness of $\mathcal{T}$ and the resulting clustering of the spectrum [4], [5], [13], [21], [20], [28], a solve using such an approach would also cost $O(N_L)$ floating point operations, but would be more expensive than one using Algorithm `nestab` because more than two Fréchet derivative-vector products would be required.

We report on computations for three problems, the data for which are summarized in Table 1. The finest spatial mesh had 2049 points for all three problems and the coarese mesh had 9 points for problems A and B and 33 points for problem C, which has reflecting boundary conditions. These meshes were the coarsest possible that allowed the loop in step 2(b) in Algorithm `nestab` to terminate after a single iteration. Since we expect (and obtain) second order accuracy [16], [24], we set

$$\epsilon_l = \|u_{l-1} - \mathcal{T}_l(u_{l-1})\|/10,$$

with a view that a reduction by a factor of 10 in the nonlinear residual will safely ensure a reduction by a factor of 4 in the error.

We compare the Atkinson-Brakhage method to a nested Newton-GMRES approach, which replaces step 2(b) with a Newton-GMRES loop. The GMRES inner iterations terminated when the relative residual had been reduced by a factor of 10. Coarse mesh solves were done with Newton-GMRES and the coarse mesh residual was reduced by a factor of 10^{-6} from that for the initial iterate of $u = 0$.

TABLE 1
Data for the test problems

Problem	τ	N_c	c	Θ_1	Θ_2	ε_1	ε_2	ρ_1^s	ρ_2^s	ρ_1^d	ρ_2^d
A	1.0	0.05	0.9	1.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0
B	1.0	0.05	0.9	1.0	0.5	1.0	1.0	0.0	0.0	0.0	0.0
C	3.0	0.05	0.9	1.0	0.5	0.7	0.6	0.1	0.3	0.2	0.1

We report both function evaluations at the fine mesh levels (FMF) and timings in Table 2. The number of function evaluations at the fine mesh levels did not vary as the mesh was refined, consistent with the theory from [2] and [5]. One can see a clear relation between the function evaluations and the timings, indicating that for both algorithms the cost of the work internal to the solver itself was minor.

TABLE 2
Timings and Function Evaluations

	Atkinson-Brakhage		**Newton-GMRES**	
Problem	Time	FMF	Time	FMF
A	10.02	2	17.78	4
B	10.27	2	17.82	4
C	16.25	2	29.95	5

The computations reported here were done on a SUN SPARCstation 5 workstation running SunOS 4.1.3_U1 version 1 with the SUN f77 compiler version 3.0.1. Some of the prelimnary computations for this work were done at the North Carolina Supercomputing Center.

REFERENCES

[1] P. M. ANSELONE, *Collectively Compact Operator Approximation Theory*, Prentice-Hall, Englewood Cliffs, NJ, 1971.

[2] K. E. ATKINSON, *Iterative variants of the Nyström method for the numerical solution of integral equations*, Numer. Math., 22 (1973), pp. 17–31.

[3] H. BRAKHAGE, *Über die numerische Behandlung von Integralgleichungen nach der Quadraturformelmethode*, Numer. Math., 2 (1960), pp. 183–196.

[4] S. L. CAMPBELL, I. C. F. IPSEN, C. T. KELLEY, AND C. D. MEYER, *GMRES and the minimal polynomial*, Tech. Report CRSC-TR94-10, North Carolina State University, Center for Research in Scientific Computation, July 1994. BIT, to appear.

[5] S. L. CAMPBELL, I. C. F. IPSEN, C. T. KELLEY, C. D. MEYER, AND Z. Q. XUE, *Convergence estimates for solution of integral equations with GMRES*, Tech. Report CRSC-TR95-13, North Carolina State University, Center for Research in Scientific Computation, March 1995. Journal of Integral Equations and Applications, to appear.

[6] J. E. DENNIS AND R. B. SCHNABEL, *Numerical Methods for Nonlinear Equations and Unconstrained Optimization*, Prentice-Hall, Englewood Cliffs, N.J, 1983.

[7] W. HACKBUSCH, *Multi-Grid Methods and Applications*, vol. 4 of Springer Series in Computational Mathematics, Springer-Verlag, New York, 1985.

[8] ———, *Multigrid methods of the second kind*, in Multigrid Methods for Integral and Differential Equations, Oxford University Press, Oxford, 1985.

[9] C. T. KELLEY, *Existence and uniqueness of solutions of nonlinear systems of conductive-radiative heat transfer equations*, Tech. Report CRSC-TR95-5, North Carolina State University, Center for Research in Scientific Computation, January 1995. Journal of Trans. Th. Stat. Phys., to appear.

[10] ———, *A fast multilevel algorithm for integral equations*, SIAM J. Numer. Anal., 32 (1995), pp. 501–513.

[11] ———, *Iterative Methods for Linear and Nonlinear Equations*, no. 16 in Frontiers in Applied Mathematics, SIAM, Philadelphia, 1995.
[12] ———, *Multilevel source iteration accelerators for the linear transport equation in slab geometry*, Trans. Th. Stat. Phys., 24 (1995), pp. 679–708.
[13] C. T. KELLEY AND Z. Q. XUE, *GMRES and integral operators*, SIAM J. on Sci. Comp., 17 (1996), pp. 217–226.
[14] E. W. LARSEN, *Unconditionally stable diffusion-synthetic acceleration methods for the slab geometry discrete ordinates equations. part I: Theory*, Nucl. Sci. Eng., 82 (1982), pp. 47–63.
[15] ———, *Diffusion-synthetic acceleration methods for discrete-ordinates problems*, Trans. Th. Stat. Phys., 13 (1984), pp. 107–126.
[16] E. W. LARSEN AND P. NELSON, *Finite difference approximations and superconvergence for the discrete ordinate equations in slab geometry*, SIAM J. Numer. Anal., 19 (1982), pp. 334–348.
[17] M. N. ÖZIŞIK, *Radiative Transfer and Interaction with Conduction and Convection*, John Wiley and Sons, New York, 1973.
[18] T. MANTEUFFEL, S. MCCORMICK, J. MOREL, S. OLIVEIRA, AND G. YANG, *Parallel multilevel methods for transport equations*, SIAM J. Sci. Comp., 15 (1994), pp. 474–493.
[19] J. E. MOREL AND T. A. MANTEUFFEL, *An angular multigrid acceleration technique for S_n equations with highly forward-peaked scattering*, Nuclear Science and Engineering, 107 (1991), pp. 330–342.
[20] N. M. NACHTIGAL, S. C. REDDY, AND L. N. TREFETHEN, *How fast are nonsymmetric matrix iterations?*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 778–795.
[21] O. NEVANLINNA, *Convergence of Iterations for Linear Equations*, Birkhäuser, Basel, 1993.
[22] E. J. NYSTRÖM, *Über die praktische Auflösung von Integralgleichungen mit Anwendungen auf Randwertaufgaben*, Acta Math., 54 (1930), pp. 185–204.
[23] J. M. ORTEGA AND W. C. RHEINBOLDT, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, 1970.
[24] J. PITKÄRANTA AND R. SCOTT, *Error estimates for the combined spatial and angular approximations of the transport equation in slab geometry*, SIAM J. Numer. Anal., 20 (1983), pp. 922–950.
[25] Y. SAAD AND M. SCHULTZ, *GMRES a generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comp., 7 (1986), pp. 856–869.
[26] C. E. SIEWERT, *An improved iterative method for solving a class of coupled conductive-radiative heat transfer problems*, J. Quant. Spectrosc. Radiat. Transfer, 54 (1995), pp. 599–605.
[27] C. E. SIEWERT AND J. R. THOMAS, *A computational method for solving a class of coupled conductive-radiative heat transfer problems*, J. Quant. Spectrosc. Radiat. Transfer, 45 (1991), pp. 273–281.
[28] Z. Q. XUE, *Mesh-independence of GMRES for Integral Equations*, PhD thesis, North Carolina State University, 1995.

An Adaptation of Krylov Subspace Methods to Path Following

Homer F. Walker*
Mathematics and Statistics Department
Utah State University
Logan, UT 84322-3900
walker@math.usu.edu

Abstract.

Krylov subspace methods at present constitute a very well known and highly developed class of iterative linear algebra methods. These have been effectively applied to nonlinear system solving through Newton–Krylov methods, in which Krylov subspace methods are used to solve the linear systems that characterize steps of Newton's method (the *Newton equations*). Here, we will discuss the application of Krylov subspace methods to path following problems, in which the object is to track a solution curve as a parameter varies. Path following methods are typically of predictor-corrector form, in which a point near the solution curve is "predicted" by some easy but relatively inaccurate means, and then a series of Newton-like corrector iterations is used to return approximately to the curve. The analogue of the Newton equation is underdetermined, and an additional linear condition must be specified to determine corrector steps uniquely. This is typically done by requiring that the steps be orthogonal to an approximate tangent direction. Augmenting the underdetermined system with this orthogonality condition in a straightforward way typically works well if direct linear algebra methods are used, but Krylov subspace methods are often ineffective with this approach. We will discuss recent work in which this orthogonality condition is imposed directly as a constraint on the corrector steps in a certain way. The means of doing this preserves problem conditioning, allows the use of preconditioners constructed for the fixed-parameter case, and has certain other advantages. Experiments on standard PDE continuation test problems indicate that this approach is effective.

* Research supported in part by United States Department of Energy Grant DE-FG03-94ER25221 and National Science Foundation Grant DMS-9400217, both with Utah State University.

A NUMERICAL METHOD FOR FINDING SIGN-CHANGING SOLUTIONS OF SUPERLINEAR DIRICHLET PROBLEMS

John M. Neuberger

In a recent result (See Castro-Cossio-Neuberger, to appear in Rocky Mountain J. of Math.), it was shown via a variational argument that a class of superlinear elliptic boundary value problems has at least three nontrivial solutions, a pair of one sign and one which sign changes exactly once. These three and all other nontrivial solutions are saddle points of an action functional, and are characterized as local minima of that functional restricted to a codimension one submanifold of the Hilbert space H-0-1-2, or an appropriate higher codimension subset of that manifold.

In this paper we present a numerical Sobolev steepest descent algorithm for finding these three solutions. Of primary interest is the method of projecting iterates of elements in our Hilbert space onto the submanifold and its subsets. When applied to the ordinary differential equation, the algorithm is extended to find additional solutions possessing a greater number of internal zeroes, and in that case the solutions are compared to independent numerical calculations obtained by Euler's method. We further test the algorithm on partial differential equations on the unit square. With or without a symmetric nonlinearity, numerical computations for PDEs on the square yield four exactly-once sign-changing solutions of Morse Index 2 and supply evidence that suggests that there may exist four more of MI 3.

In modifying the code to run on arbitrary regions such as the disk, annulus, and dumbell, we are obtaining numerical evidence complementary to our current analysis. In particular, as we show in a current paper (in submittion), the minimal action function valued sign-changing solution on the disk is nonradial. This also holds on the annulus.

This algorithm can be used to investigate the nodal structure of a wide range of superlinear elliptic PDEs and should be modifiable to include Neumann boundary conditions. Accuracy and convergence are quite satisfactory though improvements and refinements are clearly possible. Particularly gratifying are the ways we can use the algorithm to visualize the infinite dimensional submanifold's structure and its relation to the eigenfunctions of the negative Laplacian.

Topic: *Parallel*	*Session Chair:* *Loyce Adams*	*Room B*

8:00 - 8:30	A. Basermann	Parallel Preconditioning Techniques for Sparse CG Solvers
8:30 - 9:00	M. Field	Optimising a Parallel Conjugate Gradient Solver
9:00 - 9:30	A. Grama	Parallel Iterative Solvers and Preconditioners Using Approximate Hierarchical Methods
9:30 - 10:00	G. Li	A Block Variant of the GMRES Method on Massively Parallel Processors

Parallel Preconditioning Techniques for Sparse CG Solvers

A. Basermann, B. Reichel, and C. Schelthoff

Central Institute for Applied Mathematics
Research Centre Jülich GmbH, 52425 Jülich, Germany
Phone: +49/2461/61/2433, Fax: +49/2461/61/6656
Email: a.basermann@kfa-juelich.de

Conjugate gradient (*CG*) methods to solve sparse systems of linear equations play an important role in numerical methods for solving discretized partial differential equations. The large size and the condition of many technical or physical applications in this area result in the need for efficient parallelization and preconditioning techniques of the CG method. In particular for very ill-conditioned matrices, sophisticated preconditioner are necessary to obtain both acceptable convergence and accuracy of CG [2] [3]. Here, we investigate variants of polynomial and incomplete Cholesky preconditioners that markedly reduce the iterations of the simply diagonally scaled CG and are shown to be well suited for massively parallel machines.

The basic operations of the pure CG iteration as well as of the polynomially preconditioned method [1] [8] are matrix-vector products with the coefficient matrix and vector-vector computations. For incomplete Cholesky preconditioning, the factorization of the matrix before the CG iteration and a forward/back-substitution per iteration [6] are required in addition.

To parallelize these operations on a multiprocessor system with distributed memory, we present a data distribution and a communication scheme based on the analysis of the non-zero matrix elements. By a suitable block-reordering of the matrix, the overlapped execution of computation and communication is supported to reduce waiting times [4] [5]. Factorization and forward/back-substitution of the developed incomplete Cholesky preconditioner are only performed on local blocks so that these operations do not

require communication. Hence, this preconditioning technique is fully parallel. The data distribution and the communication scheme are determined before the execution of the solver by preprocessing the symbolic structure of the sparse matrix, and they both are exploited in each iteration. The schemes can be reused as long as the sparsity pattern of the matrix, which is determined by the discretization mesh and the element types, does not change. For example, they can be used in each time step of a time dependant problem or in each iterative step of a nonlinear problem that is solved by linearization.

On massively parallel systems, the computation of inner products in the CG iteration may be very costly since it needs global communication. To save synchronization points, inner products are grouped so that each iteration of the developed algorithms only requires one global communication. In addition, the contribution of inner products to the costs per iteration is considerably reduced by the preconditioning techniques used.

Performance tests of the developed parallel preconditioned CG methods were carried out on the distributed memory system PARAGON XP/S 10 of the Research Centre Jülich. We demonstrate by case studies with matrices of different sparsity from various real finite element applications that the developed data distribution and the communication scheme together with the reduction of synchronization do result in an advantageous scaling behavior of the algorithms on massively parallel machines.

With respect to the developed parallel preconditioning techniques, we observe that efficient preconditioning depends on both the structure and the condition of the coefficient matrix. To decide for which case which parallel preconditioning method is suited, we draw the following conclusions. For well-conditioned matrices, it is hard to beat diagonal scaling by more sophisticated preconditioners since this method is effective and cheap to compute (see also [7]). Polynomial preconditioning shows the best results if the coefficient matrix is so sparse that matrix-vector multiplications do not dominate the solver's total execution time. Especially for high processor numbers, this preconditioner markedly reduces synchronization costs. Moreover, the preconditioning effect is stable for varying processor numbers. Opposite to polynomial preconditioning, the developed fully parallel incomplete Cholesky preconditioner performs best if matrix-vector operations dominate the iteration's total execution time. The preconditioning effect is the better, the more significant matrix elements are close to the diagonal. These two condi-

tions are usually given for matrices from structural mechanics applications so that CG with parallel incomplete Cholesky preconditioning shows the most advantageous time behavior for this kind of matrices in our investigations.

References

[1] St. Ashby, Minimax polynomial preconditioning for Hermitian linear systems, *SIAM J. Matrix Anal. Appl.* 12 (1991) 766–789.

[2] O. Axelsson, *Iterative Solution Methods* (Cambridge University Press, 1994).

[3] R. Barrett, M. Berry, T. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods* (SIAM, Philadelphia, 1993).

[4] A. Basermann, Conjugate gradients parallelized on the hypercube, *International Journal of Modern Physics C* 4 (1993) 1295–1306.

[5] A. Basermann, Parallel sparse matrix computations in iterative solvers on distributed memory machines, in: D.H. Bailey et al., eds., *Proceedings of the Seventh SIAM Conference on Parallel Processing for Scientific Computing* (SIAM, Philadelphia, 1995) 454–459.

[6] G.H. Golub and C.F. Van Loan, *Matrix Computations* (The Johns Hopkins University Press, Baltimore, second edition, 1989).

[7] G. Pini and G. Gambolati, Is a simple diagonal scaling the best preconditioner for conjugate gradients on supercomputers?, *Adv. Water Resources* 13 (1990) 147–153.

[8] C. Schelthoff and A. Basermann, Polynomial preconditioning for the conjugate gradient method on massively parallel systems, in: K. Ecker, J. Apsel, eds., *Workshop über Parallelverarbeitung* (Informatik-Bericht 95/1, TU Clausthal, Institut für Informatik, 1995) 150–167.

Optimising a Parallel Conjugate Gradient Solver

Martyn R. Field
Hitachi Dublin Laboratory, O'Reilly Institute,
Trinity College, Dublin 2, Ireland.

Abstract

This work arises from the introduction of a parallel iterative solver to a large structural analysis finite element code. The code is called FEX and it was developed at Hitachi's Mechanical Engineering Laboratory. The FEX package can deal with a large range of structural analysis problems using a large number of finite element techniques. FEX can solve either stress or thermal analysis problems of a range of different types from plane stress to a full three-dimensional model. These problems can consist of a number of different materials which can be modelled by a range of material models. The structure being modelled can have the load applied at either a point or a surface, or by a pressure, a centrifugal force or just gravity. Alternatively a thermal load can be applied with a given initial temperature. The displacement of the structure can be constrained by having a fixed boundary or by prescribing the displacement at a boundary.

The original version of FEX uses a skyline direct solver. This solver requires a large amount of storage and limits the size of problem which can be solved on a single processor to a few thousand degrees of freedom. Using a sparse storage format with an iterative solver the size of problem which can solved increases to about 100,000 degrees of freedom. For problems in complicated domains with fine meshes this is still not sufficient. Also the time for convergence can become unacceptably large. Therefore the code was parallelised. The matrices produced by FEX are symmetric and positive definite therefore we use a conjugate gradient solver with a diagonal preconditioner.

The main results in this paper concern the optimisation of the conjugate gradient solver for both the sequential and the parallel version. A profiling of the first version of FEX with the iterative solver showed that both the sequential and parallel codes spent more than 80% of their time in the

CG subroutine and that over 95% of this time is spent in the matrix-vector product subroutine. Therefore we concentrate on optimising this subroutine first. We show that with a few small changes, such as introducing a couple of temporary variables into the main loop and using the Fortran 90 `INTENT` declaration, we achieve a 40% improvement. To gain any further improvement we need to take advantage of the structure of the matrices. In particular we look the matrices generated by FEX for three dimensional problems since these are the largest and therefore take the longest to solve. For three dimensional problems we have 3 degrees of freedom at each vertex of the mesh (the displacements in the 3 Cartesian directions). Therefore the stiffness matrix generated consists of 3 by 3 blocks. We show that we can take advantage of this structure to improve the matrix-vector subroutine by a further 20%.

We also show that, for the types of problems we are interested in, a further improvement in the execution time for conjugate algorithm can be gained by replacing the diagonal scaling preconditioner with an a priori diagonal scaling.

Finally we look at ways of optimising the communication between processors in the parallel version. The communication is performed using MPI and there are three points in the parallel version of the CG algorithm where communication is required between the processors. Two of these are in performing the vector dot-products where the `ALLREDUCE` operation is used. On a modern parallel computer this operation is performed very efficiently so there is little opportunity to improve this. The second point where communication is required is in the matrix-vector product subroutine. After performing a local matrix-vector product the values for nodes shared between two or more processors need to communicated to calculate the correct value. We show that the performance of this operation can be improved by using nonblocking sends and receives. We also show that the quality of the mesh partition is very important. Mesh partition algorithms will balance the number of elements on each processor well but this does not necessarily lead to balance in the number of nodes and hence the size of the local matrix. We tested the parallel code on Hitachi's SR4300 with up to 32 processors and we present results for problems of up to 278,000 degrees of freedom.

Parallel Iterative Solvers and Preconditioners Using Approximate Hierarchical Methods (An Extended Abstract) *

Ananth Grama, Vipin Kumar, Ahmed Sameh
Department of Computer Science,
University of Minnesota
Minneapolis, MN 55455
{ananth, kumar, sameh}@cs.umn.edu

Abstract

In this paper, we report results of the performance, convergence, and accuracy of a parallel GMRES solver for Boundary Element Methods. The solver uses a hierarchical approximate matrix-vector product based on a hybrid Barnes-Hut / Fast Multipole Method. We study the impact of various accuracy parameters on the convergence and show that with minimal loss in accuracy, our solver yields significant speedups. We demonstrate the excellent parallel efficiency and scalability of our solver. The combined speedups from approximation and parallelism represent an improvement of several orders in solution time. We also develop fast and paralellizable preconditioners for this problem. We report on the performance of an inner-outer scheme and a preconditioner based on truncated Green's function. Experimental results on a 256 processor Cray T3D are presented.

1 Introduction

Boundary Element Methods (BEM) are often used to solve integral equations using potential theory. These methods discretize the boundary of the domain into panels. Using the associated Green's function, the potential at each panel is represented as a sum of contributions of every other panel. Applying the Dirichlet boundary condition yields a large scale linear system of equations.

For an n panel boundary discretization, the $n \times n$ linear system arising from this approach is dense. Solving the system using direct factorization methods has a computational complexity of $\Theta(n^3)$. This limits the values of n to a few thousands. Therefore iterative

*This work was supported by IST/BMDO through Army Research Office contract DA/DAAH04-93-G-0080, NSF grant NSG/1RI-9216941, and by Army High Performance Computing Research Center under the auspices of the Department of the Army, Army Research Laboratory cooperative agreement number DAAH04-95-2-0003/contract number DAAH04-95-C-0008, the content of which does not necessarily reflect the position or the policy of the government, and no official endorsement should be inferred. Access to computing facilities were provided by Minnesota Supercomputer Institute, Cray Research Inc. and by the Pittsburgh Supercomputing Center. Related papers are available via WWW at URL: http://www.cs.umn.edu/kumar.

methods are used to solve these systems. The complexity of the underlying matrix-vector product (mat-vec) is $\Theta(n^2)$. For large values of n, this becomes extremely expensive. The physical properties of the underlying domain allow us to use approximate hierarchical techniques that can compute this mat-vec in $\Theta(n)$ or $\Theta(n \log n)$ time.

Parallel formulations of hierarchical methods have been explored in the context of particle dynamics for regular distributions [1, 12, 6, 8, 5] and irregular distributions [3, 4, 2, 9, 11, 10]. These techniques have been explored to solve integral equations by Nabors et al. [7]. In [3], we presented a highly scalable parallel formulation of a dense matrix-vector product for unstructured and adaptive discretizations based on hierarchical methods. Here, we present the performance and accuracy of a GMRES solver based on this parallel mat-vec. We also present techniques for preconditioning these systems and study their performance.

2 Parallel Formulation of Hierarchical Matrix-Vector Product

Hierarchical methods rely on aggregating the effect of distant panels into a series. The combined effect of these panels is then applied using the series rather than individual interactions. Algorithms such as Barnes-Hut and Fast Multipole FMM) are traditionally used for such computations. We use a hybrid of the Barnes-Hut and FMM codes for computing mat-vecs. The hierarchical representation of the domain is identical to FMM. The interactions are computed using the multipole acceptance criteria of the Barnes-Hut method.

Parallel formulations comprise two major steps: tree construction (the hierarchical representation of the domain) and tree traversal. Starting from a distribution of the panels to processors, each processor constructs its local tree. Processors communicate the top nodes in the tree to form a globally consistent image of the tree.

Each processor now proceeds to compute the potential at the panels assigned to it by traversing the tree. On encountering a node that is not locally available, there are two possible scenarios: the panel coordinates can be communicated to the remote processor that evaluates the interaction; or the node can be communicated to the requesting processor. We refer to the former as function shipping and the latter as data shipping. Our parallel formulations are based on the function shipping paradigm. We discuss the advantages of function shipping in [3]. Table 1 presents the runtimes, efficiency and computation rates for up to 266 processors of a Cray T3D for a variety of highly unstructured problems.

Our parallel formulation of the hierarchical mat-vec yields excellent performance for a wide range of processors. The speedup scales almost linearly up to 256 processors even for very small problems (4elt.xyz and g_28060). For the range of problem sizes where hierarchical methods are really useful (pscan and g_108196), the formulation yields a relative speedup of over 3.5 on going from 64 to 256 processors. Clearly, for these problem instances, the overhead due to communication and load imbalance is low. The per-processor performance of the code is about 20 MFLOPS which is respectable considering the code is highly unstructured in nature.

Problem	$p = 64$			$p = 256$		
	Runtime	Eff.	MFLOPS	Runtime	Eff.	MFLOPS
4elt.xyz	0.44	0.84	1220	0.15	0.61	3545
pscan	3.74	0.93	1352	1.00	0.87	5056
g_28060	0.53	0.89	1293	0.16	0.75	4357
g_108196	2.14	0.85	1235	0.61	0.75	4358

Table 1: Runtimes (in seconds), efficiency, and computation rates of the T3D for different problems for $p = 64$ and 256.

3 GMRES Based on Hierarchical Matrix-Vector Product

We implement a parallel restart-GMRES around the hierarchical matrix-vector product. The accuracy of the mat-vec is impacted by a number of parameters such as the degree of the multipole expansion and the multipole acceptance criteria. We study the impact of inaccuracies in the matrix-vector product on the convergence of the solver. We demonstrate that it is possible to get accurate convergence with significant savings in computation time using hierarchical methods. Table 2 presents the Log of relative residual norm for GMRES with various degrees of approximation executed on a 64 processor T3D. The parallel runtime indicates that hierarchical methods are capable of yielding significant savings in time at the expense of slight loss of accuracy.

Table 2: Convergence (Log_{10} of Relative Error Norm) and runtime (in seconds) of the GMRES solver on a 64 processor Cray T3D. The problem consists of 28060 panels.

Iter	Accurate	$\alpha = 0.5$		$\alpha = 0.667$		$\alpha = 1.0$	
degree →		4	8	4	8	4	8
0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
10	-3.015280	-3.015172	-3.015274	-3.015179	-3.015274	-3.018524	-3.015268
20	-4.066225	-4.067801	-4.069889	-4.062450	-4.068376	-4.066583	-4.073438
30	-4.964690	-4.972114	-4.976746	-4.961868	-4.973845	-4.968101	-4.986162
40	-5.722380	-5.828296	-5.836648	-5.816061	-5.832452	-5.824179	-5.858259
50	-5.895243	-6.527266	-6.527266	-6.548455	-6.570732	-6.594213	-6.619036
60		-6.673394	-6.673394	-6.770304	-6.770304	-6.895243	-6.895243
Time		53.93	64.91	35.245	41.34	21.64	25.10

Figure 1 plots the reduction in residual norm with iterations for the accurate and the worst case (most inaccurate mat-vec). It can be seen that even for the worst case accuracy, the residual norms are in near agreement until a relative residual norm of 10^{-6}. Furthermore,

the approximate iterative method runs significantly faster than the accurate one.

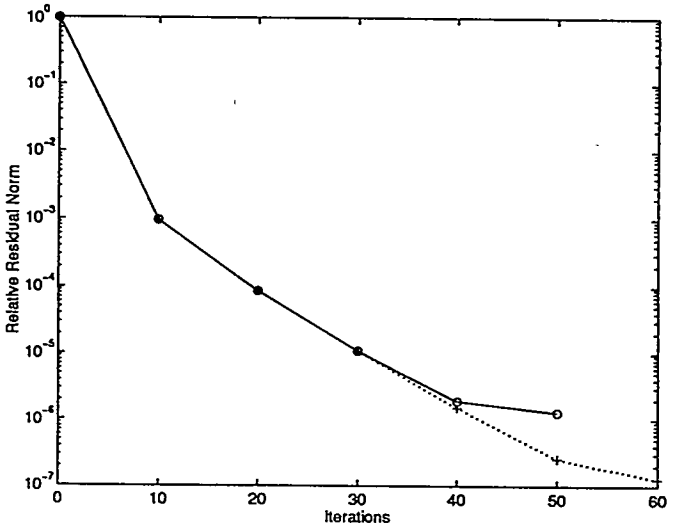


Figure 1: Relative residual norm of accurate and approximate iterative schemes.

4 Preconditioning Techniques

We implement two preconditioning strategies to accelerate the convergence of the GMRES solver:

The hierarchical representation of the domain provides us with a convenient way of approximating the system matrix. Increasing the accuracy of the mat-vec increases the number of direct interactions (and thus the runtime). Conversely, reducing the accuracy (by increasing the α parameter of the Barnes-Hut method, or by decreasing the degree of the multipole series) reduces the runtime. It is therefore possible to visualize a two level scheme in which the inner scheme runs a low accuracy mat-vec. The solve from the inner iteration can be used to accelerate the convergence of the outer iteration. The accuracy of the inner iteration can be increased as the system converges. We have implemented this scheme and preliminary results indicate that for certain accuracies of the inner iteration, this scheme works well.

A primary drawback of the two level scheme implemented above is that the inner iteration is still poorly conditioned. We have also implemented a single level scheme based on truncated Green's functions. In this scheme, the system matrix is approximated by assuming that a panel's influence on another can be neglected if it is sufficiently small. The scheme identifies the k nearest neighbors of each panel. The corresponding matrices are constructed for each panel and factorized. Since we assume that the meshes are not adaptive, the factors need to be computed just once. These factors can be used to precondition the original system. We are currently in the process of evaluating the performance of this scheme. Preliminary results suggest good convergence properties for this scheme.

We expect to present a complete evaluation of both of these schemes in the full-length version of this paper.

References

[1] J. A. Board, J. W. Causey, J. F. Leathrum, A. Windemuth, and K. Schulten. Accelerated molecular dynamics with the fast multipole algorithm. *Chem. Phys. Let.*, 198:89, 1992.

[2] Ananth Grama, Vipin Kumar, and Ahmad Sameh. Scalable parallel formulations of the barnes-hut method for n-body simulations. In *Supercomputing '94 Proceedings*, 1994.

[3] Ananth Grama, Vipin Kumar, and Ahmad Sameh. Parallel matrix-vector product using hierarchical methods. In *Proceedings of Supercomputing '95*, San Diego, CA, 1995.

[4] Ananth Grama, Vipin Kumar, and Ahmed Sameh. On n-body simulations using message passing parallel computers. In *Proceedings of the SIAM Conference on Parallel Processing*, San Francisco, 1995.

[5] L. Greengard and W. Gropp. A parallel version of the fast multipole method. *Parallel Processing for Scientific Computing*, pages 213–222, 1987.

[6] J. F. Leathrum and J. A. Board. Mapping the adaptive fast multipole algorithm into mimd systems. In P. Mehrotra and J. Saltz, editors, *Unstructured Scientific Computation on Scalable Multiprocessors*. MIT Press, Cambridge, MA, 1992.

[7] K. Nabors, F. T. Korsmeyer, F. T. Leighton, and J. White. Multipole accelerated preconditioned iterative methods for three-dimensional potential integral equations of the first kind. *J. on Sci. and Stat. Comp.*, 15(3):713–735, May 1994.

[8] K. E. Schmidt and M. A. Lee. Implementing the fast multipole method in three dimensions. *J. Stat. Phys.*, 63:1120, 1991.

[9] J. Singh, C. Holt, T. Totsuka, A. Gupta, and J. Hennessy. Load balancing and data locality in hierarchical n-body methods. *Journal of Parallel and Distributed Computing*, 1994 (to appear).

[10] M. Warren and J. Salmon. Astrophysical n-body simulations using hierarchical tree data structures. In *Proceedings of Supercomputing Conference*, 1992.

[11] M. Warren and J. Salmon. A parallel hashed oct tree n-body algorithm. In *Proceedings of Supercomputing Conference*, 1993.

[12] F. Zhao and S. L. Johnsson. The parallel multipole method on the connection machine. *SIAM J. of Sci. Stat. Comp.*, 12:1420–1437, 1991.

A Block Variant of the GMRES Method on Massively Parallel Processors

Guangye Li
Cray Research, Inc.
655E Lone Oak Drive
Eagan, MN 55121
e-mail: gli@cray.com

Abstract: This paper presents a block variant of the GMRES method for solving general unsymmetric linear systems. This algorithm generates a transformed Hessenberg matrix by solely using block matrix operations and block data communications. It is shown that this algorithm with block size s, denoted by BVGMRES(s,m), is theoretically equivalent to the GMRES(s*m) method. The numerical results show that this algorithm can be more efficient than the standard GMRES method on a cache based single CPU computer with optimized BLAS kernels. Furthermore, the gain in efficiency is more significant on MPPs due to both efficient block operations and efficient block data communications. Our numerical results also show that in comparison to the standard GMRES method, the more PEs that are used on an MPP, the more efficient the BVGMRES(s,m) algorithm is.

8:00 - 8:30	C. Brooking	Using Sparse LU Factorisation to Precondition GMRES for a Family of Similarly Structured Matrices Arising from Process Modelling
8:30 - 9:00	C.H. Guo	Incomplete Block Factorization Preconditioning for Indefinite Elliptic Problems
9:00 - 9:30	S.D. Kim	Preconditioning Cubic Spline Collocation Method by FEM and FDM for Elliptic Equations
9:30 - 10:00	S. Parter	Preconditioning Chebyshev Spectral Methods by Finite-Element and Finite-Difference Methods

Using sparse LU factorisation to precondition GMRES for a family of similarly structured matrices arising from process modelling

Chris Brooking
University of Bath, UK
cgb@maths.bath.ac.uk

January 11, 1996

Process engineering software is used to simulate the operation of large chemical plants. Such simulations are used for a variety of tasks, including operator training. For the software to be of practical use for this, dynamic simulations need to run in real-time. The models that the simulation is based upon are written in terms of Differential Algebraic Equations (DAE's). In the numerical time-integration of systems of DAE's using an implicit method such as backward Euler, the solution of nonlinear systems is required at each integration point. When solved using Newton's method, this leads to the repeated solution of nonsymmetric sparse linear systems. These systems range in size from 500 to 20,000 variables. A typical integration may require around 3000 timesteps, and if 4 Newton iterates were needed on each time step, then this means approximately 12,000 linear systems must be solved. The matrices produced by the simulations have a similar sparsity pattern throughout the integration. They are also severely ill-conditioned, and have widely-scattered spectra.

To take adventage of the similar sparsity pattern, the current solution strategy is to perform a sparse LU factorisation of the initial matrix, and re-use this factorisation on subsequent matrices. However, this direct method can lead to bottlenecks can occur in the solution procedure if subsequent matrices require additional factorisations. If several factorisations are required on a timestep, then this can result in the simulation taking much longer than real-time for that step.

Applying an iterative method such as GMRES requires effective preconditioning to obtain convergence. Numerical experiments on these matrices with incomplete factorisation methods such as ILU indicate that such preconditioners require large subspace dimensions to obtain convergence, rendering them unsuitable for application to real-time solution of problems of this type.

The method proposed here is to perform a sparse LU factorisation of the matrix from the first linear system to be solved, and to use this to precondition restarted GMRES for subsequent matrices. New preconditioners (ie additional factorisations) are calculated only when GMRES convergence is deemed too slow. The motivation for this approach is to avoid having to perform as many factorisations per timestep as required for the direct method, thus improving real-time performance. Results will be present to show that this method can outperform the direct sparse LU method for real-world simulations that are several times slower than real-time.

Incomplete block factorization preconditioning for indefinite elliptic problems

Chun-Hua Guo

Department of Mathematics and Statistics, University of Calgary, Calgary, Alberta T2N 1N4, Canada

e-mail: guo@math.ucalgary.ca

Summary. The application of the finite difference method to approximate the solution of an indefinite elliptic problem produces a linear system whose coefficient matrix is block tridiagonal and symmetric indefinite. Such a linear system can be solved efficiently by a conjugate residual method, particularly when combined with a good preconditioner. We show that specific incomplete block factorization exists for the indefinite matrix if the mesh size is reasonably small. And this factorization can serve as an efficient preconditioner. Some efforts are made to estimate the eigenvalues of the preconditioned matrix. Numerical results are also given.

Mathematics Subject Classification (1991): 65F10, 65F35, 65F50

1. Introduction

In this paper we consider the numerical solution of the elliptic problem

$$-\nabla \cdot (a(x,y)\nabla u) - p(x,y)u = f \qquad \text{in } \Omega,$$

$$u = g \qquad \text{on } \partial\Omega. \tag{1.1}$$

Here Ω is a connected bounded region in R^2, $a(x,y)$ and $p(x,y)$ are real continuous functions on $\bar{\Omega}$, while f and g are real continuous functions on $\bar{\Omega}$ and $\partial\Omega$, respectively. We further assume that $a(x,y) \geq 1$. (We may assume this without loss of generality when $a(x,y) > 0$.) The function $p(x,y)$ can take large positive values, so the problem (1.1) is generally indefinite.

We discretize (1.1) by using the standard five-point finite difference method (see [15]) with a constant mesh spacing h in both directions. (We always assume that h is small enough so that the discretization makes sense.) The region Ω is replaced by a region formed by connecting the mesh points near $\partial\Omega$ in an obvious way. The values of the approximate solution at the mesh points on the new boundary are obtained from the original boundary condition by simple transition. Arranging the unknowns in the natural ordering, we get a linear system

$$Wx = b, \tag{1.2}$$

where the matrix W (assumed to be nonsingular) is symmetric but generally indefinite, and has the block tridiagonal form

$$W = \begin{bmatrix} W_1 & F_2 & & \\ E_2 & W_2 & \ddots & \\ & \ddots & \ddots & F_m \\ & & E_m & W_m \end{bmatrix}. \tag{1.3}$$

The matrices W_i are symmetric, (point) tridiagonal, and may have zero elements on the two diagonals adjacent to the main diagonal when Ω is not convex. The matrices

$E_i(=F_i^T)$ have at most one nonzero element in each row or column. The reader may wish to form the matrix W for a fixed region Ω (not necessarily simply connected). Our analyses will rely heavily on the formation of the matrix. We denote by W_0 the matrix (1.3) corresponding to the case $p(x,y) \equiv 0$. W_0 is a symmetric M-matrix and thus positive definite (see [15]).

The linear system (1.2) can be solved efficiently by conjugate gradient type methods. In this paper we use the MCR method proposed in [5](see [14] for other efficient methods). The convergence of the MCR method will depend on the distribution of the eigenvalues of the matrix W. A good preconditioner is thus desirable. Normally we can use a symmetric definite matrix C as a preconditioner even though the matrix W may be indefinite. The MCR method is then applied to the equivalent linear system $W'x' = b'$ with $W' = C^{-\frac{1}{2}}WC^{-\frac{1}{2}}, b' = C^{-\frac{1}{2}}b$, and $x' = C^{\frac{1}{2}}x$. After rewriting the resulting algorithm we get the preconditioned MCR algorithm. This new algorithm takes the same form as the original MCR algorithm except that the solution of a linear system of the type $Cy = d$ is needed in each iterative step. Thus the preconditioning matrix C should be chosen such that the solution of $Cy = d$ is inexpensive and the distribution of the eigenvalues of $C^{-\frac{1}{2}}WC^{-\frac{1}{2}}$ (or $C^{-1}W$) is much more favorable for the MCR method.

When the values of the function $p(x,y)$ are small in magnitude, it is a good idea to use a good preconditioner for the matrix W_0 also as a preconditioner for the matrix W (cf.[17],[18]). But this strategy tends to be unsatisfactory as the values of the function $p(x,y)$ get larger. It is tempting to consider the incomplete factorization of the matrix W since it is a perturbation of an M-matrix.

The incomplete factorization method has been extensively studied over the past

eighteen years or so. The analyses have been made mostly for M- or H-matrices (see, e.g., [1],[4],[6],[8],[11],[16]). In fact, it was shown in [16] that only for H-matrices can incomplete factorizations be *universally* carried out. However, for matrices arising in applications, only specific incomplete factorizations are of practical interest. And for these incomplete factorizations to be well defined, the matrices need not necessarily be H-matrices.

In Section 2, we show that specific incomplete block factorization exists for the matrix W if the mesh size h is reasonably small. In Section 3, we give a result on bounds for eigenvalues of preconditioned matrices. In particular, we get a specific upper bound for the eigenvalues of $C^{-1}W$, where C is the preconditioner obtained from the incomplete block factorization. In Section 4, we present some numerical results to illustrate the effectiveness of incomplete block factorization as a preconditioner for indefinite matrices.

2. Existence of incomplete block factorization

We begin with some notations. If $A = [a_{ij}]$ and $B = [b_{ij}]$ are real matrices, then $A \geq B$ if $a_{ij} \geq b_{ij}$ for all i, j. For square matrix $A = [a_{ij}]$, we let $A^{(p)}$ denote the matrix $[b_{ij}]$ with

$$b_{ij} = \begin{cases} a_{ij}, & |i-j| \leq p, \\ 0, & |i-j| > p. \end{cases}$$

Let A be a square matrix in block tridiagonal form

$$A = \begin{bmatrix} A_1 & U_2 & & & \\ L_2 & A_2 & \ddots & & \\ & \ddots & \ddots & U_m & \\ & & L_m & A_m & \end{bmatrix} = D + L + U,$$

where the A_i's are square matrices, not necessarily of the same size; L and U are strictly lower and upper block triangular matrices, respectively. Consider the recursion (cf.[6])

$$X_1 = A_1,$$

$$X_r = A_r - L_r(X_{r-1}^{-1})^{(p)}U_r, \quad r = 2, 3, ..., m. \tag{2.1}$$

If the X_i's are nonsingular, $C = (X + L)X^{-1}(X + U)$ is called the *incomplete block factorization* of A, where $X = \text{diag}(X_i)_{i=1}^m$.

Now let $\widehat{D} = \text{diag}(\widehat{D}_i)_{i=1}^m$ be any nonsingular (point) diagonal matrix of the same size and partitioning as A, and

$$\tilde{A} = \widehat{D}A\widehat{D} = \begin{bmatrix} \tilde{A}_1 & \tilde{U}_2 & & & \\ \tilde{L}_2 & \tilde{A}_2 & \ddots & & \\ & \ddots & \ddots & \tilde{U}_m & \\ & & \tilde{L}_m & \tilde{A}_m & \end{bmatrix} = \tilde{D} + \tilde{L} + \tilde{U}.$$

We can then consider the recursion

$$\widetilde{X}_1 = \tilde{A}_1,$$

$$\widetilde{X}_r = \tilde{A}_r - \tilde{L}_r(\widetilde{X}_{r-1}^{-1})^{(p)}\tilde{U}_r, \quad r = 2, 3, ..., m. \tag{2.2}$$

If the $\widetilde{X}_i$'s are nonsingular, $\tilde{C} = (\widetilde{X} + \tilde{L})\widetilde{X}^{-1}(\widetilde{X} + \tilde{U})$ is the incomplete block factorization of $\tilde{A}$, where $\widetilde{X} = \text{diag}(\widetilde{X}_i)_{i=1}^m$.

It can be easily checked that the matrices $\widetilde{X}_i$ from (2.2) are nonsingular if and only if the matrices X_i from (2.1) are nonsingular. Moreover, $\widetilde{X}_i = \widehat{D}_i X_i \widehat{D}_i$ $(i = 1, 2, ..., m)$, $\widetilde{C} = \widehat{D}C\widehat{D}$, and $\widetilde{C}^{-1}\widetilde{A} = \widehat{D}^{-1}C^{-1}A\widehat{D}$.

With the above observation in mind, we turn our attention to the matrix W described in Section 1.

Since $a(x, y)$ is uniformly continuous on $\bar{\Omega}$, for every ϵ_0 $(0 < \epsilon_0 < 1)$ there exists an h_0 such that for all (x_1, y_1) and (x_2, y_2) in $\bar{\Omega}$

(2.3) $\quad |a(x_2, y_2) - a(x_1, y_1)| \leq \epsilon_0$ whenever $|x_2 - x_1| + |y_2 - y_1| \leq h_0$.

Let D' be the diagonal matrix whose diagonal elements are the values of $a(x, y)$ at the mesh points (in natural ordering) and

$$C_1 = \max_{x \in \bar{\Omega}} \frac{p(x, y)}{a(x, y)} > 0, \qquad C_2 = \min_{x \in \bar{\Omega}} \frac{p(x, y)}{a(x, y)}.$$

It is not difficult to see that for $h \leq 2h_0$ the *nonzero* elements of the matrix $\widehat{W} = ((1 + \epsilon_0)D')^{-\frac{1}{2}} W ((1 + \epsilon_0)D')^{-\frac{1}{2}}$ are such that

(2.4) $$\frac{4(1-\epsilon_0) - C_1 h^2}{1+\epsilon_0} \leq (\widehat{W})_{i,i} \leq \frac{4(1+\epsilon_0) - C_2 h^2}{1+\epsilon_0},$$

and

(2.5) $$-1 \leq (\widehat{W})_{i,j} \leq -\frac{1-\epsilon_0}{1+\epsilon_0} < 0 \ (i \neq j).$$

Let

(2.6) $$\widehat{W} = \begin{bmatrix} \widehat{W}_1 & \widehat{F}_2 & & \\ \widehat{E}_2 & \widehat{W}_2 & \ddots & \\ & \ddots & \ddots & \widehat{F}_m \\ & & \widehat{E}_m & \widehat{W}_m \end{bmatrix},$$

and consider the recursion

$$\widehat{X}_1 = \widehat{W}_1,$$

$$\widehat{X}_r = \widehat{W}_r - \widehat{E}_r(\widehat{X}_{r-1}^{-1})^{(1)}\widehat{F}_r, \quad r = 2, 3, ..., m. \tag{2.7}$$

We will show that the symmetric matrices $\widehat{X}_i$ are all M-matrices if the mesh size h is reasonably small. The following lemmas will be needed.

Lemma 2.1. (cf.[7],[12]) *Let $A \in R^{n,n}$ be an M-matrix. If the elements of $B \in R^{n,n}$ satisfy the relations*

$$b_{ii} \geq a_{ii},\ a_{ij} \leq b_{ij} \leq 0,\ i \neq j,\ 1 \leq i, j \leq n,$$

then B is also an M-matrix. Moreover, $B^{-1} \leq A^{-1}$.

Let

$$T_p = \begin{bmatrix} p & -1 & & & \\ -1 & p & \ddots & & \\ & \ddots & \ddots & -1 & \\ & & \ddots & & \\ & & & -1 & p \end{bmatrix}_{n\times n},$$

we have

Lemma 2.2. (cf.[13]) *For $j \geq i$, and $p > 2$,*

$$(T_p^{-1})_{i,j} = \frac{(r_+^i - r_-^i)(r_+^{n-j+1} - r_-^{n-j+1})}{(r_+ - r_-)(r_+^{n+1} - r_-^{n+1})},$$

where $r_\pm$ are the two solutions of the quadratic equation $r^2 - pr + 1 = 0$.

The next lemma provides a Toeplitz matrix upper bound for the matrix $(T_p^{-1})^{(1)}$. It follows readily from Lemma 2.2.

Lemma 2.3.

$$(T_p^{-1})_{i,i} \leq \frac{1}{\sqrt{p^2-4}}, \quad (T_p^{-1})_{i,i+1} = (T_p^{-1})_{i+1,i} \leq \frac{p - \sqrt{p^2-4}}{2\sqrt{p^2-4}}.$$

Proof. $(r_+^i - r_-^i)(r_+^{n-i+1} - r_-^{n-i+1}) = r_+^{n+1} + r_-^{n+1} - r_-^i r_+^{n-i+1} - r_+^i r_-^{n-i+1}$

$$\leq r_+^{n+1} + r_-^{n+1} - r_-^i r_-^{n-i+1} - r_-^i r_-^{n-i+1} = r_+^{n+1} - r_-^{n+1},$$

so

$$(T_p^{-1})_{i,i} \le \frac{1}{r_+ - r_-} = \frac{1}{\sqrt{p^2-4}}.$$

Similarly,

$$(r_+^i - r_-^i)(r_+^{n-i} - r_-^{n-i}) \le r_+^n - r_-^n \le (r_+^{n+1} - r_-^{n+1})r_-,$$

so

$$(T_p^{-1})_{i,i+1} = (T_p^{-1})_{i+1,i} \le \frac{r_-}{r_+ - r_-} = \frac{p-\sqrt{p^2-4}}{2\sqrt{p^2-4}}. \qquad \square$$

We now return to the recursion (2.7), and let

$$b = \frac{4(1-\epsilon_0) - C_1 h^2}{1+\epsilon_0}.$$

By (2.4) and (2.5),

$$\widehat{X}_1 = \widehat{W}_1 \ge \begin{bmatrix} x_1 & -y_1 & & & \\ -y_1 & x_1 & \ddots & & \\ & \ddots & \ddots & \ddots & -y_1 \\ & & \ddots & \ddots & \\ & & & -y_1 & x_1 \end{bmatrix} = Z_1$$

with $x_1 = b$ and $y_1 = 1$. If $x_1 > 2y_1$, then Z_1 is an M-matrix. By Lemma 2.1 and (2.5), $\widehat{X}_1$ is also an M-matrix, and $\widehat{X}_1^{-1} \le Z_1^{-1}$. Thus

$$\widehat{X}_2 = \widehat{W}_2 - \widehat{E}_2(\widehat{X}_1^{-1})^{(1)}\widehat{F}_2 \ge T_b - \frac{1}{y_1}\widetilde{E}_2(T_{\frac{x_1}{y_1}}^{-1})^{(1)}\widetilde{F}_2,$$

where the matrices $\widetilde{E}_2$ and $\widetilde{F}_2$ are obtained from $\widehat{E}_2$ and $\widehat{F}_2$ by replacing all of their nonzero elements by -1.

Now we have, by Lemma 2.3,

$$\widehat{X}_2 \geq \begin{bmatrix} x_2 & -y_2 & & & \\ -y_2 & x_2 & \ddots & & \\ & \ddots & \ddots & \ddots & -y_2 \\ & & \ddots & \ddots & \\ & & & -y_2 & x_2 \end{bmatrix} = Z_2,$$

where

$$x_2 = b - \frac{1}{\sqrt{x_1^2 - 4y_1^2}}, \quad y_2 = 1 + \frac{x_1 - \sqrt{x_1^2 - 4y_1^2}}{2y_1\sqrt{x_1^2 - 4y_1^2}}.$$

If $x_2 > 2y_2$, then Z_2 is an M-matrix. So $\widehat{X}_2$ is also an M-matrix, and $\widehat{X}_2^{-1} \leq Z_2^{-1}$.

Continuing with the recursion (2.7), we have the following result.

Proposition 2.1. *The symmetric matrices $\widehat{X}_i$ in (2.7) are all M-matrices if the recursion*

$$\begin{gathered} x_1 = b \ (b > 2), \ y_1 = 1; \\ x_{n+1} = b - \frac{1}{\sqrt{x_n^2 - 4y_n^2}}, \ y_{n+1} = 1 + \frac{x_n - \sqrt{x_n^2 - 4y_n^2}}{2y_n\sqrt{x_n^2 - 4y_n^2}}; \\ n = 1, 2, \ldots \end{gathered} \tag{2.8}$$

is well defined, i.e., if $x_n > 2y_n$ for all n.

We record some simple properties of the recursion (2.8).

Proposition 2.2.

1. *If $\{x_n\}_{n=1}^k$ and $\{y_n\}_{n=1}^k$ can be generated for some $b > 2$, then $x_1 > x_2 > \ldots > x_k$ and $y_1 < y_2 < \ldots < y_k$.*

2. *If $b' > b$, then $\{x_n'\}_{n=1}^k$ and $\{y_n'\}_{n=1}^k$ can also be generated for b', moreover, $x_n' > x_n$ and $y_n' < y_n$, $n = 1, 2, \ldots, k$.*

3. *If $b \geq 3.7$, then the sequences $\{x_n\}_{n=1}^\infty$ and $\{y_n\}_{n=1}^\infty$ can be generated, i.e., $x_n > 2y_n$ for all n.*

Proof. (1) and (2) can be easily checked. In view of (2), we need only prove (3) for b=3.7. But when b=3.7, we can prove by induction that $x_n > 3.233$, $y_n < 1.210$ for all n. □

Consider the matrix W in (1.3) and the recursion

$$X_1 = W_1,$$

$$X_r = W_r - E_r(X_{r-1}^{-1})^{(1)}F_r, \quad r = 2,3,...,m, \tag{2.9}$$

we have the following result.

Proposition 2.3. *The symmetric matrices X_i in (2.9) are all M-matrices if*

$$h \leq \min\left(2h_0, \sqrt{\tfrac{0.3-7.7\epsilon_0}{C_1}}\right) \tag{2.10}$$

for some pair (ϵ_0, h_0) satisfying (2.3) with $\epsilon_0 < \frac{30}{77}$.

Proof. This follows from the observation we made earlier in this section, Proposition 2.1 and Proposition 2.2(3). □

The restriction on h in the above proposition is much less strigent than in an earlier result ([9]). That result was obtained by a careful application of a more general result therein.

Example 2.1. If $\Omega = (0,1) \times (0,1)$, $a(x,y) = 1$, and $p(x,y) = 800$. then we know from Proposition 2.3 that the incomplete block factorization exists when $h \leq \frac{1}{52}$.

Example 2.2. If $\Omega = (0,1) \times (0,1)$, $a(x,y) = 1 + x^2 + y^2$, and $p(x,y) = 400$, then the incomplete block factorization exists when $h \leq \frac{1}{52}$.

3. Estimates for eigenvalues of preconditioned matrices

In Section 2, we have seen that the recursion (2.9) for the symmetric matrix W in (1.3) is well defined if condition (2.10) is satisfied. We denote by C the corresponding

incomplete block factorization of W. Clearly C is symmetric positive definite. The matrix C will then be used as a preconditioner for the linear system (1.2). Although $C^{-1}W$ has the same number of negative eigenvalues as W, the distribution of the eigenvalues will hopefully be more favorable for the iterative method.

When a matrix $A \in R^{n,n}$ is symmetric or similar to a symmetric matrix, we arrange its eigenvalues in an increasing order:

$$\lambda_1(A) \leq \lambda_2(A) \leq ... \leq \lambda_n(A).$$

The following general result gives some information about the eigenvalues of the matrix $C^{-1}W$.

Theorem 3.1. ([10]) *Let $A \in R^{n,n}$ be a symmetric matrix, $S \in R^{n,n}$ be a symmetric positive definite matrix. Then*

(1) If $\lambda_i(A) > 0$ and $\lambda_n(\mu S - A) \geq 0$ with $\mu > 0$, then

$$\lambda_i(S^{-1}A) \geq \frac{\mu\lambda_i(A)}{\lambda_i(A) + \lambda_n(\mu S - A)};$$

(2) If $\lambda_i(A) > 0$ and $\lambda_1(\mu S - A) \geq 0$ with $\mu > 0$ then

$$\lambda_i(S^{-1}A) \leq \frac{\mu\lambda_i(A)}{\lambda_i(A) + \lambda_1(\mu S - A)};$$

(3) If $\lambda_i(A) < 0$ and $\lambda_n(\mu S - A) \leq 0$ with $\mu < 0$, then

$$\lambda_i(S^{-1}A) \geq \frac{\mu\lambda_i(A)}{\lambda_i(A) + \lambda_n(\mu S - A)};$$

(4) If $\lambda_i(A) < 0$ and $\lambda_1(\mu S - A) \leq 0$ with $\mu < 0$, then

$$\lambda_i(S^{-1}A) \leq \frac{\mu\lambda_i(A)}{\lambda_i(A) + \lambda_1(\mu S - A)}.$$

Next we will apply part (2) of the theorem to get a more specific upper bound for the eigenvalues of $C^{-1}W$.

We express the matrix $\widehat{W}$ in (2.6) as $\widehat{W} = \widehat{D} + \hat{L} + \hat{U}$, where $\hat{L}$ and $\hat{U}$ are strictly lower and upper block triangular matrices, respectively. When condition (2.10) is satisfied, we let $\hat{C}$ be the incomplete block factorization of $\widehat{W}$ obtained from the recursion (2.7). So $\hat{C} = (\widehat{X} + \hat{L})\widehat{X}^{-1}(\widehat{X} + \hat{U})$ with $\widehat{X} = \text{diag}(\widehat{X}_i)_{i=1}^m$. The matrix $\hat{C}$ is again symmetric positive definite. Since $\hat{C}^{-1}\widehat{W}$ is similar to $C^{-1}W$, we will apply Theorem 3.1(2) to the pair of matrices $\widehat{W}$ and $\hat{C}$.

As in [2], we let $V = (1 - 1/\mu)\widehat{X} + \hat{L}$ and find

$$\mu\hat{C} - \widehat{W} = \mu V\widehat{X}^{-1}V^T + (2 - \frac{1}{\mu})\widehat{X} - \widehat{D}.$$

Since $\mu V\widehat{X}^{-1}V^T$ is positive semidefinite for any $\mu > 0$, it follows that

$$\lambda_1(\mu\hat{C} - \widehat{W}) \geq \lambda_1((2 - \frac{1}{\mu})\widehat{X} - \widehat{D}).$$

We will find $\mu \geq 1$ such that

$$\lambda_1((2 - \tfrac{1}{\mu})\widehat{X} - \widehat{D}) \geq 0. \tag{3.1}$$

This μ will then be an upper bound for the eigenvalues of $C^{-1}W$.

(3.1) is clearly equivalent to

$$\lambda_1((2 - \tfrac{1}{\mu})\widehat{X}_i - \widehat{W}_i) \geq 0,\ 1 \leq i \leq m. \tag{3.2}$$

Consider the recursion (2.8) with $3.7 \leq b \leq 4$; let $x^* = \lim_{n\to\infty} x_n$, $y^* = \lim_{n\to\infty} y_n$ (the limits exist by Proposition 2.2). It is readily found that $3.234 < x^* < 3.654$, $1.134 < y^* < 1.210$. If $\widehat{W}_i$ has no zero elements on the two diagonals adjacent to the

main diagonal, we have by (2.4) and (2.5)

$$\widehat{W}_i \le \begin{bmatrix} a_1 & -b_1 & & & \\ -b_1 & a_1 & \ddots & & \\ & \ddots & \ddots & \ddots & -b_1 \\ & & \ddots & \ddots & \\ & & & -b_1 & a_1 \end{bmatrix},$$

with

$$a_1 = \frac{4(1+\epsilon_0) - C_2 h^2}{1+\epsilon_0}, \quad b_1 = \frac{1-\epsilon_0}{1+\epsilon_0}.$$

By the argument leading to Proposition 2.1 and in view of Proposition 2.2, we have

$$\widehat{X}_i \ge \begin{bmatrix} x^* & -y^* & & & \\ -y^* & x^* & \ddots & & \\ & \ddots & \ddots & \ddots & -y^* \\ & & \ddots & \ddots & \\ & & & -y^* & x^* \end{bmatrix}.$$

Thus

$$(2-\frac{1}{\mu})\widehat{X}_i - \widehat{W}_i \ge \begin{bmatrix} a_2 & -b_2 & & & \\ -b_2 & a_2 & \ddots & & \\ & \ddots & \ddots & \ddots & -b_2 \\ & & \ddots & \ddots & \\ & & & -b_2 & a_2 \end{bmatrix},$$

with

$$a_2 = (2-\frac{1}{\mu})X^* - a_1, \quad b_2 = (2-\frac{1}{\mu})y^* - b_1.$$

Since $\mu \ge 1$, the offdiagonal elements of $(2-1/\mu)\widehat{X}_i - \widehat{W}_i$ are nonpositive. Therefore, (3.2) is satisfied for the given i if $a_2 \ge 2b_2$, or $(2x^* - 4y^* - a_1 + 2b_1)\mu \ge x^* - 2y^*$. If $\widehat{W}_i$ has zero elements on the two diagonals adjacent to the main diagonal, the corresponding elements in the above lower bound for $\widehat{X}_i$ can also be changed to zero

(this can be seen by examining the procedure leading to Proposition 2.1). And the above analysis can be made on proper submatrices of $\widehat{X}_i$ and $\widehat{W}_i$.

We have thus reached the following conclusion.

Proposition 3.1. *With previous notations, if*

$$h \leq \min(2h_0, \sqrt{\frac{0.3 - 7.7\epsilon_0}{C_1}}),$$

and

$$\mu = \frac{x^* - 2y^*}{2x^* - 4y^* - a_1 + 2b_1} \geq 1,$$

then μ is an upper bound for the eigenvalues of $C^{-1}W$.

From the above proposition we can conclude that, in the $h \to 0$ limit, $\mu = 1.7$ is an upper bound for the eigenvalues of $C^{-1}W$.

Example 3.1. If $\Omega = (0,1) \times (0,1)$, $a(x,y) = 1 + x^2 + y^2$, $p(x,y) = 400$, and $h = \frac{1}{100}$, we can take $h_0 = \frac{1}{200}$, and $\epsilon_0 = \frac{1}{100}$. We find by application of Proposition 3.1 that $\mu = 2.87$ is an upper bound for the eigenvalues of $C^{-1}W$.

4. Numerical results

For test purposes we consider the following special case of problem (1.1):

$$-\triangle u - \sigma u = 1 \qquad \text{in } \Omega = (0,1) \times (0,1),$$

$$u = 0 \qquad \text{on } \partial\Omega,$$

where σ is a real constant.

The corresponding linear system $Wx = b$ will be solved by the MCR method, with or without preconditioning. The matrices W and W_0 are now related by $W = W_0 - \sigma h^2 I$.

For preconditioning, we use the following two preconditioners:

- Method 1: Preconditioner C is the incomplete block factorization of W. Its existence will be guaranteed by Proposition 2.3 (cf. Example 2.1).
- Method 2: Preconditioner C_0 is the modified incomplete block factorization of W_0. C_0 is a very good preconditioner for W_0 (see [2,p346]).

We note that (for variable coefficient problems) the computational work per iteration for the preconditioned MCR method is less than twice that for the unpreconditioned MCR method.

In our numerical experiments we use double precision and use the zero vector as initial guess. The algorithm is terminated as soon as $\| r_k \|_2 / \| r_0 \|_2 \leq 10^{-6}$, where r_0 and r_k are the residuals at the initial step and the kth iterative step, respectively. We give the number of MCR iterations in Tables 1 and 2. The numbers in parentheses for Method 1 are the upper bounds for the eigenvalues of $C^{-1}W$, obtained by application of Proposition 3.1.

From the test results we observe that Method 2 works well only when σ is relatively small. When σ gets larger, Method 1 gives much better performance than Method 2. Compared with the unpreconditioned MCR method, Method 1 becomes less effective as σ gets larger. But it still gives improvement when σ is as large as 800. When the mesh size is halved, the number of MCR iterations increases normally by a multiple of 2 if no preconditioner is used. But the increase is generally much slower for both Method 1 and Method 2.

Table 1. The number of MCR iterations for $h = \frac{1}{96}$

σ	No preconditioning	Method 1	Method 2
0	148	29 (1.70)	19
50	158	30 (1.72)	28
100	188	49 (1.73)	41
150	182	46 (1.74)	44
200	191	48 (1.76)	46
250	201	72 (1.77)	79
300	200	71 (1.78)	70
350	207	73 (1.80)	85
400	207	78 (1.82)	109
450	226	84 (1.83)	119
500	257	103 (1.85)	156
550	250	101 (1.87)	157
600	248	95 (1.89)	174
650	248	98 (1.91)	198
700	255	111 (1.93)	208
750	262	117 (1.95)	220
800	291	147 (1.97)	298

Table 2. The number of MCR iterations for $h = \frac{1}{192}$

σ	No preconditioning	Method 1	Method 2
0	297	49 (1.70)	28
50	316	55 (1.71)	43
100	375	92 (1.71)	64

150	354	71 (1.71)	66
200	382	87 (1.72)	78
250	428	106 (1.72)	122
300	425	107 (1.72)	116
350	436	133 (1.73)	164
400	436	115 (1.73)	176
450	471	128 (1.73)	196
500	516	183 (1.73)	250
550	495	149 (1.74)	244
600	491	141 (1.74)	364
650	489	159 (1.74)	353
700	510	158 (1.75)	338
750	523	187 (1.75)	384
800	572	210 (1.76)	452

Acknowledgement. The author would like to thank Dr. Peter Lancaster for reading the manuscript and providing useful suggestions.

References

1. Axelsson, O. (1986): A general incomplete block-matrix factorization method. Linear Algebra Appl. 74, 179-190

2. Axelsson, O. (1992): Bounds of eigenvalues of preconditioned matrices. SIAM J. Matrix Anal. Appl. 13, 847-862

3. Axelsson, O. (1994): Iterative Solution Methods. Cambridge University Press

4. Beauwens, R., Ben Bouzid, M. (1987): On sparse block factorization iterative methods. SIAM J. Numer. Anal. 24, 1066-1076

5. Chandra, R., Eisenstat, S.C., Schultz, M.H. (1977): The modified conjugate residual method for partial differential equations. In: R. Vichnevetsky, ed., Advances in Computer Methods for Partial Differential Equations II, pp. 13-19. IMACS

6. Concus, P., Golub, G.H., Meurant, G. (1985): Block preconditioning for the conjugate gradient method. SIAM J. Sci. Stat. Comput. 6, 220-252

7. Fan, K. (1960): Note on M-matrices. Quart. J. Math. Oxford (2) 11, 43-49

8. Guo, C.-H. (1991): Some results on sparse block factorization iterative methods. Linear Algebra Appl. 145, 187-199

9. Guo, C.-H. : Incomplete block factorization preconditioning for linear systems arising in the numerical solution of the Helmholtz equation. Appl. Numer. Math., to appear

10. Guo, C.-H. : Some observations on bounds for eigenvalues of preconditioned matrices. Linear Algebra Appl., submitted

11. Manteuffel, T.A. (1980): An incomplete factorization technique for positive definite linear systems. Math. Comput. 34, 473-497

12. Meijerink, J.A., van der Vorst, H.A. (1977): An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. Math. Comput. 31, 148-162

13. Meurant, G. (1992): A review on the inverse of symmetric tridiagonal and block tridiagonal matrices. SIAM J. Matrix Anal. Appl. 13, 707-728

14. Stoer, J., Freund, R. (1982): On the solution of large indefinite systems of linear equations by conjugate gradient algorithms. In: R. Glowinski, J.L. Lions, eds., Computing Methods in Applied Sciences and Engineering V, pp. 35-53. North-Holland, Amsterdam

15. Varga, R.S. (1962): Matrix Iterative Analysis. Prentice-Hall, Englewood Cliffs, New Jersey

16. Varga, R.S., Saff, E.B., Mehrmann, V. (1980): Incomplete factorizations of matrices and connections with H-matrices. SIAM J. Numer. Anal. 17, 787-793

17. Vassilevski, P.S. (1992): Indefinite elliptic problem preconditioning. Commun. Appl. Numer. Methods 8, 257-264

18. Yserentant, H. (1989): Preconditioning indefinite discretization matrices. Numer. Math. 54, 719-734

Preconditioning Cubic Spline Collocation method by FEM and FDM for Elliptic Equations

Sang Dong Kim *

January 17, 1996

ABSTRACT

In this talk we discuss the finite element and finite difference technique for the cubic spline collocation method. For this purpose, we consider the uniformly elliptic operator A defined by $Au := -\Delta u + a_1 u_x + a_2 u_y + a_0 u$ in Ω (the unit square) with Dirichlet or Neumann boundary conditions and its discretization based on Hermite cubic spline spaces and collocation at the Gauss points. Using an interpolatory basis with support on the Gauss points one obtains the matrix $\hat{A}_N(h = \frac{1}{N})$.

We discuss the H^1 condition numbers and the distribution of $\tilde{\beta}_N$-singular values of the preconditioned matrices $(\tilde{\beta}_N)^{-1}\hat{A}_N$ or $(\tilde{L}_N)^{-1}\hat{A}_N$ where $\tilde{\beta}_N$ and $\tilde{L}_N$ are the stiffness matrix and the matrix of finite difference operator respectively associated with the finite element or finite difference discretization of the positive definite uniformly elliptic operator B given by $Bv := -\Delta v + b_0 v$ in Ω with boundary conditions matching to the boundary conditions of A.

The finite element and the finite difference spaces are considered as the space of continuous functions which are bilinear on the rectangles determined by Gauss points. When $A = B$ we obtain results on the eigenvalues of $\tilde{\beta}_N^{-1}\hat{A}_N$. Such estimates are used for the iterative method like CGM, damped Jacobi iteration, GMRESS or other methods.

*Mathematics Department, Teachers College, KyungPook National University, Taegu, Korea. Supported by KOSEF, Korea

Preconditioning Chebyshev Spectral Methods by Finite-Element and Finite-Difference Methods

Seymour V. Parter (Speaker)
Mathematics Department
University of Wisconsin–Madison
Madison, WI 53706

and

Sang Dong Kim
Department of Mathematics
Kyungpook National University
Taegu, 702-701, Korea

Abstract

Let Ω be the square $[-1,1] \times [-1,1]$ and let Γ be its boundary. Consider the elliptic boundary value problem

$$Au := -[u_{xx} + u_{yy}] + a_1(x,y)u_x + a_2(x,y)u_y + a(x,y)u = f, \text{ in } \Omega \tag{1.1a}$$

with Dirichlet boundary conditions

$$u = 0 \quad \text{on } \Gamma. \tag{1.1a}$$

The coefficients a_1, a_2, a are smooth and $f(x,y) \in C(\overline{\Omega})$. We assume the operator A is invertible. A popular and very accurate method for obtaining approximate solutions of this boundary value problem is the method of Chebyshev Spectral Collocation.

Let $\mathbb{P}_N^0$ denote the space of polynomials which are of degree N in x, of degree N in y and satisfy the boundary condition (1.1b). Let

$$x_j = -\cos\frac{\pi_j}{N}, \qquad j = 0,1,2,\ldots,N \tag{1.2a}$$

so that

$$-1 = x_0 < x_1 < \cdots x_{N-1} < x_N = 1. \tag{1.2b}$$

These are the Chebyshev-Gauss-Lobatto [CGL] points. The two dimensional [CGL] points are defined as

$$q_\eta = (x_i, x_j), \qquad \eta = i + (N-1)(j-1). \tag{1.2c}$$

The Chebyshev (spectral) collocation method for the solution of (1.1a), (1.1b), is: find $p(x,y) \in \mathbb{P}_N^0$ such that

$$[Ap](q_\eta) = f(q_\eta) = f(x_i, x_j). \tag{1.3}$$

The solution $p(x,y) \in \mathbb{P}_N^0$ is a very good approximation to $u(x,y)$, the solution of (1.1a), (1.1b). Indeed, we have "spectral accuracy". That is, there is a constant $c > 0$ such that if the solution $u(x,y) \in H_w^m(\Omega)$, then

$$|u-p|_{1,w} + N\|u-p\|_{0,w} \le cN^{1-m}\|u\|_{m,w} \tag{1.4}$$

where

$$\|u\|_{0,w}^2 = \int\int_\Omega u^2 w_2 dxdy \tag{1.5a}$$

$$|u|_{1,w}^2 = \int\int_\Omega |\nabla u|^2 w_2 dxdy \tag{1.5b}$$

and

$$w_2(x,y) = [(1-x^2)(1-y^2)]^{-1/2}. \tag{1.5c}$$

We choose the Lagrange basis for $\mathbb{P}_N^0$. That is

$$\varphi_k(x) = \frac{\prod_{j\neq k}(x-x_j)}{\prod_{j\neq k}(x_k - x_j)}, \quad k = 1, 2, \ldots, (N-1).$$

Then

$$\Phi_\eta(x,y) = \varphi_i(x), \varphi_j(y), \qquad \eta = i + (N-1)j$$

is the Lagrange basis for $\mathbb{P}_N^0$. Using this basis the system (1.3) is represented as a system of linear equation

$$A_N P = F. \tag{1.6}$$

The matrix A_N is badly conditioned and (essentially) full.

In this talk we consider two preconditioners based on the space of continuous piecewise bilinear functions determined by their values at the two dimensional [CGL]

points (x_i, x_j). Specifically, let $\psi_i(x)$ be the continuous piecewise linear hat function which satisfies

$$\psi_i(x_j) = \delta_{ij}, \quad i = 1, 2, \ldots, N-1, \quad j = 0, 1, \ldots, N. \tag{1.7a}$$

and let

$$\Psi_\eta(x, y) = \psi_i(x)\psi_j(y), \quad \eta = i + (N-1)j. \tag{1.7b}$$

Our first preconditioner is the symmetric positive definite matrix Q_N given by

$$(Q_N)_{\eta,\mu} = \int\int_\Omega \nabla\Psi_\eta \cdot \nabla\Psi_\mu w_2 dxdy. \tag{1.8}$$

Our second preconditioner is the matrix L_N associated with the finite-difference operator for the Lapace operator bsed on the two dimensional [CGL] points.

In both these cases we are dealing with preconditioners which arise from different finite element spaces and different discretization approaches.

The main results are

Theorem 1: *Consider the special case where A is a Helmholtz operator, i.e. $a_1 = a_2 = 0$ and*

$$a \geq 0. \tag{1.9}$$

There are constants $0 < \Lambda_0 < \Lambda_1$ (independent of N) such that

$$Re\ \lambda_j \geq \Lambda_0, \quad j = 1, 2, \ldots, N^2 \tag{1.10a}$$

$$|\lambda_j| \leq \Lambda_1, \quad j = 1, 2, \ldots, N^2 \tag{1.10b}$$

where $\{\lambda_j, j = 1, 2, \ldots, N^2\}$ are the eigenvalues of either the preconditioned matrix

$$Q_N^{-1} A_N \tag{1.11a}$$

or the preconditioned matrix

$$L_N^{-1} A_N. \tag{1.11b}$$

□

Theorem 2: *Let A be the general invertible operator given by (1.1a), (1.1b). There is an integer N_0 and two constants α, β such that; for all $N \geq N_0$ we have*

$$0 < \alpha \leq \sigma_j \leq \beta, \qquad j = 1, 2, \ldots, N^2$$

where $\{\sigma_j; j = 1, 2, \ldots, N^2\}$ are the Q_N singular values of $Q_N^{-1} A_N$ or $L_N^{-1} A_N$. □

Topic: *Nonlinear*	*Session Chair:* *Homer Walker*	*Room A*
10:30 - 11:00	D. Knoll	Enhanced Nonlinear Iterative Techniques Applied to a Non-Equilibrium Plasma Flow
11:00 - 11:30	V. Pan	Newton's Iteration for Inversion of Cauchy-like and Other Structured Matrices
11:30 - 12:00	M. Drexler	Fractal Aspects and Convergence of Newton's Method

Enhanced Nonlinear Iterative Techniques Applied to a Non-equilibrium Plasma Flow

D.A. Knoll and P.R. McHugh
Idaho National Engineering Laboratory
P.O. Box 1625
Idaho Falls, ID 83415-3808
email: nol@inel.gov

Abstract

We study the application of enhanced nonlinear iterative methods to the steady-state solution of a system of two-dimensional convection-diffusion-reaction partial differential equations that describe the partially-ionized plasma flow in the boundary layer of a tokamak fusion reactor. This system of equations is characterized by multiple time and spatial scales, and contains highly anisotropic transport coefficients due to a strong imposed magnetic field. We use Newton's method to linearize the nonlinear system of equations resulting from an implicit, finite volume discretization of the governing partial differential equations, on a staggered Cartesian mesh. The resulting linear systems are neither symmetric nor positive definite, and are poorly conditioned. Preconditioned Krylov iterative techniques are employed to solve these linear systems. We investigate both a modified and a matrix-free Newton-Krylov implementation, with the goal of reducing CPU cost associated with the numerical formation of the Jacobian. A combination of a damped iteration, one-way multigrid and a pseudo-transient continuation technique are used to enhance global nonlinear convergence and CPU efficiency. GMRES is employed as the Krylov method with Incomplete Lower-Upper(ILU) factorization preconditioning. The goal is to construct a combination of nonlinear and linear iterative techniques for this complex physical problem that optimizes trade-offs between robustness, CPU time, memory requirements, and code complexity. It is shown that a one-way multigrid implementation provides significant CPU savings for fine grid calculations. Performance comparisons of the modified Newton-Krylov and matrix-free Newton-Krylov algorithms will be presented.

1 Introduction

In this paper, we study the performance of Newton-Krylov algorithms applied to an advanced fluid transport model of the partially-ionized tokamak edge plasma. The tokamak is a toroidally shaped magnetic confinement fusion device, and the tokamak edge plasma fluid equations are a highly nonlinear set of convection-diffusion-reaction equations that describe the boundary layer plasma of this device. They contain widely varying time and spatial scales, and the transport coefficients and reaction rates are strong functions of density and temperature. These equations describe the flow of plasma particles and energy from the edge of the reactor core into what is called the divertor region, which serves as the exhaust system. The transport model used in this paper is unique in that it is the first to include a

full Navier-Stokes treatment of momentum in the neutral component of the partially-ionized plasma [1].

The presence of multiple time scales and the desire for a steady-state solution motivates a highly implicit algorithm. The nonlinear coupling between conservation equations motivates the use of a strongly convergent Newton-type method. However, this choice presents formidable obstacles such as: the potentially small radius of convergence of Newton's method; the solution of very large, ill-conditioned, nonsymmetric linear systems; and the CPU cost associated with the numerical formation of a complicated Jacobian matrix.

A combination of damped iteration, pseudo-transient continuation, and one-way multigrid are used to help overcome the small radius of convergence issue and to increase CPU efficiency [2, 3]. The preconditioned GMRES algorithm [4], employed within an inexact Newton framework [5, 6] , is used to solve the linear systems that arise on each Newton step. We investigate both a modified Newton-Krylov method [7, 8] and a matrix-free Newton-Krylov method [9, 10] to help reduce the CPU cost associated with repeated numerical Jacobian evaluations.

2 Solution Algorithm

Finite volumes are used to discretize the conservation equations on a staggered Cartesian grid with velocities located at cell faces and thermodynamic variables located at cell centers. This produces a nonlinear system of algebraic equations, which is then solved using the Newton- Krylov methods discussed below.

2.1 Nonlinear Iterative Method

Application of Newton's method requires the solution of the linear system

$$\mathbf{J}^k \delta \mathbf{x}^k = -\mathbf{F}(\mathbf{x}^k), \tag{1}$$

where $\mathbf{J}$ is the Jacobian matrix, $\mathbf{F}(\mathbf{x})$ is the nonlinear system of equations, and $\mathbf{x}$ is the state vector. The new solution approximation at iteration $k+1$ is obtained from,

$$\mathbf{x}^{k+1} = \mathbf{x}^k + s\delta \mathbf{x}^k, \tag{2}$$

where s is a damping scalar, which is adaptively chosen to be less than or equal to one [2]. This iteration is continued until the norm of $\mathbf{F}(\mathbf{x}^k)$ and/or $\delta \mathbf{x}^k$ falls below a specified tolerance level.

A pseudo-transient relaxation technique is used to increase the radius of convergence, and is implemented as:

$$(\frac{\mathbf{V}}{\Delta t^k} + \mathbf{J}^k)\delta \mathbf{x}^k = -\mathbf{F}(\mathbf{x}^k). \tag{3}$$

It is important to note that in Eq.(4), $\mathbf{F}(\mathbf{x}^k)$ is the steady-state residual, and $\mathbf{V}$is a diagonal matrix whose entries consist of the volumes of the computational cell corresponding to that row of the Jacobian. The time step is adaptively varied based upon the current level of convergence in a manner similar to the switched evolution relaxation (SER) algorithm of Mulder and Van Leer [11]. Here, Δt^k is determined from

$$\Delta t^k = \eta \frac{||\mathbf{F}(\mathbf{x}^0)||_\infty \Delta t^0}{||\mathbf{F}(\mathbf{x}^{k-1})||_\infty}, \tag{4}$$

where Δt^0 is the initial time step and η is a chosen scaling constant. Use of Eq. (4) enables the time step to automatically increase as the steady-state residuals decrease.

A one-way multigrid technique is also used to improve the convergence of the nonlinear iteration. In this case, an interpolated coarse grid solution is used as an initial guess on a finner mesh. Thus, less expensive coarse grid calculations are used to furnish an improved starting point on the fine grid, where iterations are more expensive.

2.2 Linear Iterative Method

We use the restarted Generalized Minimal RESidual (GMRES) algorithm [4] to solve Eq. (3). The linear system is preconditioned in order to improve the performance of the GMRES algorithm. Since our Jacobian exhibits a sparse banded structure, only nonzero diagonals are stored. These nonzero diagonals then constitute the Jacobian sparsity pattern, whether or not some diagonal entries are zero. Thus, the ILU(k) preconditioners used here are derived assuming this diagonal sparsity pattern using the "level-of-fill" idea of Watts [12].

Since the use of an iterative technique to solve Eq. (3) does not require the exact solution of the linear system, the resulting algorithm is labeled an "inexact" Newton's method [5, 6]. This feature is advantageous in the sense that the tolerance of the linear equation solve can be relaxed when far from the true solution, and tightened as the true solution is approached. This behavior is controlled using an linear iteration convergence criteria similar to that proposed by Averick and Ortega [5] and Dembo et. al. [6]. Specifically, the GMRES iteration is assumed converged when,

$$\frac{\| \mathbf{J}^k \delta \mathbf{x}^k + \mathbf{F}(\mathbf{x}^k) \|_2}{\| \mathbf{F}(\mathbf{x}^k) \|_2} < \gamma_k. \tag{5}$$

The effect of γ_k on CPU performance will be studied.

2.3 Reduced Jacobian Evaluation Techniques

The GMRES algorithm requires the action of the Jacobian only in the form of matrix-vector products, which may be approximated by [9],

$$\mathbf{J}\mathbf{v} \approx \frac{\mathbf{F}(\mathbf{x} + \epsilon \mathbf{v}) - \mathbf{F}(\mathbf{x})}{\epsilon}, \tag{6}$$

where $\mathbf{J}$ is the Jacobian matrix, $\mathbf{v}$ is a Krylov vector, $\mathbf{x}$ is the state vector, ϵ is a perturbation constant, and $\mathbf{F}$ represents the system of nonlinear equations. Eq. (6) enables the action of the Jacobian without explicitly forming or storing the matrix. This property can be extremely advantageous in problems where forming the Jacobian represents a significant fraction of the total CPU time, and/or storing the Jacobian matrix is prohibitive. In many instances, however, the Jacobian or parts thereof are still needed to generate an effective preconditioner. In this situation, the primary advantage of this matrix-free implementation may lie in reducing the total number of required discrete function evaluations by amortizing the cost of forming the preconditioner over several Newton steps [10]. Note that although storage is required for the preconditioner, storage is not required for the Jacobian.

In the full length paper, this matrix-free method will be compared to a modified Newton method [7, 8], where the same "frozen" Jacobian is used for a number of Newton iterations

without employing Eq.(6). This is in contrast to the matrix-free implementation where the same "frozen" preconditioner is used for a number of Newton iterations, but the effects of the current Jacobian are captured within the accuracy of the approximation in Eq. (6). Note that the time step, Δt^k, in the pseudo-transient is updated only when a new Jacobian and preconditioner are formed.

3 Equation System

The tokamak edge plasma (scrape-off layer, boundary layer) is that plasma which lies between the last closed magnetic flux surface, called the separatrix, and the vessel wall. This plasma is made up of hydrogen ions, electrons, impurity ions from the vessel structure, such as carbon, as well as hydrogen and impurity neutral atoms and molecules. The system of equations solved in this paper only models hydrogen ions, atoms and electrons. For more details regarding the equations system, boundary conditions, and geometry see [1]. The following equations are solved on a two-dimensional (x, y) Cartesian grid after they are first nondimensionalized to improve scaling.

Plasma continuity:

$$\frac{\partial n_i}{\partial t} + \nabla \cdot (n_i \vec{U}_i) = \nu_{ion} n_o - \nu_{rec} n_i. \tag{7}$$

Neutral continuity:

$$\frac{\partial n_o}{\partial t} + \nabla \cdot (n_o \vec{U}_o) = -\nu_{ion} n_o + \nu_{rec} n_i. \tag{8}$$

Ion parallel momentum:

$$\begin{aligned}\frac{\partial m n_i u_\parallel}{\partial t} + \nabla \cdot (m n_i \vec{U}_i u_\parallel - \tilde{\eta}_i \cdot \nabla u_\parallel) = -\frac{B_x}{B}(\frac{\partial P_i}{\partial x} + \frac{\partial P_e}{\partial x}) \\ +\nu_{ion} m n_o u_{\parallel o} - \nu_{rec} m n_i u_\parallel + \nu_{i-n} m n_i (u_{\parallel o} - u_\parallel).\end{aligned} \tag{9}$$

Neutral vector momentum (three components):

$$\begin{aligned}\frac{\partial m n_i \vec{U}_o}{\partial t} + \nabla \cdot (m n_i \vec{U}_o \vec{U}_o) = \nabla \cdot \tau_{i,j} \\ -\nu_{ion} m n_o \vec{U}_o + \nu_{rec} m n_i \vec{U}_i - \nu_{i-n} m n_i (\vec{U}_o - \vec{U}_i).\end{aligned} \tag{10}$$

Ion + neutral internal energy:

$$\begin{aligned}\frac{\partial}{\partial t}(\frac{3}{2}(n_i + n_o)T_i) + \nabla \cdot (\frac{3}{2} T_i (n_i \vec{U}_i + n_o \vec{U}_o) - (\tilde{\kappa}_i + \tilde{I}\kappa_o) \cdot \nabla T_i) = \\ -(P_i \nabla \vec{U}_i + P_o \nabla \vec{U}_o) + \kappa_{eq}(T_e - T_i).\end{aligned} \tag{11}$$

Electron internal energy:

$$\begin{aligned}\frac{\partial}{\partial t}(\frac{3}{2} n_e T_e) + \nabla \cdot (\frac{3}{2} n_e \vec{U}_i T_e - \tilde{\kappa}_e \cdot \nabla T_e) = \\ -P_e \nabla \vec{U}_i - \kappa_{eq}(T_e - T_i) - I_p(n_o \nu_{ion} - n_i \nu_{rec}) - L_{rad}.\end{aligned} \tag{12}$$

Here, n_i is the ion number density, n_e is the electron number density (equal to n_i assuming quasi-neutrality), n_o is the neutral atom number density. $u_\parallel$ is the ion velocity along the

total magnetic field (B), with the parallel direction being a combination of the x and z directions. u is the x-direction ion velocity which is equal to $\frac{B_x}{B}u_{\parallel}$, where $\frac{B_x}{B}$ is the magnetic field pitch. v is the y-direction ion velocity which is obtained from a diffusive approximation. Ambipolar flow is assumed so that the electron velocity is equal to the ion velocity. $\vec{U}_o$ is the neutral atom velocity, with $\vec{U}_o = u_o\hat{x} + v_o\hat{y} + w_o\hat{z}$. m is the ion/atom mass. T_i is the ion temperature, T_e is the electron temperature. The equation of state gives the following relations for pressure, $P_e = n_eT_e, P_i = n_iT_i$, and $P_o = n_oT_i$. ν_{ion} and ν_{rec} are the ionization and recombination frequencies, and ν_{i-n} is the ion-atom elastic collision frequency. η is the viscosity, κ is the thermal conductivity, $\tau_{i,j}$ is the neutral (atom) fluid stress tensor, and κ_{eq} is a thermal energy transfer coefficient. I_p is the ionization potential energy, and L_{rad} represents energy loss due to atomic line radiation. Because we solve for three components of neutral momentum there is a total of eight equations with the dependant variables being $n_i, n_o, u_{\parallel}, u_o, v_o, w_o, T_i$, and T_e. These equations are strongly coupled through nonlinear source/sink terms which represent the effects of ionization, recombination, ion-neutral elastic collisions, and ion-electron coulomb collisions. The transport coefficients are highly nonlinear and some are anisotropic due to the magnetic field. Additionally, the problem exhibits large spreads in time scales and space scales.

4 Algorithm Performance Summary

Fig. 1 shows the convergence history for a 3-grid solution starting on a 32x16 grid and finishing on a 128x64 grid. This simulation was run using the matrix-free implementation and evaluating the Jacobian/preconditioner every 10 Newton steps. GMRES(40) was used with point ILU(1) preconditioning. The inexact Newton convergence tolerance γ_k, was set equal to 0.05. The initial time step on the coarse grid was very small ($5 \times 10^{-8} sec$) due to the poor initial guess. However, the time step at convergence on the 32x16 grid was approximately $1 \times 10^{-4} sec$. η, in Eq. 4 was set to 1.0. On each successive grid the time step was initialized 1 order of magnitude below the final value on the previous grid. The total number of Newton iterations on each grid was 74, 19, and 7 respectively. The gap, or delay, between the convergence of the 32x16 grid and the first triangle for the 64x32 grid represents the time to form the first Jacobian and preconditioner on the 64x32 grid. The distance between the last triangle and the first circle represents the time required to form the first, and only, Jacobian and preconditioner on the 128x64 grid. We can see that by the time this grid is reached, nearly two thirds of the total solution time required for the grid is spent forming the Jacobian and preconditioner. When plotted as a function of Newton iteration, one observes an increasingly rapid convergence as we move up in grid.

Fig. 2 is used to quantify the cost incured to obtain the 64x32 grid solution when the intermediate 32x16 grid is not employed. For completeness, we include the 32x16 and 64x32 grid convergence plots from Fig. 1, in addition to displaying the convergence plot for the 64x32 grid solution with a poor initial guess. A CPU savings of a factor of 4 is observed by employing a coarse grid. Note that while the two 64x32 grid convergence plots have roughly the same maximum steady-state residual at approximately 0.5 hours, the one obtained using one-way multigrid exhibits a much faster convergence rate beyond that point.

5 Acknowledgments

This work was supported under the auspices of the U.S. Department of Energy under DOE Idaho Field Office Contract DE-AC07-94ID13223.

References

[1] D.A. Knoll et al. Simulation of dense recombining divertor plasmas with a navier-stokes neutral transport model. *Physics of Plasmas*, 3:293–303, 1996.

[2] D.A. Knoll, A.K. Prinja, and R.B. Campbell. A Direct Newton Solver for the Two-Dimensional Tokamak Edge Plasma Fluid Equations. *J. Comput. Phys.*, 104:418–426, 1993.

[3] D.A. Knoll and P.R. McHugh. An Inexact Newton Algorithm for Solving the Tokamak Edge Plasma Fluid Equations on a Multiply Connected Domain. *J. Comput. Phys.*, 116:281–291, 1995.

[4] Y. Saad and M.H. Schultz. GMRES: A generalized minimal residual algorithm for solving non-symetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7:856, 1986.

[5] B.M Averick and J.M. Ortega. Solutions of nonlinear poisson-type equations. *Appl. Numer. Math.*, 8:443–455, 1991.

[6] R. Dembo et al. Inexact newton methods. *SIAM J. Numer. Anal.*, 19:400–408, 1982.

[7] M. D. Smooke. Solution of Burner-Stabilized Premixed Laminar Flames by Boundary Value Methods. *J. Comput. Phys.*, 48:72–105, 1982.

[8] D.A. Knoll and P.R. McHugh. A fully implicit direct newton solver for the navier-stokes equations. *Int. J. Num. Meth. Fluids*, 17:449–461, 1993.

[9] P. N. Brown and Y. Saad. Hybrid krylov methods for nonlinear systems of equations. *SIAM J. Sci. Stat. Comput.*, 11:450–481, 1990.

[10] D.A. Knoll and P.R. McHugh. Newton-Krylov methods applied to a system of convection-diffusion-reaction equations. *Comput. Phys. Comm.*, 88:141–160, 1995.

[11] W.A. Mulder and B. Van Leer. Experiments with Implicit Upwind Methods for the Euler Equations. *J. Comput. Phys.*, 59:232–246, 1985.

[12] James W. Watts III. A conjugate gradient-truncated direct method for the iterative solution of the reservoir simulation pressure equation. *Society of Petroleum Engineers Journal*, pages 345–353, June 1981.

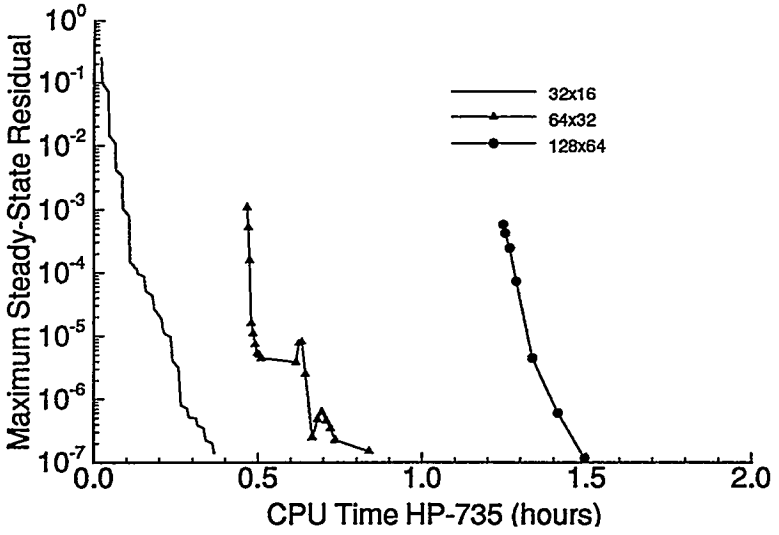


Figure 1: Convergence history for 128x64 grid problem using one-way multigrid

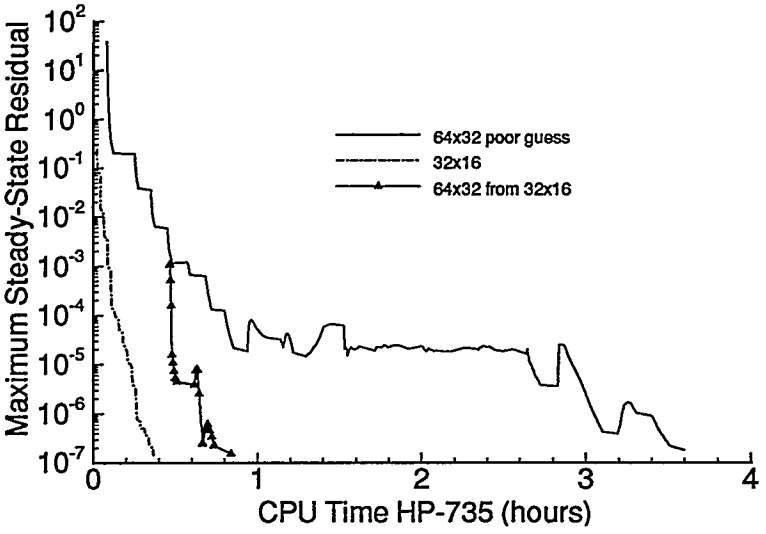


Figure 2: Advantage of using one-way multigrid technique in obtaining 64x32 grid solution

Newton's Iteration for Inversion of Cauchy-like and Other Structured Matrices

Victor Y. Pan[1,a], Ailong Zheng[2,b], Xiaohan Huang[2,c], Olen Dias[2]

[1] Department of Mathematics and Computer Science
Lehman College, CUNY, Bronx, NY 10468, USA
[a] Internet: vpan@lcvax.lehman.cuny.edu

[2] Ph.D. Program in Mathematics
Graduate School and University Center, CUNY
33 West 42nd Street, New York, NY 10036, USA
[b] Internet: zaugc@cunyvm.cuny.edu
[c] Internet: xhhgc@cunyvm.cuny.edu

February 6, 1996

Abstract

We specify some initial assumptions that guarantee rapid refinement of a rough initial approximation to the inverse of a Cauchy-like matrix, by mean of our new modification of Newton's iteration, where the input, output, and all the auxiliary matrices are represented with their short generators defined by the associated scaling operators. The computations are performed fast since they are confined to operations with short generators of the given and computed matrices. Because of the known correlations among various structured matrices, the algorithm is immediately extended to rapid refinement of rough initial approximations to the inverses of Vandermonde–like, Chebyshev–Vandermonde–like and Toeplitz–like matrices, where again, the computations are confined to operations with short generators of the involved matrices.

1 Introduction

Computations with structured matrices (such as Toeplitz, Cauchy, and Vandermonde matrices) can be facilitated (so that the computational time and memory space decrease dramatically) by means of representing these matrices with their low rank generators associated with operators of displacement (shift) and/or scaling [KVM], [KKM], [CKL-A], [GKK], [GKKL], [HR], [P90], [GO], [BP], [H], [GKO], [GO1], [KO], [KS].

Such a generator is a matrix associated with a given structured $n \times n$ matrix A and representable in one of the forms

$$\Delta_{\{R,S\}}(A) = A - RAS = GH^T, \tag{1.1}$$

$$\nabla_{\{R,S\}}(A) = RA - AS = GH^T, \tag{1.2}$$

for a fixed natural α, for two fixed matrices R and S, representing scaling and/or displacement, and for some $n \times \alpha$ matrices G and H. In this paper we will deal with $\nabla_{\{R,S\}}$-generators of A, of the form (1.2), and will specify R and S according to the next equations (1.3) – (1.5): Scaling is represented by diagonal matrices

$$D_x = diag(\, x_1, x_2, \cdots, x_n \,), \tag{1.3}$$

for fixed $x_1, x_2, \cdots, x_n$, whereas such $n \times n$ matrices as

$$Z_f = (z_{ij}), \quad z_{i,i-1} = 1, \quad z_{1,n} = f, \quad z_{i,j} = 0 \text{ if } j \neq i - 1 \bmod n, \tag{1.4}$$

or

$$aZ_f + bZ_\varphi^T + cZ_\psi^{-1}\,, \tag{1.5}$$

for some fixed scalars a, b, c, f, φ, and $\psi \neq 0$, represent displacement.

The key-idea is that the original matrix A can be easily recovered from its generators (1.1) or (1.2); moreover, the basic operations (such as multiplication, addition, subtraction, and inversion) with $n \times n$ structured matrices of certain classes can be reduced to operations with their generators that are represented by $O(\alpha n)$ parameters. This leads to a dramatic saving of the computational time and memory space, if α is much less than n.

For Toeplitz, Cauchy (generalized Hilbert), and Vandermonde matrices, the length α of the associated generators (1.1) and (1.2), for some appropriate choices of the matrices R and S among the matrices of the classes (1.3) and (1.4), is as small as 1 or 2. These 3 classes of matrices are naturally extended to Toeplitz-like, Cauchy-like, and Vandermonde-like matrices, for which α is bounded by a fixed (and not too large) constant. (These classes include or are closely related to some other well-known classes of structured matrices, such as Sylvester, subresultant, Hankel, Hankel-like, Lowener, Bezout, and Chebyshev-Vandermonde matrices [HR], [BP], [H], [GO1], [KO].) It was observed in [P90] that some simple correlations among the operators associated with the matrices of the 3 cited classes can be exploited in order to reduce the computations for matrices of any of the 3 classes to computations with matrices of the class for which most effective algorithms are available.

On the other hand, due to a simple reduction (available via FFT) from Toeplitz-like and Vandermonde-like matrices to Cauchy-like matrices, [H] and [GKO], effective algorithms for computations with Cauchy-like matrices could play most fundamental role.

In this paper, we consider the solution of a nonsingular Cauchy-like linear system, $C\vec{x} = \vec{v}$, and the inversion of a nonsingular Cauchy-like matrix, C. These operations can be immediately extended to the Vandermonde-like and Toeplitz-like cases, as well as to the Chebyshev-Vandermonde cases [GO1] and [KO].

It is well-known that Newton's iteration rapidly improves a rough initial approximation to the matrix inverse (cf. e.g. [BP]), but such an iteration also rapidly destroys the structure of Cauchy-like, Toeplitz-like, and Vandermonde-like matrices. In [P92], [P93], and [P93a], Newton's iteration has

been modified so as to preserve the initial displacement structure of a Toeplitz-like input matrix during the iteration. The idea was to control the growth of the length of short displacement generators by periodically chopping-off their components corresponding to the smallest singular values in the SVD of these generators. This made all the iteration steps of the resulting algorithms computationally simple; moreover, such a simplification was achieved with no significant slowdown of the convergence.

In the present paper, we consider a similar problem of controlling the length of the associated generators in a modification of Newton's iteration, where the inverse of a fixed Cauchy-like input matrix C is sought. Our solution of the problem is substantially simplified in this case (in particular, we do not need to involve the SVD), due to the formula for the inverse matrix C^{-1} available from [H]. This may motivate the reduction of the Toeplitz-like case to the Cauchy-like case (by means of FFTs) and applying the techniques of the present paper, instead of the inversion of Toeplitz-like matrices by applying the techniques of [P92], [P93], and [P93a].

2 Modified Newton's Iteration for the Inversion of Cauchy-like Matrices

Let C be an $n \times n$ nonsingular Cauchy-like matrix with the associated scaling operator

$$\nabla_{\{D_t,D_s\}}(C) = D_tC - CD_s = ZY^T \ , \tag{2.1}$$

$$C = \left[\frac{z_i^T y_j}{t_i - s_j} \right], \qquad Z^T = [\ z_1, \cdots, z_n\ ], \qquad Y^T = [\ y_1, \cdots, y_n\ ] \ , \tag{2.2}$$

where $z_i,\ y_j \in C^{\alpha \times 1}$, $D_t = diag(\ t_1, \cdots, t_n\ )$, $D_s = diag(\ s_1, \cdots, s_n\ )$, $t_i \neq s_j$, for all $i,\ j = 1, 2, \cdots, n$ (cf. [H] and [GKO]). Assume that an initial approximation X_0 to C^{-1} is available with its $\nabla_{\{D_t,D_s\}}$-generator of length at most α. Then we recursively define matrices $X_1^*,\ X_1,\ X_2^*,\ X_2,\ \cdots$, as follows:

$$X_{k+1}^* = X_k\ (\ 2I - CX_k\ ), \qquad k = 0, 1, \cdots, \tag{2.3}$$

$$X_{k+1} = -\left[\frac{(u_i^{k+1})^T w_j^{k+1}}{s_i - t_j} \right]_{i,j=1}^{n}, \qquad k = 0, 1, \cdots, \tag{2.4}$$

where the vectors $u_i^{k+1},\ w_j^{k+1} \in C^{\alpha \times 1}$ are defined by

$$U_{k+1} = \begin{pmatrix} (u_1^{k+1})^T \\ \vdots \\ (u_n^{k+1})^T \end{pmatrix} = X_{k+1}^* Z \ , \qquad Z = \begin{pmatrix} z_1^T \\ \vdots \\ z_n^T \end{pmatrix} \ , \tag{2.5}$$

$$W_{k+1}^T = [\ w_1^{k+1}, \cdots, w_n^{k+1}\ ] = Y^T X_{k+1}^* \ , \qquad Y^T = [\ y_1, \cdots, y_n\ ] \ . \tag{2.6}$$

Equation (2.3) represents a step of Newton's iteration for matrix inversion (cf. e.g. [BP]), and equation (2.4) "corrects" the results X_{k+1}^* of (2.3) so as to turn X_{k+1}^* into a Cauchy-like matrix, associated with the same scaling operators as C^{-1}. Namely,

$$\nabla_{\{D_s,D_t\}}(X_{k+1}) = -U_{k+1}W_{k+1}^T \ ,$$

that is, X_{k+1} is a Cauchy-like matrix whose $\nabla_{\{D_s,D_t\}}$- generator has a length of at most α. Furthermore, we have

Proposition 2.1 *For any $k = 0, 1, \cdots$, the matrix $X^*_{k+1} = 2X_k - X_k C X_k$ is a Cauchy-like matrix whose $\nabla_{\{D_s, D_t\}}$-generator has a length of at most 3α.*

3 Computational Complexity of an Iteration Step

Next, we estimate the arithmetic cost of computing the matrices X_{k+1}, U_{k+1}, W_{k+1}, and X^*_{k+1}, based on the equations (2.3)–(2.6). Given Cauchy–like matrices C and X_k, the computation of the $\nabla_{\{D_s, D_t\}}$-generator of length 3α for X^*_{k+1} (according to (2.3)) uses $O(\alpha^2 n \log^2 n)$ ops (see e.g. [GO] or [BP], chapter 2, section 4, 11, and 12). (Here and hereafter, ops stands for arithmetic operations). Therefore, (2.5) and (2.6) together enable us to compute the $n \times \alpha$ matrices U_{k+1} and W_{k+1} by using $O(\alpha^2 n \log^2 n)$ ops. An additional attractive feature of this computation is the economization of computer memory, that is, representation of all involved matrices by means of their short generators requires only $O(\alpha n)$ words of memory. We also refer the reader to [BP], pages 130, 261–262, on some alternative methods for faster numerical approximation of the product of a Cauchy matrix C by a vector, which may lead to a further decrease of the computational cost of our iteration steps.

4 Estimating Convergence Rate of Newton's Iteration

In the following, we will estimate how rapidly X_k approaches C^{-1}. From [H], we have

$$C^{-1} = -\left[\frac{u_i^T w_j}{s_i - t_j} \right] , \tag{4.1}$$

$$U = [\, u_1^T, \cdots, u_n^T \,]^T = C^{-1} Z \,, \qquad W^T = [\, w_1, \cdots, w_n \,] = Y^T C^{-1} \,, \tag{4.2}$$

where $Z^T = [\, z_1, \cdots, z_n \,]$, and $Y^T = [\, y_1, \cdots, y_n \,]$ (cf. (2.1) and (2.2)).
We recall from (2.4)–(2.6) and (4.2) that

$$\begin{aligned} X_k &= -\left[\frac{(u_i^k)^T w_j^{(k)}}{s_i - t_j} \right] , \\ U_k &= X^*_k Z = (\, X^*_k - C^{-1} \,) Z + C^{-1} Z = (\, X^*_k - C^{-1} \,) Z + U \,, \\ W_k^T &= Y^T X^*_k = Y^T (\, X^*_k - C^{-1} \,) + W^T \,. \end{aligned}$$

Then, we obtain the following matrix equation:

$$U_k W_k^T = (X^*_k - C^{-1}) Z Y^T (X^*_k - C^{-1}) + U W^T + U Y^T (X^*_k - C^{-1}) + (X^*_k - C^{-1}) Z W^T .$$

From this equation and (4.2), we obtain that

$$\begin{aligned} E_k &= U_k W_k^T - U W^T \\ &= (X^*_k - C^{-1}) Z Y^T (X^*_k - C^{-1}) + C^{-1} Z Y^T (X^*_k - C^{-1}) + (X^*_k - C^{-1}) Z Y^T C^{-1} . \end{aligned} \tag{4.3}$$

Hereafter, we will use the column-norm of matrices $\|W\| = \|W\|_1$ ([GL], p.57).

Proposition 4.1 *Let* $e_k^* = \| X_k^* - C^{-1} \|$. *Then*

$$\|E_k\| = \| U_k W_k^T - UW^T \| \leq \|ZY^T\| e_k^* (e_k^* + 2\|C^{-1}\|) .$$

Proof: Proposition 4.1 immediately follows from (4.3) . □

Proposition 4.2 *Let* $e_k = \| X_k - C^{-1} \|$, *for a nonsingular matrix* C, *and let* X_{k+1}^* *be defined by* (2.3), *for* $k = 0, 1, 2, \cdots$. *Then we have*

$$e_{k+1}^* \leq \|C\| e_k^2 . \tag{4.4}$$

Proposition 4.3 *For any* $k = 1, 2, \cdots$, *we have* $e_k \leq s_k \, e_k^*$, $e_{k+1}^* \leq (s_k e_k^*)^2 \|C\|$, *where*

$$s_k = \rho \|ZY^T\| (e_k^* + 2\|C^{-1}\|), \qquad \rho = \max_{i,j} \frac{1}{|s_i - t_j|} . \tag{4.5}$$

Proposition 4.4 *If*

$$e_1^* \leq 1 \tag{4.6}$$

and if

$$(e_1^*)^\theta \|C\| s_1^2 \leq 1 , \tag{4.7}$$

for $\theta \leq 1$ *and for* s_1 *of proposition* 4.3, *then*

$$e_{k+1}^* \leq (e_k^*)^{2-\theta} \leq \cdots \leq (e_1^*)^{(2-\theta)^k} , \qquad \text{for} \quad k = 1, 2, \cdots . \tag{4.8}$$

Proposition 4.5 *If* $e_0 \sqrt{\|C\|} \leq 1$ *and if*

$$e_0^{2\theta} \|C\|^{1+\theta} s^2 \leq 1 , \tag{4.9}$$

for $\theta \leq 1$, *for* $s = \rho \|ZY^T\| (e_0^2 \|C\| + 2\|C^{-1}\|)$, *and for* ρ *of* (4.5), *then*

$$e_{k+1}^* \leq (e_0^2 \|C\|)^{(2-\theta)^k} , \qquad \text{for} \quad k = 0, 1, \cdots .$$

We write

$$e_0^+ = \frac{r_0 \|X_0\|}{1 - r_0} , \tag{4.10}$$

$$s_{k+1}^+ = \rho \|ZY^T\| \, [\, (\, (e_0^+)^2 \|C\| \,)^{(2-\theta)^k} + \frac{2\|X_0\|}{1 - r_0} \,] , \quad k = 0, 1, \cdots , \tag{4.11}$$

so that $e_0^+ \geq e_0$ and $s_{k+1}^+ \geq s_{k+1}$ if $s_{k+1}^* \leq (\, (e_0^+)^2 \|C\| \,)^{(2-\theta)^k}$, and

$$s_{k+1}^+ \leq \rho \|ZY^T\| \, (\, 1 + \frac{2\|X_0\|}{1 - r_0} \,) , \quad \text{for all } k, \quad \text{if} \quad e_0^+ \sqrt{\|C\|} \leq 1 . \tag{4.12}$$

We now summarize our results, including the bound $e_{k+1} \leq s_{k+1} e_{k+1}^*$ (from proposition 4.3), which enables us to estimate e_{k+1} as soon as we estimate e_{k+1}^* .

Corollary 4.1 *Let*

$$r_0 = ||I - CX_0|| < 1, \quad e_0^+\sqrt{||C||} \le 1, \quad \theta \le 1, \tag{4.13}$$

$$(e_0^+)^{2\theta}||C||^{1+\theta}\ (s_1^+)^2 \le 1; \tag{4.14}$$

for e_0^+ *of* (4.10) *and* s_1^+ *of* (4.11). *Then*

$$e_{k+1}^* \le (\ (e_0^+)^2||C||\ )^{(2-\theta)^k}, \qquad k = 0, 1, \cdots,$$

$$e_{k+1} \le s_{k+1}^+ e_{k+1}^*, \qquad k = 0, 1, \cdots.$$

Let us write

$$k^* = \lceil\ (\log \frac{\log \epsilon^*}{\log(\ (e_0^+)^2||C||\ )}\ )\ /\log(2-\theta)\ \rceil\ , \tag{4.15}$$

$$\hat{k} = \lceil\ (\log \frac{\log \epsilon}{\log(\ (e_0^+)^2||C||\ )}\ )\ /\log(2-\theta)\ \rceil\ , \tag{4.16}$$

where $\theta < 1$, e_0^+ and s_{k+1}^+ are defined by (4.10) and (4.11), so that (4.12) holds. Then, under the assumptions (4.13) and (4.14) of corollary 4.1, it suffices to perform $k+1 \ge k^*+1$ recursive steps of the iteration (2.3), (2.4) in order to ensure that

$$e_{k+1}^* = ||\ X_{k+1}^* - C^{-1}\ || \le \epsilon^*\ ,$$

and it suffices to perform $k+1 \ge \hat{k}+1$ recursive steps in order to ensure that

$$e_{k+1} = ||\ X_{k+1} - C^{-1}\ || \le \epsilon s_{k+1}^+\ .$$

[Note that (4.13) implies, in particular, the bounds of (4.12).]

References

[BP] D. Bini and V. Y. Pan, *Polynomial and Matrix Computations, Volume 1: Fundamental Algorithms*, Birkhauser, Boston, 1994.

[CKL-A] J. Chun, T. Kailath, and H. Lev-Ari, Fast Parallel Algorithm for QR-factorization of Structured Matrices, *SIAM J. Sci. Stat. Comput.*, 8, 6, pp. 899-913, 1987.

[GKK] I. Gohberg, T. Kailath, and I. Koltracht, Efficient Solution of Linear Systems of Equations with Recursive Structure, *Linear Algebra Appl.*, 80, pp. 81-113, 1986.

[GKKL] I. Gohberg, T. Kailath, I. Koltracht, and P. Lancaster, Linear Complexity Parallel Algorithms for Linear Systems of Equations with Recursive Structure, *Linear Algebra Appl.*, 88/89, pp. 271-315, 1987.

[GKO] I. Gohberg, T. Kailath, and V. Olshevsky, Fast Gaussian Elimination with Partial Pivoting for Matrices with Displacement Structure, *submitted for publication*, 1994.

[GL] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins Univ. Press, Baltimore, Maryland, 1989.

[GO] I. Gohberg and V. Olshevsky, Complexity of Multiplication with Vectors for Structured Matrices, *Linear Algebra Appl.*, 202, pp. 163-192, 1994.

[GO1] I. Gohberg and V. Olshevsky, Fast Inversion of Chebyshev-Vandermonde Matrices, *Numer. Math.*, 67, 1, pp. 71-92, 1994.

[H] G. Heinig, Inversion of Generalized Cauchy Matrices and the Other Classes of Structured Matrices, *Linear Algebra for Signal Processing, IMA Volume in Math. and Its Applications*, 69, pp. 95-114, Springer, 1994.

[HR] H. Heinig and K. Rost, *Algebraic Methods for Toeplitz-like Matrices and Operators, Operator Theory*, 13, Birkhauser, 1984.

[KKM] T. Kailath, S.-Y. Kung, and M. Morf, Displacement Ranks of Matrices and Linear Equations, *J. Math. Anal. Appl.*, 68, 2, pp. 395-407, 1979.

[KO] T. Kailath and V. Olshevsky, Displacement Structure Approach to Chebyshev-Vandermonde and Related Matrix, *submitted for publication*, 1994.

[KS] T. Kailath and A. Sayed, Displacement Structure: Theory and Applications, *SIAM Review*, to appear.

[KVM] T. Kailath, A. Viera, and M. Morf, Inverses of Toeplitz Operators, Innovations, and Orthogonal Polynomials, *SIAM Review*, 20, 1, pp. 106-119, 1978.

[P90] V. Y. Pan, Computations with Dense Structured Matrices, *Math. of Computation*, 55, pp.179-190, 1990.

[P92] V. Y. Pan, Parallel Solution of Toeplitz-like Linear Systems, *J. of Complexity*, 8, pp. 1-21, 1992.

[P93] V. Y. Pan, Concurrent Iterative Algorithms for Toeplitz-like Linear Systems, *IEEE Trans. on Parallel and Distributive Systems*, 4, 5, pp. 592-600, 1993.

[P93a] V. Y. Pan, Decreasing the Displacement Rank of a Matrix, *SIAM J. Matrix Anal. Appl.*, 14, 1, pp. 118-121, 1993.

[W] D. H. Wood, Product Rules for the Displacement of Near-Toeplitz Matrices, *Linear Algebra Appl.*, 188, 189, pp. 641-663, 1993.

Fractal Aspects and Convergence of Newton's Method

M. Drexler
Numerical Analysis Group
Oxford University Computing Laboratory
Wolfson Building, Parks Road, Oxford, OX1 3QD, England
e-mail: namd@comlab.ox.ac.uk

1 Introduction

Newton's Method is a widely established iterative algorithm for solving non-linear systems. Its appeal lies in its great simplicity (see [8] for a geometric interpretation), easy generalisation to multiple dimensions and a quadratic local convergence rate. Despite these features, little is known about its global behaviour. In this paper, we will explain a seemingly random global convergence pattern using fractal concepts and show that the behaviour of the residual is entirely explicable. We will also establish quantitative results for the convergence rates. Knowing the mechanism of fractal generation, we present a stabilisation to the orthodox Newton method that remedies the fractal behaviour and improves convergence.

It is well known that the basins of attraction of different roots have fractal boundaries when Newton's method is used to determine the complex roots of polynomials; the simplest example being *Cayley's problem* $z^3 = 1$ that arose as early as 1879. Basic properties of the boundaries were explained by Julia [5] in a substantial paper. With the advent of computer graphics, interest in iterated polynomial mappings re-emerged, but much attention was devoted to the Mandelbrot set ([1], [7], [9]) and Newton's method was treated historically as an accessible example to introduce the general concept of Julia sets [10].

In this work, we take a different approach and do not seek to establish general properties related to Julia sets, but the mechanism by which Newton's method generates a fractal structure and how modifications affect that mechanism. Using these results, we are able to close the gap between the theoretical results of fractal geometry and the problems arising in numerical analysis. In addition, we are able to give quantitative results on Cayley's problem that are hitherto unknown.

2 The Orthodox Newton Method

In this section, we will give results for applying the orthodox Newton method to complex polynomials of the form $z^n = 1$. After briefly defining the necessary terms, we give results for Cayley's problem and discuss their generalisation for higher-order polynomials. This work will be restricted to functions of a complex variable, which can always be stated as two-dimensional systems via a splitting into real and imaginary part. However, these are only exemplary for general two-dimensional systems which can just as well exhibit fractal behaviour.

2.1 Definitions

The *orthodox Newton method* on $f(z)$ is defined by the iteration

$$z^{(k+1)} = z^{(k)} - \frac{f(z^{(k)})}{f_z(z^{(k)})} \tag{1}$$

with $z^{(k)}$ denoting the k^{th} iterate, and f_z denoting $\frac{df}{dz}$. The view we shall take of the Newton method is slightly different to that implied by (1). Rather than finding the next iterate given a

starting point $z^{(k)}$, we shall ask which points $z^{(k)} = z = (x + iy)$ are mapped into a given point $z^{(k+1)} = z_0 = (a + ib)$ by (1).

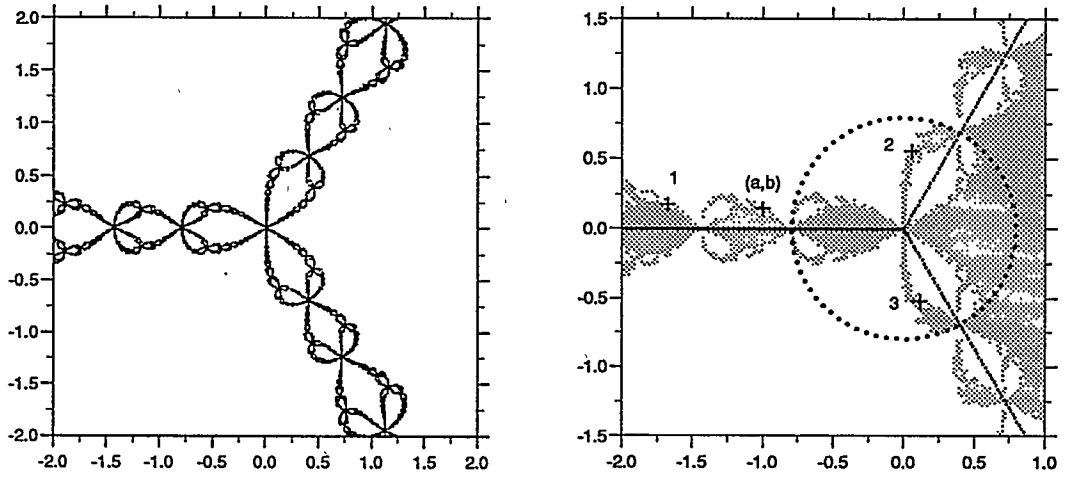
For $f(z) = z^\nu - 1$, we then define the complex *Newton polynomial of order* ν to be

$$(\nu - 1)z^\nu - \nu z_0 z^{\nu-1} + 1 = 0, \qquad z, z_0 \in \mathbf{C}. \tag{2}$$

For the rest of this paper, we assume $f(z)$ to be the aforementioned polynomial unless stated otherwise.

To give a brief, yet comprehensive definition of a fractal is not a straightforward task. We refer to the very instructive discussion in [3] and, coherent with the concepts presented there, define the *Newton fractal of order* ν as the union of all points that are mapped into the singular origin (the Julia points) by the Newton mapping (2). Fig. 1a depicts the Newton fractal of order 3 in the complex plane.

We also assume general facts about Julia sets as given in [3], [1]. In particular, we denote the *order of a Julia point* by the number of Newton iterations it takes until its image coincides with the origin.



(a) (b)

Figure 1: (a) Newton Fractal $\nu = 3$ (b) Images of the inverse Newton mapping

2.2 Generation of the Fractal for Cayley's Problem

Throughout this subsection, we shall be concerned with the case $f(z) = z^3 - 1$. We can write the general solution $\{\mathbf{x_k}\}$ to (2) for $\nu = 3$ in the complex plane as

$$\begin{aligned} \mathbf{x_1} &= \mathbf{b_1} + \mathbf{b_{23}} + \mathbf{t} \\ \mathbf{x_2} &= \mathbf{R_{120}}\mathbf{b_1} + \mathbf{R_{240}}\mathbf{b_{23}} + \mathbf{t} \\ \mathbf{x_3} &= \mathbf{R_{240}}\mathbf{b_1} + \mathbf{R_{120}}\mathbf{b_{23}} + \mathbf{t} \end{aligned} \tag{3}$$

where

$$\begin{aligned} \varphi &= (z_0)^3 - 2 + 2\sqrt{1 - (z_0)^3} \\ \xi &= \mathrm{Re}(\sqrt[3]{\varphi}), & \eta &= \mathrm{Im}(\sqrt[3]{\varphi}) \\ \mathbf{x_k} &= \begin{pmatrix} x_k \\ y_k \end{pmatrix} & \mathbf{b_1} &= \tfrac{1}{2}\begin{pmatrix} \xi \\ \eta \end{pmatrix} \\ \mathbf{b_{23}} &= \tfrac{1}{2(\xi^2+\eta^2)}\begin{pmatrix} 2ab\eta - (b^2 - a^2)\xi \\ 2ab\xi + (b^2 - a^2)\eta \end{pmatrix} & \mathbf{t} &= \tfrac{1}{2}\begin{pmatrix} a \\ b \end{pmatrix} \end{aligned} \tag{4}$$

$\mathbf{R_{120}}$ and $\mathbf{R_{240}}$ denote the two-dimensional rotation matrices about an angle of $\frac{2\pi}{3}$ and $\frac{4\pi}{3}$, respectively. In accordance with its definition, the fractal emerges by successive mappings (3) from the images of the trivial real Julia point $\chi_1 = -\frac{1}{\sqrt[3]{2}}$ (obtained from (2) for $z_0 = 0$) and its rotations by $\pm\frac{2\pi}{3}$. By substituting $b = 0$ for the real axis, we can get a parametric description of the image of the real axis and determine that points on the real axis approach the origin as a geometric progression, and the image of the real axis sufficiently close to the origin can be approximated by

$$\begin{aligned} y &\approx \pm\sqrt[4]{x}, \qquad x < \tfrac{1}{2\sqrt[3]{2}} \\ x &\approx \tfrac{1}{9a^2}, \qquad a \gg 1. \end{aligned} \tag{5}$$

This arc and its rotations by $\pm\frac{2\pi}{3}$ determine the first-order approximation to the fractal structure. It can also be seen that the origin and points at infinity are images of each other. From

$$\|\mathbf{b_{23}}\| \cdot \|\mathbf{b_1}\| = \|\mathbf{t}\|^2, \tag{6}$$

we can infer that $\mathbf{t}$ will never be the dominant vector and hence that the $\mathbf{x_k}$ will always be separated by angles of approximately $\frac{2\pi}{3}$. These define three branches of the fractal at angles of $\frac{\pi}{3}, \pi, \frac{5\pi}{5}$ with respect to the positive real axis, as can be seen in Fig. 1a. A closer analysis for $|z_0| < 1$ shows that the maximal perturbation of this angle is $\gamma \approx 0.62134$. We also obtain from (3) that $\mathbf{t}$ is the bisector of the angle between $\mathbf{b_{23}}$ and $\mathbf{b_1}$.

For large $|z_0|$, we can approximate

$$\varphi \approx (z_0)^3, \quad \xi \approx a, \quad \eta \approx b. \tag{7}$$

In this case, all three vectors $\mathbf{b_1}, \mathbf{b_{23}}, \mathbf{t}$ point in the same direction and are of approximately equal length. Therefore, one of the three mappings in (3) generates an image of z_0 in the same general direction, but scaled by a factor of

$$\lambda_{3,\infty} = \frac{3}{2}. \tag{8}$$

As the mappings (3) are locally conformal, this scale factor determines the far-field similarity of the fractal. As can be seen from a geometrical argument, the two other mappings generate points close to the origin.

For small $|z_0|$, that is $\|(a,b)\|^3 \ll 1$, we can approximate by binomial expansion of φ

$$\varphi \approx -\tfrac{(z_0)^6}{4}, \quad \xi \approx -\tfrac{a^2-b^2}{\sqrt[3]{4}}, \quad \eta \approx -\sqrt[3]{2}ab. \tag{9}$$

We denote the circle with radius $\frac{1}{\sqrt[3]{2}}$ as the *attractive circle*. Choosing z_0 inside the attractive circle, it can be established that of the three mappings in (3), one generates an image outside the circle (*prolongation*) in the same direction as (a,b) and the other two generate an image on each of the two remaining branches (*rotation*). One of the images will be reflected with respect to the central symmetry axis $\mathcal{L}_s$ of the branch. Fig. 1b depicts the arrangement of the images for (a,b) lying close to the negative real axis. The bold lines are the symmetry axes $\mathcal{L}_s$. The dotted circle is the attractive circle. Point 1 is a result of prolongation and points 2 and 3 emerge from rotations. The basin of attraction for the root $(1,0)$ is greyshaded in the background.

From this description, we can infer that the Newton iterates can only change from one branch of the fractal to another inside the attractive circle. Any movement of iterates outside the attractive circle is confined to a lateral movement along the branch towards the origin. Due to local conformity, the iterate between Julia points of order $(n+1)$ and n can only move to the

region between Julia points of order n and $(n-1)$ - a point that will be useful in understanding the convergence rate of the orthodox Newton method.

By the same argument about the order of Julia points, we see that the changing of branches via rotations of the iterate inside the attractive circle leads to a "nesting" of images. The images are basically slightly deformed copies of the arc in (5). Therefore the self-similar structure can be thought of as emerging in the attractive circle and then being copied to the infinite series of lobes on each branch outside the attractive circle which grows as a geometric progression with ratio $\lambda_{3,\infty}$.

Concerning the change of branch inside the attractive circle, we examine the fixed points of (2) for cycle 2 and 3 (cycle 1 yields the trivial solution of (2)). Using a local linear approximation around z_0, we obtain the scale factor associated with such a change of branch for an infinitesimal ϵ as

$$\Lambda_3(z_0) = \lim_{\epsilon \to 0} \left| \frac{z(z_0+\epsilon) - z(z_0)}{\epsilon} \right| = \frac{1}{2} \left| \frac{z}{z - z_0} \right|. \tag{10}$$

Choosing z and z_0 according to the definition of fixed points of cycle 2 or 3 (for example $z(z(\zeta_i)) = \zeta_i$, $z(\zeta_1) = \zeta_2$, $z(\zeta_2) = \zeta_1$ for cycle 2), we arrive after some algebra at scale factors of

$$\Lambda_{3,2} = \tfrac{1}{\sqrt{6}}, \qquad \Lambda_{3,3} = \tfrac{1}{\sqrt{12}} \tag{11}$$

for mappings associated with branch changes. These factors (which can be confirmed numerically) can be used together with $\lambda_{3,\infty}$ and the formation of the fractal explained above to give an estimate of the fractal dimension of 1.801.

2.2.1 Generalisation to Higher-order problems

We remark that many of the results obtained for $\nu = 3$ generalise to the case of larger ν. For example, it is possible to state global and local symmetries (3- and 6-fold for $\nu = 3$) for the general case, obtain the global scale factor $\lambda_{\nu,\infty}$ and also several local scale factors. The fact that the ν^{th} order fractal exhibits a $\nu(\nu-1)$-fold local symmetry is worth mentioning, as it means that the fractal gets more dense with ν increasing. It can also be established that the fractal generation mechanism is similar to the case $\nu = 3$, although an explicit form of the inverse mapping as in (3) is not available for the general case. However useful in predicting properties of the ν^{th} order fractal, these results add no further insight into the numerical behaviour over what can already be inferred from $\nu = 3$. They are therefore not considered further in this paper.

3 A Stabilised Newton Method

After having established the fractal properties of the orthodox Newton method, we consider stabilising modifications that have been established for engineering problems ([2], [4], [6]) and examine how their use affects the fractal pattern.

3.1 Definitions

We define two positive limiters s_u, s_d and with these state Newton's method with dynamic shift scaling in one dimension for a scalar function $h(x)$ as

$$s = \frac{h(x^{(k)})}{h_x(x^{(k)})} \tag{12}$$

$$s_N = \begin{cases} -\min\left(\left|\frac{s}{x^{(k)}}\right|, s_u\right) & \text{if } s < 0 \\ \min\left(\frac{s}{x^{(k)}}, s_d\right) & \text{if } s > 0 \end{cases} \tag{13}$$

$$x^{(k+1)} = x^{(k)}(1 - s_N). \tag{14}$$

In multiple dimensions, the Newton shift s is of course a vector obtained by solving the Jacobian system $\mathbf{Js} = \mathbf{f}(\mathbf{x}^{(k)})$. The scaling restrictions s_N are then found independently for all components of s and the most restrictive one is used.

We note that in the limit $s_u, s_d \to \infty$, the method with shift scaling is equivalent to the orthodox Newton method. We also remark that a change of sign in any component of the updated iterate $\mathbf{x}^{(k)}$ is impossible for positive components for $s_d \leq 1$ and for negative components for $s_u \leq 1$.

3.2 Changes to the Fractal Structure

We consider Cayley's problem with $\nu = 3$. From the discussion of the orthodox Newton method we know that the fractal is generated by rotations of $\pm\frac{2\pi}{3}$, thus involving changes of the quadrant by the iterates in the complex plane. By the above argument, these changes cannot occur for $s_d \leq 1$ or $s_u \leq 1$, and hence the fractal structure cannot emerge. However, points with positive real part that would normally converge to one of the roots in the negative half-plane are forced to stay positive. Therefore, convergence is greatly impaired by such radical shift limits. It is therefore not sensible to allow the limits to be less than 1.

For limits greater than 1, the chain of Julia points leading to the attractive circle on each branch of the fractal can still exist. For a more detailed description of this case, we consider Newton mappings from close to the origin on the arc defined by (5) inside the attractive circle onto the negative real axis. In this case, we can write for the Newton shift

$$s = -(a + x) \tag{15}$$

with $-a$ being the image of x after one Newton step. Using the approximation (5) close to the origin, we obtain the shift limiting condition

$$s_d < 1 + 9a^3 \tag{16}$$

and hence

$$a_d = \sqrt[3]{\tfrac{1}{9}(s_d - 1)}. \tag{17}$$

This means that for a given limit s_d, no point left of $-a_d$ on the negative real axis will have an image inside the attractive circle under the inverse Newton mapping (3). Therefore, none of the Julia points to the left of $-a_d$ can have an image on another branch and be the 'parent' of a sequence of Julia points on that branch. The closer s_d gets to 1, the shorter the sequence of Julia points approaching the origin on the arc described by (5) becomes. For some value $s_d = 1 + \epsilon$, the first Julia point on the negative real axis χ_1 has no longer an image inside the attractive circle, and the fractal chain on this side breaks down completely. The corresponding value can be determined from (17) as $s_d = 5.5$. As the shift scaling breaks the global symmetry, the critical values for the other branches of the fractal are lower and also involve conditions on s_u. The general argument, however, prevails and can be verified in Fig. 2. As every n^{th} order Julia point is by definition an image of the origin after n locally conformal inverse Newton mappings (3), the presented argument holds throughout the fractal structure.

4 Numerical Relevance

Having understood the formation of the fractal, we now turn to numerical experiments. Relating the theoretical results presented in section 2 to numerically observable quantities, a case study on the orthodox method and $\nu = 3$ is discussed. Then, the method with dynamic shift scaling is compared to the orthodox method for several ν and finally a problem involving a transcendental function is solved to demonstrate that the presented results generalise beyond simple polynomials.

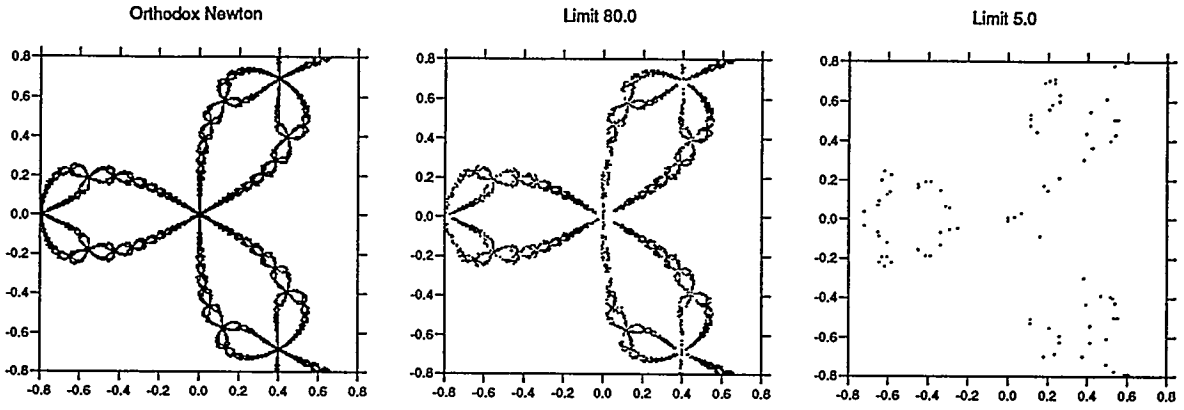


Figure 2: Julia sets for $z^3 = 1$ with various shift limits.

4.1 The orthodox method

We start numerical experiments from the points given in table 1. The distance to the nearest root in the L_2 norm is denoted by d.

	x	y	d
z_1	-0.593573840026322	-0.146431149219563	0.7257
z_2	0.538607827019894	-0.417205032532133	0.6220
z_3	0.538607827019894	-0.35	0.5791
z_a	0.538607827019894	-0.417205031032133	0.6220
z_b	0.538607829019894	-0.417205031032133	0.6220

Table 1: Starting points for numerical experiments

We see that the three points z_2, z_a and z_b only differ after the 8^{th} digit of one of their coordinates. Their convergence history is depicted in Fig. 3. As expected, their residuals are

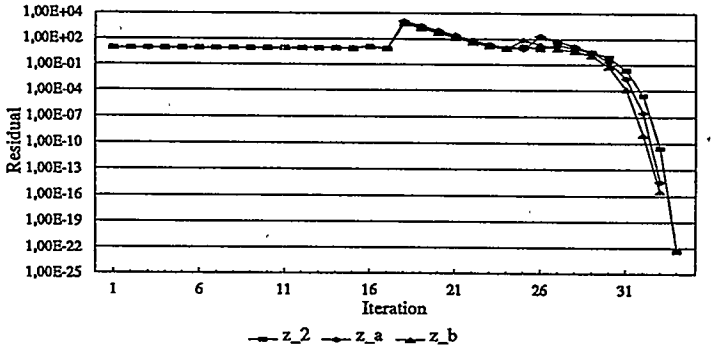


Figure 3: Convergence history for three starting points close to a Julia point

identical in the beginning and differ slightly after iteration 18. Then the path of their iterates leaves the attractive circle, on what can be thought of as an inverse prolongation step, hence the increase in residual. It emerges that the starting points converge to different roots, namely

$$z_2 \to (1,0), \quad z_a \to (-\tfrac{1}{2}, \tfrac{\sqrt{3}}{2}), \quad z_b \to (-\tfrac{1}{2}, -\tfrac{\sqrt{3}}{2}). \tag{18}$$

From the slight differences in residual after step 18, we can infer that the inverse prolongation was by almost the same distance, but onto three different branches of the fractal. As the three

points have been chosen to lie close to a Julia point, this confirms the well-known result that each Julia point lies on the boundary of the basin of attraction of all roots. It should also be noted that, as each Julia point by definition eventually is mapped onto the origin, a fractal pattern contains infinitely many singularities.

From the principles of fractal formation, we can predict four convergence patterns.

- A stationary residual with magnitude 1, when the iterates stay inside the attractive circle and constantly change branches.
- A sharp increase in residual when the iterates leave the attractive circle with a change of branch. This *has to be* followed by
- Linear convergence with an approximate rate $(\lambda_{3,\infty})^{-1}$ as the iterates move laterally along the branch. This is the *only* type of convergence possible outside the attractive circle for large iterates.
- Quadratic convergence once the iterates get sufficiently close to the root.

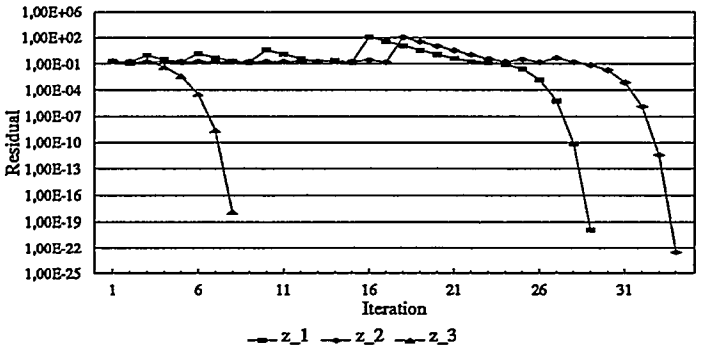


Figure 4: Convergence history for starting points inside the attractive circle

These patterns are confirmed by Fig. 4. The iterates starting from z_2 stay inside the attractive circle until iteration 18, when they leave for the branch with angle π and after some lateral movement along that branch converge to (1,0). The sequence of iterates starting from z_1 leaves the attractive circle four times before finally arriving on the correct branch and then converging to (1,0). Finally, z_3 converges immediately as it was chosen to lie outside the fractal region.

4.2 The stabilised method

The main advantage of a method with no fractal character is a drastic reduction in the number of singularities. Another point is a higher predictability of the convergence path, avoiding situations as depicted in Fig. 3 where a small initial perturbation leads to completely different results. To examine this, an array of 240×240 starting points was cast over the region $[-2,2] \times [-2,2]$ and each point converged to a root. The number of iterations for that was averaged to give a measure of how fast the method converges and times to convergence were taken. Table 2 lists the results. The limits given refer to the upward and downward limits $s_d = s_u$. The hardware used was an Intel-80486/100 based system. The results confirm the predictions of the theory. Limiting the shifts depletes the fractal structure and therefore improves convergence - however, excessively restrictive limits penalise the change of quadrants and slow convergence down.

The positive effects of stabilisation are more obvious for increasingly ill-conditioned problems, for which the fractal structure is more dense. This is confirmed by the results in table 3. A

Method	limits	average iteration #	time [sec]
orthodox	-	7.997	23.4
stabilised	10.0	7.529	27.4
	1.1	7.667	27.7
	1.01	8.194	30.0

Table 2: Comparison of convergence for $z^3 = 1$

breakdown denotes failure to converge within 500 iterations and is excluded from the averaging (in favour of the method showing the breakdown).

Method	limits	av. iteration #	time [sec]	breakdowns
orthodox	-	10.948	36.7	84
stabilised	5.0	8.910	36.4	4
	1.1	8.789	36.3	0
	1.01	9.458	39.5	0

Table 3: Comparison of convergence for $z^4 = 1$

As an example of a transcendental function, as such not being a finite polynomial,

$$\sin z = \frac{z}{2} \tag{19}$$

was chosen. There exist three roots of this system obtained by spliting into real and imaginary part, all of which are real. The non-trivial roots are the symmetric solutions of $\sin x = \frac{x}{2}$, with $x \approx \pm 1.9$. Table 4 gives the convergence results which are remarkably similar to those of polynomials. Even more striking is the fact that a fractal depletion similar to that described for polynomials can be observed. Fig. 5 shows this effect in the complex plane. For clarity, the black area has been chosen as the union of the basins of attraction for the two non-zero roots.

Method	limits	av. iteration #	time [sec]	breakdowns
orthodox	-	6.087	77.7	30
stabilised	10.0	5.918	78.4	10
	5.0	5.845	77.2	6
	1.1	5.911	74.8	0

Table 4: Comparison of convergence for $\sin z = \frac{z}{2}$

5 Concluding Remarks

In this paper, we have established an understanding of how Newton's method generates fractal structures. Quantitative results for the orthodox Newton method applied to polynomials $z^n = 1$ have been given and modifications proposed which can remedy the fractal structure gradually. The same modifications are used as stabilisations for engineering problems and their convergence behaviour in these cases suggests a connection with the fractal explanation we have given. For example, in many applied problems, sudden sharp jumps in residual followed by a long period of

linear convergence or stagnation phases of the residual occur. Analysing the convergence path of a test problem, it was shown that the fractal results for this problem translate directly into numerical properties.

This work has been mostly concerned with polynomials in two dimensions, but the numerical effects of fractal properties can also be observed for transcendental functions. It appears therefore very likely that the results obtained generalise to other classes of problems, for example higher-dimensional systems, and that fractal properties of Newton's method play an important role in engineering problems. Although quantitative results for these will be much harder, if not impossible, to obtain, it is hoped that the qualitative understanding gained from simple model problems such as polynomials will lead to better adaptions of Newton's method for engineering and physics applications.

Acknowledgements

The author would like to acknowledge the financial support of the German Academic Exchange Service (DAAD) through the programme HSPII/AUFE. He also wishes to express his gratitude to Dr. Ian Sobey, Numerical Analysis Group, Oxford University, and Christian Bracher, Dpt. of Theoretical Physics, Technical University at Munich, for many fruitful discussions and their continued interest in the results.

References

[1] Barnsley, M. *Fractals everywhere.* Academic Press. London. 1988.

[2] Drexler, M. and Rollett, J.S. *An adapted Newton Method for the turbulent $k - \epsilon$ Equations.* in: Morton, K.W. and Baines, M.J. *Numerical Methods for Fluid Dynamics.* Oxford University Press. 1996.

[3] Falconer, K. *Fractal Geometry.* John Wiley and Sons. Chichester. 1990.

[4] Gwilliam, C.S. *Parallel Algorithms for Navier-Stokes Modelling.* D.Phil. thesis. Numerical Analysis Group, University of Oxford. 1993.

[5] Julia, G. *Sur l'iteration des fonctions rationelles.* Journal de Mathématique Pure et Appliquée sér. 8, 47-245. 1918.

[6] Knoll, D.A. and McHugh, P.R. *An inexact Newton Algorithm for Solving the Tokamak Edge Plasma Fluid Equations on a Multiply-Connected Domain.* Journal of Computational Physics 116, 281-291. 1995.

[7] Mandelbrot, B. *The Fractal Geometry of Nature.* W.H. Freeman and Company. New York. 1983.

[8] Ortega, J. and Rheinboldt, W. *Iterative Solution of Nonlinear Equations in Several Variables.* Academic Press. New York. 1970.

[9] Peitgen, H.O. and Saupe, D. *The Science of Fractal Images.* Springer-Verlag. New York. 1988.

[10] Peitgen, H.O., Saupe, D. and Haeseler, F.v. *Cayley's Problem and Julia Sets.* The Mathematical Intelligencer Vol. 6, No. 2. Springer-Verlag. New York. 1984.

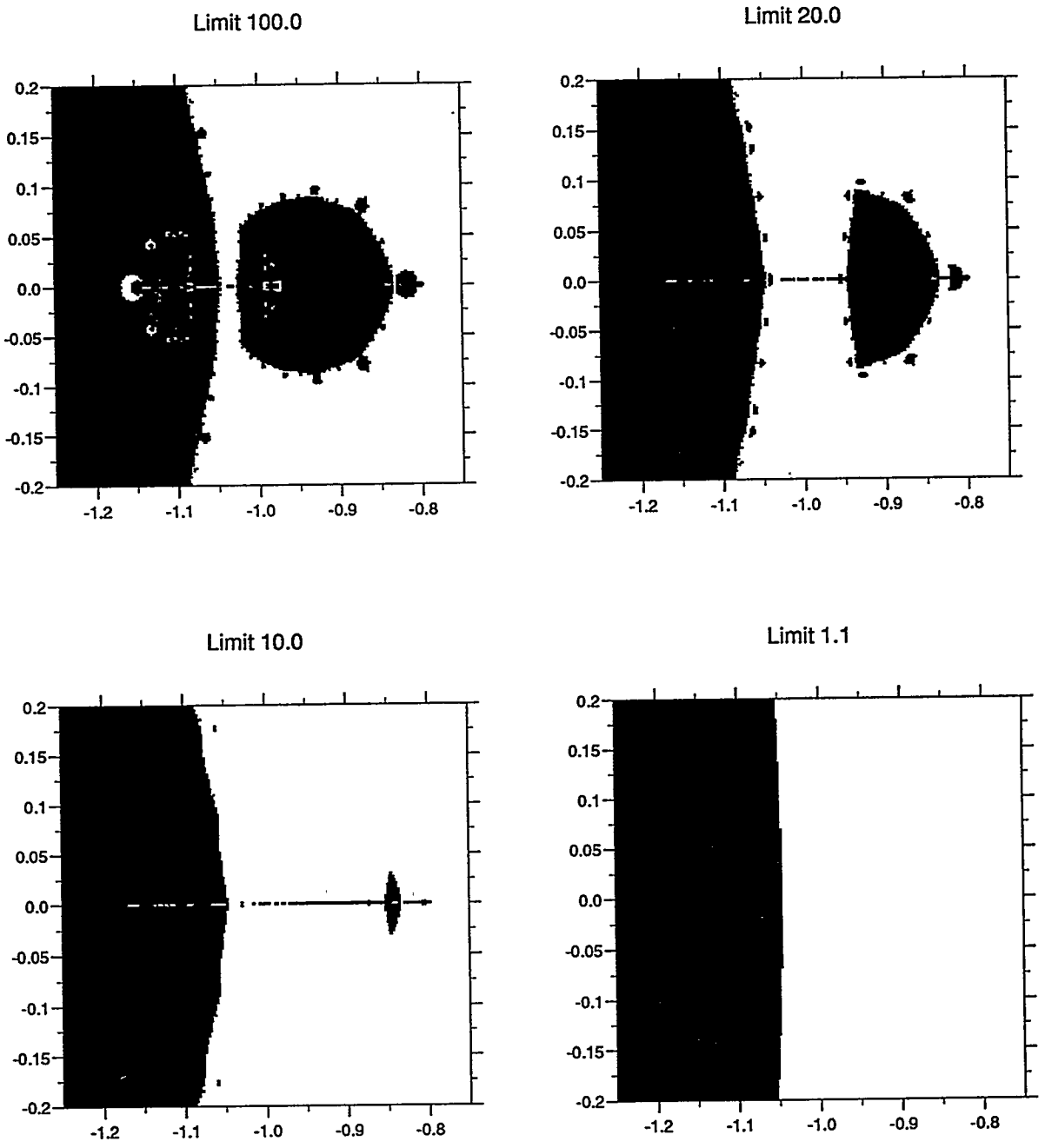


Figure 5: Basins of attraction of non-trivial roots for $\sin z = \frac{z}{2}$, stabilised Newton method.

Topic: *Parallel*	*Session Chair:* *Loyce Adams*	*Room B*
10:30 - 11:00	G.C. Lo	Iterative Solution of General Sparse Linear Systems on Clusters of Workstations
11:00 - 11:30	Q. Yao	New Concurrent Iterative Methods with Monotonic Convergence
11:30 - 12:00	R. McLay	Improving Matrix-Vector Product Performance and Multi-Level Preconditioning for the Parallel PCG Package

Iterative Solution of General Sparse Linear Systems on Clusters of Workstations*

Gen-Ching Lo and Yousef Saad
Department of Computer Science, and
Minnesota Supercomputer Institute
University of Minnesota
Minneapolis, MN 55455

March 15, 1996

Abstract

Solving sparse irregularly structured linear systems on parallel platforms poses several challenges. First, sparsity makes it difficult to exploit data locality, whether in a distributed or shared memory environment. A second, perhaps more serious challenge, is to find efficient ways to precondition the system. Preconditioning techniques which have a large degree of parallelism, such as multicolor SSOR, often have a slower rate of convergence than their sequential counterparts. Finally, a number of other computational kernels such as inner products could ruin any gains gained from parallel speed-ups, and this is especially true on workstation clusters where start-up times may be high. In this paper we discuss these issues and report on our experience with PSPARSLIB, an on-going project for building a library of parallel iterative sparse matrix solvers.

1 Introduction

In the past few years, there has been a flurry of activity on the use of distributed memory computers to solve challenging scientific problems. Of particular interest is the recent surge of activity on the use of workstations clusters. Connecting a small number of workstations by fast communication networks is increasingly gaining acceptance as a low-cost and effective alternative to large supercomputer engines. However, in using iterative methods to solve large sparse linear systems on workstation clusters several problems emerge. First, if few processors are used then any gains that are made from parallelism can easily be wiped out by the overhead, particularly communication. This is not untypical of parallel processing in general but the problem is more acute with workstation clusters because communication costs can be quite high. Second, inner products and other global operations which normally cause few problems can now cause a serious bottleneck. The reasons are the same: the overhead in doing the global sum of the smaller inner products can overwhelm the actual time to carry out the arithmetic. On the other hand, workstation clusters present several advantages over traditional distributed computers. For example, we found that one of the main advantages is that memory is not a serious limitation: in massively parallel architectures, the global memory can be very large but local memory is often small, being limited by physical constraints. In earlier experiments on the CM-5 for example we found

*Work supported in part by ARPA under grant number NIST 60NANB2D1272 and in part by the National Science Foundation under grant NSF/CCR-9214116

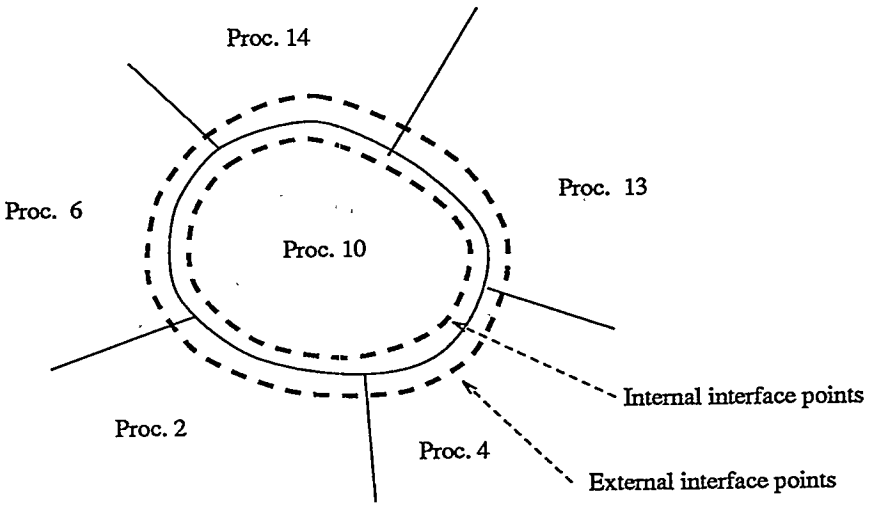


Figure 1: A local view of a distributed sparse matrix.

that in order to obtain satisfactory speed-ups, the problems to be solved would have to be very large, so large in fact that they would not fit in memory given the relatively small size of local memory available. This is not the case on wortkstation clusters.

2 Iterative Solution of Distributed Sparse Systems

We consider a linear system of the form

$$Ax = b, \tag{1}$$

where A is a large sparse nonsymmetric real matrix of size n. When mapping such a system into a parallel computer, it is natural to assign pairs of equations-unknowns of a given sparse linear system to the same processor. Each processor holds a set of equations (rows of a sparse linear system) and a vector of the variables associated with these rows. These sets, i.e., the mapping of the unknowns to processors, can be determined by a graph partitioner or *ad hoc* from knowledge of the problem. We assume (for simplicity only) that each processor is assigned only one subgraph (or subdomain, in the PDE literature). This natural way of distributing a sparse linear system is fairly general and is closely related to the physical viewpoint. Rather than starting from the standard natural ordering used in the sequential setting, we look at the system as a distributed object and try to develop techniques for the global system. This is often done in the domain decomposition literature – but it has rarely been exploited in sparse matrix methodology.

2.1 Basic Concepts

Figure 1 illustrates the viewpoint adopted in defining solution techniques for distributed sparse linear systems. This representation borrows from the domain decomposition literature – so the term 'subdomain' is often used instead of the more proper term 'subgraph'. Each point (node) in a 'subdomain' in the figure) is actually a pair representing an equation and an associated unknown. It is important to distinguish between three types of unknowns: (1) Interior variables are those coupled with local variables only; (2) Local interface variables are those coupled with non-local (external) variables as well as local variables; and (3)

External interface variables are those variables in other processors which are coupled with local variables.

For example, a block SOR iteration can be easily carried out as a sequence of annihilations or eliminations of the residual components of the system which are local to the processor. Each elimination provides a correction to these local variables of the unknown vector. These variables are then updated as well as the global residual vector. In order to implement this, all that is required is a global order in which to perform these eliminations as well as some global stopping criterion.

2.2 Distributed Krylov accelerators

In this paper we only consider the GMRES algorithm introduced in [12], which is a technique that minimizes the 2-norm of the residual vector $b - Ax$ over x in the so-called Krylov subspace

$$K_m = \text{Span}\{r_0, Ar_0, \ldots, A^{m-1}r_0\},$$

where r_0 is the initial residual vector $b - Ax_0$.

The main operations in a standard Krylov subspace acceleration are (1) vector updates, (2) dot-products, (3) matrix-vector products and (4) preconditioning operations. In some algorithms, such as GMRES, most of the operations (1) and (2) are imbedded in an orthogonalization procedure such as Gram-Schmidt.

2.3 Preconditioning

Most of the preconditioning discussed in this paper are block preconditioners in which blocking is based on the domains. When using these block preconditioners, we solve with large blocks associated with entire domains and it becomes important to allow the preconditioner itself to be an iterative solver. This means that the GMRES iteration should allow the preconditioner to vary from step to step within the inner GMRES process. One variant of GMRES which allows this is called the Flexible variant of GMRES (FGMRES) [8]. It is derived by observing that in the last step of the standard GMRES algorithm, the approximate solution is formed as a linear combination of the preconditioned vectors $z_i = M^{-1}v_i, i = 1, \ldots, m$, where the v_i's are the Arnoldi vectors [12]. Since these vectors are all obtained by applying the same preconditioning matrix M^{-1} to the v's, we need not save them. We only need to apply M^{-1} to the linear combination of the $v's$. If the preconditioner varies at every step, then we need to save the 'preconditioned' vectors $z_j = M_j^{-1}v_j$ to use them when computing the approximate solution. For further details on the algorithm, see [8].

Our implementation of FGMRES exploits a feature referred to as a "reverse communication mechanism" whose goal is to avoid data structures. When calling a standard FORTRAN subroutine implementation of an iterative solver, we normally need to pass a list of arguments related to the matrix A and to the preconditioner. This can be a burden on the programmer because of the rich variety of existing data structures. The solution is not to pass the matrices in any form. Whenever a matrix – vector product or a preconditioning operation is needed, we can simply exit the subroutine and have the subroutine caller perform the desired operation and call the subroutine again, after placing the desired result in one of the vector arguments of the subroutine.

Distributed Block Jacobi Preconditioner One of the most natural preconditioners to use for solving distributed linear systems is the block Jacobi approach. The blocks here refer to a solve with an entire domain. We use a standard (sequential) ILUT preconditioner [10] combined with GMRES. A factor which can affect convergence is the tolerance used for the inner solve. As accuracy increases the cost increases with little additional reductions in the step number. On the other hand using too inaccurate inner solves increases the number of outer iterations and increases the cost. This is illustrated in table 3 in the next section. We should note that the implementation of a Block-Jacobi step is very similar to that of a matrix-vector product.

SOR and SSOR preconditioners As was mentioned earlier, a block SOR iteration can be easily implemented provided a global ordering for the domains is chosen. Thus, if the global ordering of the domains is the ordering defined by their labels, the SOR iteration as executed in Processor *loc* would be as follows:

ALGORITHM **2.1**

1. *Receive external interface variables from domains j with $j < loc$*
2. *Update local residual $r_{loc} = (b - Ax)_{loc}$*
3. *Compute $x_{loc} := x_{loc} + A_{loc}^{-1} r_{loc}$*
4. *Send external interface variables to domains j with $j > loc$*

Algorithm 2.1 is executed on each processor and a convergence test on the global residual or some measure of the error must be included. This block SOR algorithm is the simplest form of the Multiplicative Schwarz procedure used in domain decomposition techniques [1, 6, 7, 5]. Many variations are possible, including overlapping of the domains, inaccurate solves in step 4, inclusion of a relaxation parameter ω, etc.

The global ordering can be based on an arbitrary labeling of the processings provided two neighboring domains have a different label. The most common global ordering is a multi-coloring of the domains, which maximizes parallelism [4, 3, 2, 9].

2.4 Matrix-by-vector products

Consider the matrix-vector (matvec) operation. For a distributed matrix, we can view a matrix-vector product essentially as a local operation. To be more precise, in order to carry-out a matvec, we need to perform the local-matrix by local-vector product and then add to the result any contributions from external interface points. The rows of the matrix assigned to a certain processor, say processor k, can be split in two pieces: a *local* matrix A_i which acts on the local variables and a *interface* matrix B_i which acts on remote variables. These remote variables must be first communicated to the processor(s) which need them before the matrix-vector product can be completed in these processors.

Thus, a matrix-vector product can be performed by the following sequence of operations: (1) Multiply the local matrix B_{loc} by the local variables; (2) Receive the external variables; (3) Multiply these external variables by the external matrix B_{ext} associated with them and add the result to that obtained from the first multiplication. The first and second steps can be performed simultaneously: A processor can be multiplying B_{loc} by the local variables while waiting for the external variables to be received. For details on the implementation and data structures used in P_SPARSLIB see [11].

2.5 Iterative Solvers

One of the potential bottlenecks in a parallel implementation of GMRES is the orthogonalization of the Krylov vectors in the Arnoldi procedure. As will be seen, the parallel performance of FGMRES can be strongly affected by the (parallel) performance of the orthogonalization procedure used.

In sequential implementations, orthogonalization is typically carried out by a Modified Gram-Schmidt Orthogonalization (MGSO) procedure which is usually sufficient. However, as it is well-known, a difficulty with MGSO in a parallel computing environment is that each of the inner products must be done in sequence. Each global inner product requires one global communication and this counts as one synchronization point. At step j of GMRES we would have exactly $j+1$ synchronization points. This overhead can overwhelm the whole computation when the dimension of the matrix is small. The Classical Gram-Shmidt Orthogonalization (CGSO) algorithm is an alternative to MGSO which reduces these synchronization points and is thus more attractive in parallel environments. In contrast with MGSO, we now have two synchronization points at each step. The first is when we compute all inner products with previous vectors and the second is required when we normalize the vector resulting from the linear combination with previous vectors.

In fact we can further reduce these two synchronization points to only one by reordering the computations in the CGSO algorithm. This 'synchronized version' (referred to as the 'Synchronized Classical Gram-Schmidt Orthogonalization' (SCGSO)) is based on the observation that the normalization of the vector resulting from the linear combination mentioned above can be delayed until the next step of the algorithm.

It is known that CGSO is numerically not as viable as MGSO. Thus, CGSO with reorthogonalization is used. For larger Krylov subspace sizes, orthogonality between basis vectors generated is quickly lost. This can be remedied by selecting a smaller subspace dimension which, unfortunately, often leads to more steps to achieve convergence. A better alternative is to use reorthogonalization.

Table 1 compares MGSO, CGSO, and SCGSO on the SGI cluster with two different machine configurations. In the 4 workstations case, 4 processors means running one processor on each machine; while 8 processors means running two processors on each machine.

3 Results

We report some of our results obtained on the IBM SP2 and the SGI Challenge cluster. The IBM SP2 available to us has a maximum of 10 processors. The SGI challenge workstation cluster consists of three 4-processors Challenge workstations one 8-processors Challenge workstation. First, we compare the iterative solver FGMRES using the distributed block Jacobi preconditioner with three different ways of overlapping domains. The results obtained are summarized in table 1. In the table, Jaco_no stands for block Jacobi with no overlapping, Jaco_ov_av for block Jacobi with overlapping and averaging the overlapping data, and Jaco_ov for block Jacobi with overlapping and exchange of overlapped data. In the table *its* is the number of FGMRES iterations. The comparison shows that overlapping can reduce total number of iterations. Comparing Jaco_ov with Jaco_ov_av, we find that Jaco_ov is faster.

The next results are obtained on the IBM SP2. Table 3 shows how the accuracy of the inner solver can affect the outer solver on both number of iterations and time.

Figures 2 presents the the break-down of the times spent in a typical solution. The re-

Table 1: The behavior of Gram-Schmidt algorithms on SGI cluster

1 workstation				
Matrix Size	**Procs**	**MGSO**	**CGSO**	**SCGSO**
21200	1	107.8	107.8	109.5
	4	87.3	53.1	46.7
	8	102.6	44.8	36.0
62424	1	335.1	357.0	358.2
	4	170.2	116.5	111.5
	8	138.8	80.1	69.3
4 workstations				
21200	1	107.8	107.8	109.5
	4	167.3	110.5	70.9
	8	184.6	117.0	67.1
62424	1	335.1	357.0	358.2
	4	261.5	272.0	225.8
	8	208.8	159.4	96.1

sults are for the matrix VENKAT01, using a relative tolerance of $eps = 10^{-6}$ and a Krylov subspace dimension of $im = 15$. The figure shows that the total time decreases as the number of processors increases. The FGMRES time (which is dominated by orthogonalization) increases slightly because of the inner products.

4 Conclusion

Thanks to the availability of communication libraries such as MPI and PVM, and inexpensive network technologies, workstation clusters have recently become one of the most promising paradigms for high performance computing. Our main conclusion from the experiments in this work is that workstation clusters can be effectively used to solve very large sparse linear systems. Small systems can be handled more efficiently on a single workstation. The break-even point depends on many factors and is a function of the architecture parameters and the iterative solution techniques used. For example, any improvements in latency and bandwith will allow us to solve smaller problems efficiently. For very large problems, communication becomes less of an issue as it is small relative to computation. For these problems, avoiding idle time and achieving a more effective utilization of the memory and the processors becomes more important.

References

[1] P. E. Bjørstad and O. B. Widlund. Iterative methods for the solution of elliptic problems on regions partitioned into substructures. *SIAM Journal on Numerical Analysis*, 23(6):1093–1120, 1986.

[2] X. C. Cai and Y. Saad. Overlapping domain decomposition algorithms for general sparse matrices. *Numerical Linear Algebra with Applications*, 1996. To appear.

Table 2: The comparison of FGMRES with distributed block Jacobi preconditioner and three different domain overlappings

Matrix Size	Algorithm	Number of Processors									
		1		2		4		8		16	
		its	time	its	time	its	time	its	time	its	time
SHERMAN5	Jac_no	6	0.42	6	0.42	19	1.16	31	2.32	80	27.08
$n = 3312$	Jac_ov_av	7	0.50	7	0.49	12	0.72	16	1.44	23	8.10
$nnz = 20793$	Jac_ov	7	0.49	7	0.49	12	0.70	16	1.22	30	10.41
RAEFSKY3	Jac_no	4	12.71	9	19.44	11	11.73	13	6.58	8	3.63
$n = 21200$	Jac_ov_av	4	14.88	5	10.36	6	8.72	7	6.26	6	3.98
$nnz = 1488768$	Jac_ov	4	14.78	5	9.71	7	8.16	7	4.05	7	3.22
VENKAT01	Jac_no	5	30.71	13	45.29	14	23.97	16	15.20	16	12.04
$n = 62424$	Jac_ov_av	5	30.92	8	28.37	9	28.80	11	16.79	11	12.69
$nnz = 1717792$	Jac_ov	5	30.97	9	30.80	9	16.39	11	11.26	11	8.25

Table 3: The comparison of different accuracies for the inner solver on IBM SP2

Matrix size	inner its	Number of Processors							
		1		2		4		8	
		its	time	its	time	its	time	its	time
ex37.mat	1	5	0.17	5	0.15	6	0.14	7	0.25
$n = 3565$	3	5	0.14	5	0.13	6	0.11	7	0.12
$nnz = 67591$	5	5	0.14	5	0.13	6	0.11	7	0.11
RAEFSKY3	1	9	4.52	8	2.26	7	1.14	7	0.73
$n = 21200$	3	6	3.92	5	1.75	7	1.60	6	0.71
$nnz = 1488768$	5	4	2.77	4	1.59	7	2.07	6	0.91
	10	4	3.96	4	2.08	7	2.39	6	1.31

[3] X. C. Cai and O. Widlund. Multiplicative Schwarz algorithms for some nonsymmetric and indefinite problems. *SIAM Journal on Numerical Analysis*, 30(4), August 1993.

[4] Xiao-Chuan Cai, William D. Gropp, and David E. Keyes. A comparison of some domain decomposition and ILU preconditioned iterative methods for nonsymmetric elliptic problems. *J. Numer. Lin. Alg. Appl.*, June 1993.

[5] T. F. Chan and T. P. Mathew. Domain decomposition algorithms. *Acta Numerica*, pages 61–143, 1994.

[6] Maksymilian Dryja and Olof B. Widlund. Towards a unified theory of domain decomposition algorithms for elliptic problems. In Tony Chan, Roland Glowinski, Jacques Périaux, and Olof Widlund, editors, *Third International Symposium on Domain Decomposition Methods for Partial Differential Equations, held in Houston, Texas, March 20-22, 1989*. SIAM, Philadelphia, PA, 1990.

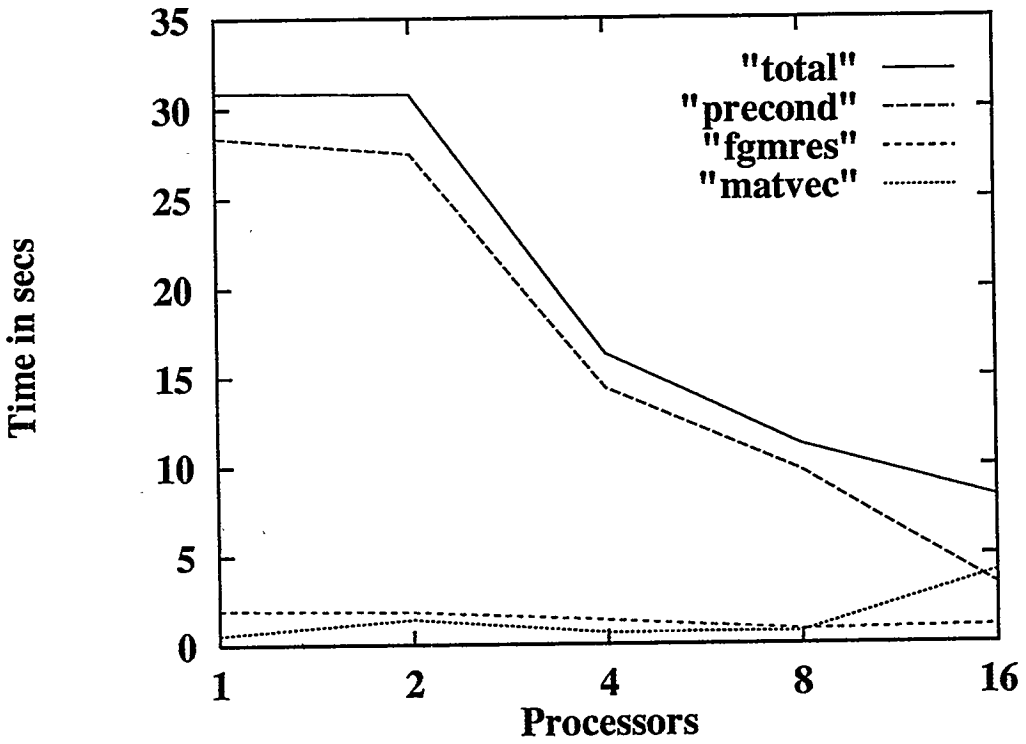


Figure 2: Distributed block Jacobi preconditioner with overlapping

[7] Maksymilian Dryja and Olof B. Widlund. Some recent results on Schwarz type domain decomposition algorithms. In Alfio Quarteroni, editor, *Sixth Conference on Domain Decomposition Methods for Partial Differential Equations*. AMS, 1993. Held in Como, Italy, June 15–19,1992. To appear. Technical report 615, Department of Computer Science, Courant Institute.

[8] Y. Saad. A flexible inner-outer preconditioned GMRES algorithm. *SIAM Journal on Scientific and Statistical Computing*, 14:461–469, 1993.

[9] Y. Saad. Highly parallel preconditioners for general sparse matrices. In G. Golub, M. Luskin, and A. Greenbaum, editors, *Recent Advances in Iterative Methods, IMA Volumes in Mathematics and Its Applications*, volume 60, pages 165–199, New York, 1994. Springer Verlag.

[10] Y. Saad. ILUT: a dual threshold incomplete ILU factorization. *Numerical Linear Algebra with Applications*, 1:387–402, 1994.

[11] Y. Saad and A. Malevsky. PSPARSLIB: A portable library of distributed memory sparse iterative solvers. In V. E. Malyshkin et al., editor, *Proceedings of Parallel Computing Technologies (PaCT-95), 3-rd international conference, St. Petersburg, Sept. 1995*, 1995.

[12] Y. Saad and M. H. Schultz. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7:856–869, 1986.

New Concurrent Iterative Methods With Monotonic Convergence

Qingchuan Yao

Department of Mathematics
Michigan State University
E. Lansing, MI 48824
yao@math.msu.edu

Abstract

This paper proposes the new concurrent iterative methods without using any derivatives for finding all zeros of polynomials simultaneously. The new methods are of monotonic convergence for both simple and multiple real-zeros of polynomials and are quadratically convergent. The corresponding accelerated concurrent iterative methods are obtained too. The new methods are good candidates for the application in solving symmetric eigenproblems.

1 Introduction

Several iterative methods for concurrent inclusion of polynomial zeros can be found in the Petkovic's book [1]. But none of them are of monotonic convergence for finding real zeros of polynomials. In this paper we propose new concurrent iterative methods with monotonic convergence for both simple real zeros and multiple real zeros of polynomials. Our new concurrent iterative methods without derivatives are quadratically convergent.

Recently Li and Zeng have used Laguerre iteration in solving symmetric tridiagonal eigenproblems successfully.The monotonic convergence of the iteration has played an important role. The comprehensive numerical results in this direction can be found in [2]. But in Laguerre iteration, one needs to compute first derivative and second derivative for each iteration. In the application of solving symmetric eigenproblems, it is very expensive to compute first and second derivatives. In this direction, our new concurrent iterative methods with monotonic convergence and without any derivatives are good candidates. The numerical results of our new concurrent iterative methods in solving symmetric eigenproblems will be reported in a separate article.

In §2 we propose a new concurrent iterative method without derivative for simple real-zero of polynomials. We show that the new method is of monotonic convergence and is quadratically convergent. Then we introduce an alternative scheme which reduces the computational cost of each iteration. In §3 we present the corresponding

concurrent iterative method for multiple real-zero of polynomials. In §4, by applying the Gauss-Seidel approach, the accelerated concurrent iterative methods called Single-step Methods are obtained both for simple-zero and multiple-zero of polynomials.

2 Monotonic convergence of the concurrent iterative method and its alternative scheme

In this section we propose a new concurrent iterative method without derivative. The new concurrent iterative method has the monotonic convergence with quadratic convergent rate for approximating the zeros of a polynomial.

Consider a monic polynomial of degree $n \geq 3$

$$P(z) = \prod_{j=1}^{n}(z - \lambda_j) \tag{1}$$

with simple real zeros

$$\lambda_1 < \lambda_2 < \cdots < \lambda_n. \tag{2}$$

We define the pair of sequences $\{x_i^{(k)}\}_{i=1}^n$ and $\{y_i^{(k)}\}_{i=1}^n$ by means of the following iterations:

$$x_i^{(k+1)} = x_i^{(k)} - \frac{P(x_i^{(k)})}{\prod_{j=1}^{i-1}(x_i^{(k)} - x_j^{(k)})\prod_{j=i+1}^{n}(x_i^{(k)} - y_j^{(k)})}, i = 1, \cdots, n, \tag{3}$$

$$y_i^{(k+1)} = y_i^{(k)} - \frac{P(y_i^{(k)})}{\prod_{j=1}^{i-1}(y_i^{(k)} - x_j^{(k)})\prod_{j=i+1}^{n}(y_i^{(k)} - y_j^{(k)})}, i = 1, \cdots, n. \tag{4}$$

with the initial pair of values $\{x_i^{(0)}\}_{i=1}^n$ and $\{y_i^{(0)}\}_{i=1}^n$. The above (3) and (4) are concurrent iterations which can be used to find the approximation zeros of the polynomial $P(z)$ simultaneously. In the following theorem, we will show that the concurrent iterations (3) and (4) are monotonically convergent.

Theorem 1 *For the polynomial (1), if the initial pair of values* $\{x_i^{(0)}\}_{i=1}^n$ *and* $\{y_i^{(0)}\}_{i=1}^n$ *satisfy*

$$x_1^{(0)} \leq \lambda_1 \leq y_1^{(0)} \leq x_2^{(0)} \leq \lambda_2 \leq y_2^{(0)} \leq \cdots \leq x_n^{(0)} \leq \lambda_n \leq y_n^{(0)}, \tag{5}$$

then

(a) *The sequence* $\{x_i^{(k)}\}_{k=0}^{\infty}$ *generated by* (3) *converges to* λ_i *monotonically. That is*

$$x_i^{(0)} < x_i^{(1)} < \cdots < x_i^{(k)} \overset{k\to\infty}{\longrightarrow} \lambda_i, i = 1, \cdots, n. \tag{6}$$

(b) *The sequence* $\{y_i^{(k)}\}_{k=0}^{\infty}$ *generated by* (4) *converges to* λ_i *monotonically. That is*

$$\lambda_i \overset{k\to\infty}{\longleftarrow} y_i^{(k)} < y_i^{(k-1)} < \cdots < y_i^{(1)} < y_i^{(0)}, i = 1, \cdots, n. \tag{7}$$

Proof Let $\varepsilon_i^{(k)} = \lambda_i - x_i^{(k)}$ and $\delta_i^{(k)} = y_i^{(k)} - \lambda_i$ for $i = 1, \cdots, n$. By (3) we have

$$\begin{aligned}\varepsilon_i^{(k+1)} &= \varepsilon_i^{(k)} + (x_i^{(k)} - \lambda_i) \prod_{j=1}^{i-1} \frac{x_i^{(k)} - \lambda_j}{x_i^{(k)} - x_j^{(k)}} \prod_{j=i+1}^{n} \frac{x_i^{(k)} - \lambda_j}{x_i^{(k)} - y_j^{(k)}} \\ &= \varepsilon_i^{(k)} \left[1 - \prod_{j=1}^{i-1} \frac{x_i^{(k)} - \lambda_j}{x_i^{(k)} - x_j^{(k)}} \prod_{j=i+1}^{n} \frac{x_i^{(k)} - \lambda_j}{x_i^{(k)} - y_j^{(k)}} \right]. \end{aligned} \tag{8}$$

Then let $k = 0$ and by (5), if $\varepsilon_i^{(0)} \neq 0$,

$$0 < \varepsilon_i^{(1)} = \varepsilon_i^{(0)} \left[1 - \prod_{j=1}^{i-1} \frac{x_i^{(0)} - \lambda_j}{x_i^{(0)} - x_j^{(0)}} \prod_{j=i+1}^{n} \frac{x_i^{(0)} - \lambda_j}{x_i^{(0)} - y_j^{(0)}} \right] < \varepsilon_i^{(0)};$$

or $\varepsilon_i^{(1)} = 0$ if $\varepsilon_i^{(0)} = 0$. So

$$x_i^{(0)} < x_i^{(1)} < \lambda_i \quad \text{or} \quad x_i^{(0)} = x_i^{(1)} = \lambda_i.$$

Similarly, by (4), we have

$$\begin{aligned}\delta_i^{(k+1)} &= \delta_i^{(k)} - (y_i^{(k)} - \lambda_i) \prod_{j=1}^{i-1} \frac{y_i^{(k)} - \lambda_j}{y_i^{(k)} - x_j^{(k)}} \prod_{j=i+1}^{n} \frac{y_i^{(k)} - \lambda_j}{y_i^{(k)} - y_j^{(k)}} \\ &= \delta_i^{(k)} \left[1 - \prod_{j=1}^{i-1} \frac{y_i^{(k)} - \lambda_j}{y_i^{(k)} - x_j^{(k)}} \prod_{j=i+1}^{n} \frac{y_i^{(k)} - \lambda_j}{y_i^{(k)} - y_j^{(k)}} \right]. \end{aligned} \tag{9}$$

Then let $k = 0$ and by (5), if $\delta_i^{(0)} \neq 0$,

$$0 < \delta_i^{(1)} = \delta_i^{(0)} \left[1 - \prod_{j=1}^{i-1} \frac{y_i^{(0)} - \lambda_j}{y_i^{(0)} - x_j^{(0)}} \prod_{j=i+1}^{n} \frac{y_i^{(0)} - \lambda_j}{y_i^{(0)} - y_j^{(0)}} \right] < \delta_i^{(0)};$$

or $\delta_i^{(1)} = 0$ if $\delta_i^{(0)} = 0$. So

$$\lambda_i < y_i^{(1)} < y_i^{(0)} \quad \text{or} \quad y_i^{(1)} = y_i^{(0)} = \lambda_i.$$

Therefore, by mathematical induction and the above expressions (8) and (9), we can easily obtain (6) and (7). □

Theorem 2 *The convergent rate of the concurrent iterations (3) and (4) is quadratic for the simple zeros of the polynomial (1).*

Proof Let $\varepsilon_i^{(k)} = \lambda_i - x_i^{(k)}$ and $\delta_i^{(k)} = y_i^{(k)} - \lambda_i$ for $i = 1, \cdots, n$. Then, by (8), we have

$$\varepsilon_i^{(k+1)} = \varepsilon_i^{(k)} \frac{N_1}{D_1}$$

where

$$\begin{aligned} N_1 &= \prod_{j=1}^{i-1}(x_i^{(k)} - x_j^{(k)}) \prod_{j=i+1}^{n}(x_i^{(k)} - y_j^{(k)}) - \prod_{j=1}^{i-1}(x_i^{(k)} - \lambda_j) \prod_{j=i+1}^{n}(x_i^{(k)} - \lambda_j) \\ &= \prod_{j=1}^{i-1}[(x_i^{(k)} - \lambda_j) + \varepsilon_j^{(k)}] \prod_{j=i+1}^{n}[(x_i^{(k)} - \lambda_j) - \delta_j^{(k)}] - \prod_{j=1}^{i-1}(x_i^{(k)} - \lambda_j) \prod_{j=i+1}^{n}(x_i^{(k)} - \lambda_j) \\ &= \sum_{l=1}^{i-1}[\varepsilon_l^{(k)} \prod_{j\neq i,l}^{n}(x_i^{(k)} - \lambda_j)] - \sum_{l=i+1}^{n}[\delta_l^{(k)} \prod_{j\neq i,l}^{n}(x_i^{(k)} - \lambda_j)] + O(\varepsilon_i^{(k)}\delta_j^{(k)} + \varepsilon_i^{(k)}\varepsilon_j^{(k)} + \delta_i^{(k)}\delta_j^{(k)}) \end{aligned}$$

and

$$D_1 = \prod_{j=1}^{i-1}(x_i^{(k)} - x_j^{(k)}) \prod_{j=i+1}^{n}(x_i^{(k)} - y_j^{(k)}).$$

Since $\lambda_1, \cdots, \lambda_n$ are distinct, $\forall k$, $\exists d$ and D such that

$$d < |x_i^{(k)} - x_j^{(k)}|, |x_i^{(k)} - y_j^{(k)}|, |y_i^{(k)} - y_j^{(k)}| < D, \forall i \neq j, \forall k;$$

$$d < |x_i^{(k)} - \lambda_j|, |y_i^{(k)} - \lambda_j| < D, \forall i \neq j, \forall k.$$

Let $\varepsilon^{(k)} = max\{\varepsilon_i^{(k)}, \delta_i^{(k)} | i = 1, \cdots, n\}$, then

$$\begin{aligned} \varepsilon_i^{(k+1)} &\leq \varepsilon_i^{(k)} \left|\frac{N}{D}\right| \\ &\leq (\varepsilon^{(k)})^2(n-1)\frac{D^{n-2}}{d^{n-1}} + O\big((\varepsilon^{(k)})^3\big). \end{aligned}$$

Similarly, we can obtain

$$\delta_i^{(k+1)} \leq (\varepsilon^{(k)})^2(n-1)\frac{D^{n-2}}{d^{n-1}} + O\big((\varepsilon^{(k)})^3\big).$$

Thus,

$$\varepsilon^{(k+1)} \leq (n-1)\frac{D^{n-2}}{d^{n-1}}(\varepsilon^{(k)})^2 + O\big((\varepsilon^{(k)})^3\big).$$

□

In the rest of this section, we introduce a suitable alternative scheme of the concurrent iterations (3) and (4) which reduces the computational cost of each iteration. For $i = 1, \cdots, n$, consider the midpoint $c_i^{(k)} = (x_i^{(k)} + y_i^{(k)})/2$ of the interval $[x_i^{(k)}, y_i^{(k)}]$. We may detect if the zero λ_i belongs either to the left half or to the right half of the interval $[x_i^{(k)}, y_i^{(k)}]$. In the former case we apply (4) in the later case we apply (3). That is:

(1) If $c_i^{(k)} > \lambda_i$, then

$$x_i^{(k+1)} = x_i^{(k)}; \tag{10}$$

$$y_i^{(k+1)} = c_i^{(k)} - \frac{P(c_i^{(k)})}{\prod_{j=1}^{i-1}(c_i^{(k)} - x_j^{(k)}) \prod_{j=i+1}^{n}(c_i^{(k)} - y_j^{(k)})}. \tag{11}$$

(2) If $c_i^{(k)} < \lambda_i$, then

$$x_i^{(k+1)} = c_i^{(k)} - \frac{P(c_i^{(k)})}{\prod_{j=1}^{i-1}(c_i^{(k)} - x_j^{(k)})\prod_{j=i+1}^{n}(c_i^{(k)} - y_j^{(k)})}; \quad (12)$$

$$y_i^{(k+1)} = y_i^{(k)}. \quad (13)$$

It is easy to see that the above alternative scheme has monotonically convergence for the zeros of the polynomial (1) too.

3 Monotonic Convergence for Multiple Zeros and Non-overshoot Properties for Cluster of Zeros

From Theorem 2 we know that the concurrent iterations (3) and (4) are of quadratic convergence only for simple zeros. In this section we will consider the multiple-zero case. For a monic polynomial of degree $n \geq 3$

$$P(z) = \prod_{j=1}^{m}(z - \lambda_j)^{n_j} \quad (14)$$

with zeros

$$\lambda_1 < \lambda_2 < \cdots < \lambda_m \text{ and } \sum_{j=1}^{m} n_j = n, \quad (15)$$

we can modify the iterations (3) and (4) to the following Modified Concurrent Iterations:

$$x_i^{(k+1)} = x_i^{(k)} + \sqrt[n_i]{\left|\frac{P(x_i^{(k)})}{\prod_{j=1}^{i-1}(x_i^{(k)} - x_j^{(k)})^{n_j}\prod_{j=i+1}^{m}(x_i^{(k)} - y_j^{(k)})^{n_j}}\right|} \equiv A_{n_i}(x_i^{(k)}); \quad (16)$$

$$y_i^{(k+1)} = y_i^{(k)} - \sqrt[n_i]{\frac{P(y_i^{(k)})}{\prod_{j=1}^{i-1}(y_i^{(k)} - x_j^{(k)})^{n_j}\prod_{j=i+1}^{m}(y_i^{(k)} - y_j^{(k)})^{n_j}}} \equiv B_{n_i}(y_i^{(k)}), \quad (17)$$

for $i = 1, \cdots, m$ and $k = 0, 1, \cdots$. In the next theorem, we show that the Modified Concurrent Iterations (16) and (17) are monotonically convergent for the multiple zeros of the polynomial (14).

Theorem 3 *For the polynomial (14), if the initial pair of values* $\{x_i^{(0)}\}_{i=1}^m$ *and* $\{y_i^{(0)}\}_{i=1}^m$ *satisfy :*

$$x_1^{(0)} \leq \lambda_1 \leq y_1^{(0)} \leq x_2^{(0)} \leq \lambda_2 \leq y_2^{(0)} \leq \cdots \leq x_m^{(0)} \leq \lambda_m \leq y_m^{(0)}, \quad (18)$$

then

(a) *The sequence* $\{x_i^{(k)}\}_{k=0}^{\infty}$ *generated by* (16) *converges to* λ_i *monotonically. That is*

$$x_i^{(0)} < x_i^{(1)} < \cdots < x_i^{(k)} \xrightarrow{k\to\infty} \lambda_i, i = 1, \cdots, m. \quad (19)$$

(b) *The sequence* $\{y_i^{(k)}\}_{k=0}^{\infty}$ *generated by* (17) *converges to* λ_i *monotonically. That is*

$$\lambda_i \overset{k\to\infty}{\longleftarrow} y_i^{(k)} < y_i^{(k-1)} < \cdots < y_i^{(1)} < y_i^{(0)}, i = 1, \cdots, m. \tag{20}$$

Proof Let $\varepsilon_i^{(k)} = \lambda_i - x_i^{(k)}$ and $\delta_i^{(k)} = y_i^{(k)} - \lambda_i$ for $i = 1, \cdots, m$. By (16) we have

$$\begin{aligned}
\varepsilon_i^{(k+1)} &= \varepsilon_i^{(k)} - |x_i^{(k)} - \lambda_i| \sqrt[n_i]{\prod_{j=1}^{i-1} \left(\frac{x_i^{(k)} - \lambda_j}{x_i^{(k)} - x_j^{(k)}}\right)^{n_j} \prod_{j=i+1}^{m} \left(\frac{x_i^{(k)} - \lambda_j}{x_i^{(k)} - y_j^{(k)}}\right)^{n_j}} \\
&= \varepsilon_i^{(k)} \left[1 - \sqrt[n_i]{\prod_{j=1}^{i-1} \left(\frac{x_i^{(k)} - \lambda_j}{x_i^{(k)} - x_j^{(k)}}\right)^{n_j} \prod_{j=i+1}^{m} \left(\frac{x_i^{(k)} - \lambda_j}{x_i^{(k)} - y_j^{(k)}}\right)^{n_j}}\right].
\end{aligned} \tag{21}$$

Then let $k = 0$ and by (18), if $\delta_i^{(0)} \neq 0$,

$$0 < \varepsilon_i^{(1)} = \varepsilon_i^{(0)} \left[1 - \sqrt[n_i]{\prod_{j=1}^{i-1} \left(\frac{x_i^{(0)} - \lambda_j}{x_i^{(0)} - x_j^{(0)}}\right)^{n_j} \prod_{j=i+1}^{m} \left(\frac{x_i^{(0)} - \lambda_j}{x_i^{(0)} - y_j^{(0)}}\right)^{n_j}}\right] < \varepsilon_i^{(0)};$$

or $\varepsilon_i^{(1)} = 0$ if $\varepsilon_i^{(0)} = 0$. So

$$x_i^{(0)} < x_i^{(1)} < \lambda_i \quad \text{or} \quad x_i^{(0)} = x_i^{(1)} = \lambda_i.$$

Similarly, by (17), we have

$$\begin{aligned}
\delta_i^{(k+1)} &= \delta_i^{(k)} - (y_i^{(k)} - \lambda_i) \sqrt[n_i]{\prod_{j=1}^{i-1} \left(\frac{y_i^{(k)} - \lambda_j}{y_i^{(k)} - x_j^{(k)}}\right)^{n_j} \prod_{j=i+1}^{m} \left(\frac{y_i^{(k)} - \lambda_j}{y_i^{(k)} - y_j^{(k)}}\right)^{n_j}} \\
&= \delta_i^{(k)} \left[1 - \sqrt[n_i]{\prod_{j=1}^{i-1} \left(\frac{y_i^{(k)} - \lambda_j}{y_i^{(k)} - x_j^{(k)}}\right)^{n_j} \prod_{j=i+1}^{m} \left(\frac{y_i^{(k)} - \lambda_j}{y_i^{(k)} - y_j^{(k)}}\right)^{n_j}}\right].
\end{aligned} \tag{22}$$

Then let $k = 0$ and by (18), if $\delta_i^{(0)} \neq 0$,

$$0 < \delta_i^{(1)} = \delta_i^{(0)} \left[1 - \sqrt[n_i]{\prod_{j=1}^{i-1} \left(\frac{y_i^{(0)} - \lambda_j}{y_i^{(0)} - x_j^{(0)}}\right)^{n_j} \prod_{j=i+1}^{m} \left(\frac{y_i^{(0)} - \lambda_j}{y_i^{(0)} - y_j^{(0)}}\right)^{n_j}}\right] < \delta_i^{(0)};$$

or $\delta_i^{(1)} = 0$ if $\delta_i^{(0)} = 0$. So

$$\lambda_i < y_i^{(1)} < y_i^{(0)} \quad \text{or} \quad y_i^{(1)} = y_i^{(0)} = \lambda_i.$$

Therefore, by mathematical induction and the above expressions (21) and (22), we can easily obtain (19) and (20). □

In the next theorem we will show that the modified iterations (16) and (17) have non-overshoot properties for a polynomial with cluster of zeros. That means the modified iterations (16) and (17) will not overshoot n_i zeros of a polynomial. The non-overshoot properties are important for the application of finding the eigenvalues of a symmetric matrix.

Theorem 4 *If $P(z)=\prod_{j=1}^{n}(z-\lambda_j)$ has the cluster of zeros $\{\lambda_i^{(1)},\lambda_i^{(2)},\cdots,\lambda_i^{(n_i)}\}_{i=1}^{m}$ with $\lambda_i^{(1)}\le\lambda_i^{(2)}\le\cdots\le\lambda_i^{(n_i)}<\lambda_{i+1}^{(1)}$ and the initial pair of values $\{x_i\}_{i=1}^{m}$ and $\{y_i\}_{i=1}^{m}$ satisfy*

$$\begin{aligned} x_1 &\le \lambda_1^{(1)}\le\cdots\le\lambda_1^{(n_1)}\le y_1\le x_2\le\lambda_2^{(1)}\le\cdots \\ &\le \lambda_2^{(n_2)}\le y_2\le\cdots\le x_m\le\lambda_m^{(1)}\le\cdots\le\lambda_m^{(n_m)}\le y_m \end{aligned} \tag{23}$$

where $\sum_{j=1}^{m}n_j=n$, then

$$x_i<A_1(x_i)<A_2(x_i)<\cdots<A_{n_i}(x_i)<\lambda_i^{(n_i)},i=1,\cdots,m \tag{24}$$

and

$$\lambda_i^{(1)}<B_{n_i}(y_i)<B_{n_i-1}(y_i)<\cdots<B_1(y_i)<y_i,i=1,\cdots,m \tag{25}$$

where $A_{n_i}(x_i)$ and $B_{n_i}(y_i)$ are the same as in the iterations (16) and (17).

Proof By (16), (17) and (23), we have

$$\begin{aligned} \lambda_i^{(n_i)}-A_{n_i}(x_i) &> (\lambda_i^{(n_i)}-x_i)-\sqrt[n_i]{|x_i-\lambda_i^{(n_i)}|^{n_i}\prod_{j=1}^{i-1}\left(\frac{x_i-\lambda_j^{(1)}}{x_i-x_j}\right)^{n_j}\prod_{j=i+1}^{m}\left(\frac{x_i-\lambda_j^{(n_j)}}{x_i-y_j}\right)^{n_j}} \\ &= (\lambda_i^{(n_i)}-x_i)\left[1-\sqrt[n_i]{\prod_{j=1}^{i-1}\left(\frac{x_i-\lambda_j^{(1)}}{x_i-x_j}\right)^{n_j}\prod_{j=i+1}^{m}\left(\frac{x_i-\lambda_j^{(n_j)}}{x_i-y_j}\right)^{n_j}}\right] \\ &> 0; \end{aligned}$$

$$\begin{aligned} B_{n_i}(y_i)-\lambda_i^{(1)} &> (y_i-\lambda_i^{(1)})-\sqrt[n_i]{(y_i-\lambda_i^{(1)})^{n_i}\prod_{j=1}^{i-1}\left(\frac{y_i-\lambda_j^{(1)}}{y_i-x_j}\right)^{n_j}\prod_{j=i+1}^{m}\left(\frac{y_i-\lambda_j^{(n_j)}}{y_i-y_j}\right)^{n_j}} \\ &= (y_i-\lambda_i^{(1)})\left[1-\sqrt[n_i]{\prod_{j=1}^{i-1}\left(\frac{y_i-\lambda_j^{(1)}}{y_i-x_j}\right)^{n_j}\prod_{j=i+1}^{m}\left(\frac{y_i-\lambda_j^{(n_j)}}{y_i-y_j}\right)^{n_j}}\right] \\ &> 0. \end{aligned}$$

and it is easy to check the followings:

$$\frac{\partial A_{n_i}(x_i)}{\partial n_i}>0 \quad\text{and}\quad \frac{\partial B_{n_i}(y_i)}{\partial n_i}<0.$$

Therefore, we have proved (24) and (25). □

Theorem 5 *The convergent rate of the Modified Concurrent Iterations (16) and (17) are quadratic for a n_i-fold zero λ_i of the polynomial (14).*

Proof It is similar to the proof of Theorem 2. □

4 Single-step Methods

We call the concurrent iterations (3) and (4) for simple-zero or the modified concurrent iterations (16) and (17) for multiple-zero *Total-step Methods*. In this section we consider their corresponding *Single-step Methods* which have the similar convergent properties, but have faster convergence speed.

First consider the simple-zero case. For the polynomial (1), we define the following Single-step Concurrent Iterations (denoted by SSCIS):

$$x_i^{(k+1)} = x_i^{(k)} - \frac{P(x_i^{(k)})}{\prod_{j=1}^{i-1}(x_i^{(k)} - x_j^{(k+1)})\prod_{j=i+1}^{n}(x_i^{(k)} - y_j^{(k)})}, i = 1, \cdots, n, \quad (26)$$

$$y_i^{(k+1)} = y_i^{(k)} - \frac{P(y_i^{(k)})}{\prod_{j=1}^{i-1}(y_i^{(k)} - x_j^{(k+1)})\prod_{j=i+1}^{n}(y_i^{(k)} - y_j^{(k)})}, i = 1, \cdots, n. \quad (27)$$

Similarly to Theorem 1, we can prove the following monotonic convergence theorem.

Theorem 6 *For the polynomial (1), if the initial pair of values* $\{x_i^{(0)}\}_{i=1}^n$ *and* $\{y_i^{(0)}\}_{i=1}^n$ *satisfy*

$$x_1^{(0)} \le \lambda_1 \le y_1^{(0)} \le x_2^{(0)} \le \lambda_2 \le y_2^{(0)} \le \cdots \le x_n^{(0)} \le \lambda_n \le y_n^{(0)}, \quad (28)$$

then

(a) *The sequence* $\{x_i^{(k)}\}_{k=0}^{\infty}$ *generated by* (26) *converges to* λ_i *monotonically. That is*

$$x_i^{(0)} < x_i^{(1)} < \cdots < x_i^{(k)} \overset{k\to\infty}{\longrightarrow} \lambda_i, i = 1, \cdots, n. \quad (29)$$

(b) *The sequence* $\{y_i^{(k)}\}_{k=0}^{\infty}$ *generated by* (27) *converges to* λ_i *monotonically. That is*

$$\lambda_i \overset{k\to\infty}{\longleftarrow} y_i^{(k)} < y_i^{(k-1)} < \cdots < y_i^{(1)} < y_i^{(0)}, i = 1, \cdots, n. \quad (30)$$

For the convergent speed of the single-step iterations (26) and (27), we use the **R**-order of convergence concept by Ortega and Rheinboldt in [4].

Theorem 7 *Let* $\sigma_n > 1$ *be the unique positive root of* $q_n(\sigma) = \sigma^n - \sigma - 1 = 0$. *Then for the* **R***-order of (SSCIS), we have*

$$O_{\mathbf{R}}((\mathrm{SSCIS}), \lambda_i) \ge 1 + \sigma_n.$$

Proof The proof is very long. Omitted here. The idea is similar to the one in [3]. □

Now consider the multiple-zero case. For the polynomial (14), we define the following Single-step Concurrent Iterations (denoted by SSCIM):

$$x_i^{(k+1)} = x_i^{(k)} + \sqrt[n_i]{\left|\frac{P(x_i^{(k)})}{\prod_{j=1}^{i-1}(x_i^{(k)} - x_j^{(k+1)})^{n_j}\prod_{j=i+1}^{m}(x_i^{(k)} - y_j^{(k)})^{n_j}}\right|} \equiv C_{n_i}(x_i^{(k)}); \quad (31)$$

$$y_i^{(k+1)} = y_i^{(k)} - \sqrt[n_i]{\frac{P(y_i^{(k)})}{\prod_{j=1}^{i-1}(y_i^{(k)} - x_j^{(k+1)})^{n_j}\prod_{j=i+1}^{m}(y_i^{(k)} - y_j^{(k)})^{n_j}}} \equiv D_{n_i}(y_i^{(k)}), \quad (32)$$

for $i = 1, \cdots, m$ and $k = 0, 1, \cdots$.

Similarly to Theorem 3 and Theorem 4 we can show the following two theorems.

Theorem 8 *For the polynomial (14), if the initial pair of values* $\{x_i^{(0)}\}_{i=1}^m$ *and* $\{y_i^{(0)}\}_{i=1}^m$ *satisfy :*

$$x_1^{(0)} \le \lambda_1 \le y_1^{(0)} \le x_2^{(0)} \le \lambda_2 \le y_2^{(0)} \le \cdots \le x_m^{(0)} \le \lambda_m \le y_m^{(0)}, \tag{33}$$

then

(a) *The sequence* $\{x_i^{(k)}\}_{k=0}^{\infty}$ *generated by* (31) *converges to* λ_i *monotonically. That is*

$$x_i^{(0)} < x_i^{(1)} < \cdots < x_i^{(k)} \overset{k\to\infty}{\longrightarrow} \lambda_i, i = 1, \cdots, m. \tag{34}$$

(b) *The sequence* $\{y_i^{(k)}\}_{k=0}^{\infty}$ *generated by* (32) *converges to* λ_i *monotonically. That is*

$$\lambda_i \overset{k\to\infty}{\longleftarrow} y_i^{(k)} < y_i^{(k-1)} < \cdots < y_i^{(1)} < y_i^{(0)}, i = 1, \cdots, m. \tag{35}$$

Theorem 9 *If* $P(z) = \prod_{j=1}^n (z - \lambda_j)$ *has the cluster of zeros* $\{\lambda_i^{(1)}, \lambda_i^{(2)}, \cdots, \lambda_i^{(n_i)}\}_{i=1}^m$ *with* $\lambda_i^{(1)} \le \lambda_i^{(2)} \le \cdots \le \lambda_i^{(n_i)} < \lambda_{i+1}^{(1)}$ *and the initial pair of values* $\{x_i\}_{i=1}^m$ *and* $\{y_i\}_{i=1}^m$ *satisfy*

$$\begin{aligned} x_1 \;&\le\; \lambda_1^{(1)} \le \cdots \le \lambda_1^{(n_1)} \le y_1 \le x_2 \le \lambda_2^{(1)} \le \cdots \\ &\le\; \lambda_2^{(n_2)} \le y_2 \le \cdots \le x_m \le \lambda_m^{(1)} \le \cdots \le \lambda_m^{(n_m)} \le y_m \end{aligned} \tag{36}$$

where $\sum_{j=1}^m n_j = n$, *then*

$$x_i < C_1(x_i) < C_2(x_i) < \cdots < C_{n_i}(x_i) < \lambda_i^{(n_i)}, i = 1, \cdots, m \tag{37}$$

and

$$\lambda_i^{(1)} < D_{n_i}(y_i) < D_{n_i-1}(y_i) < \cdots < D_1(y_i) < y_i, i = 1, \cdots, m \tag{38}$$

where $C_{n_i}(x_i)$ *and* $D_{n_i}(y_i)$ *are the same as in the iterations (31) and (32).*

Similarly to Theorem 7 we can show the following theorem.

Theorem 10 *Let* $\sigma_m > 1$ *be the unique positive root of* $q_m(\sigma) = \sigma^m - \sigma - 1 = 0$. *Then for the* **R***-order of (SSCIM), we have*

$$O_{\mathbf{R}}((\text{SSCIM}), \lambda_i) \ge 1 + \sigma_m.$$

References

[1] Miodrag Petkovic, *Iterative Methods for Simultaneous Inclusion of Polynomial Zeros*, Lecture Notes in Mathematics, No.1387, Springer-Verlag, Berlin 1989.

[2] Li, T.Y. and Z. Zeng , *Laguerre's iteration in solving the symmetric tridiagonal eigenproblem—revisited*, SIAM J. Sci. Comput., Vol.15, No.5 (1994) , 1445–1473.

[3] G. Alefeld and J. Herzberger, *On the Convergence Speed of Some Algorithms for the Simultaneous Approximation of Polynomial Roots*, SIAM J. Numer.Anal., Vol.11, No.2 (1974), 237–243.

[4] Ortega, J.M. and W.C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, 1970.

ABSTRACT

Improving Matrix-Vector Product Performance and Multi-Level Preconditioning for the Parallel PCG Package.

R. T. McLay and G. F. Carey

In this study we consider parallel solution of sparse linear systems arising from discretized PDE's. As part of our continuing work on our parallel PCG Solver package, we have made improvements in two areas. The first is improving the performance of the matrix-vector product. Here on regular finite-difference grids, we are able to use the cache memory more efficiently for smaller domains or where there are multiple degrees of freedom. The second problem of interest in the present work is the construction of preconditioners in the context of the parallel PCG solver we are developing [1, 2]. Here the problem is partitioned over a set of processors subdomains and the matrix–vector product for PCG is carried out in parallel for overlapping grid subblocks. For problems of scaled speedup, the actual rate of convergence of the unpreconditioned system deteriorates as the mesh is refined. Multigrid and subdomain strategies provide a logical approach to resolving the problem. We consider the parallel trade-offs between communication and computation and provide a complexity analysis of a representative algorithm. Some preliminary calculations using the parallel package and comparisons with other preconditioners are provided together with parallel performance results.

Acknowledgement: This research is supported by ARPA Grant DABT63-92-C-0024 and the Advanced Technology Program from Texas.

References

[1] Joubert, W. *et al*, **PCG Reference Manual**, CNA-274, Center for Numerical Analysis, University of Texas at Austin, January 1995.

[2] McLay, R. T., S. Swift and G. F. Carey, "Maximizing Sparse Matrix–Vector Product Performance in MIMD Computers", Colorado Conference of Iterative Methods, Breckenridge CO, April 1994.

Topic: *Session Chair:* *Room C*

Preconditioning *Seymour Parter*

10:30 - 11:00	M. Tuma	Approximate Inverse Preconditioning of Iterative Methods for Nonsymmetric Linear Systems
11:00 - 11:30	I. Mishev	Comparison of Different Preconditioners for Nonsymmetric Finite Volume Element Methods
11:30 - 12:00	C.W. Oosterlee	An Evaluation of Parallel Multigrid as a Solver and a Preconditioner for Singular Perturbed Problems

APPROXIMATE INVERSE PRECONDITIONING OF ITERATIVE METHODS FOR NONSYMMETRIC LINEAR SYSTEMS

MICHELE BENZI * AND MIROSLAV TŮMA †

Abstract. A method for computing an incomplete factorization of the inverse of a nonsymmetric matrix A is presented. The resulting factorized sparse approximate inverse is used as a preconditioner in the iterative solution of $Ax = b$ by Krylov subspace methods.

1. Introduction. We describe a method for computing an incomplete factorization of the inverse of a general sparse matrix $A \in \mathbb{R}^{n\times n}$. The resulting factorized sparse approximate inverse, which is guaranteed to exist if A is an H-matrix, can be used as an explicit preconditioner for the solution of $Ax = b$ by conjugate gradient-type methods. The application of the preconditioner only requires matrix-vector products, which can be advantageous on vector and parallel architectures. In contrast, implicit preconditioners (such as ILU) involve the solution of triangular linear systems, whose efficient implementation can be problematic in a parallel environment. The results of our experiments with a sequential implementation of the new method indicate that the approximate inverse preconditioner results in good convergence rates (comparable to those obtained with standard ILU techniques). A detailed study of this method can be found in [3].

2. An incomplete inverse triangular factorization method. A standard approach for constructing an approximate inverse $G \approx A^{-1}$ is to compute G as the matrix which minimizes $\|I - AG\|$ (or $\|I - GA\|$), subject to some sparsity constraint. Here the matrix norm is usually the Frobenius norm and the sparsity structure of G is determined adaptively (see, e.g., [4], [9], [8]; [3] contains a fairly complete list of references). The optimization approach to constructing approximate inverses is not the only possible one. In this section we describe an alternative procedure based on a direct method of matrix inversion, performed incompletely in order to preserve sparsity. This results in a factorized sparse $G \approx A^{-1}$. Being an incomplete matrix factorization method, our procedure resembles classical ILU-type implicit techniques. The construction of the preconditioner is based on an algorithm which computes two sets of vectors $\{z_i\}_{i=1}^n$, $\{w_i\}_{i=1}^n$, which are A-biconjugate, i.e. such that $w_i^T A z_j = 0$ if and only if $i \neq j$. Given a nonsingular matrix $A \in \mathbb{R}^{n\times n}$, there is a close relationship between the problem of inverting A and that of computing two sets of A-biconjugate vectors $\{z_i\}_{i=1}^n$ and $\{w_i\}_{i=1}^n$. If we introduce the matrices

$$Z = [z_1,\ z_2, \ldots, z_n] \quad \text{and} \quad W = [w_1,\ w_2, \ldots, w_n],$$

* Dipartimento di Matematica, Università di Bologna, Italy, and CERFACS, 42 Ave. G. Coriolis, 31057 Toulouse Cedex, France (benzi@cerfacs.fr).

† Institute of Computer Science, Academy of Sciences of the Czech Republic, Pod vodárenskou věží 2, 182 07 Prague 8 - Libeň, Czech Republic (tuma@uivt.cas.cz).

then

$$W^T AZ = D = \mathrm{diag}(p_1, p_2, \ldots, p_n)$$

where $p_i = w_i^T A z_i \neq 0$. It follows that W and Z are necessarily nonsingular and

$$A^{-1} = ZD^{-1}W^T = \sum_{i=1}^{n} \frac{z_i w_i^T}{p_i} \quad . \tag{1}$$

Hence, the inverse of A is known if two complete sets of A-biconjugate vectors are known. Note that there are infinitely many such sets. Matrices W and Z whose columns are A-biconjugate can be explicitly computed by means of a biconjugation process applied to the standard basis vectors $e_1, \ldots, e_n$. This algorithm is essentially due to L. Fox, see Ch. 6 of [7]. In order to describe the procedure, let a_i^T and c_i^T denote the ith row of A and A^T, respectively (i.e., c_i is the ith column of A). The basic A-biconjugation procedure can be written as follows.

THE BICONJUGATION ALGORITHM

(1) Let $w_i^{(0)} = z_i^{(0)} = e_i \quad (1 \leq i \leq n)$

(2) **for** $i = 1, 2, \ldots, n$

for $j = i, i+1, \ldots, n$

$$p_j^{(i-1)} := a_i^T z_j^{(i-1)} \; ; \qquad q_j^{(i-1)} := c_i^T w_j^{(i-1)}$$

end

if $i = n$ **go to** (3)

for $j = i+1, \ldots, n$

$$z_j^{(i)} := z_j^{(i-1)} - \left(\frac{p_j^{(i-1)}}{p_i^{(i-1)}} \right) z_i^{(i-1)} \; ; \qquad w_j^{(i)} := w_j^{(i-1)} - \left(\frac{q_j^{(i-1)}}{q_i^{(i-1)}} \right) w_i^{(i-1)}$$

end

end

(3) Let $z_i := z_i^{(i-1)}$, $w_i := w_i^{(i-1)}$ and $p_i := p_i^{(i-1)}$, for $1 \leq i \leq n$.

Return $Z = [z_1, z_2, \ldots, z_n]$, $W = [w_1, w_2, \ldots, w_n]$ and $D = \mathrm{diag}(p_1, p_2, \ldots, p_n)$.

Some observations regarding this algorithm are in order. In the first place we note that the above formulation contains some redundancy, since in exact arithmetic

$$p_i = w_i^T A z_i = z_i^T A^T w_i = q_i.$$

Therefore, at step i the computation of the dot product $q_i^{(i-1)} = c_i^T w_i^{(i-1)}$ may be replaced by the assignment $q_i^{(i-1)} := p_i^{(i-1)}$. Another observation is the fact that the procedure, as it stands, is vulnerable to breakdown (division by zero), which occurs whenever any of the quantities $p_i^{(i-1)}$

$(= q_i^{(i-1)})$ happens to be zero. It can be shown that in exact arithmetic, the biconjugation algorithm will not break down if and only if all the leading principal minors of A are nonzero. For any nonsingular matrix A there exists a permutation matrix P (or Q) such that the procedure applied to PA (or to AQ) will not break down. As in the LU decomposition with pivoting, such permutation matrices represent row (or column) interchanges on A which can be performed, if needed, in the course of the computation. In finite precision arithmetic, pivoting may be required to promote numerical stability. If the biconjugation process can be carried to completion without interchanges, the resulting Z and W matrices are upper triangular, they both have all diagonal entries equal to one, and satisfy the identity

$$A = W^{-T} D Z^{-1}. \tag{2}$$

We recognize in (2) the familiar LDU decomposition $A = LDU$, where L is unit lower triangular, U is unit upper triangular and D is the diagonal matrix with the pivots down the main diagonal. Because this factorization is unique, we have that the biconjugation algorithm explicitly computes $W = L^{-T}$, $Z = U^{-1}$ and the matrix D, which is exactly the same in (2) and in $A = LDU$. Hence, the process produces an inverse triangular decomposition of A or, equivalently, a triangular decomposition (of the UDL type) of A^{-1}. In exact arithmetic, the p_i's returned by the algorithm are equal to the pivots in the LDU factorization of A.

In [3] we introduced an explicit preconditioning strategy based on the biconjugation process described above. Sparsity in the Z and W factors of A^{-1} is preserved by removing "small" fill in the z- and w-vectors. Because we are interested in general (unstructured) sparse matrices, we use a drop tolerance $T > 0$ to decide which fill-in is to be discarded: new nonzeros are dropped if smaller than T in absolute value. In a previous paper [2] this technique was applied to symmetric positive definite matrices (for which $W = Z$) to construct factorized approximate inverse preconditioners for the conjugate gradient method. A trivial extension of the results in [2] shows that the incomplete biconjugation process (incomplete inverse factorization) will never break down, in exact arithmetic, if A is an H-matrix. For more general matrices it is necessary to safeguard the computation in order to avoid breakdowns. This requires pivot modifications and perhaps some form of pivoting —we refer to [3] for details. The incomplete biconjugation algorithm computes sparse unit upper triangular matrices $\bar{Z} \approx Z$, $\bar{W} \approx W$ and a nonsingular diagonal matrix $\bar{D} \approx D$ such that

$$G := \bar{Z} \bar{D}^{-1} \bar{W}^T \approx A^{-1}$$

is a factorized sparse approximate inverse of A which can be used as an explicit preconditioner for conjugate gradient-type methods for the solution of $Ax = b$. Theoretical properties of the incomplete inverse factorization are studied in [3].

3. Numerical experiments. In this section we present the results of numerical experiments on a range of problems from the Harwell-Boeing collection [6] and from Tim Davis' collection [5]. A comparison between preconditioners based on Frobenius norm minimization and the no-fill ILU(0) preconditioner (see [1]) was carried out in [8]. Here we make a similar comparison between ILU(0) and the approximate inverse preconditioner based on the biconjugation process. Additional experiments can be found in [3], together with implementation details.

On input, all our codes for the computation of the preconditioners check whether the coefficient matrix has a zero-free diagonal. If not, row reordering of the matrix is used to permute nonzeros on the diagonal. For both the ILU(0) and the approximate inverse factorization, we introduced a simple pivot modification to avoid breakdown. Whenever some diagonal element in any of our algorithms to compute a preconditioner was found to be small, in our case less in absolute value than the IEEE machine precision $\epsilon \approx 2.2 \cdot 10^{-16}$, we increased it to 10^{-3}. It is worth mentioning that in our numerical experiments, this safeguarding measure was required far more often for ILU(0) than for the approximate inverse factorization. All matrices used were rescaled by dividing their elements by the absolute value of their largest nonzero entry. No other scaling was used. The right-hand side of each linear system was computed from the solution vector x^* of all ones.

We present results for the biconjugate gradient method (denoted BCG in the tables), the conjugate gradient-squared method (CGS), and GMRES with no restarting and Householder orthogonalization (denoted GMR in the tables). See [1] for a description of these algorithms. We deliberately avoided the use of restarting with GMRES because we are interested in the effect of preconditioning on the rate of convergence rather than in the performance of a particular solver. However, in practical applications restarting should be used in order to reduce the cost of GMRES.

			Its			Time		
MATRIX	N	NNZ	BCG	CGS	GMR	BCG	CGS	GMR
ADD20	2395	17319	388	255	337	7.562	5.494	198.6
FS1836	183	1069	355	336	24	0.446	0.460	0.770
FS5414	541	4285	872	813	213	4.953	4.679	14.19
FS7601	760	5976	100	87	46	1.193	6.075	2.515
HOR131	434	4710	†	†	425	†	†	42.35
JPWH991	991	6027	60	39	56	0.797	0.806	4.817
ORSIRR1	1030	6858	857	†	408	6.721	†	119.0
ORSIRR2	886	5970	593	726	312	3.966	5.371	56.37
ORSREG1	2205	14133	166	180	145	2.699	3.258	33.99
PORES2	1224	9613	†	†	†	†	†	†
RAEFSKY1	3242	294276	277	366	254	68.43	95.51	180.9
SAYLR3	1000	3750	501	437	340	2.837	2.794	75.12
SAYLR4	3564	22316	†	†	†	†	†	†
SHERMAN1	1000	3750	505	449	340	2.975	3.122	75.29
SHERMAN3	5005	20033	†	†	†	†	†	†
SHERMAN4	1104	3786	135	106	119	1.433	1.271	22.46
SHERMAN5	3312	20793	†	†	†	†	†	†
SWANG1	3169	20841	31	18	29	0.797	0.514	2.411
WATT1	1856	11360	174	144	157	4.759	3.896	10.02
WATT2	1865	11550	335	473	46	7.819	8.447	6.534

Table 1: Test problems (N =order of matrix, NNZ =nonzeros in matrix) and convergence results for the iterative methods without preconditioning.

The order N and number NNZ of nonzeros for each test problem are given in Table 1, together with the number of iterations and computing times for the unpreconditioned iterative methods. A † means that convergence was not attained in 1000 iterations for BCG and CGS, and 500 iterations for GMRES. All tests were performed on a SGI Crimson workstation with RISC processor R4000 using double precision arithmetic. Codes were written in standard Fortran 77 and compiled with the optimization option *-O4*. CPU time is given in seconds. The initial guess for the iterative solvers was always $x_0 = 0$. The stopping criterion used was $||r_k||_2 < 10^{-8}$, where r_k is the (unpreconditioned) updated residual.

The following two tables present the results of experiments with the ILU(0) preconditioner and with the approximate inverse preconditioner based on the biconjugation process (hereafter referred to as AIBC). Observe that the number of nonzeros in the ILU(0) preconditioner is equal to the number NNZ of nonzeros in the original matrix, whereas for the AIBC preconditioner fill-in is given by the total number of nonzeros in the factors $\bar{Z}$, $\bar{W}$ and $\bar{D}$. In the tables, the number of nonzeros in AIBC is denoted by *Fill*. Right preconditioning was used for all the experiments. In Table 2 we give the timings for the preconditioner computation, iteration counts and timings for the three iterative solvers preconditioned with ILU(0). The same information is given in Table 3 for AIBC corresponding to two choices of the drop tolerance T. For AIBC we give two timings for the construction of the preconditioner, the first for the DDS implementation using dynamic data structures and the second for the SDS implementation using only static data structures (see [3]).

		ILU(0) – Its			ILU(0) – Time		
MATRIX	P-time	BCG	CGS	GMR	BCG	CGS	GMR
ADD20	0.071	172	119	149	7.002	5.008	44.21
FS1836	0.003	7	5	5	0.016	0.012	0.015
FS5414	0.007	7	4	6	0.067	0.042	0.063
FS7601	0.014	2	2	2	0.031	0.032	0.036
HOR131	0.008	62	46	39	0.566	0.427	0.618
JPWH991	0.009	19	12	18	0.286	0.191	0.434
ORSIRR1	0.009	42	24	38	0.717	0.418	1.450
ORSIRR2	0.008	44	24	37	0.634	0.361	1.216
ORSREG1	0.019	50	27	42	1.828	1.049	4.475
PORES2	0.013	40	27	32	0.884	0.608	1.412
RAEFSKY1	2.457	37	30	35	17.93	14.30	15.06
SAYLR3	0.005	46	35	45	0.491	0.427	1.639
SAYLR4	0.030	41	38	39	2.413	2.357	5.746
SHERMAN1	0.005	46	34	44	0.502	0.393	1.578
SHERMAN3	0.134	82	71	79	5.030	4.630	26.91
SHERMAN4	0.006	34	27	32	0.365	0.340	1.056
SHERMAN5	0.034	35	29	33	1.801	1.576	3.940
SWANG1	0.031	12	7	10	0.661	0.404	0.766
WATT1	0.015	22	14	9	0.771	0.191	0.345
WATT2	0.016	64	57	54	1.892	1.757	4.992

Table 2: Time to form the ILU(0) preconditioner (P-time), number of iterations and time for BCG, CGS and GMRES with ILU(0) preconditioning.

MATRIX	Fill	P-time		AIBC – Its			AIBC – Time		
		DDS	SDS	BCG	CGS	GMR	BCG	CGS	GMR
ADD20	7525	0.639	1.470	15	8	14	0.529	0.306	0.747
	9752	0.639	1.499	15	8	14	0.561	0.321	0.774
FS1836	2191	0.180	0.067	7	4	7	0.022	0.013	0.027
	4169	0.226	0.095	7	4	6	0.026	0.014	0.026
FS5414	4199	0.298	0.148	42	37	28	0.385	0.354	0.457
	5204	0.339	0.168	18	10	15	0.175	0.102	0.198
FS7601	859	0.173	0.128	2	1	2	0.022	0.013	0.027
	1428	0.201	0.139	1	1	1	0.013	0.014	0.018
HOR131	5196	0.234	0.121	65	41	53	0.592	0.389	1.020
	8394	0.339	0.203	45	29	40	0.499	0.331	0.750
JPWH991	7063	0.309	0.262	27	17	25	0.411	0.280	0.733
	11981	0.367	0.309	24	16	22	0.438	0.312	0.671
ORSIRR1	5219	0.249	0.245	46	27	43	0.706	0.439	1.736
	13117	0.346	0.307	26	15	24	0.517	0.316	0.788
ORSIRR2	5284	0.233	0.193	47	27	41	0.646	0.388	1.439
	12634	0.316	0.250	23	15	22	0.417	0.287	0.638
ORSREG1	11886	0.452	0.975	59	29	49	2.007	1.056	5.093
	24454	0.598	1.073	44	24	36	1.809	1.051	3.315
PORES2	18691	0.547	0.504	102	85	90	2.721	2.363	8.660
	23867	0.655	0.568	103	82	87	3.062	2.521	8.448
RAEFSKY1	56607	5.494	12.62	75	62	72	22.24	19.75	25.10
	145951	16.48	25.14	52	42	50	18.88	16.30	18.54
SAYLR3	3650	0.174	0.197	64	43	56	0.724	0.528	2.492
	11002	0.236	0.239	35	25	34	0.547	0.414	1.203
SAYLR4	42768	0.665	2.571	46	37	42	3.172	2.736	7.012
	48362	0.676	2.631	44	37	41	3.183	2.891	6.870
SHERMAN1	3650	0.170	0.198	64	43	56	0.751	0.530	2.466
	8692	0.239	0.228	36	25	36	0.509	0.378	1.268
SHERMAN3	24439	0.755	5.111	121	115	109	8.233	8.331	50.02
	36296	0.882	5.195	100	95	95	7.579	7.697	39.38
SHERMAN4	3936	0.197	0.248	50	40	48	0.622	0.530	2.115
	4957	0.203	0.157	46	35	44	0.592	0.484	1.871
SHERMAN5	21387	0.765	2.358	56	43	52	2.881	2.425	8.574
	26654	0.891	2.447	48	35	43	2.671	2.141	6.309
SWANG1	7723	0.508	1.938	13	8	13	0.577	0.397	0.892
	13252	0.640	1.988	10	6	10	0.481	0.323	0.662
WATT1	10215	0.369	0.693	31	19	12	0.848	0.565	0.487
	17998	0.463	0.763	19	16	10	0.604	0.558	0.432
WATT2	9394	0.402	0.762	72	52	13	1.969	1.536	0.541
	13547	0.458	0.760	63	48	11	1.844	1.504	0.460

Table 3: Time to form the AIBC preconditioner (P-time) using DDS and SDS implementations, number of iterations and time for BCG, CGS and GMRES with AIBC preconditioning.

It appears from these results that the ILU(0) and AIBC preconditioners are roughly equivalent from the point of view of the rate of convergence, with ILU(0) having a slight edge. We also notice that using a more dense approximate inverse preconditioner (obtained with a smaller value of T)

nearly always reduces the number of iterations, although this does not necessarily mean a reduced computing time since it takes longer to compute the preconditioner and the cost of each iteration is increased.

As for the time required to compute the preconditioners, it is obvious that ILU(0) can be computed more quickly. On the other hand, the computation of the AIBC preconditioner is not prohibitive. Our experiments with AIBC show that the overall solution time is almost always dominated by the iterative part, unless convergence is extremely rapid, in which case the iteration part takes slightly less than the computation of the preconditioner. This is in contrast with the *sequential* behavior of approximate inverse preconditioners based on the optimization approach, for which the construction phase can be extremely time-consuming (see the experiments reported in [8]). However, the situation could be reversed in a parallel environment.

4. Conclusions and future work. We have described a new approach to explicit preconditioning with sparse approximate inverses, based on an inverse triangular factorization of A. The results of numerical experiments show that this technique can be quite effective for accelerating the convergence of conjugate gradient-type methods, with convergence rates comparable to those obtained with a standard ILU(0) implicit preconditioner. We conclude that the new preconditioner shows some promise and is competitive with other approximate inverse preconditioners.

We are currently working on vector and parallel versions of our code. In the near future, we plan to carry out a comparison between the technique described in this paper and the methods based on the optimization approach.

REFERENCES

[1] R. Barret, M. Berry, T. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine and H. van der Vorst. *Templates for the Solution of Linear Systems.* SIAM, Philadelphia, 1994.

[2] M. Benzi, C. D. Meyer, and M. Tůma. A sparse approximate inverse preconditioner for the conjugate gradient method. *SIAM J. Sci. Comput.*, 17, 1996 (to appear).

[3] M. Benzi and M. Tůma. A sparse approximate inverse preconditioner for nonsymmetric linear systems. Research Report No. 653, Institute of Computer Science, Czech Academy of Sciences, Prague, October 1995. Submitted for publication.

[4] E. Chow and Y. Saad. Approximate inverse preconditioners for general sparse matrices. Research Report UMSI 94/101, University of Minnesota Supercomputer Institute, Minneapolis, Minnesota, USA, 1994.

[5] T. Davis. Sparse matrix collection. *NA Digest*, Volume 94, Issue 42, October 1994.

[6] I. S. Duff, R. G. Grimes and J. G. Lewis. Users' guide for the Harwell-Boeing sparse matrix collection. Technical Report RAL-92-086, Rutherford Appleton Laboratory, Chilton, England, 1992.

[7] L. Fox. *An Introduction to Numerical Linear Algebra.* Oxford University Press, Oxford, 1964.

[8] N. I. M. Gould and J. A. Scott. On approximate-inverse preconditioners. Technical Report RAL-95-026, Rutherford Appleton Laboratory, Chilton, England, June 1995.

[9] M. Grote and T. Huckle. Parallel preconditioning with sparse approximate inverses. *SIAM J. Sci. Comput.*, 17, 1996 (to appear).

Comparison of Different Preconitioners for Nonsymmtric Finite Volume Element Methods

Ilya D. Mishev

We consider a few different preconditioners for the linear systems arising from the discretization of 3-D convection-diffusion problems with the finite volume element method. Their theoretical and computational convergence rates are compared and discussed.

An evaluation of parallel multigrid as a solver and a preconditioner for singular perturbed problems
Part I: The standard grid sequence

C.W. Oosterlee * and T. Washio ‡

Abstract:

In this paper we try to achieve h-independent convergence with preconditioned GMRES ([13]) and BiCGSTAB ([18]) for 2D singular perturbed equations. Three recently developed multigrid methods are adopted as a preconditioner. They are also used as solution methods in order to compare the performance of the methods as solvers and as preconditioners.
Two of the multigrid methods differ only in the transfer operators. One uses standard matrix-dependent prolongation operators from [3], [5]. The second uses "upwind" prolongation operators, developed in [24]. Both employ the Galerkin coarse grid approximation and an alternating zebra line Gauss-Seidel smoother. The third method is based on the block LU decomposition of a matrix and on an approximate Schur complement. This multigrid variant is presented in [11]. All three multigrid algorithms are algebraic methods.
The eigenvalue spectra of the three multigrid iteration matrices are analyzed for the equations solved on a 33^2 grid, in order to understand the convergence of the three algorithms. Furthermore, the construction of the search directions for the multigrid preconditioned GMRES solvers is investigated by the calculation and solution of the minimal residual polynomials.
For Poisson and convection-diffusion problems all solution methods are investigated and evaluated for finite volume discretizations on fine grids. The methods have been parallelized with a grid partitioning technique, and are compared on an MIMD machine.

AMS(MOS) 65N55, 65F10, 65Y05.

Keywords: Krylov subspace methods, multigrid, robustness, parallel computing, grid partitioning

*GMD, Institute for Algorithms and Scientific Computing, D-53754 Sankt Augustin, Germany
(email: oosterlee @ gmd.de)

‡C&C Research Laboratories, NEC Europe Ltd., D-53757 Sankt Augustin, Germany
(email: washio @ ccrl-nece.technopark.gmd.de)

Topic: *Nonlinear*	*Session Chair:* *Homer Walker*	*Room A*
4:45 - 5:15	M. Pernice	NITSOL: A Newton Iterative Solver for Nonlinear Systems
5:15 - 5:45	M. Trummer	Nonsymmetric Systems Arising in the Computation of Invariant Tori
5:45 - 6:15	R. Renaut	Multisplitting for Linear, Least Squares and Nonlinear Problems

NITSOL: a Newton Iterative Solver for Nonlinear Systems

Michael Pernice*
Utah Supercomputing Institute
University of Utah
Salt Lake City, UT 84112
usimap@sneffels.usi.utah.edu

Homer F. Walker†
Mathematics and Statistics Department
Utah State University
Logan, UT 84322-3900
walker@math.usu.edu

Abstract.

Newton iterative methods, also known as truncated Newton methods, are implementations of Newton's method in which the linear systems that characterize Newton steps are solved approximately using iterative linear algebra methods. Here, we outline a well-developed Newton iterative algorithm together with a Fortran implementation called NITSOL. The basic algorithm is an inexact Newton method globalized by backtracking, in which each initial trial step is determined by applying an iterative linear solver until an inexact Newton criterion is satisfied. In the implementation, the user can specify inexact Newton criteria in several ways and select an iterative linear solver from among several popular "transpose-free" Krylov subspace methods. Jacobian-vector products used by the Krylov solver can be either evaluated analytically with a user-supplied routine or approximated using finite differences of function values. A flexible interface permits a wide variety of preconditioning strategies and allows the user to define a preconditioner and optionally update it periodically. We give details of these and other features and demonstrate the performance of the implementation on a representative set of test problems.

*Presenting author.

†The work of this author was supported in part by United States Department of Energy Grant DE-FG03-94ER25221 and National Science Foundation Grant DMS-9400217, both with Utah State University.

Nonsymmetric systems arising in the Computation of Invariant Tori

Manfred R. Trummer
Department of Mathematics and Statistics
Simon Fraser University
Burnaby, British Columbia, Canada V5A 1S6
trummer@sfu.ca

January 11, 1996

Abstract. We introduce two new spectral implementations for computing invariant tori. The underlying nonlinear partial differential equation although hyperbolic by nature, has periodic boundary conditions in both space and time. In our first approach we discretize the spatial variable, and find the solution via a shooting method. In our second approach, a full two-dimensional Fourier spectral discretization and Newton's method lead to very large, sparse, nonsymmetric systems. These matrices are highly structured, but the sparsity pattern prohibits the use of direct solvers. A modified conjugate gradient type iterative solver appears to perform best for this type of problems. The two methods are applied to the van der Pol oscillator, and compared to previous algorithms. Several preconditioners are investigated .

Key words. dynamical systems, nonlinear ODEs, invariant torus, pseudospectral method, iterative methods for nonsymmetric systems .

AMS(MOS) subject classifications. 65P05, 65N30, 35K55.

Multisplitting For Linear, Least Squares and Nonlinear Problems

Rosemary Renaut
Arizona State University

In earlier work, presented at the 1994 Iterative Methods meeting, a multisplitting (MS) method of block relaxation type was utilised for the solution of the least squares problem, and nonlinear unconstrained problems. This talk will focus on recent developments of the general approach and represents joint work both with Andreas Frommer, University of Wupertal for the linear problems and with Hans Mittelmann, Arizona State University for the nonlinear problems.

The least squares problem can be solved using MS at the domain level. This amounts to a column partitioning of the underlying system matrix. Equivalently, it can be shown that the MS solution actually finds the solution of the normal equations using MS on the system matrix defined by the normal equations. Hence, this corresponds to the solution of a system of equations with SPD system matrix. According to the standard theory for MS methods this iteration will converge if and only if the underlying split is P-regular. The theory for convergence of the least squares problem shows, however, that convergence is in this case guaranteed. Hence the splitting must be P-regular. This leads us to suppose that the MS of any SPD linear system is necessarily P-regular, and provides a convergent iteration. In this talk I will indicate the convergence theory that leads to this conclusion and investigate how we might seek an alternative proof using the usual techniques from linear algebra.

Block-Jacobi methods are practical for parallel implementations but the rate of convergence is limited. To attempt to remedy this inherent problem MS can be reformulated as a two-phase process. Two-stage methods, in which a splitting is applied to the already split method have already shown to be advantageous in some cases, see Szyld et al. Here a different strategy, which originates in the Parallel Variable Distribution idea of Mangasarian et al, is suggested. At each iterative step there is some flexibility in the manner in which the updated global solution is obtained. Standard MS uses a convex combination of the recent local solutions with a fixed choice of weights. Instead the minimal solution can be found by carrying out a local optimization with respect to these weights. Typically the number of subproblems indicated by the MS is substantially less than the original size of the problem. Therefore this optimization can be carried out locally to one processor with minimal extra cost. For the linear problems this minimization can actually be formulated as a least squares problem with a system matrix that only needs to be factorized at the very first iteration. Results demonstrating the increased efficiency will be presented.

These ideas have also been applied for the solution of large-scale unconstrained convex minimization problems. Again, in order to not have the convergence order limited to linear we have started to explore an alternative approach. This combines an exact augmented Lagrangian with a truncated Newton iteration. Preliminary results will be presented.

Topic: *Parallel*	*Session Chair:* *Loyce Adams*	*Room B*
4:45 - 5:15	V. Menkov	Solving Block Linear Systems with Low-Rank Off-Diagonal Blocks is Easily Parallelizable
5:15 - 5:45	Y. Shapira	Parallelizable Approximate Solvers for Recursions Arising in Preconditioning
5:45 - 6:15	D. Xie	New Parallel SOR Method by Domain Partitioning

SOLVING BLOCK LINEAR SYSTEMS WITH LOW-RANK OFF-DIAGONAL BLOCKS IS EASILY PARALLELIZABLE

V. MEŃKOV
DEPARTMENT OF COMPUTER SCIENCE
INDIANA UNIVERSITY - BLOOMINGTON*

Abstract. An easily and efficiently parallelizable direct method is given for solving a block linear system $Bx = y$, where $B = D + Q$ is the sum of a non-singular block diagonal matrix D and a matrix Q with low-rank blocks. This implicitly defines a new preconditioning method with an operation count close to the cost of calculating a matrix-vector product Qw for some w, plus at most twice the cost of calculating $D^{-1}w$ for some w. When implemented on a parallel machine the processor utilization can be as good as that of those operations. Order estimates are given for the general case, and an an implementation is compared to block SSOR preconditioning.

1. Introduction. Solving a large linear system of equations

$$Ax = y \tag{1}$$

is a common subproblem encountered in the numerical solution of differential and integral equations. Such linear systems are usually solved with preconditioned conjugate gradient (PCG) like methods, which involve solving an auxiliary system

$$Cx = y, \tag{2}$$

with a preconditioner C on each iteration.

This paper examines parallel preconditioners of the form

$$C_1 = (D + Q_L)D^{-1}(D + Q_U), \tag{3}$$

or

$$C_2 = D + Q. \tag{4}$$

Here D is a non-singular block diagonal matrix; Q_L and Q_U are block strictly lower and upper triangular matrices composed of low-rank blocks; and Q has zero diagonal blocks and low-rank off-diagonal blocks (ODBs).

Both preconditioners require solving one or two block linear systems of the form

$$Bx = y, \tag{5}$$

where the non-singular $n \times n$ matrix B has the splitting $B = D+Q$ with a non-singular block-diagonal matrix $D = \mathrm{diag}\{D_{11}, D_{22}, \ldots, D_{pp}\}$ and some block matrix Q. [1]

Block diagonal preconditioning (C_2 with $Q = 0$) is typically used in parallel computations. If diagonal blocks are assigned to processors, the preconditioner can be applied without interprocessor synchronization or communication. Unfortunately, block

* WORK SUPPORTED BY NSF GRANTS CDA-9303189, ASC-9502292 AND ROME LABS AF 30602-92-C-0135

[1] In fact, the preconditioner C_1 (3) can be more efficiently dealt with as a product of two matrices ($C_1 = B_1B_2$, with $B_1 = (D + Q_L)D^{-1}$ and $B_2 = D + Q_U$), only one of which (B_2) is of the form $D + Q$. But the system with the matrix B_1 (which has the form $(D + Q)D^{-1}$, rather than $D + Q$) can be solved in almost the same way as (5); one only needs to remove step 5 from the SMW solve algorithm (Sec. 5, p. 5).

diagonal preconditioned systems often fail to converge, especially for non-symmetric A. Block symmetric Gauss-Seidel (C_1 with Q_L and Q_U the corresponding parts of A) is significantly more robust, but the block lower and upper triangular system solves are inherently sequential. One way to recover parallelism is to increase the number of blocks in the partitioning of A and use a level scheduling [1, 2]. however, this also causes the quality of preconditioning to fall, and in the limiting case becomes pointwise symmetrized Gauss-Seidel, which typically provides poor quality of preconditioning.

In this paper we borrow an idea from the solution of integral equations, and use low-rank approximations (LRA) of the off-diagonal blocks. In [3] the characteristics of such an approximation are analyzed; this paper shows how such a preconditioner can be implemented with parallelism almost as good as that of block diagonal preconditioning. By varying the rank of the ODBs the new method can be parametrized to vary in quality between that of block diagonal and block SSOR [9] preconditioner.

We assume that the n variables are divided into p blocks (so that B and other matrices of the same dimension consist of $p \times p$ blocks). A typical source of the blocking would be from domain decomposition, with $p \ll n$.

Let $r_{kl} = \operatorname{rank} Q_{kl}$, let $M = \sum_{k,l} r_{kl}$ be the sum of ranks of all blocks of Q, let $m_l = \sum_k r_{kl}$ be the sum of ranks of the blocks of Q in the l-th block column, and let m be the maximum of the sum of the ranks of the blocks in any one block column or block row of Q. If no block of Q has rank higher than one, then M becomes the number of non-zero blocks in Q, and m the maximum number of non-zero blocks in any one row or column of Q. Let $\mathcal{P}$ be the set of all the pairs (k, l) such that $Q_{kl} \neq 0$.

Each block of Q can be represented as

$$Q_{kl} = \sum_{i=1}^{r_{kl}} u_{kl}^i {v_{kl}^i}^{\mathrm{T}} = U_{kl} V_{kl}^T, \tag{6}$$

with $U_{kl} = [\, u_{kl}^1 \quad u_{kl}^2 \quad \cdots \quad u_{kl}^{r_{kl}} \,]$ and $V_{kl} = [\, v_{kl}^1 \quad v_{kl}^2 \quad \cdots \quad v_{kl}^{r_{kl}} \,]$.

Since (5) is used in iterative solving of (1), we assume that problem (5) needs to be solved for many right-hand sides y, which are not available at the same time. Blocks of D and Q are constructed and represented to allow: (1) solving systems with the diagonal blocks D_{kk}, and (2) performing matrix-vectors multiplications with the blocks Q_{kl}. E.g. each block of D is represented as a product of a lower and an upper triangular matrices obtained by incomplete LU factorization of the diagonal blocks of A.

The goal is a direct preconditioner whose operation count is comparable to that of any method that uses the two above-mentioned operations (such as, e.g., the usual block elimination method, in the case when B is block-triangular), but which can be efficiently parallelized.

2. Data representation and distribution. The data are distributed among the processors with

- D_{kk} stored on processor k;
- U_{kl} (composed of the vectors $\{u_{kl}^s\}_{s=1,\ldots,r_{kl}}$) stored on processor k;
- V_{kl} (composed of the vectors $\{v_{kl}^s\}_{s=1,\ldots,r_{kl}}$) is stored on processor l.

Here processor k may be a virtual processor, and if fewer than p processors are available, several virtual processors are mapped to one physical processor.

3. Representation of B^{-1}. From (6), $Q = UV^{\mathrm{T}}$, with M-column matrices U and V:

$$(7)\quad U = \begin{bmatrix} U_{11} & 0 & \cdots & 0 & U_{12} & 0 & \cdots & 0 & \cdots & U_{1p} & 0 & \cdots & 0 \\ 0 & U_{21} & \cdots & 0 & 0 & U_{22} & \cdots & 0 & \cdots & 0 & U_{2p} & \cdots & 0 \\ \vdots & & \ddots & \vdots & \vdots & & \ddots & \vdots & \cdots & \vdots & & \ddots & \vdots \\ 0 & & 0 & U_{p1} & 0 & & 0 & U_{p2} & \cdots & 0 & & 0 & U_{pp} \end{bmatrix}$$

$$(8)\quad V = \begin{bmatrix} V_{11} & V_{21} & \cdots & V_{p1} & 0 & 0 & \cdots & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & V_{12} & V_{22} & \cdots & V_{p2} & \cdots & \vdots & 0 & & \vdots \\ \vdots & & & & \vdots & & & & \cdots & 0 & & & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & \cdots & V_{1p} & V_{2p} & \cdots & V_{pp} \end{bmatrix}$$

The block column ordering shown above is chosen so that G will have the same block structure as Q, see Section 4. Block columns in (7) and (8) corresponding to zero blocks Q_{kl} are absent (have no columns).

Since $B = D + UV^{\mathrm{T}}$, the Sherman–Morrison–Woodbury formula [8] gives

$$(9)\qquad B^{-1} = (D + UV^{\mathrm{T}})^{-1} = D^{-1} - D^{-1}U(I + G)^{-1}V^{\mathrm{T}}D^{-1},$$

with $G = V^{\mathrm{T}}D^{-1}U$ of order M.

Since the non-zero eigenvalues of G are the same as those of $D^{-1}UV^{\mathrm{T}}$,

$$\sigma(I + G) \setminus \{1\} = \sigma(D^{-1}B) \setminus \{1\},$$

where $\sigma(H)$ is the set of eigenvalues of the square matrix H. Since B is non-singular, so is $I+G$. Furthermore, if $M < n$ (i.e. if the ranks of the blocks of Q are low enough), then $\operatorname{rank}(UV^{\mathrm{T}}) \le M < n$, and $1 \in \sigma(D^{-1}B)$. Therefore $\sigma(I + G) \subset \sigma(D^{-1}B)$, and $\operatorname{cond}(I + G) \le \operatorname{cond}(D^{-1}B)$.

4. Structure of G. The following proposition shows that the block sparsity pattern of G is contained in that of Q:

PROPOSITION 4.1. *If $Q_{kl} = 0$, then $G_{kl} = 0$.*

Proof. Let G can be partitioned as

$$G = \begin{bmatrix} G_{11} & \cdots & G_{1p} \\ G_{21} & \cdots & G_{2p} \\ \vdots & & \vdots \\ G_{p1} & \cdots & G_{pp} \end{bmatrix},$$

where an $m_k \times m_l$ block G_{kl} has the structure

$$G_{kl} = \begin{bmatrix} 0 & \cdots & 0 & V_{1k}^{\mathrm{T}} & 0 & \cdots & 0 \\ 0 & \cdots & 0 & V_{2k}^{\mathrm{T}} & 0 & \cdots & 0 \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & \cdots & 0 & V_{pk}^{\mathrm{T}} & 0 & \cdots & 0 \end{bmatrix} \cdot \begin{bmatrix} D_{11}^{-1}U_{11} & 0 & \cdots & 0 \\ 0 & D_{22}^{-1}U_{21} & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & D_{pp}^{-1}U_{p1} \end{bmatrix}$$

$$(10)\qquad = \begin{bmatrix} 0 & \cdots & 0 & H_{1kl} & 0 & \cdots & 0 \\ 0 & \cdots & 0 & H_{2kl} & 0 & \cdots & 0 \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & \cdots & 0 & H_{pkl} & 0 & \cdots & 0 \end{bmatrix},$$

with zeros in the first $k-1$ block columns, and the last $p-k$ block columns, and $H_{jkl} = V_{jk}^{\mathrm{T}} D_{kk}^{-1} U_{kl}$ of size $r_{jk} \times r_{kl}$. The equation for H_{jkl} shows that the block G_{kl} can be non-zero only if U_{kl} is non-zero, which, in its turn, can be non-zero only if Q_{kl} is non-zero. □

If we use *no-cancellation assumption* [6, Section 2.2.2] (i.e., disregard the possibility that a potentially non-zero element becomes zero simply because a given dot product happens to be zero), it follows from Proposition 4.1 that G has the same non-zero block structure as Q.

COROLLARY 4.2. *If Q is strictly block upper or lower triangular, then so is G.*

Since H_{jkl} is $r_{jk} \times r_{kl}$, the total number of non-zero elements in G is $\mathrm{nnz}(G) = \sum_k \left((\sum_j r_{jk})(\sum_l r_{kl}) \right) = \sum_{kl} (r_{kl} m_k) \leq mM$.

If no block of Q has rank higher than one, then $mM \leq p^3$. If Q is block-tridiagonal, with all non-zero blocks of rank one, then $mM < 3p$.

5. Outline of the solution method. The parallel solution method for the solving $Bx = y$ is based on (9), and will be referred to later as LRA SMW.

Computing the preconditioner.
begin
1. **foreach** $k = 1 : p$ *(in parallel)*
Preprocess D_{kk} for solving $D_{kk} x_k = y_k$ (if necessary)
endfor
2. **foreach** $k = 1 : p$ *(in parallel)* ** Compute entries of G **
for $l = 1 : p$ *such that* $(k,l) \in \mathcal{P}$
2a. *Solve* $D_{kk} \tilde{U}_{kl} = U_{kl}$
for $j = 1 : p$ *such that* $(j,k) \in \mathcal{P}$
2b. *Set* $H_{jkl} = V_{jk}^{\mathrm{T}} \tilde{U}_{kl}$
endfor
endfor
endfor
2c. If desired, gather all block rows of G on a single processor
3. LU-factor $I + G$
end

Computing G consists of two steps. On step 2a, processor k computes $\tilde{U}_{kl}$ for all l's by solving the linear system

$$D_{kk} [\tilde{U}_{k1} \quad \tilde{U}_{k2} \quad \cdots \quad \tilde{U}_{kp}] = [U_{k1} \quad U_{k2} \quad \cdots \quad U_{kp}]$$

with $\sum_l r_{kl}$ right-hand sides. On step 2b, processor k computes the blocks G_{kl} for all values l, i.e. the blocks H_{jkl} in the matrix

$$\begin{bmatrix} H_{1k1} & \cdots & H_{1kp} \\ H_{2k1} & \cdots & H_{2kp} \\ \vdots & & \vdots \\ H_{pk1} & \cdots & H_{pkp} \end{bmatrix} = \begin{bmatrix} V_{1k}^{\mathrm{T}} \\ V_{2k}^{\mathrm{T}} \\ \vdots \\ V_{pk}^{\mathrm{T}} \end{bmatrix} \cdot [\tilde{U}_{k1} \quad \tilde{U}_{k2} \quad \cdots \quad \tilde{U}_{kp}]$$

As a result of this step, G is distributed among the processors by block row. A gather step is needed if one wants G to be stored on a single processor.

Unless G is triangular, some LU-factorization of $I + G$ is needed (step 3). If G is stored on a single processor, factoring is done by that processor; otherwise, it is done in parallel.

Applying the preconditioner. Applying the preconditioner (4) or (3) involves solving one or two systems of the form (5), using formula (9):

begin Solving $Bx = y$

1. **foreach** $k = 1 : p$ *(in parallel)* /* $z = D^{-1}y$ */
  Solve $D_{kk}z_k = y_k$
endfor

2. **foreach** $l = 1 : p$ *(in parallel)* /* $t = V^{\mathrm{T}}z$ */
  for $k = 1 : p$
    Set $t_{kl} = V_{kl}^{\mathrm{T}} z_l$
  endfor
endfor

3. Solve $(I + G)s = t$ /* $s = (I + G)^{-1}\, t$ */

4. **foreach** $k = 1 : p$ *(in parallel)* /* $y' = y - Us$ */
  Set $y'_k = y_k - \sum_l U_{kl}s_{kl}$
endfor

5. **foreach** $k = 1 : p$ *(in parallel)* /* $x = D^{-1}y'$ */
  Solve $D_{kk}x_k = y'_k$
endfor

end

Step 3 solves

$$(I + G)s = t, \tag{11}$$

where s is partitioned conformally with t.

Only step 3 involves interprocessor communications. On step 2, the j-th processor produces all components of t in its block column (t_{kj} for all k); on step 4, the j-th processor needs all components of s in its block row (s_{jl} for all l). Depending on how (11) is solved, step 3 can be implemented in different ways:

1. Store the entire factors of $I + G$ on a single processor, which solves the system (11). First, t is gathered on this processor; after solving, components of s are scattered to the processors that will need them at the next step. Both steps 2 and 4 send M floating-point values between the processors.
   What processor should be used for solving (11)? Normally, it can be one of the processors $1, \ldots, p$ that perform all other operations. However, if minimizing the memory allocation per processor is a priority, and there are extra processors available, it may be expedient to use $p + 1$ processors, the extra $(p + 1)$-st processor being used to store the factors of $I + G$ and solve (11).
2. Have each processor, redundantly, solve (11). (This option was used in the test.)
3. Use a parallel method for solving (11), distributing the factors of $I + G$ among all processors. At most pM floating-point numbers are sent between processors. Factors of $I + G$ can be distributed among the p processors by block row. If this scheme results in poor data balance (as in the case of a block-triangular or block-tridiagonal Q), linear speed-up can be achieved by distributing the factors of $I + G$ cyclically by row.

The preconditioner can be applied with one D-solve per iteration instead of two, by replacing steps 4 and 5 with

4′. **foreach** $k = 1 : p$ *(in parallel)* /* $x = z - \tilde{U}\, s$ */
  Set $x_k = z_k - \sum_l \tilde{U}_{kl}s_{kl}$

endfor

where $\tilde{U} = D^{-1}U$ (see step 2a of the preconditioner-preparing algorithm). The disadvantage is that the DAXPY operation in $4'$ is done with dense vectors, while that in 5 uses sparse ones. In the remainder of the article, we use 4 and 5 instead of $4'$.

6. Operation count analysis. Solving (5) requires the following number of operations:

$$C_B = C_D + C_V + C_G + C_U + C_D = 2C_D + C_m + C_G.$$

The five terms of the first sum correspond to the five steps of the algorithm. C_D is for solving $Dz = w$, C_V is for one local dot product for each column of the blocks V_{kl}, C_U is for one DAXPY with each column of the blocks U_{kl}'s, and C_G is for solving (11). The sum $C_m = C_V + C_U$ is the number of operations needed to perform a matrix-vector multiplication $Qw = UV^{\mathrm{T}}w$.

7. Timing models. During solving $Bx = y$, only solving system (11) requires interprocessor communications; all other operations are parallel. From the operation count data in Section 6, the time t_B the method needs to solve (5) is approximately the following:

$$t_B = 2t_D + t_V + t_U + t_G = 2t_D + t_m + t_G. \tag{12}$$

The three main terms here, t_D, t_V, and t_U, are times this parallel computer will take, respectively

t_D: to solve $Dz = w$,

t_V: to calculate a dot product with each column of each block V_{kl},

t_U: to perform a DAXPY operation with each column of each block U_{kl},

with the matrices distributed as described in Section 2. The sum $t_m = t_V + t_U$ is the time needed to calculate a matrix-vector product $Qw = UV^{\mathrm{T}}w$.

Cost of solving (11). The term t_G is the time needed to solve (11). When Q is strictly block triangular, so is G, and $t_G = O(Mm)$.

When the matrix G is not triangular, t_G it still does not depend on the right-hand side y, and so $I + G$ needs to be LU-factored once for all future B-solves. The sequential solve time t_G^{seq} is $O(M^2)$; parallel solve time $t_G^{\mathrm{par}} = O(mM)$. This t_G can be reduced if the block structure of Q is such such that fill-in is limited in some way. E.g., for block-tridiagonal Q, t_G^{seq} is $O(mM)$ and $t_G^{\mathrm{par}} = O(M \max\{1, m/p\})$.

Costs of preparing G. Computing G involves solving M linear systems of the form $D_{jj}\tilde{u}_{ji} = u_{ji}$, and then calculating no more than Mm dot products of the form $v_{kj}^{\mathrm{T}}\tilde{u}_{ji}$. The total number of operations involved is under $m(C_D + C_U) \approx \frac{m}{2}C_B$. Since both D-solves and dot products can be done in parallel,

$$t_{\mathrm{prepG}} \approx \frac{m}{2}t_B. \tag{13}$$

The cost of LU-factoring the matrix $I+G$ depends greatly on the structure of this matrix (and thus on the structure of Q). For a general Q, the operation count for this factoring is, is the worst case, $C_{\mathrm{fact(I+G)}} = \mathcal{O}(M^3)$, which implies $t^{\mathrm{seq}}_{\mathrm{fact(I+G)}} = \mathcal{O}(M^3)$ for sequential factoring, and

$$t^{\mathrm{par}}_{\mathrm{fact(I+G)}} = \mathcal{O}(mM^2) \tag{14}$$

for factoring on p processors in parallel (the p-th processor calculates the p-th block row of the factors). This result may be better if Q has some special structure making LU-factoring G less expensive than $\mathcal{O}(M^3)$ operations. For example, if Q is block-tridiagonal, $t^{\text{seq}}_{\text{fact(I+G)}} = \mathcal{O}(m^2 M)$ and

$$t^{\text{par}}_{\text{fact(I+G)}} = \mathcal{O}(mM \max 1, m/p). \tag{15}$$

The timing model provides a practical definition of "low-rank" for blocks of Q:

(a) The time spent preparing and factoring $I+G$ must be small compared to the iterative solve time for (1), and

(b) t_G must be small relative to $t_D + t_U$, so that $t_B \approx 2t_D + t_m$.

Below we will show what restrictions this conditions impose on m, for a general matrix Q, as well as for the special cases of block-triangular and block-tridiagonal Q. We suppose that:

1. p processors are used.
2. The maximum dimension of a block D_{kk} is $\mathcal{O}(n/p)$.
3. $m = \mathcal{O}(M/p)$, i.e. the maximum sum of block ranks in a block row of Q is of the same order as the average sum.
4. Factoring $I+G$ and solving (11) is done in parallel on p processors. (Estimates for sequential operations can be obtained in a similar way.)
5. t_D is the dominant term in $t_D + t_m$, i.e. $t_D + t_m = \mathcal{O}(t_D)$. This is usually the case with many matrices; and if off-diagonal block computations are expensive relative to the diagonal block computations, i.e. if $t_m > pt_D$, then parallelizing the solution of the system (5) is easy.

The time t_D spent solving the system $Dz = w$ depends greatly on the structure of D. The two obvious limiting cases are

- $t_D = \Theta(n/p)$ (D_{kk} is a diagonal matrix, or a product of LU-factors with few non-zeros per row);
- $t_D = \Theta(n^2/p^2)$ (D_{kk} is a product of dense L- and U-factors).

Estimates below will be made for a Q of arbitrary block structure, as well as for block-triangular and block-tridiagonal Q.

Equation (13) indicates that t_{prepG} is on the order of the time taken by m iterations. Thus the criterion (a) implies that m needs to be small in comparison to the total number of iteration.

In order for the term $t_{\text{fact(I+G)}}$ not to make this ratio much worse, it needs to be of the same order as t_{prepG}, i.e. the condition

$$t_{\text{fact(I+G)}} = \mathcal{O}(mt_D) \tag{16}$$

should hold. For a block-triangular Q, this is not an issue (no factoring is needed); for a general Q, the parallel LU-factoring time estimate (14) is used to rewrite the condition (16) as

$$m = \begin{cases} \mathcal{O}\left(\frac{n^{1/2}}{p^{3/2}}\right), & \text{if } t_D = \Theta\left(n/p\right), \\ \mathcal{O}\left(\frac{n}{p^2}\right), & \text{if } t_D = \Theta\left(n^2/p^2\right) \end{cases} \tag{17}$$

For a block-tridiagonal Q, the parallel LU-factoring time estimate (15) gives

$$m = \begin{cases} \mathcal{O}\left(\max\{\frac{n}{p^2}, \frac{n^{1/2}}{p^{1/2}}\}\right), & \text{if } t_D = \Theta\left(n/p\right), \\ \mathcal{O}\left(\max\{\frac{n^2}{p^3}, \frac{n}{p}\}\right), & \text{if } t_D = \Theta\left(n^2/p^2\right). \end{cases} \tag{18}$$

If Q is ...	To ensure that...	If $t_D = \Theta(n/p)$	If $t_D = \Theta(n^2/p^2)$
arbitrary	$t_{\text{fact(I+G)}} = \mathcal{O}(mt_D)$	$m = \mathcal{O}\left(\frac{n^{1/2}}{p^{3/2}}\right)$	$m = \mathcal{O}\left(\frac{n}{p^2}\right)$
	$t_G = o(t_D + t_m)$	$m = o\left(\frac{n^{1/2}}{p}\right)$	$m = o\left(\frac{n}{p^{3/2}}\right)$
blk-tridiag.	$t_{\text{fact(I+G)}} = \mathcal{O}(mt_D)$	$m = \mathcal{O}\left(\max\{\frac{n}{p^2}, \frac{n^{1/2}}{p^{1/2}}\}\right)$	$m = \mathcal{O}\left(\max\{\frac{n^2}{p^3}, \frac{n}{p}\}\right)$
	$t_G = o(t_D + t_m)$	$m = o\left(\max\{\frac{n}{p^2}, \frac{n^{1/2}}{p^{1/2}}\}\right)$	$m = o\left(\max\{\frac{n^2}{p^3}, \frac{n}{p}\}\right)$
blk-triang.	$t_G = o(t_D + t_m)$	$m = o\left(\max\{\frac{n}{p^2}, \frac{n^{1/2}}{p^{1/2}}\}\right)$	$m = o\left(\max\{\frac{n^2}{p^3}, \frac{n}{p}\}\right)$

TABLE 1

Low-rank criteria (in the case of parallel solving of $(I + G)s = t$).

If Q is ...	To ensure that...	If $t_D = \Theta(n/p)$	If $t_D = \Theta(n^2/p^2)$
arbitrary	$t_{\text{fact(I+G)}} = \mathcal{O}(mt_D)$	$m = \mathcal{O}\left(\frac{n^{1/2}}{p^2}\right)$	$m = \mathcal{O}\left(\frac{n}{p^{5/2}}\right)$
	$t_G = o(t_D + t_m)$	$m = o\left(\frac{n^{1/2}}{p^{3/2}}\right)$	$m = o\left(\frac{n}{p^2}\right)$
block-tridiagonal	$t_{\text{fact(I+G)}} = \mathcal{O}(mt_D)$	$m = \mathcal{O}\left(\frac{n^{1/2}}{p}\right)$	$m = \mathcal{O}\left(\frac{n}{p^{3/2}}\right)$
	$t_G = o(t_D + t_m)$	$m = o\left(\frac{n^{1/2}}{p}\right)$	$m = o\left(\frac{n}{p^{3/2}}\right)$
block-triangular	$t_G = o(t_D + t_m)$	$m = o\left(\frac{n^{1/2}}{p}\right)$	$m = o\left(\frac{n}{p^{3/2}}\right)$

TABLE 2

Low-rank criteria (in the case of sequential solving of $(I + G)s = t$).

Restrictions on m that are necessary to satisfy criterion (b) can be derived from the estimates of t_G given earlier in this section.

These results for parallel factoring of $I + G$ and parallel $(I + G)$-solve are summarized in Table 1. Table 2 presents analogously obtained results for sequential LU-factoring and $(I + G)$-solve.

If the symbol o in the restrictions on m is replaced with $\mathcal{O}$, then instead of $t_B \approx 2t_D + t_m$ we will have $t_B = \mathcal{O}(t_D + t_m)$.

8. Preliminary computational results. The LRA SMW preconditioner with conjugate gradients stabilized [4] iteration was implemented in pC++ [5] on an SGI Power Challenge. Only sequential solving of $(I + G)s = t$ was implemented in this version.

Two test problems with $p = 8$, coming from a 2D finite element problem in CFD, were solved using a number of preconditioners:

1. BSSOR with low-rank ODB ($C = (Q_L + D)D^{-1}(Q_U + D)$; blocks of the strict lower and upper triangular Q_L and Q_U are obtained by a lumping-based low-rank approximation method [3] of the respective blocks of A, with $\operatorname{rank} Q_{kl} \leq 3$);
2. SMW with low-rank ODB ($C = D + (Q_L + Q_U)$, with the same Q_L and Q_U as above), SMW method used;
3. BSSOR with original ODB ($C = (L + D)D^{-1}(U + D)$; strict lower and upper triangular L and U are composed from the ODBs of A);
4. SMW with full-rank ODBs ($C = D + (L + U)$ with the same L and U as above).

In all preconditioners, blocks of D were obtained by an incomplete LU-factorization [7] of the diagonal blocks of A with 3 levels of fill allowed.

In Tables 3 and 4, the first number in each cell is the total time spent to solve the problem on the specified number of processors with the specified preconditioner; it is followed by the preconditioner preparation time (LU-factoring D, generating Q, generating and factoring $I + G$) plus the number of iterations times the time per iteration. All times are in seconds.

Table 5 presents partial breakdown of SMW iteration time. For four iterative processes, it gives the total time spent during the iterations, as well as the amount of time that was spent solving $Dz = w$ and solving $(I + G)s = t$. (The rest of the time was spent mainly doing matrix-vector multiplications).

Data in Tables 3 and 4 show that in the low rank case, LRA SMW has a parallel efficiency of 0.91 or better, compared to 0.19 of block SSOR. For higher ranks, LRA SMW only slightly increases the parallel efficiency from 0.20 to 0.22 or 0.26; the main reason for this is that, as Table 5 shows, while D-solve time decreases as the inverse of the number of processors, the $(I + G)$-solve time stays constant. Thus, when M is large, parallel $(I + G)$-solve needs to be implemented.

Further note that fewer iterations are required for LRA SMW, indicating that it provides a better quality preconditioner. These results confirm that LRA SMW can provide parallel efficiency while maintaining and even improving the quality of preconditioning that block SSOR provides.

REFERENCES

[1] F. ALVARADO, *Ordering schemes for partitioned sparse inverses*, 1989. SIAM Symposium on Sparse Matrices, Salishan Lodge, Gleneden Beach, OR, May 22–24 1989.

[2] E. ANDERSON, *Parallel implementation of preconditioned conjugate gradient methods for solving sparse systems of linear equations*, Master's thesis, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., 1988.

[3] R. BRAMLEY AND V.MEŇKOV, *Low rank off-diagonal block preconditioners for solving sparse linear systems on parallel computers*, Tech. Rep. 446, Department of Computer Science, Indiana University, Bloomington, IN, 1996. (To be published).

[4] H. V. DER VORST, *Bi–CGSTAB: A fast and smoothly converging variant of Bi–CG for the solution of nonsymmetric linear systems*, SIAM Journal of Scientific and Statistical Computing, 13 (1992), pp. 631–644.

[5] D. GANNON, S. X. YANG, AND P. BECKMAN, *User Guide for a Portable Parallel C++ Programming System, pC++*, Department of Computer Science and CICA, Indiana University, Bloomington, Indiana 47405, U.S.A., 1994. Available via World Wide Web at http://www.extreme.indiana.edu/sage/pcxx_ug/pcxx_ug.html.

[6] A. GEORGE AND W.-H. LIU, *The Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, Engelwood Cliffs, NJ, 1981.

[7] J. MEIJERINK AND H. VAN DER VORST, *An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix*, Math. Comp., 31 (1977), pp. 148–162.

[8] J. M. ORTEGA AND W. C. RHEINBOLDT, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, 1970.

[9] A. YEREMIN AND L. KOLOTILINA, *On a family of two-level preconditionings of the incomplete block factorization type*, Sov. J. Numer. Anal. Math. Modeling, 1 (1986), pp. 293–320.

Num. proc.	SSOR, low-r. ODB (M=84)	LRA SMW, low-r. ODB (M=84)	SSOR, orig. ODB	LRA SMW, full-r. ODB (M=1234)
1	$95.3 = 8.6 + +67 \cdot 1.29$	$78.8 = 10.8 + +52 \cdot 1.31$	$40.9 = 7.9 + +23 \cdot 1.44$	$78.3 = 64.7 + +6 \cdot 2.26$
2	$67.5 = 4.7 + +60 \cdot 1.05$	$39.1 = 5.9 + +53 \cdot 0.63$	$28.7 = 4.1 + +23 \cdot 1.07$	$55.8 = 46.3 + +6 \cdot 1.59$
4	$64.2 = 3.6 + +66 \cdot 0.92$	$20.6 = 4.1 + +51 \cdot 0.32$	$24.1 = 3.0 + +22 \cdot 0.96$	$47.1 = 39.6 + +6 \cdot 1.24$
8	$60.8 = 2.7 + +66 \cdot 0.88$	$12.4 = 3.2 + +51 \cdot 0.18$	$21.9 = 2.0 + +22 \cdot 0.90$	$42.2 = 35.7 + +6 \cdot 1.09$

TABLE 3

Timing results for Problem `Str389`. $n = 9275$.

Num. proc.	SSOR, low-r. ODB (M=66)	LRA SMW, low-r. ODB (M=66)	SSOR, orig. ODB	LRA SMW, full-r. ODB (M=2214)
1	$230.2 = 18.9 + +72 \cdot 2.93$	$223.7 = 22.6 + +68 \cdot 2.96$	$141.4 = 16.1 + +39 \cdot 3.21$	$453.1 = 349.0 + +19 \cdot 5.48$
2	$181.9 = 10.3 + +70 \cdot 2.45$	$112.4 = 12.1 + +68 \cdot 1.47$	$105.4 = 8.2 + +39 \cdot 2.49$	$357.7 = 282.5 + +19 \cdot 3.95$
4	$167.7 = 6.2 + +77 \cdot 2.10$	$54.5 = 7.5 + +68 \cdot 0.69$	$95.4 = 4.2 + +43 \cdot 2.12$	$315.2 = 251.1 + +19 \cdot 3.38$
8	$156.1 = 4.2 + +77 \cdot 1.97$	$28.3 = 5.0 + +68 \cdot 0.34$	$85.9 = 2.2 + +43 \cdot 1.95$	$359.6 = 301.3 + +19 \cdot 3.07$

TABLE 4

Timing results for Problem `20284.mlp`, $n = 20284$.

Num proc	$n = 9275$ $M = 84$			$n = 9275$ $M = 1234$			$n = 20284$ $M = 66$			$n = 20284$ $M = 2214$		
	D	I+G	Tot	D	I+G	Tot	D	I+G	Tot	D	I+G	Tot
1	47.7	0.21	68.0	6.1	4.83	13.5	142.3	0.18	201.1	40.6	46.49	104.1
2	22.5	0.18	33.2	3.0	4.82	9.6	68.4	0.18	100.3	19.8	46.05	75.1
4	10.8	0.17	16.5	1.5	4.82	7.4	30.7	0.16	47.0	9.7	46.08	64.1
8	5.0	0.17	9.2	0.7	4.80	6.6	14.3	0.16	23.3	4.5	46.08	58.3

TABLE 5

Breakdown of iteration time. "D" is the total time spent solving $Dz = w$; "I+G" is the total time spent solving $(I + G)s = t$; "Tot" is the total time spent during iterations.

Parallelizable Approximate Solvers for Recursions Arising in Preconditioning

Yair Shapira *

Abstract

For the recursions used in the Modified Incomplete LU (MILU) preconditioner, namely, the incomplete decomposition, forward elimination and back substitution processes, a parallelizable approximate solver is presented. The present analysis shows that the solutions of the recursions depend only weakly on their initial conditions and may be interpreted to indicate that the inexact solution is close, in some sense, to the exact one. The method is based on a domain decomposition approach, suitable for parallel implementations with message passing architectures. It requires a fixed number of communication steps per preconditioned iteration, independently of the number of subdomains or the size of the problem. The overlapping subdomains are either cubes (suitable for mesh-connected arrays of processors) or constructed by the data-flow rule of the recursions (suitable for line-connected arrays with possibly SIMD or vector processors). Numerical examples show that, in both cases, the overhead in the number of iterations required for convergence of the preconditioned iteration is small relatively to the speed-up gained.

1 Introduction and Main Results

The Incomplete LU (ILU) decomposition of a sparse matrix [14] [15] [21] is considered as one of the most powerful and robust methods for the solution of sparse linear systems of equations. The idea is to construct sparse triangular matrices L and U such that LU approximates the coefficient matrix in some sense. Then LU serves as a preconditioner in a Krylov-space acceleration method. The ILU decomposition process, as well as the back substitution and forward elimination in the triangular matrices L and U, is basically sequential, and parallelizable and vectorizable algorithms for solving it are of great interest. In the following, we list some of the well known approaches. A vectorization based on the "data flow" of the recursions is presented in [3]. In this implementation, the recursion is done data flow front by data flow front, where a data flow front is a set of variables which can be computed simultaneously in a certain step of the recursion. A parallel "pipeline" strategy is proposed in [4]. An inexact vectorizable algorithm is presented in [22]. All these approaches are compared in [23].

A case of special interest is the one for which the coefficient matrix may be considered an operator acting in the space of grid functions defined on a d-dimensional grid. This situation arises

*Computer Science Department, Technion – Israel Institute of Technology, Haifa 32000, Israel, e-mail: yairs@csa.technion.ac.il.

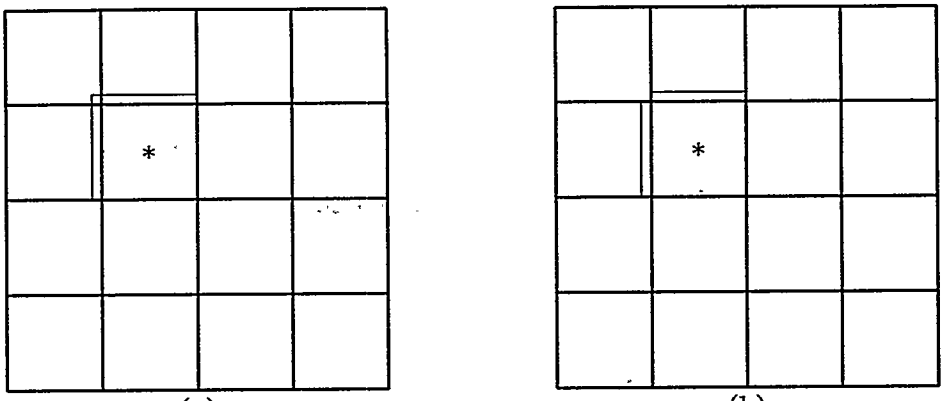
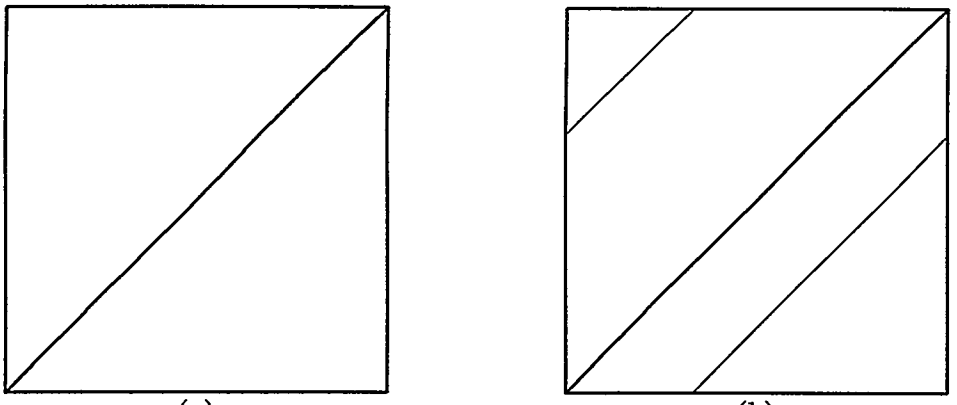


Figure 1: A domain decomposition with overlapping square subdomains. It is assumed that the recursion goes left to right and downward. The truncated recursion for, e.g., the subdomain including the $*$ is initiated on the thin lines.

often in the numerical solution of elliptic PDEs. The domain decomposition approach, based on substructuring the grid into a collection of subgrids, is considered suitable to this case, especially when implemented on message passing parallel architectures with mesh connected processor arrays. In [8], a domain decomposition approach based on an iterative solution (using ILU) of the Schur complement system corresponding to the interface unknowns is presented. In [17], an alternating Swartz algorithm with inexact solvers for the overlapping subdomains is presented. We will also consider "star" schemes, namely, $(2d+1)$-coefficient stencils arising, for example, from central second order finite difference discretizations of second order elliptic PDEs. For such problems, one may order the variables in a red-black ordering and obtain an efficiently parallelizable ILU method. However, with this ordering the convergence is much slower than with the standard (lexicographic) one [3]. A nested dissection ordering may give better convergence rates in the price of a more complicated parallel implementation [5].

Next, we consider some inexact solvers for the recursions used in the ILU preconditioning. In [20] a domain decompositioned inexact solver based on d-dimensional cube subdomains is developed. It is shown there that the solutions of the exact recursions depend only weakly on their initial conditions; a good approximation is thus obtainable when an inexact (truncated) recursion is performed in all the subdomains simultaneously. For a triangular system, this is equivalent to performing one block Jacobi iteration, where a block corresponds to a subdomain. This, of course, requires the transfer of internal boundary conditions from adjacent subdomains. For message-passing parallel architectures with mesh-connected processor arrays, though, this requires only one communication step per approximate recursion. Numerical experiments show that the method converges nicely for standard ILU, but deteriorates for the Modified ILU (MILU) version of [11]. It is also mentioned in [20] that it is possible to use extended subdomains as in Figure 1(a); this, however, requires d communication steps per recursion.



(a) (b)

Figure 2: A domain decomposition with triangular subdomains, suitable for vector computers. The values for variables on the thin lines in Picture (b) are computed by one SIMD or vector operation.

A vectorizable inexact solver for the recursions, which may be also viewed as a domain decompositioned one, is presented in [24]. Its implementation to star schemes is illustrated in Figure 2. The recursion is done in both domains simultaneously. It is vectorizable on each oblique line parallel to the interface between the subdomains. When such a line reaches a boundary, it is continued as if the region was periodic, exploiting the maximal vector length of the machine. As in [20], the method deteriorates when applied to MILU; this is cured by using overlapping (see Figure 3).

In this work we present convergence analysis which shows that, under some algebraic conditions, the solution of the recursion for a certain variable converges when the initial conditions of the recursion are given on a data flow front farer and farer away from the location of this variable. An estimate for the rate of convergence of this limit process is also given. This analysis is related to the theory of continued fractions [19] and may also serve as a justification for the infinite domain analysis used in [7] and [13]. The results are stated in pure algebraic manner, avoiding the use of the grids, orders and norms defined in [20]. In the case of [24], this convergence means that when the overlapping in Figure 3 is large enough, the approximate values in the lower right domain almost reach their exact values. Consequently, the analysis supports the numerical results of [24]. Furthermore, it motivates the definition of a domain decompositioned approximate solver for the recursions, in which each subdomain consists of several consecutive data flow fronts. For star schemes, this results in the oblique domain decomposition of Figure 4. This algorithm is especially suitable to message passing parallel architectures with line connected arrays of vector or SIMD processors. For such architectures, the solution in each subdomain may be optimally vectorized as in [24], and additional speed-up is gained via parallelism. Of course, it is also suitable to shared memory architectures with several vector processors, such as that of the CRAY. Our numerical examples show that the rate of convergence of the algorithm is not far from that of the standard one. Finally, we present an implementation of the method of [20] which is efficient also for MILU and requires only one communication step per approximate recursion (see Figure 1(b)).

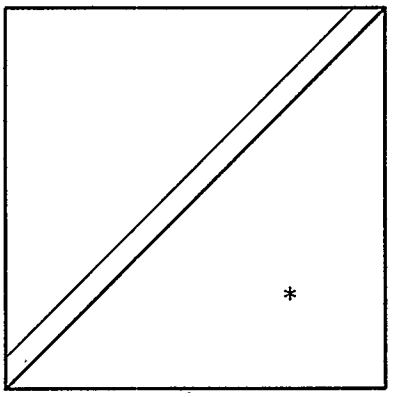


Figure 3: A domain decomposition with overlapping triangular subdomains. The truncated recursion for, e.g., the subdomain including the $*$ is initiated on the thin line.

We do not deal in this paper with the problem of parallelizing the inner product computations required in the acceleration scheme. One may assume that, for this task, some broadcasting or shared memory device is available, which takes care of collecting the pieces of inner product computed over the subdomains.

2 Numerical Examples

Here we present a numerical comparison of the standard and parallelizable versions of MILU. The following 2-d examples are tested: (1) the Poisson equation, (2) an anisotropic diffusion equation, (3) a circulating convection equation, (4) a diffusion equation with discontinuous coefficients and (5) an anisotropic diffusion equation with discontinuous coefficients. The 5-coefficient discretization schemes of [1] and [6] are used on a 128×128 grid. The MILU parameter is 0.95 (except of Example 1, where the optimal parameter of [7] is used). The amount of overlapping is 8 grid points in each spatial direction. The Transpose Free Quasi Minimal Residual method (Algorithm 5.2 in [10]) is used for acceleration. The numbers of iterations required for reducing the l_2 norm of the residual by 6 orders of magnitude are displayed in Table 1.

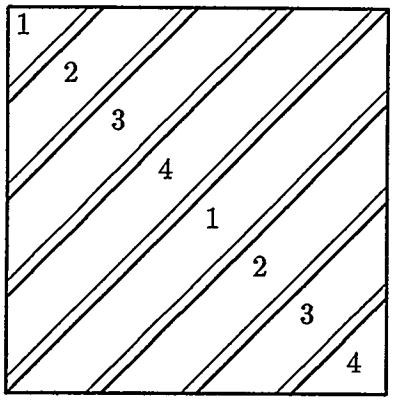


Figure 4: A domain decomposition with oblique subdomains, suitable for a parallel computer consisting of four vector processors. The subdomains denoted by 1, 2, 3 and 4 are assigned to the first, second, third and fourth processor, respectively. The vectorization is done as in Figure 2(b). Overlapping is used; the recursion for a subdomain is initiated on the thin line on its left.

Table 1: A comparison of the standard and parallelizable versions of MILU.

example	standard MILU	Figure 1(a)	Figure 1(b)	Figure 3	Figure 4
1	40	47	53	63	54
2	33	54	53	74	59
3	97	145	152	112	135
4	160	162	155	152	166
5	149	144	166	187	188

3 Conclusions

The present analysis shows that the recursions used in the ILU iteration depend only weakly on their initial values. This motivates the construction of inexact algorithms using truncated recursions. These approximate recursions can be done in various subdomains simultaneously. When overlapping subdomains are used, the method applies also to Modified ILU (MILU). Two types of domain decomposition, suitable to two types of parallel architectures, are introduced. The first one, based on [24], is suitable to line connected arrays of SIMD or vector processors. The other one, based on [20], is suitable to mesh connected processor arrays. The analysis may be interpreted to indicate that these algorithms converge in almost the same rate as does the standard one. This is illustrated numerically for several examples.

References

[1] Alcouffe, R.; Brandt, A.; Dendy, J. E.; and Painter J.: The Multigrid Method for the Diffusion Equation with Strongly Discontinuous Coefficients. *SIAM J. Sci. Stat. Comput.,* vol. 2, 1981, pp. 430-454.

[2] Alvarado, F.L., Schreiber, R., "Optimal Parallel Solution of Sparse Triangular Systems", *SIAM J. Sci. Stat. Comput.* 14 (1993), 446-460.

[3] Aschcraft C., Grimes R., "On Vectorizing Incomplete Factorizations and SSOR Preconditioners", *SIAM J. Sci. Stat. Comput.* 9 (1988), 122-151.

[4] Basttian P., Horton G., "Parallelization of Robust Multigrid Methods: ILU Factorization and Frequency Decomposition Method", *SIAM. J. Sci. Stat. Comput.* 12 (1991), 1457-1470.

[5] Brand, C. W.: An Incomplete Factorization Preconditioning Using Repeated Red Black Ordering. *Numer. Math.* 61 (1992), pp. 433-454.

[6] Brandt, A.; and Yavneh, I.: Accelerated Multigrid Convergence and High Reynolds Recirculating Flows, *SIAM J. Sci. Stat. Comput.* 14, 1993, pp. 607-626.

[7] Chan T.F., "Fourier Analysis of Relaxed Incomplete Factorization Preconditioners", *SIAM J. Sci. Stat. Comput.* 12 (1991), 668-680.

[8] Chan, T.F., and Goovaert, D., "A Note on the Efficiency of Domain Decomposed Incomplete Factorizations". *SIAM J. Sci. Stat. Comput.* 11 (1990), pp. 794-803.

[9] Eisenstat, S. C.: Efficient Implementation of a Class of Preconditioned Conjugate Gradients Methods. *SIAM J. Sci. Stat. Comput.* 2 (1981), pp. 1-4.

[10] Freund R., W., "Transpose Free Quasi-Minimal Residual Algorithm for Non-Hermitian Linear Systems", *SIAM J. Sci. Stat. Comput.* 14 (1993), pp. 470-482.

[11] Gustafsson S., "Modified Incomplete Methods", in Preconditioning Methods, Evans D.J., *ed.*, Gordon and Breach, N.Y., 1983.

[12] Gustafsson S., "On modified incomplete factorization methods", *Lecture Notes in Mathematics* 968, pp. 334-351.

[13] Kettler R., "Analysis and Comparison of Relaxation Schemes in Robust Multigrid and Preconditioned Conjugate Gradients Methods", in *Multigrid Methods*, Hackbusch W. and Trottenberg U. *eds., Lecture Notes in Mathematics* 960, Springer-Verlag, Berlin, Heidelberg, 1982.

[14] Manteuffel T.A., "An Incomplete Factorization Technique for Positive Definite Linear Systems", *Math. Comp.* 34 (1980), pp. 473-497.

[15] Meijerink J.A., Van der Vorst H.A., "An Iterative Solution Method for Linear Systems of which the Coefficients Matrix is a Symmetric M-matrix", *Math. Comp.* 31 (1977), 148-162.

[16] Meijerink J.A., Van der Vorst H.A., "Guidelines for the Usage of Incomplete Decompositions in Solving Sets of Linear Equations as they Occur in Practical Problems", *J. Comp. Phys.* 44 (1981), 134-155.

[17] Radicati, G. and Robert, Y.; "Parallel Conjugate Gradient Like Algorithms for Solving Sparse Nonsymmetric Linear Systems on a Vector Multiprocessor", Parallel Computing 11 (1989), pp. 223-239.

[18] Shapira, Y., "Coloring Update Methods", Technical Report #851, Computer Science Department, Technion – Israel Institute of Technology, Sept. 1995, submitted to *J. Comput. Appl. Math.*.

[19] Shapira, Y., Sidi, A., Israeli, M., "Optimal Error Bounds for Convergents of a Family of Continued Fractions", *J. Math. Anal. Appl.* 98, vol. 1, Feb. 1996.

[20] Shapira, Y., Sidi, A., Israeli, M., "A Parallelizable Incomplete *LU*-type Preconditioner for the Solution of Sparse Linear Systems", Technical Report #799, Computer Science Department, Technion – Israel Institute of Technology, January 1994, submitted to *SIAM Journal on Matrix Analysis and Applications.*

[21] Van der Vorst H.A., "Iterative Solution Methods for Certain Sparse Linear Systems with a Non-symmetric Matrix Arising from PDE-problems", *J. Comp. Phys.* 49 (1982), 1-19.

[22] Van der Vorst H.A., "a Vectorizable Version of some Incomplete Conjugate Gradients Methods", *SIAM J. Sci. Stat. Comput.* 3 (1982), 350-356.

[23] Van der Vorst H.A., "High Performance Preconditioning", *SIAM J. Sci. Stat. Comput.* 10 (1989), 1175-1184.

[24] Washio, T.; and Hayami, K.: Overlapping Multicolor MILU Preconditioning. *SIAM J. Sci. Comput.* 16 (1995), pp. 636-650.

NEW PARALLEL SOR METHOD BY DOMAIN PARTITIONING *

DEXUAN XIE†

Abstract. In this paper, we propose and analyze a new parallel SOR method, the PSOR method, formulated by using domain partitioning together with an interprocessor data-communication technique. For the 5-point approximation to the Poisson equation on a square, we show that the ordering of the PSOR based on the strip partition leads to a consistently ordered matrix, and hence the PSOR and the SOR using the row-wise ordering have the same convergence rate. However, in general, the ordering used in PSOR may not be "consistently ordered". So, there is a need to analyze the convergence of PSOR directly. In this paper, we present a PSOR theory, and show that the PSOR method can have the same asymptotic rate of convergence as the corresponding sequential SOR method for a wide class of linear systems in which the matrix is "consistently ordered". Finally, we demonstrate the parallel performance of the PSOR method on four different message passing multiprocessors (a KSR1, the Intel Delta, an Intel Paragon and an IBM SP2), along with a comparison with the point Red-Black and four-color SOR methods.

Key words. parallel computing, SOR, JSOR, PSOR, convergence analysis

AMS subject classifications. Primary 65Y05; Secondary 65F10.

1. Introduction. The successive over-relaxation (SOR) iterative method is an important solver for large linear systems [21]. It is also a robust smoother in the multigrid method [19]. However, the SOR method is essentially sequential in its original form. With the increasing use of parallel computers, several parallel versions of the SOR method have been studied by a number of authors. Adams and Ortega [1], Adams and Jordan [2], and Adams, Leveque and Young [3] have written about the multicolor SOR method; Block, Frommer and Mayer [5] wrote about a general block multicolor SOR method; and White [18] wrote about the multisplitting SOR method. We also note the papers by Evans [6] and Patel and Jordan [12], which presented two parallel SOR methods for particular parallel computers. Moreover, several different parallel SOR methods which are defined by domain decomposition have been described in Farhat [7], Rice [14], and White [18].

Recently, we analyzed a simple parallel version of the SOR method, the JSOR method, in [15]. The idea of JSOR can be simply described below. Let a grid mesh domain be partitioned into p disjoint subgrids. The JSOR method is obtained by concurrently applying one sweep of the SOR algorithm to each subgrid, using the previous iterates as the "boundary values". Because it is based on a domain decomposition, JSOR can be easily implemented on MIMD computers. Moreover, each JSOR iteration requires interprocessor-date-communication only once. However, since its convergence speed may be slow down almost linearly with respect to the number p, JSOR is rarely used as a parallel solver for linear systems. Instead, JSOR can be a efficient smoother for parallel multigrid methods as shown in [19] and [20].

In this paper we will show that the convergence rate of the JSOR method can be greatly improved by only reordering the positions of the data communication. The resulted algorithm is referred to as the PSOR (parallel SOR) method.

* Submitted to 1996 COPPER MOUNTAIN CONFERENCE ON ITERATIVE METHODS This work was supported in part by the National Science Foundation through award number DMS-9105437.

† Courant Institute of Mathematical Sciences New York University, 251 Mercer Street, New York, NY 100112, (xie@math.uh.edu).

Although PSOR is generated from the JSOR method [15], it can be regarded as a SOR method using a new ordering. For the 5-point stencil model problem, we show that the new ordering based on a strip partition leads to a consistently ordered matrix, and hence the SOR theory in [21] follows that PSOR has the same asymptotic rate of convergence as the SOR using the row-wise ordering. However, in general, the ordering used in PSOR cannot lead to a consistently ordered matrix and so the SOR theory does not apply. Thus, there is a need to study the convergence of PSOR directly. In this paper, we propose a PSOR theory, and prove that the spectral radius of the PSOR iteration matrix can be the same as that of the SOR iteration matrix for a wide class of linear systems in which the matrix is "consistently ordered". ,

Moreover, we demonstrate the parallel performance of the PSOR method on a shared memory MIMD computer (a KSR1) and three distributed memory MIMD computers (the Intel Delta, an Intel Paragon L38 and an IBM POWERparallel System 9076 SP2). The numerical results show that PSOR is very efficient on these distinct multiprocessor machines.

Finally, we present a comparison of the parallel performance between the PSOR method and the multicolor SOR method. We considered the Red/Black SOR method for the 5-point stencil and the four-color SOR method for the 9-point stencil [3]. We let both PSOR and the multicolor SOR have the same date structure as the SOR with a natural ordering. Our numerical results show that PSOR has a much better performance than multicolor SOR on both computation and date-communication.

The remainder of the paper is organized as follows. In Section 2 we consider the PSOR algorithm for the 5-point difference stencil of the Laplacian equation on a square. In Section 3 we give a general description of the PSOR method. Section 4 is devoted to a convergence analysis of the PSOR method. In Section 5 we present the numerical results on four different MIMD computers to show the parallel performance of PSOR. We conclude in Section 6 with a comparison between PSOR and multi-color SOR.

2. The Model Problem.

3. The PSOR Method.

4. PSOR Analysis.

5. Numerical Examples.

6. A Comparison between PSOR and Multi-Color SOR.

REFERENCES

[1] Loyce M. Adams and J. M. Ortega, *A multi-color SOR method for parallel computation*, Proc. 1982 International Conference on Parallel Processing, Bellaire, MI, August 1982, pp. 53–58.

[2] Loyce M. Adams and Harry F. Jordan, *Is SOR color-blind?* SIAM J. Sci. Stat. Comput. 7 (1986), pp. 490–506.

[3] Loyce M. Adams, Randall J. Leveque, and David M. Young, *Analysis of the SOR iteration for the 9-point Laplacian*, SIAM J. Numer. Anal. 25 (1988), pp. 1156–1180.

[4] B. Bagheri, T. W. Clark and L. R. Scott, *A Parallel Dialect of* Fortran, Fortran Forum, Volume 11, 20-31, Sept. 1992.

[5] U. BLOCK, A. FROMMER AND G. MAYER, *Block coloring schemes for the SOR method on local memory parallel computers.* Parallel Computing, 14 (1990), pp. 61-75.
[6] D. J. EVANS: *Parallel SOR Iterative Methods,* Parallel computing, 1 (1984), 3-18.
[7] FARHAT: Thesis in Civil Engineering at Berkeley, 1987.
[8] LOUIS A. HAGEMAN AND DAVID M. YOUNG, *Applied iterative methods,* Academic press, New York, (1981).
[9] RAMI G. MELHEM , *Determination of stripe structures for finite element matrices,* SIAM J. Numer. Anal. 24 (1987), pp. 1419–1433.
[10] W. NIETHAMMER, *The SOR method on parallel computers.* Numer. Math, 56 (1989), pp. 247-254.
[11] J. M. ORTEGA AND R. G. VOIGT, Solution of partial differential equations on vector and parallel computers. SIAM Rev. 27 (1985), 149-270.
[12] N. R. PATEL AND H. F. JORDAN, *A parallelized point row-wise successive over-relaxation method on a multiprocessor,* Parallel Computing, 1 (1984), pp. 207-222.
[13] MICHAEL J. QUINN, *Designing Efficient Algorithms for Parallel Computers,* McGraw-Hill, Inc., (1987).
[14] JOHN RICE: Purdue University, Ellpack, 1987.
[15] L. R. SCOTT AND DEXUAN XIE, *Parallel Linear Stationary Iterative Methods,* to be submitted to SIAM J. Sci. Comput., 1995.
[16] UNIVERSITY OF TENNESSEE, *MPI: A Message-Passing Interface Standard,* Version 1.0, May 5, 1994.
[17] R. VARGA, *Matrix Iterative Analysis,* Prentice Hall, Englewood Cliffs, New Jersey, (1962).
[18] R. E. WHITE: *Multisplittings and Parallel Iterative Methods,* Computer Methods in Applied Mechanics and Engineering, 64 (1987), 567-577.
[19] DEXUAN XIE, *Parallel Multiplicative Smoother and Analysis,* Math. Comp., submitted (1995).
[20] DEXUAN XIE, Ph. D. thesis, University of Houston, (1995).
[21] DAVID M. YOUNG, *Iterative Solution of Large Linear System,* Academic press, New York, (1971).

Topic: *Preconditioning*	*Session Chair:* *Seymour Parter*	*Room C*

4:45 - 5:15	J.D. Moulton	Approximate Schur Complement Preconditioning of the Lowest Order Nodal Discretizations
5:15 - 5:45	K. Chen	On Preconditioning Techniques for Dense Linear Systems Arising from Singular Boundary Integral Equations
5:45 - 6:15	W.L. Wan	Fast Wavelet Based Sparse Approximate Inverse Preconditioner
6:15 - 6:45	E. Meese	Combined Incomplete LU and Strongly Implicit Procedure Preconditioning

Approximate Schur Complement Preconditioning of the Lowest Order Nodal Discretizations

J. David Moulton [1], Jim E. Morel [2] and Uri M. Ascher [3]

Abstract

Particular classes of *nodal* methods and *mixed hybrid* finite element methods lead to equivalent, robust and accurate discretizations of 2^{nd} order elliptic PDEs. However, widespread popularity of these discretizations has been hindered by the awkward linear systems which result. The present work exploits this awkwardness, which provides a natural partitioning of the linear system, by defining two optimal preconditioners based on approximate Schur complements. Central to the optimal performance of these preconditioners is their sparsity structure which is compatible with Dendy's *black box* multigrid code.

1 Introduction

Nodal methods have long been one of the most popular discretization techniques employed within the reactor physics community to solve multi-group diffusion problems [4]. This success is a result of their exceptional accuracy which may be attributed to three distinct aspects of the *nodal ideology*. Specifically, akin to finite volume methods, nodal methods are physically motivated cell-based discretizations that by construction, rigorously enforce cell balance. They utilize an intriguing choice of unknowns which, in two dimensions for example, consists of cell and edge moments, making the nodal discretization naturally compatible with the various homogenization techniques that are crucial to reactor modelling. Moreover, comparable to *mixed finite element* methods, the neutron current is obtained explicitly from the discretization thereby avoiding problematic finite difference approximations. Thus, it is not surprising that most *nodal methods* may be derived using *non-conforming* and *mixed hybrid finite element* formulations [3], hence extending their realm of interest to the entire numerical analysis community.

Since our primary interest is the development of preconditioners which exploit the underlying structure of the resulting sparse linear system, we may avoid the unnecessary complications of multi-group diffusion and restrict our discussion to linear 2^{nd} order elliptic PDEs of the form (ie. conservative one-group diffusion),

$$\nabla \cdot \mathbf{J} = Q(x,y) \tag{1a}$$

$$\mathbf{J} = -D(x,y)\nabla\phi \tag{1b}$$

for $(x,y) \in \Omega$ and subject to the Dirichlet boundary condition,

$$\phi = g(x,y) \quad \forall\,(x,y) \in \partial\Omega \tag{2}$$

The discretizations which follow assume Ω is a rectangular domain and employ a tensor product mesh having cells, $\Omega_{i,j} = \left(x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}\right) \times \left(y_{j-\frac{1}{2}}, y_{j+\frac{1}{2}}\right)$ with $\Omega_h \equiv \bigcup \Omega_{i,j}$ for $i = 1,\ldots,L$ and $j = 1,\ldots,M$. It is convenient to denote the mesh spacing, $\Delta x_i = x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}}$ and $\Delta y_j = y_{j+\frac{1}{2}} - y_{j-\frac{1}{2}}$.

1.1 The Nodal Discretization

Common to all *nodal* discretizations is the choice of cell and edge based unknowns. Generally these are taken to be moments up to some specified order, hence in the lowest order case simple averages are employed. Specifically, the cell based unknowns are averages defined by,

$$\phi_{i,j} = \frac{1}{\Delta x_i \Delta y_j} \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \int_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}} \phi(x,y)\,dx\,dy \tag{3}$$

[1]Department of Mathematics, University of British Columbia, Vancouver, British Columbia, Canada
[2]Scientific Computation, CIC-19, Los Alamos National Laboratory, Los Alamos, New Mexico, USA
[3]Department of Computer Science, University of British Columbia, Vancouver, British Columbia Canada

while the edge based scalar unknowns, namely edge averages of the scalar flux are given by,

$$\phi_{i+\frac{1}{2},j} = \lim_{x \to x_{i+\frac{1}{2}}} \phi_j(x), \qquad \phi_j(x) = \frac{1}{\Delta y_j} \int_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}} \phi(x,y) dy \tag{4}$$

with analogous definitions of $\phi_{i,j+\frac{1}{2}}$ and $\phi_i(y)$. Finally the edge integrated currents are written,

$$J^{\pm}_{i+\frac{1}{2},j} = \lim_{x \to x^{\pm}_{i+\frac{1}{2}}} \int_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}} \left[-D(x,y) \frac{\partial}{\partial x} \{\phi(x,y)\} \right] dy \tag{5}$$

while $J^{\pm}_{i,j+\frac{1}{2}}$ is defined analogously.

Since all lowest order members of the *nodal method* family yield equivalent discretizations of (1) we have chosen to present a brief discussion of the Nodal Integration Method (NIM). The first step in the NIM discretization is the posing of the cell based transverse integrated ODEs which govern $\phi_j(x)$ and $\phi_i(y)$. To this end we assume an homogenized diffusion coefficient, $D_{i,j}$ is defined on each cell and transverse integrate (1) to obtain a set of two ODEs which are coupled through *pseudo–source* terms. The definition of the edge averages, (4) naturally yields Dirichlet boundary conditions for each cell. Moreover, we are deriving the lowest order or *constant–constant* NIM and hence the *pseudo–source* terms (ie. $J^{\pm}_{j\mp\frac{1}{2}}(x)$ and $J^{\pm}_{i\mp\frac{1}{2}}(y)$) are assumed to be constant along their respective cell edge. By further assuming the source $Q(x,y)$ is constant over each cell, we finally obtain a set of two *constant* coefficient ODEs, with a constant *source* and Dirichlet boundary conditions that are readily solved.

With expressions for $\phi_j(x)$ and $\phi_i(y)$ in hand we begin the construction of the discrete relationships which comprise the discretization. Specifically we first note two independent definitions of the cell average (ie. $2LM$ equations) are possible. Furthermore, under the assumption of an homogenized diffusion coefficient the order of integration and differentiation may be reversed in (5) to yield expressions for $J^{\pm}_{i\mp\frac{1}{2},j}$, $J^{\pm}_{i,j\mp\frac{1}{2}}$ on each cell. Although these comprise only $3LM$ independent equations as the *balance* equation arises from both $J^{-}_{i+\frac{1}{2},j} - J^{+}_{i-\frac{1}{2},j}$ and $J^{-}_{i,j+\frac{1}{2}} - J^{+}_{i,j-\frac{1}{2}}$. Imposing continuity of $\mathbf{J} \cdot \mathbf{n}$ yields an equation for all interior edges (ie. $(L-1)M + L(M-1)$ equations) while the boundary conditions give rise to $2L + 2M$ discrete boundary equations. Thus we have $7LM + L + M$ equations in as many unknowns.

Although the *constant–constant* NIM discretization is complete at this point it is seldom used in this form. Specifically, for our work we proceed by eliminating the currents, $J_{i+\frac{1}{2},j}$ and $J_{i,j+\frac{1}{2}}$, followed by the trivial elimination of the cell averages, $\phi_{i,j}$ to obtain the 7-point nearest neighbour hexagonally coupled stencils that govern $\phi_{i+\frac{1}{2},j}$ and $\phi_{i,j+\frac{1}{2}}$. Using a finite difference style analysis it is possible to prove that with constant mesh spacing and constant diffusivity the *constant-constant* NIM discretization is second order convergent in the scalar variable.

1.2 Mixed Hybrid Finite Element Methods

The most common *mixed* finite element methods (FEMs) arise in problems that contain an equality constraint. In the context of (1) the mixed method ideology views (1a) as the differential equation and (1b) as the constraint. This has the distinct advantage of obtaining independent approximations of ϕ and $\mathbf{J}$. However, the resulting system is difficult to solve as it is indefinite. In contrast *hybrid* FEMs are a special class of *mixed* FEMs which temporarily relax certain inter-element continuity conditions, ultimately enforcing them in only the weak variational sense as constraints. Of particular interest to this work is the inter-element continuity of the normal current, $\mathbf{J} \cdot \mathbf{n}$, which may be treated as a constraint. Remarkably *hybridization* of the *mixed* finite element formulation yields an indefinite system, that may be recast as a positive definite system, in which the sparsity structure is largely preserved.

To initiate the discretization process we first define the function spaces for each unknown. In particular, to accommodate the definition of spaces involving edge-based unknowns or constraints we denote the set of

edges E of the rectangles, $\Omega_{i,j}$ as $\mathcal{E}_h$ such that defining $\mathcal{E}_h^\partial = \{E \in \mathcal{E}_h | E \subset \partial\Omega\}$ yields $\mathcal{E}_h^0 = \mathcal{E}_h \backslash \mathcal{E}_h^\partial$. Thus, the lowest order Raviart–Thomas space may be written,

$$RT_0^0(\Omega_h) = \left\{ \mathbf{q} \,|\, \mathbf{q} \in RT_{-1}^0(\Omega_h),\ \int_{E^+} \mathbf{q}\cdot\mathbf{n}\, ds = \int_{E^-} \mathbf{q}\cdot\mathbf{n}\, ds \quad \forall E \in \mathcal{E}_h^0 \right\}$$
$$RT_{-1}^0(\Omega_h) = \left\{ \mathbf{q} \,|\, \mathbf{q} \in L^2(\Omega) \times L^2(\Omega),\ \mathbf{q}|_{\Omega_{i,j}} \in \{\{1,\xi\},\{1,\eta\}\}\, \forall \Omega_{i,j} \in \Omega_h \right\}$$

where $RT_0^0 \subset H(\mathrm{div};\Omega)$ is the conforming subspace in which the solution $\mathbf{J}_h$ will ultimately be found. The space of Lagrange multipliers, $M_{-1}^0(\Omega_h)$ is defined as,

$$M_{-1}^0(\Omega_h) = \{\mu \,|\, \mu \in L^2(\Omega),\ \mu|_E \in \{1\}\ \ \forall E \in \mathcal{E}_h^0;\ \mu|_E = 0\ \forall E \in \mathcal{E}_h^\partial\}$$

while the cell-based unknowns are defined over,

$$U_{-1}^0(\Omega_h) = \left\{ \varphi \,|\, \varphi \in L^2(\Omega),\ v|_{\Omega_{i,j}} \in \{1\}\ \ \forall \Omega_{i,j} \in \Omega_h \right\}$$

Thus the discrete *mixed hybrid* variational formulation of (1) may be expressed as,

find $(\mathbf{J}_h, \phi_h, \mu_h) \in RT_{-1}^0(\Omega_h) \times U_{-1}^0(\Omega_h) \times M_{-1}^0(\mathcal{E}_h^0)$ *such that,*

$$\int_\Omega [D^{-1}]\, \mathbf{J}_h \cdot \mathbf{q}_h dxdy - \sum_{i,j} \left\{ \int_{\Omega_{i,j}} \phi_h \nabla\cdot \mathbf{q}_h dxdy - \int_{\partial\Omega_{i,j}} \mu_h (\mathbf{q}_h\cdot\mathbf{n})\, ds \right\} = -\int_{\partial\Omega} g(\mathbf{q}_h\cdot\mathbf{n})\, ds \qquad \forall \mathbf{q}_h \in RT_{-1}^0(\Omega_h)$$

$$-\sum_{i,j} \int_{\Omega_{i,j}} \varphi_h \nabla\cdot \mathbf{J}_h dxdy = -\int_\Omega Q(x,y)\varphi_h dxdy \qquad \forall \varphi_h \in U_{-1}^0(\Omega_h)$$

$$\sum_{i,j} \int_{\partial\Omega_{i,j}} \nu(\mathbf{J}_h\cdot\mathbf{n})\, ds = 0 \qquad \forall \nu_h \in M_{-2}^k(\mathcal{E}_h^0)$$

Substitution of the corresponding basis functions and integrating then yields a system of the following form,

$$\begin{bmatrix} A & B^T & C^T \\ B & 0 & 0 \\ C & 0 & 0 \end{bmatrix} \begin{bmatrix} J_h \\ \phi_h \\ \mu_h \end{bmatrix} = - \begin{bmatrix} Q_J \\ Q_\phi \\ 0 \end{bmatrix} \tag{6}$$

Certainly this system is indefinite, however, unlike the *mixed* FEMs A is block diagonal with each block corresponding to a cell, $\Omega_{i,j}$. Moreover, each block of A is itself block diagonal, with each block corresponding to a coordinate direction, hence A^{-1} retains the sparsity structure of A. The underlying accuracy has also improved over *mixed* FEMs as it is possible to show that the Lagrange multipliers, μ_h are nothing more than the edge unknowns introduced in (4) and are second order convergent. Combining this with the cell average, ϕ_h which is also second order convergent it is possible to construct a *non-conforming* approximation of $\phi(x,y)$ over each cell, $\Omega_{i,j}$, which is second order.

2 Approximate Schur Complements

2.1 Lumping the Edge Based Schur Complement

The block elimination of J_h and ϕ_h in (6) leads to the Schur complement of the edge unknowns,

$$S_\mu \mu_h = Q_{S_\mu} \tag{7}$$

where $S_\mu = C(A^{-1} - A^{-1}B^T S_B^{-1} B A^{-1})C^T$, $Q_{S_\mu} = C(A^{-1}B^T S_B^{-1} B A^{-1} - A^{-1})Q_J - CA^{-1}B^T S_B^{-1} Q_\phi$ and the partial Schur complement is $S_B = BA^{-1}B^T$. The relative sparsity of the system is preserved in S_μ

because $\mathcal{S}_B$ is strictly diagonal. Moreover, it is possible to prove that $\mathcal{S}_\mu$ is symmetric positive definite. In fact, it is this combination that has made (7) the most common formulation of the *mixed hybrid* finite element system. Finally we note that (7) is precisely the edge based formulation of the *constant-constant* NIM discretization.

To understand the inherent complexity of creating fast iterative solvers for (7) and to motivate our approach we partition the edge unknowns by orientation, vertical and horizontal, $\mu_h = \{u, v\}$ such that (7) may be rewritten,

$$\begin{bmatrix} A_{uu} & A_{uv} \\ A_{uv}^T & A_{vv} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} Q_u \\ Q_v \end{bmatrix} \tag{8}$$

where A_{uu}, A_{vv} are tridiagonal and A_{uv} is bidiagonal. ADI methods appear to be a natural choice and have been investigated by a number of authors. In addition most standard preconditioners such as polynomial and incomplete factorizations have been studied. However, none of these methods can attain the efficiency of multi-level solvers, the development of which has been hindered by the difficult hierarchy of grids and inter-grid transfer operators which must be defined for these edge based stencils. Certainly multi-level solvers would be much easier to develop if only a single orientation of the edge unknowns remained. However, the choice to eliminate either u or v is arbitrary and moreover it is apparent that the respective Schur Complements,

$$\mathcal{S}_u = A_{vv} - A_{uv}^T \left(A_{uu}\right)^{-1} A_{uv}, \qquad \mathcal{S}_v = A_{uu} - A_{uv} \left(A_{vv}\right)^{-1} A_{uv}^T$$

suffer an equivalent loss of sparsity as a result of $(A_{uu})^{-1}$ and $(A_{vv})^{-1}$, respectively. Thus posed purely as a problem in linear algebra, a suitable objective is to minimize the fill generated during the formation of $\mathcal{S}_u$ (or $\mathcal{S}_v$). Further noting that $(A_{uu})^{-1}$ is *pre* and *post* multiplied by bidiagonal matrices the minimal fill attainable through approximations of $(A_{uu})^{-1}$ results when it is diagonal. Determining the *optimal* diagonal approximation is a foreboding task, however, if we recall the best incomplete Cholesky factorizations preserve *row sums* it is reasonable to construct a diagonal approximation, $\widetilde{A_{uu}}$ which is composed simply of *row sums* of A_{uu}. A simple inversion then gives $(\widetilde{A_{uu}})^{-1} \approx (A_{uu})^{-1}$. Hence the preconditioner may be written,

$$\widetilde{\mathcal{S}_{\mu;u}} = \begin{bmatrix} \widetilde{A_{uu}} & A_{uv} \\ A_{vu} & A_{vv} \end{bmatrix} \tag{9}$$

The incredible potential of this approximation becomes apparent if we note that $\widetilde{\mathcal{S}_u} = A_{vv} - A_{uv}^T (\widetilde{A_{uu}})^{-1} A_{uv}$ is not only symmetric positive definite but also a second order 9-point approximation of an elliptic PDE. Thus, Dendy's *black box* multigrid code [2], a robust and optimal solver, can be employed to invert it.

2.2 Lumping the Cell Based Schur Complement

Conversely, the Schur complement for the cell based unknowns may be written,

$$\mathcal{S}_\phi \phi_h = Q_{\mathcal{S}_\phi} \tag{10}$$

where $\mathcal{S}_\phi = B\left(A^{-1} - A^{-1}C^T\mathcal{S}_C^{-1}CA^{-1}\right)B^T$, $Q_{\mathcal{S}_\phi} = Q_\phi - B\left(A^{-1} - A^{-1}C^T\mathcal{S}_C^{-1}CA^{-1}\right)Q_J$ and the partial Schur complement is $\mathcal{S}_C = CA^{-1}C^T$. It is possible to prove that $\mathcal{S}_\phi$ is symmetric positive definite and hence an interesting candidate for iterative solvers. Of particular importance to the robustness of the resulting solvers is that this reduced system (10) governs only the cell-based unknowns. However, the relative sparsity structure of the system has been compromised because $\mathcal{S}_C$ is block diagonal, with each block itself tridiagonal. Thus the matrix-vector multiplication, $\mathcal{S}_\phi \phi_h$ will involve the inversion of L, $(L-1) \times (L-1)$ and M, $(M-1) \times (M-1)$ tridiagonal matrices.

To develop a preconditioner for $\mathcal{S}_\phi$, and in particular one which is suitable for multigrid inversion we return to the full system (6) and note that $\mathcal{S}_\phi > 0$ if $A > 0$ and $\text{range}(B) \bigcap \text{range}(C) = \{\emptyset\}$. Hence the primary objective is the minimization of fill through symmetric positive definite approximations of A. To this end we follow the success of the previous section and replace A, a block diagonal matrix, with $\tilde{A}$ a

strictly diagonal lumped approximation,

$$A_{i,j} = \left\{ \frac{\Delta x_i \Delta y_i}{6 D_{i,j}} \right\} \begin{bmatrix} 2 & 1 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & 1 & 2 \end{bmatrix} \quad \longrightarrow \quad \tilde{A}_{i,j} = \left\{ \frac{\Delta x_i \Delta y_i}{2 D_{i,j}} \right\} I_4$$

where I_4 is the 4×4 identity matrix. This simple approximation has the profound effect of transforming $\mathcal{S}_\phi \rightarrow \widetilde{\mathcal{S}_\phi}$ where $\widetilde{\mathcal{S}_\phi}$ is in fact the well known 5-point cell-centred discretization of (1). Thus we have derived an approximate Schur complement which may be used as a preconditioner and which may be inverted with Dendy's *black box* multigrid code [2].

We also note that $\tilde{A}$ may be derived by approximating the linear basis of $\mathbf{J}_h$ (ie. $\{\{1,\xi\},\{1,\eta\}\}$) with a piecewise constant basis. Such a basis preserves the 0^{th} moment of $\mathbf{J}_h$ but introduces an error in the first moment. It is this property in combination with the mutual second order consistency observed by $\mathcal{S}_\phi$ and $\widetilde{\mathcal{S}_\phi}$ which suggests a true multi-level solver should be pursued in the future.

2.3 Approximate Multi-level Inversion of the Preconditioner

Using either of the aforementioned preconditioners is clearly impractical if we actually intended to invert them to some strict level of accuracy. However, this is unnecessary as multi-level solvers provide a uniform reduction of the error over all frequencies and hence it is sufficient to employ only a single V or W cycle of *black box* multigrid per conjugate gradient iteration.

3 Numerical Tests

Tests of the robustness and efficiency of the preconditioners developed in Section 2 have been performed over a large class of problems. In both cases these tests have demonstrated that on a uniform mesh the resulting condition number appears to be independent of the grid spacing. In addition it is only weakly dependent on the spatial dependence of the diffusion coefficient. However, the condition number arising from the simple lumping of A_{uu} is sensitive to high aspect ratio cells with, $r = \Delta x / \Delta y > 1$. Conversely, the cell-centred preconditioner is not hindered by high aspect ratio cells.

This failing is readily understood if we employ Theorem 2.4 of Demko et al. [1] which bounds the exponential decay rate of elements in the inverse of a banded matrix. Specifically we consider constant mesh spacing and constant diffusivity with Dirichlet boundary conditions such that the diagonal blocks of A_{uu} are given by,

$$(A_{uu})_{jj} = D \left(\frac{r}{1+r^2} \right) \mathrm{tri}\left[2 - r^2, 8 + 2r^2, 2 - r^2 \right], \quad j = 1, \ldots, M \tag{11}$$

and of dimension $(L-1) \times (L-1)$. A bound on the elemental decay rate may be written in the form,

$$\left| \left\{ (A_{uu})_{jj}^{-1} \right\}_{kl} \right| \leq C \lambda(r)^{|k-l|} \tag{12}$$

where the decay rate is itself bounded,

$$\lambda(r) \leq \lambda_\infty(r) = \frac{\alpha^+ - \alpha^-}{\alpha^+ + \alpha^-}, \quad \alpha^\pm = \sqrt{(4+r^2) \pm |2 - r^2|} \tag{13}$$

One interesting property of the discretization, namely that the 5-point approximation is exact for $r = \sqrt{2}$, is observed here with $\lambda(\sqrt{2}) = 0$. In addition we note that $\lambda_\infty(1) = 0.101\ldots$ which is a fantastic bound. Perhaps most importantly, Figure 1 reveals that not only does $\lambda_\infty(r) \rightarrow 1$ as $r \rightarrow \infty$ but $\lambda_\infty(r)$ is disastrously close to 1 for even moderate values of r (eg. $\lambda_\infty(8) = 0.65\ldots$).

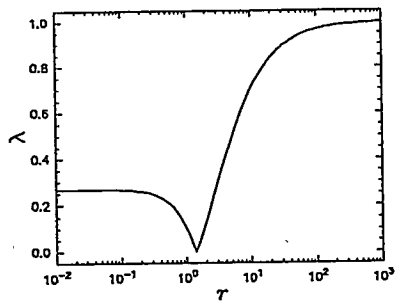


Figure 1: Upper bound on the decay rate $\lambda_\infty(r)$.

3.1 A Toy Problem

Consider the following Dirichlet problem with constant diffusivity, $D(x,y) \equiv 1$ and solution,

$$\phi(x,y) = 2 + \sin(\alpha\pi x)\sin(\beta\pi y)$$

where $\{\alpha, \beta\}$ are free parameters. Substitution of the *solution* into (1) yields the required source,

$$Q(x,y) = (\alpha^2 + \beta^2)\,\pi^2 \sin(\alpha\pi x)\sin(\beta\pi y)$$

A comparison of the edge-based, $\widetilde{\mathcal{S}_{\mu;u}}$ and the cell-based $\widetilde{\mathcal{S}_\phi}$ preconditioners is presented in Table 1 for the domain $[0,1]\times[0,1]$ with $\alpha = \beta = 2$. It is clear from these results that both preconditioners generate an average residual reduction which is independent of the mesh size. Moreover, there is essentially no difference in the iteration counts for those runs which inverted the preconditioner *exactly* (ie. columns marked, C) and those which only used a single V(2,1,1) cycle (ie. columns marked, S). Thus, it is possible to claim that these solvers are indeed optimal.

Unfortunately we must also demonstrate the vulnerability that $\widetilde{\mathcal{S}_{\mu;u}}$ has regarding high aspect ratio cells and hence we conducted several runs in which the size of the physical domain ($[0,a]\times[0,b]$) was varied while the mesh size remained fixed. The results for the preconditioned edge-based solver are summarized in Table 2. Here it is apparent that for r sufficiently close to 1, this solver demonstrates optimal efficiency. Performance is still excellent as r approaches zero. However, just as the bound on λ_∞ predicts, the efficiency is degraded as r increases. Conversely, it is apparent from Table 3 that high aspect ratio cells do not hinder the cell-based preconditioner.

Table 1: A comparison of iteration counts for the two preconditioners, $\widetilde{\mathcal{S}_{\mu;u}}$ and $\widetilde{\mathcal{S}_\phi}$ for the *toy problem* with constant mesh spacing on the domain $[0,1]\times[0,1]$. (Convergence criteria of 10^{-6}).

Mesh Size	CG($\mathcal{S}_\mu$)	$\widetilde{\mathcal{S}_{\mu;u}}$		$\widetilde{\mathcal{S}_\phi}$	
		C	S	C	S
10×10	47	7	7	11	12
20×20	101	7	7	12	12
40×40	201	7	7	12	12

4 Conclusions

The family of *nodal methods* and *mixed hybrid* finite element methods embody many desirable properties of the underlying PDE which contribute to their robustness and accuracy. Unfortunately their very design

Table 2: Iteration counts for the lumped preconditioner, $\widetilde{\mathcal{S}_{\mu;u}}$ with constant mesh spacing on the domain $[0,a] \times [0,b]$ and a convergence criteria of 10^{-6}. {Note: $\alpha = 2/a, \beta = 2/b$}

Mesh Size	Domain Parameters: (a,b) with $r = b/a$													
	$(8,1)$		$(4,1)$		$(2,1)$		$(1,1)$		$(1,2)$		$(1,4)$		$(1,8)$	
	C	S	C	S	C	S	C	S	C	S	C	S	C	S
10×10	9	9	9	9	9	9	7	7	7	7	14	15	21	24
20×20	9	9	9	10	9	9	7	7	7	7	14	15	26	27
40×40	9	9	10	10	9	9	7	7	7	7	15	15	27	28

Table 3: Iteration counts for the cell-based preconditioner, $\widetilde{\mathcal{S}_{\phi}}$ with constant mesh spacing on the domain $[0,a] \times [0,b]$ and a convergence criteria of 10^{-6}. {Note: $\alpha = 2/a, \beta = 2/b$}

Mesh Size	Domain Parameters: (a,b) with $r = b/a$													
	$(8,1)$		$(4,1)$		$(2,1)$		$(1,1)$		$(1,2)$		$(1,4)$		$(1,8)$	
	C	S	C	S	C	S	C	S	C	S	C	S	C	S
10×10	11	11	12	12	12	12	11	12	12	12	12	12	11	11
20×20	13	13	12	12	12	12	12	12	12	12	12	12	13	13
40×40	13	13	13	13	12	12	12	12	12	13	13	13	13	13

makes their solution awkward and costly, thwarting many potential users. Yet we have demonstrated that this awkwardness may be exploited in the development of preconditioners as it provides a natural partitioning of the system. Clearly the existence of such a partitioning, suggests the investigation of various reduced systems (eg. Schur complements) in conjunction with suitably sparse approximations. In particular, we presented two such approximations, $\widetilde{\mathcal{S}_{\mu;u}}$ and $\widetilde{\mathcal{S}_{\phi}}$, and demonstrated that the preconditioning which resulted was optimal in the sense that a fixed number of iterations, independent of the mesh spacing, was required to reduce the residual by a fixed amount. But ultimately these solvers are competitive with multi-level methods because the preconditioner is only approximately inverted with a single V or W cycle of Dendy's *black box multigrid.*

Unfortunately, $\widetilde{\mathcal{S}_{\mu;u}}$ which preconditions the most popular form of the discretization, is sensitive to high aspect ratio cells having $r > 1$. This shortcoming, an artifact of arbitrarily approximating A_{uu} as opposed to A_{vv}, can in fact be alleviated by defining a preconditioner which is composed of two *half* steps. Specifically, the first *half* step approximates $A_{uu} \rightarrow \widetilde{A_{uu}}$ while the second approximates $A_{vv} \rightarrow \widetilde{A_{vv}}$. Conversely, an identical approximation is made in both coordinate directions during the development of $\widetilde{\mathcal{S}_{\phi}}$ and hence it's preconditioning is insensitive to high aspect ratio cells. A result which suggests that this form of the discretization may be of significant practical interest. Moreover both preconditioners are robust with respect to spatial variations in the diffusion coefficient.

References

[1] S. Demko, W. F. Moss, and P. W. Smith. Decay rates for inverses of band matrices. *Mathematics of Computation*, 43(168):491–499, 1984.

[2] J. E. Dendy. Black box multigrid. *J. Comput. Phys.*, 48:366–386, 1982.

[3] J. P. Hennart and E. del Valle. On the relationship between nodal schemes and mixed-hybrid finite elements. *Numerical Methods for Partial Differential Equations*, 9:411–430, 1993.

[4] R. D. Lawrence. Progress in nodal methods for the solution of the neutron diffusion and transport equations. *Progress in Nuclear Energy*, 17(3):271–301, 1986.

"ON PRECONDITIONING TECHNIQUES FOR DENSE LINEAR SYSTEMS ARISING FROM SINGULAR BOUNDARY INTEGRAL EQUATIONS"

KE CHEN *

Abstract. We study various preconditioning techniques for the iterative solution of boundary integral equations, and aim to provide a theory for a class of sparse preconditioners. Two related ideas are explored here: singularity separation and inverse approximation. Our preliminary conclusion is that singularity separation based preconditioners perform better than approximate inverse based while it is desirable to have both features.

Key words. Dense and Sparse matrices, Boundary integral equations, Preconditioners, Iterative methods.

1. Introduction. The numerical solution of boundary integral equations gives rise to dense linear systems that often have unsymmetric matrices. As all boundary integral equations posses some kind of singularity, iterative methods (two-grid and multi-grid type, CG-type) generally do require preconditioning for any convergence. Only in the case of weak singularities, has preconditioning been found non-essential.

Recently various practical preconditioning techniques have been proposed for the difficult case of strong singularities that include [1-4] among others. Such preconditioners or their inverses are sparse, requiring only $O(n)$ flops to implement at each step. They were mostly proposed based on efficiency considerations and heuristic arguments and have not been fully justified in theory.

Here we present two new contributions towards understanding and designing such sparse preconditioners.

Firstly we have generalized the work of [5] on singular integral equations to singular boundary integral equations and have shown that operator splitting offers a new way of regularizations into compact operators. Essentially, by studying the discretization process, we can identify part of an integral operator representing all singularities and follow its corresponding discretization leading to part of the full matrix, that is usually sparse. Then such a sparse matrix is used to construct a suitable preconditioner. With this approach, we can see the close link between matrix splitting and integral operator splitting. More importantly the preconditioned matrix like the preconditioned integral operator can be shown to have its eigenvalues clustered at a fixed point.

Secondly we have investigated the possibility of constructing an approximate inverse preconditioner [2,4,6] that may involve the solution of a least squares problem. We show that the method of [1], while approximately solving such a least squares problem, possesses the nice property of singularity separation and leads to a robust preconditioner that clusters eigenvalues.

* Computational Mathematics Group, Department of Mathematical Sciences, The University of Liverpool, Liverpool L69 3BX, UK. *Email: K.Chen@liverpool.ac.uk. URL: http://www.liv.ac.uk/~cmchenke/cm.html*

Numerical experiments that support the theoretical results will be reported. For comparison purpose, we also report results on using the fast wavelet transform (FWT) as a preconditioner. Our conclusion is that singularity separation based preconditioners perform better than approximate inverse based while it is desirable to have both features.

2. Preconditioning techniques. To give a flavour of some of these techniques, we illustrate each sparse preconditioner by a small system so that their implementations may become more transparent. Further and complete details will be given in a full paper in preparation. For equation $Au = f$, denote a preconditioner by M. For boundary element equations, we have $A = \alpha I - K$, where α may be 0 for first kind equations.

Then a (left) preconditioned system can be written as $MAu = Mf$. In general, two requirements are imposed. Firstly a system $Mx = y$ for any vectors x, y should be efficiently solved in less than $O(n^2)$ operations or $O(n)$ operations if a more sophisticated method like the panel clustering is also used. This normally means that M or its inverse must be sparse and so it natural to seek sparse preconditioners. Secondly the preconditioned matrix MA should possess some better properties than A e.g. eigenvalues are more clustered. Often we hope the condition number of MA to be smaller than that of A. In this sense, the best preconditioner should approach the inverse of A.

2.1. Two grid based sparse column preconditioners. Let $\alpha = 1$ and $n = \eta m$ with η, m integers. Then from matrix $K = (k_1, k_2, \cdots, k_n)$, construct column vectors by

$$k_i' = \begin{cases} k_i, & \text{if } i = \ell\eta, \quad 1 \le \ell \le m, \\ 0, & \text{otherwise,} \end{cases}$$

and define a new matrix by $K' = (k_1', k_2', \cdots, k_n')$. Then we use $M = A' = (I + \eta K')$ as a preconditioner. With $n = 9$,

$$M = \begin{pmatrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \\ B_{31} & B_{32} & B_{33} \end{pmatrix} = \begin{pmatrix} \times & & \times & & & \times & & & \times \\ & \times & \times & & & \times & & & \times \\ & & \boxed{\times} & & & \boxed{\times} & & & \boxed{\times} \\ & & \times & \times & & \times & & & \times \\ & & \times & & \times & \times & & & \times \\ & & \boxed{\times} & & & \boxed{\times} & & & \boxed{\times} \\ & & \times & & & \times & \times & & \times \\ & & \times & & & \times & & \times & \times \\ & & \boxed{\times} & & & \boxed{\times} & & & \boxed{\times} \end{pmatrix},$$

where each B_{ij} denotes a 3×3 block matrix.

2.2. Mesh neighbour based approximate inverses. Here we wish to find a matrix such that $AM \approx I$ and this M should have its diagonal elements and immediate neighbours possess more importance as A usually comes from singular integral

equations. In particular, assume that M is a quasi-tridiagonal matrix; for $n = 9$ we have

$$M = \begin{pmatrix} \times & \times & & & & & & & \times \\ \times & \times & \times & & & & & & \\ & \times & \times & \times & & & & & \\ & & \times & \times & \times & & & & \\ & & & \times & \times & \times & & & \\ & & & & \times & \times & \times & & \\ & & & & & \times & \times & \times & \\ & & & & & & \times & \times & \times \\ \times & & & & & & & \times & \times \end{pmatrix}. \tag{1}$$

Then to find $M = (p_1, p_2, \cdots, p_n)$ in terms of $A = (a_1, a_2, \cdots, a_n)$, a direct approach is used. Let $AM = I \iff Ap_i = e_i$. Then, as p_i only has three nonzero positions i.e. $p_{i_1}, p_{i_2}, p_{i_3}$, an approximate solution is obtained by solving a 3×3 system

$$\begin{pmatrix} A_{i_1 i_1} & A_{i_1 i_2} & A_{i_1 i_3} \\ A_{i_2 i_1} & A_{i_2 i_2} & A_{i_2 i_3} \\ A_{i_3 i_1} & A_{i_3 i_2} & A_{i_3 i_3} \end{pmatrix} \begin{pmatrix} p_{i_1} \\ p_{i_2} \\ p_{i_3} \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix},$$

where $i_1 = i-1, i_2 = i, i_3 = i+1$ for column $i = 2, \cdots, n-1$, $i_1 = n, i_2 = 1, i_3 = 2$ for $i = 1$ and $i_1 = n-1, i_2 = n, i_3 = 1$ for $i = n$ due to the wrap around nature of M.

2.3. Approximate inversion techniques. Here we wish to construct approximations, M, to the inverse of matrix A for which $\|AM - I\|$ is small in some norm. Once such an M has been constructed, we solve the preconditioned linear system $AMw = f$, $u = Mw$.

Consider all possible choices of M that have the same sparsity as in (1). To this end, let $\mathcal{S}$ be the set of (i,j)'s corresponding to all nonzeros in (1), and $\mathcal{G}_{\mathcal{S}}$ be the space of all $n \times n$ matrices that have entries in positions indexed by $\mathcal{S}$. For each column index $j = 1, \cdots, n$, define its row indexes $\mathcal{S}_j = \{i : (i,j) \in \mathcal{S}\}$ and let its vector space $\mathcal{G}_{\mathcal{S}_j}$ contain all vectors that have entries in positions indexed by $\mathcal{S}_j$. Then the approximate inverse M is calculated by solving the least squares problem (LS) in the Frobenius norm

$$\min_{M \in \mathcal{G}_{\mathcal{S}}} \|AM - I\|_F^2 = \sum_{j=1}^{n} \min_{m_j \in \mathcal{G}_{\mathcal{S}_j}} \|Am_j - e_j\|_2^2 = \sum_{j=1}^{n} \min_{m_j \in \mathcal{G}_{\mathcal{S}_j}} \|A_j m_j - e_j\|_2^2 \tag{2}$$

where $M = [m_1, \cdots, m_n]$, $I = [e_1, \cdots, e_n]$ and A_j is a $n \times 3$ submatrix of A. The full problem of finding M is thus reduced to n standard least squares problems for its column vectors.

2.4. Sparse LU decompositions. Instead of looking for M of form (1) that approximates the inverse of A, we choose M to be explicitly that part of A that has the same sparsity as (1). Refer to [3] and [5]. To illustrate, for $n = 5$, we have $M = LU$ or

$$\begin{bmatrix} A_{1,1} & A_{1,2} & & & A_{1,5} \\ A_{2,1} & A_{2,2} & A_{2,3} & & \\ & A_{3,2} & A_{3,3} & A_{3,4} & \\ & & A_{4,3} & A_{4,4} & A_{4,5} \\ A_{5,1} & & & A_{5,4} & A_{5,5} \end{bmatrix} = \begin{bmatrix} L_{1,1} & & & & \\ L_{2,1} & L_{2,2} & & & \\ & L_{3,2} & L_{3,3} & & \\ & & L_{4,3} & L_{4,4} & \\ L_{5,1} & L_{5,2} & L_{5,3} & L_{5,4} & L_{5,5} \end{bmatrix} \begin{bmatrix} 1 & U_{1,2} & & & U_{1,5} \\ & 1 & U_{2,3} & & U_{2,5} \\ & & 1 & U_{3,4} & U_{3,5} \\ & & & 1 & U_{4,5} \\ & & & & 1 \end{bmatrix}.$$

Note that the inverse of a sparse matrix of form (1) is, though sparse, not of the same sparsity pattern; see Figs 1-2 where the inverse pattern of a 32×32 matrix $M = I - K$ (entries of K are randomly generated in $(0,1)$) is illustrated.

FIG. 1. *Inverse of a quasi-triangular matrix* M $(|M_{ij}^{-1}| \geq 10^{-5})$

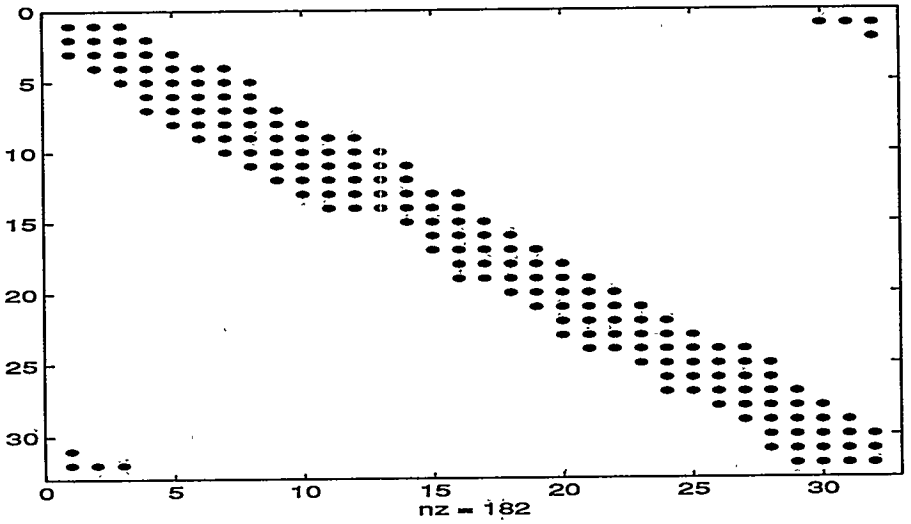
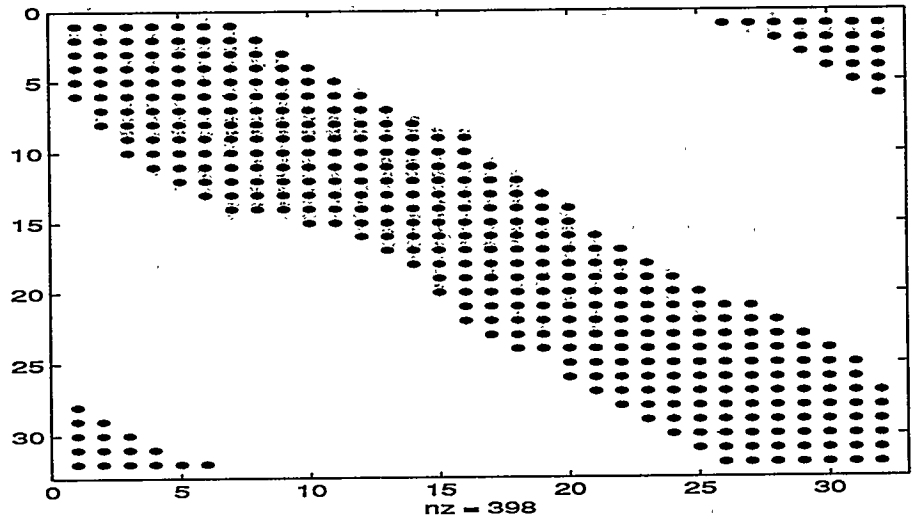


FIG. 2. *Inverse of a quasi-triangular matrix* M $(|M_{ij}^{-1}| \geq 10^{-10})$



2.5. Singularity separation preconditioner. Our recent work, attempting to devise a theory suitable for singular boundary integral equations, has suggested the following choice of M as a good preconditioner

$$M = \begin{bmatrix} A_{1,1} & & & & A_{1,n} \\ A_{2,1} & A_{2,2} & & & \\ & A_{3,2} & \ddots & & \\ & & \ddots & \ddots & \\ & & & A_{n,n-1} & A_{n,n} \end{bmatrix}.$$

Such a matrix can be factorized into L, U efficiently.

3. Operator and matrix splitting. Corresponding to every preconditioning technique for matrices, there must exist an underlying preconditioning strategy for the integral operator. But it is not always obvious to work out such a strategy. It turns out that, for most of the above mentioned preconditioners, there exist operator splittings giving rise to matrix splittings.

Further eigenvalue properties of the preconditioned matrix equations follow that of the preconditioned operator equations. This leads to a theory of preconditioned matrices and, more importantly, opens up new possibilities for new constructions.

4. Variants of the normal equation. All preconditioning techniques have been implemented on test problems using the conjugate gradient method based on the normal equation. What is interesting about the normal equation is that, when a preconditioner M is involved, the normal equation can take any of the four forms

$$(MA)^T MAu = (MA)^T Mf,$$

$$MA(MA)^T z = Mf,$$

$$(AM)^T AMy = (AM)^T f,$$

$$AM(AM)^T w = f,$$

where $u = My = (MA)^T z = M(AM)^T w$. Here there are two issues. Firstly one may use left or right preconditioners. But once a preconditioner is available, it might be applied either way. Secondly for non-normal matrices, left and right multiplications produce different results. So a left preconditioner may be applied on the right to have better results and vice versa.

REFERENCES

[1] Vavasis, S. (1992) SIAM J. Matr. Anal. Appl., V.13.
[2] Cosgrave at al, (1992), Int. J. Comp. Math., V.44.
[3] Amini, S. and Maines, N. (1994) Proceedings of BEM 14.
[4] Yan, Y. (1994) SIAM J. Sci. Comp., V.15.
[5] Chen, K. (1994) Elec. Tran. Numer. Anal. (ETNA), V.2.
[6] Huckle T. and Grote M. (1994) Preprint.

FAST WAVELET BASED SPARSE APPROXIMATE INVERSE PRECONDITIONER

W. L. WAN*

Abstract. Incomplete LU factorization is a robust preconditioner for both general and PDE problems but unfortunately not easy to parallelize. Recent study of Huckle and Grote [15] and Chow and Saad [6] showed that sparse approximate inverse could be a potential alternative while readily parallelizable. However, for special class of matrix A that comes from elliptic PDE problems, their preconditioners are not optimal in the sense that independent of mesh size. A reason may be that no good *sparse* approximate inverse exists for the dense inverse matrix. Our observation is that for this kind of matrices, its inverse entries typically have piecewise smooth changes. We can take advantage of this fact and use wavelet compression techniques to construct a better sparse approximate inverse preconditioner. We shall show numerically that our approach is effective for this kind of matrices.

1. Introduction. Consider solving the right preconditioned linear system,

$$AMy = b, \quad x = My, \tag{1}$$

where A is large, sparse matrix and M is a right preconditioner. We want to find a sparse M so that $\|AM - I\|$ is small. This approach has been studied by Benson [2] and Benson and Frederickson [3]. More precisely, we want to minimize the residual,

$$\min_M \|AM - I\|_F^2. \tag{2}$$

The Frobenius norm is particularly chosen for parallelism. Notice that

$$\|AM - I\|_F^2 = \sum_{j=1}^{n} \|Am_j - e_j\|_2^2,$$

where m_j and e_j are the jth column of M and I respectively. Thus solving (2) leads to solving n independent least square problems,

$$\min_{m_j} \|Am_j - e_j\|_2, \qquad j = 1, \ldots, n, \tag{3}$$

which can be done in parallel.

In practice, it is desirable to look for sparse solution of (3). However, this poses two difficulties: how to determine the sparsity pattern of M and how to solve (3) efficiently. Recently, there are two approaches. One is discussed by Cosgrove et al [7] and Huckle and Grote [15] and the other is by Chow and Saad [6]. For the former approach, they solve the least square problems (3) by QR factorization, which apparently sounds costly. But because m_j is sparse, the cost of QRF can be greatly reduced. Moreover, they derive some algorithm to determine the positions of fill-in adaptively. Similar approaches can be found in [17], [16], [18], [13], [14], in which case, the sparsity pattern of M is fixed.

For Chow and Saad's approach, they use standard iterative method to find an approximate solution to

$$Am_j = e_j,$$

* Dept. of Mathematics, Univ. of Calif. at Los Angeles, Los Angeles, CA 90024. E-mail: wlwan@math.ucla.edua.

and apply some dropping strategy to m_j to control the number of fill-in. We should also mention that they have suggested several other possible approaches of solving (2) in [6].

For more details in classical approximate inverse methods, we refer the reader to [1].

There is a potential difficulty in approximate inverse preconditioning. Its idea bases on the assumption that A^{-1} can be approximated by a sparse matrix M. It is well known that A^{-1} can be structurally dense even A is sparse. Yet one might expect rapid decay in the A^{-1} entries away from the diagonal [10], [5], [11] for some class of matrix A. However, as mentioned in [7], [6], if we require $\|AM-I\|_1 < 1$, we can always find a sparse matrix A while M has to be dense. Even if A is symmetric positive definite or A comes from PDE problems, the inverse entries still need not decay.

For example, consider the following near tridiagonal symmetric positive definite matrix of size 40×40, derived from some periodic boundary problem,

$$A = \begin{bmatrix} 2.01 & -1 & & & & & -1 \\ -1 & 2.01 & -1 & & & & \\ & -1 & 2.01 & -1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & -1 & 2.01 & -1 & \\ -1 & & & & -1 & 2.01 & \end{bmatrix}.$$

The inverse entries are all greater than one and hence have no decay at all, as we can see from fig 1.

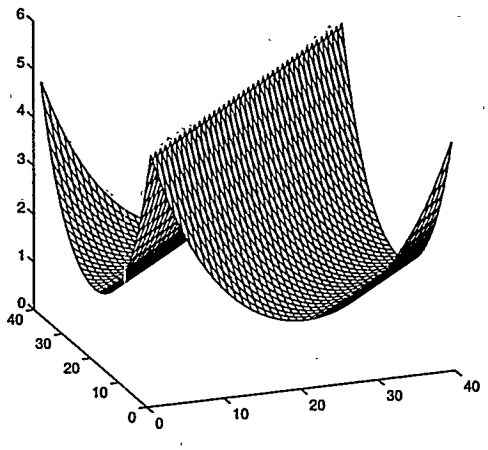


FIG. 1. *Mesh plot of* A^{-1}.

In such case, approximate inverse preconditioners may not work very well. Actually, it is too much to ask for decay of inverse entries. We want to weaken this assumption and still get a sparse approximate inverse. One observation is that if A corresponds to a differential operator of some elliptic PDE, the inverse would be the corresponding Green's function. See appendix for a simple example illustrating this idea. Similar observation in the case of solving integral equation can be found in [4] and in the case of hyperbolic and parabolic PDE can be found in [12]. Since A^{-1} corresponds to the Green's function, we would expect piecewise smooth changes in the inverse entries. In other words, if we treat the inverse matrix as a graph of a function of two variables, then we will get a piecewise smooth graph. Note that for piecewise smooth inverse, the entries need not have decay as shown in fig 1.

For matrix A that comes from elliptic problems, its inverse is typically piecewise smooth with singularity along the diagonal. Its entries may or may not decay. From now on, we focus our attention to this kind of matrices and try to derive an efficient approximate inverse preconditioner. In fact the idea is similar to that of Beylkin et al [4] and Engquist et al [12]. Our strategy is to apply wavelet transform to compress these piecewise smooth inverse matrices. Then we combine this idea with Huckle and Grote's implementation to construct the sparse approximate inverse. We shall show that our new approach gives rise to a more effective preconditioner in this class of matrices.

In section 2, we show how to adopt wavelet transform in the least square approach of solving (3). In section 3, we estimate the extra cost for the wavelet transform. In section 4, we present several numerical examples to compare various methods. Finally, we conclude our approach in section 5.

2. Fast Wavelet Based Approximate Inverse. Assuming A^{-1} is piecewise smooth, our idea is to apply wavelet transform to compress A^{-1} and then use it as a preconditioner. Apparently, this is impossible since we don't even have A^{-1}. Our trick is the observation that

$$\widetilde{A^{-1}} \equiv WA^{-1}W^T = (WAW^T)^{-1} = \tilde{A}^{-1},$$

where W is an orthogonal wavelet transform matrix. We first transform A to its wavelet basis representation $\tilde{A}$ and then apply Huckle and Grote's method to find an approximate inverse for $\tilde{A}$, which is the preconditioner that we want to compute. We will make all these ideas precise in the next two sections.

2.1. Wavelet Formulation. We shall show how we adopt wavelet transform in the least square approach. Consider equation (2) again. Let W be an orthogonal wavelet transform matrix, i.e. $\tilde{x} = Wx$ is the vector x in the wavelet basis. (Note that W can be 1-level or full $\log_2 n$-level wavelet transform matrix.) Then

$$\begin{aligned} \min_M \|AM - I\|_F &= \min_M \|WAW^T WMW^T - I\|_F \\ &= \min_{\tilde{M}} \|\tilde{A}\tilde{M} - I\|_F, \end{aligned}$$

where $\tilde{A} = WAW^T$ and $\tilde{M} = WMW^T$ are representations of A and M in the wavelet basis respectively. Thus, our n least square problems become

$$\min_{\tilde{m}_j} \|\tilde{A}\tilde{m}_j - e_j\|_2, \qquad j = 1, 2, \ldots, n. \tag{4}$$

$\tilde{A}$ is sparse (but probably denser than A) since A is. Because of the wavelet basis representation, if M is piecewise smooth, we would expect $\tilde{M}$, neglecting small entries, to be sparse too. Proof can be seen in [12]. Therefore, the sparse solution of (4) would probably give a more effective approximate inverse than both previous approaches. We shall justify this numerically in section 4.

2.2. Algorithm. In previous section, we derive the wavelet formulation of approximate inverse. Then we can simply apply Huckle and Grote's method, which is called SPAI algorithm [15], to solve the n least square problems (4) in parallel. Here is the algorithm.

Wavelet Based SPAI Algorithm

(a) Wavelet transform A to get $\tilde{A} = WAW^T$.
(b) Apply SPAI algorithm to solve for $\tilde{M}$.
(c) Use $\tilde{M}$ as preconditioner to solve: $\tilde{A}\tilde{x} = \tilde{b}$, where $\tilde{b} = Wb$.
(d) Apply backward wavelet transform to $\tilde{x}$ to get $x = W^T\tilde{x}$.

It should be noted that if we know the sparsity pattern of $\tilde{M}$ *a priori*, we can simply use that pattern to solve the least square problems (4) instead of using Huckle and Grote's adaptive approach. It will then be much more efficient. Actually, for M being piecewise smooth, the significant nonzero entries of $\tilde{M}$ are determined by the location of the singularities of M. In the special case that A is the 1D or 2D Laplacian operator, then the singularity of M will concentrate at the block diagonal. We can simply use block diagonal as the sparsity pattern for $\tilde{M}$ and save time for searching new fill-in elements.

3. Complexity and Implementation of Algorithm. In this section, we try to analyze the complexity of each step of our algorithm. We shall show that it is not much more expensive than the SPAI algorithm although it seems to be. In the following discussion, we assume that the wavelet used is of compact support, [8], [9].

Step (a). In general, to compute wavelet transform of a vector requires $O(n)$ operations. To compute $\tilde{A} = WAW^T$, it is equivalent to transform the columns and then the rows of A (or vice versa). Thus it will cost $O(n^2)$ operations. However, since A is sparse, if we assume that there are only $O(1)$ nonzeros in each column and each row, the cost will reduce to $O(kn)$ where k, ranging from 1 to $\log_2 n$, is the number of levels in the wavelet transform. In fact, in the implementation, we do not need to form $\tilde{A}$ explicitly. Notice that solving each least square problem only need a few columns of $\tilde{A}$. We just form those columns and the cost will then become $O(n)$.

Step (b). In each level of transform, we will introduce a fixed amount of fill-in. Even though there are only $O(1)$ nonzeros in each column and row of A, there will be $O(k)$ nonzeros in each column and row of $\tilde{A}$. We can choose k so small that the number of nonzero introduced is not significant. We may further reduce the cost by solving the least square problem (4) only locally since the singularities of M cluster around the diagonal. More precisely, first we fix the sparsity pattern of $\tilde{M}$ to be block diagonal. Second, we use the nonzero indices, i, of $\tilde{m}_j$ for the column and row indices of $\tilde{A}$ and then we compute the QR factorization for the submatrix $\tilde{A}(i,i)$ when solving (4). This reduces the cost to $O(1)$ when solving each column.

Step (c). When we solve $\tilde{A}\tilde{x} = \tilde{b}$ by some iterative method, we need to perform $\tilde{A}$ times a vector. If we do it directly, the cost will be $O(kn)$. Note that

$$\tilde{A}v = WAW^T v.$$

If we first backward transform v, apply A and then transform back, the overall process will only be $O(n)$.

Hence, with above modifications, the overall complexity of the wavelet based algorithm is only $O(n)$.

4. Numerical Results. In this section, we compare our preconditioner with those by Huckle and Grote's SPAI and ILU(0). We choose several matrices that come from different elliptic PDE. The inverses of all these matrices are piecewise smooth and the singularities are cluster around the diagonal. For efficiency, instead of applying SPAI to solve for $\tilde{M}$ adaptively in step (a) of our algorithm, we specify a block diagonal structure *a priori* for $\tilde{M}$ and then solve (4) by QR. In all the test, we use the compact support wavelet D_4 by I. Daubechies [8], [9]. We apply 6 levels of wavelet transform to matrix of order 1024 and 8 levels of transform to matrix of order 4096. Note that the number of levels is arbitrary. One could use different number in different situation.

We apply these preconditioners to GMRES(20) and BICGSTAB. The initial guess was $x_0 = 0, r_0 = b$ and the stopping criterion was $\|r_n\|/\|r_0\| < 10^{-6}$. All the experiments were done in MATLAB.

Example 1: We first consider a slightly modified artificial case of (4) where the diagonal entries are changed from 2.01 to 2.00001 and the size is 1024×1024. The bandwidth is 0,0,0,0,5,5 for the 1st to 6th level of the block diagonal structure of $\tilde{M}$ respectively. The convergence of different preconditioners is shown in fig 2. We can see that SPAI(0.2) and SPAI(0.4) converge very slowly in this somewhat artificial but illustrating case. On the other hand, the wavelet based preconditioner converges rapidly. This shows the importance of taking advantage of the piecewise smoothness of A^{-1}. We do not show the convergence of ILU(0) since it only takes 3 iterations for convergence. This is exceptional because of the special near tridiagonal structure of A. Table 1 shows the number of nonzero (nnz) for the preconditioners. Moreover the wavelet based preconditioner requires the least number of nonzero as expected.

Example 2: In this case, A is the 2D Laplacian operator with size 1024×1024 and 4096×4096. For n=1024, we choose the bandwidth as before. For n=4096, we choose the bandwidth = 0,0,0,0,5,5,5,5 for the 1st to 8th level of the block diagonals respectively. The convergence is shown in fig 3. We can see the wavelet based preconditioner is still very efficient. Fig 4 and table 2 show similar result for n=4096. We can see that the wavelet based preconditioner saves much more storage than the others. Actually, this is typical throughout our tests.

Example 3: We try to solve something more complicated than the Laplace equation but still have piecewise smooth inverse. Consider the following PDE with variable coefficients,

$$((1+x^2)u_x)_x + u_{yy} + (\tan y)^2 u_y = -100x^2.$$

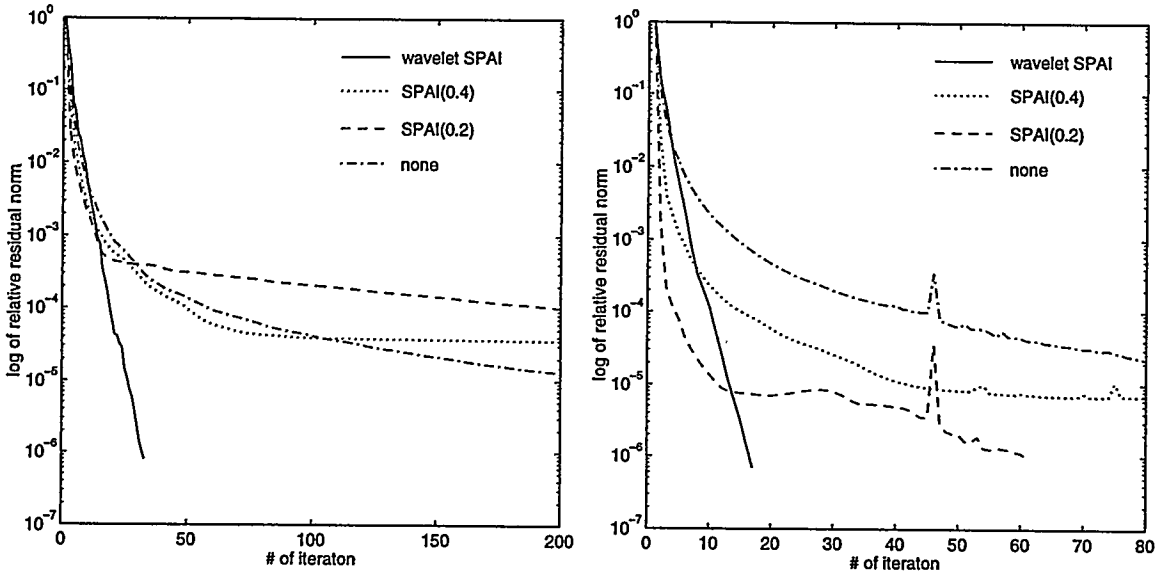


FIG. 2. *Convergence behavior of (a) GMRES(20) (b) BICGSTAB using different preconditioners when solving the artificial problem.*

	n=1024		
	GMRES	BICGSTAB	nnz
Wavelet SPAI	32	16	3544
SPAI(0.4)	>200	>200	5120
SPAI(0.2)	>200	60	21504
ILU(0)	3	3	4096
No preconditioning	>200	>200	~

TABLE 1
Number of iterations for artificial case

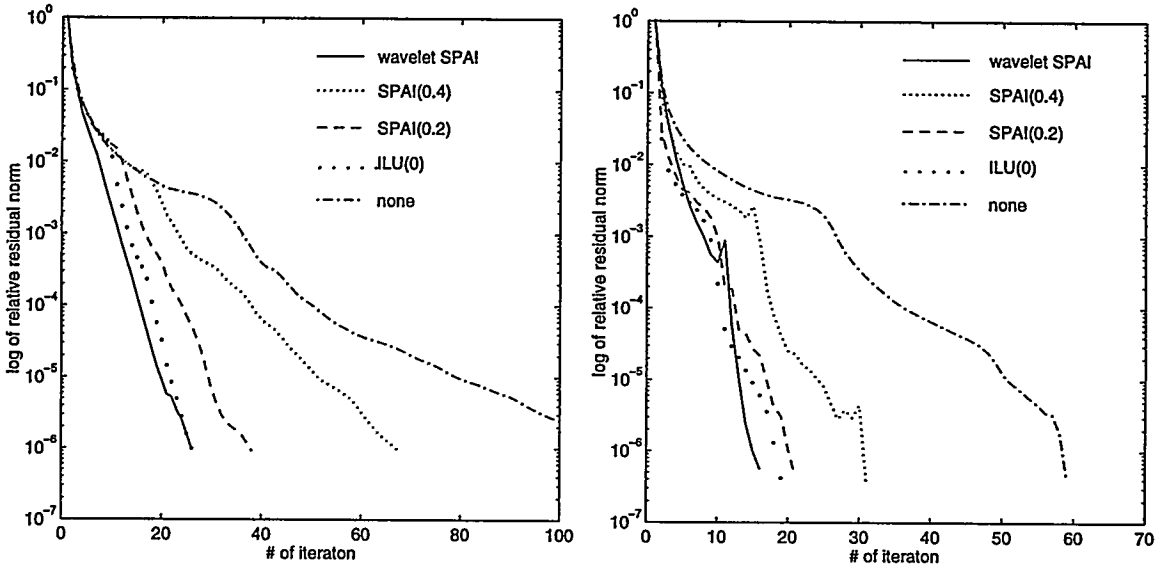


FIG. 3. *Convergence behavior of (a) GMRES(20) (b) BICGSTAB using different preconditioners when solving 2D Laplacian. n=1024.*

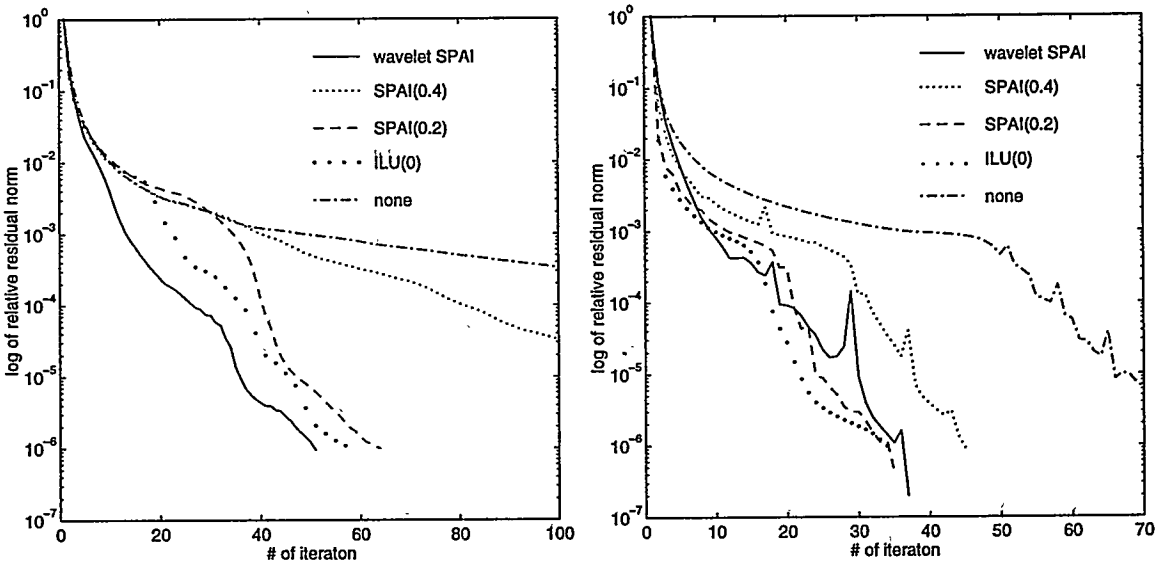


FIG. 4. *Convergence behavior of (a) GMRES(20) (b) BICGSTAB using different preconditioners when solving 2D Laplacian. n=4096.*

	n=1024			n=4096		
	GMRES	BICGSTAB	nnz	GMRES	BICGSTAB	nnz
Wavelet SPAI	25	17	3544	50	36	6616
SPAI(0.4)	67	30	4624	160	44	19472
SPAI(0.2)	37	19	16440	63	34	69688
ILU(0)	25	17	6016	57	33	24320
No preconditioning	116	55	~	>200	88	~

TABLE 2
Number of iterations for 2D Laplacian

We solve the 32×32 and 64×64 grid cases. The bandwidth of the block diagonal of $\tilde{M}$ is the same as before. Again, we can see from fig 5, 6 and table 3 that a significant amount of storage is saved but still getting best performance.

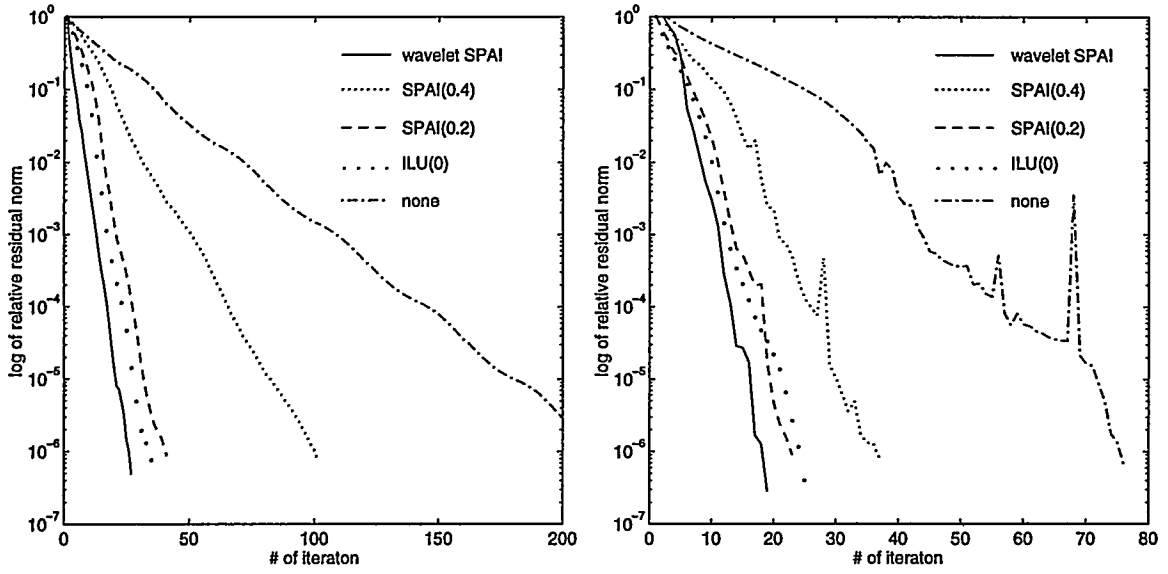


FIG. 5. *Convergence behavior of (a) GMRES(20) (b) BICGSTAB using different preconditioners when solving variable coefficient problem. n=1024.*

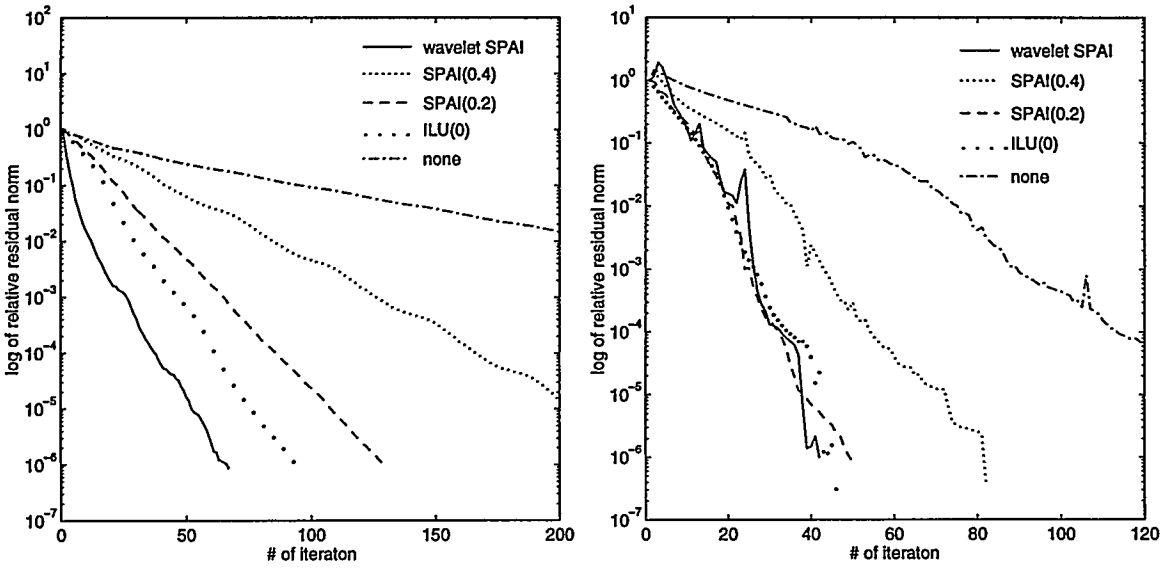


FIG. 6. *Convergence behavior of (a) GMRES(20) (b) BICGSTAB using different preconditioners when solving variable coefficient problem. n=4096.*

Example 4: In this case, A comes from a PDE of helical spring. Same setting as before. The wavelet preconditioner does not outperform the others very much but still among the best. Yet, the amount of storage is much less than the others.

5. Conclusion. We have shown that SPAI could fail if there is no decay in the inverse entries, even though the inverse may be piecewise smooth. We have also shown that the fast wavelet based

	n=1024			n=4096		
	GMRES	BICGSTAB	nnz	GMRES	BICGSTAB	nnz
Wavelet SPAI	26	18	3544	66	41	6616
SPAI(0.4)	100	36	4514	>200	81	18616
SPAI(0.2)	40	22	17260	129	49	73936
ILU(0)	34	24	6016	93	45	24320
No preconditioning	>200	75	~	>200	163	~

TABLE 3
Number of iterations for variable coefficient problem

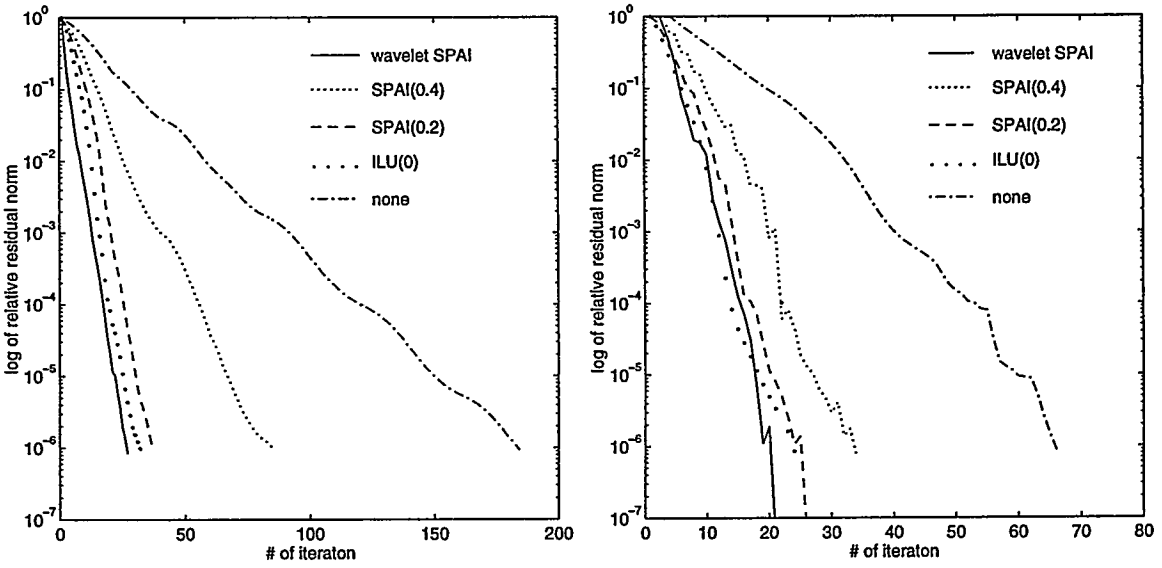


FIG. 7. *Convergence behavior of (a) GMRES(20) (b) BICGSTAB using different preconditioners when solving helical spring problem. n=1024.*

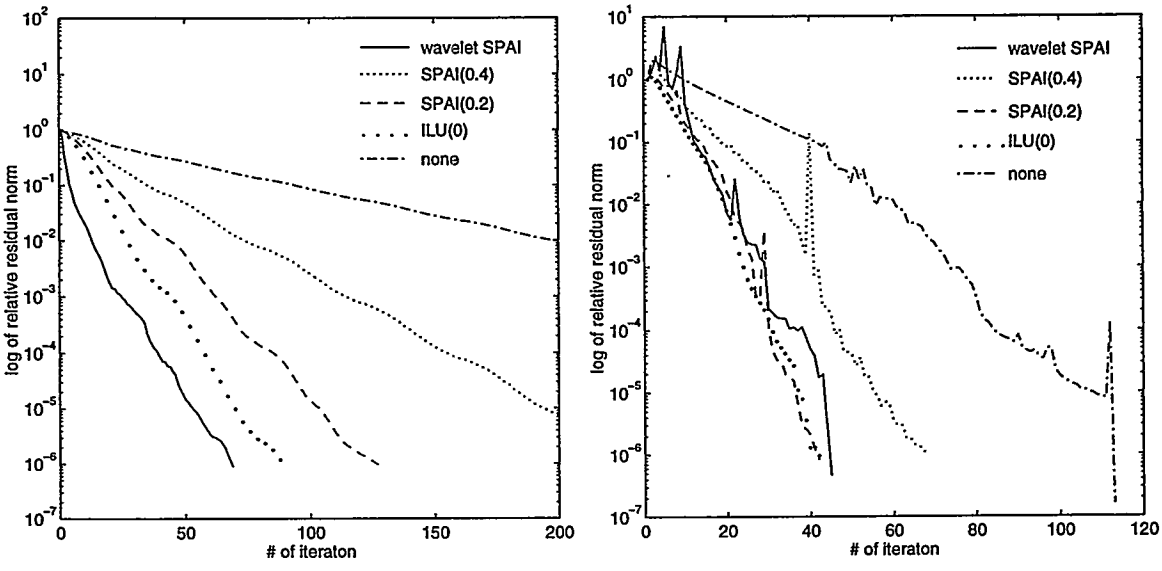


FIG. 8. *Convergence behavior of (a) GMRES(20) (b) BICGSTAB using different preconditioners when solving helical spring problem. n=4096.*

	n=1024			n=4096		
	GMRES	BICGSTAB	nnz	GMRES	BICGSTAB	nnz
Wavelet SPAI	26	20	3544	68	44	6616
SPAI(0.4)	84	33	4624	>200	67	19472
SPAI(0.2)	36	25	16387	126	41	69628
ILU(0)	31	23	6016	89	41	24320
No preconditioning	183	65	~	>200	112	~

TABLE 4
Number of iterations for helical spring problem

preconditioner is a very effective alternative if A^{-1} is piecewise smooth, especially when the singularities are clustered around the diagonal. Moreover, significant amount of storage can be saved. However, for more general case when the number of singularities of A^{-1} is large and they are not cluster around the diagonal, further study and investigation is needed.

Acknowledgments. We would like to thank Tony Chan and Barry Smith for their helpful comments throughout the paper and also W.P. Tang for his active discussion and suggestion on the implementation of our algorithm.

Appendix. We shall show that for 1D Dirichlet problem, the inverse of the discrete Laplacian operator A is actually a discrete Green's function. Consider the following problem,

$$\begin{aligned} -u''(x) &= f(x) \qquad\qquad \forall x \in (0,1) \\ u(0) &= y(1) = 0. \end{aligned}$$

Then the solution is given by $u(x) = \int_0^1 G(x,y)f(y)dy$, where $G(x,y)$ is a Green's function defined as

$$G(x,y) = \begin{cases} y(1-x) & 0 \le y \le x, \\ x(1-y) & x \le y \le 1. \end{cases}$$

Let a partition of [0,1] be $\{x_0, x_1, \ldots, x_{n+1}\}$. We want to evaluate y at the points $\{x_i\}$. By definition of u,

$$u(x_i) = \int_0^1 G(x_i,y)f(y)dy.$$

We compute the integral by trapezoidal rule and we obtain

$$\begin{aligned} u(x_i) &\approx \frac{1}{n+1}\sum_{j=1}^{n} G(x_i,x_j)f(x_j) \\ &= \sum_{j=1}^{n} \frac{1}{n+1} G(\frac{i}{n+1}, \frac{j}{n+1}) f(\frac{j}{n+1}) \\ &= \sum_{j=1}^{n} G_{ij} f_j \end{aligned}$$

where

$$\begin{aligned} G_{ij} &= \frac{1}{n+1} G(\frac{i}{n+1}, \frac{j}{n+1}) \\ &= \begin{cases} \frac{1}{(n+1)^3}[j(n+1-i)] & j \le i, \\ \frac{1}{(n+1)^3}[i(n+1-j)] & i \le j. \end{cases} \end{aligned}$$

On the other hand, let A be the corresponding discrete Laplacian operator (second order central differencing) and $b = (b_j) = (f_j)$. Then the solution for $Ax = b$ is precisely $x = Gb$, where $G = (G_{ij})$ is defined above. Thus, rows of A^{-1} can be viewed as a discrete Green's function.

REFERENCES

[1] O. Axelsson. *Iterative Solution Methods.* Cambridge University Press, Cambridge, 1994.

[2] M.W. Benson. Iterative solution of large scale linear systems. Master's thesis, Lakehead University, Thunder Bay, Ontario, 1973.

[3] M.W. Benson and P.O. Frederickson. Iterative solution of large sparse linear systems arising in certain multidimensional approximation problems. *Utilitas Math.*, 22:127–140, 1982.

[4] G Beylkin, R. Coifman, and V. Rokhlin. Fast wavelet transforms and numerical algorithms I. *Comm. Pure Appl. Math.*, 44:141–184, 1991.

[5] C. De Boor. Dichotomies for band matrices. *SIAM J. Numer. Anal.*, 1980.

[6] E. Chow and Y. Saad. Approximate inverse preconditioners for general sparse matrices. *Colorado Conference on Iterative Methods, April 5-9*, 1994.

[7] J.D.F. Cosgrove, J.C. Diaz, and A. Griewank. Approximate inverse preconditionings for sparse linear systems. *Intern. J. Computer Math*, 1992.

[8] I. Daubechies. Orthonormal bases of compactly supported wavelets. *Comm. Pure Appl. Math.*, 41:909–996, 1988.

[9] I. Daubechies. *Ten Lectures on Wavelets, CBMS-NSF Series Appl. Math.* SIAM, 1991.

[10] S. Demko. Inverses of band matrices and local convergence of spline projections. *SIAM J. Numer. Anal.*, 1977.

[11] S. Demko, W.F. Moss, and P.W. Smith. Decay rates for inverses of band matrices. *Math. Comp.*, 1984.

[12] Bjorn Engquist, Stanley Osher, and Sifen Zhong. Fast wavelet basd algorithms for linear evolution equations. *SIAM J. Sci. Comput.*, 15:755–775, 1994.

[13] M. Grote and H. Simon. Parallel preconditioning and approximate inverses on the connection machine. In *Scalable High Performance Computing Conference (SHPCC), 1992 Williamsburg, VA*. IEEE Computer Science Press, 1992.

[14] M. Grote and H. Simon. Parallel preconditioning and approximate inverses on the connection machine. In Richard Sincover et al, editor, *Sixth SIAM Conference on Parallel Processing for Scientific Computing II*. SIAM, 1993.

[15] T. Huckle and M. Grote. A new approach to parallel preconditioning with sparse approximate inverses. Technical Report SCCM-94-03, Scientific Computing and Computational Mathematics Program, Stanford University, Stanford, California, 1994.

[16] L.Yu. Kolotilina, A.A. Nikishin, and A.Yu. Yeremin. Factorized sparse approximate inverse (FSAI) preconditionings for solving 3d fe systems on massively parallel computers II. In R. Beauwens and P.de Groen, editors, *Iterative Methods in Linear Algebra, Proc of the IMACS International Symposium, Brussels, April 2-4, 1991*, pages 311–312, 1992.

[17] L.Yu. Kolotilina and A.Yu. Yeremin. Factorized sparse approximate inverse preconditionings I. theory. *SIAM J. Matrix Anal. Appl.*, 14:45–58, 1993.

[18] Ju.B. Lifshitz, A.A. Nikishin, and A.Yu. Yeremin. Sparse approximate inverse (FSAI) preconditionings for solving 3D CFD problems on massively parallel computers. In R. Beauwens and P.de Groen, editors, *Iterative Methods in Linear Algebra, Proc of the IMACS International Symposium, Brussels, April 2-4, 1991*, pages 83–84, 1992.

Combined Incomplete LU and Strongly Implicit Procedure Preconditioning

Ernst A. Meese*

December 14, 1995

Abstract

For the solution of large sparse linear systems of equations, the Krylov-subspace methods have gained great merit. Their efficiency are, however, largely dependent upon preconditioning of the equation-system. A family of matrix factorisations often used for preconditioning, is obtained from a truncated Gaussian elimination, ILU(p). Less common, supposedly due to it's restriction to certain sparsity patterns, is factorisations generated by the strongly implicit procedure (SIP). The ideas from ILU(p) and SIP are used in this paper to construct a generalised strongly implicit procedure, applicable to matrices with any sparsity pattern. The new algorithm has been run on some test equations, and efficiency improvements over ILU(p) was found.

1 Introduction

The discretisation of systems of partial differential equations often leads to very large sparse systems of linear equations,

$$A\,x = b, \tag{1}$$

$A \in \mathbb{R}^{n\times n}$, $x, b \in \mathbb{R}^n$. One of the most prominent classes of methods for the solution of these equations, is the Krylov-subspace methods. However, crucial for the efficiency of these methods is the preconditioning of the system (1).

If we represent a right preconditioner by the matrix $M \in \mathbb{R}^{n\times n}$, the Krylov-subspace method should solve the system

$$A\,M^{-1}\,y = b, \quad x = M^{-1}y, \tag{2}$$

instead of system (1). The matrix M is often realised in the form of an iterative method, e.g. Jacobi iteration, SSOR, et c., or even a member of the Krylov-subspace methods.

Another common approach, which is pursued in this paper, is to express M as a product LU, where L and U are lower and upper triangular matrices respectively. The solution of $(LU)\,x = y$ is then easily obtained by successively applying a forward and a backward substitution. For a good preconditioner, LU should approximate A well. However, for the substitution to be computationally cheap, L and U should also be sparse.

To compute such sparse LU factors, some form of truncated Gaussian elimination may be used [7, 8, 16]. As we shall see, the incomplete LU factorisation ILU(p) by Watts [16] combines naturally with the strongly implicit procedure (SIP) developed by Stone [14].

SIP was originally developed as a iterative solver based on a splitting matrix. However, this splitting matrix has also been used as a preconditioner [2, 3, 5]. But, a major limitation to SIP is the restriction to a limited number of differential equations, to specific discretisation stencils, and to certain ordering of the

*Student for a Dr. Ing. degree at the Department of Applied Mechanics, Thermo and Fluiddynamics, Div. of Mechanical Engineering, The Norwegian Institute of Technology, The University of Trondheim, Norway. Student member of SIAM.

grid nodes. Common are one equation on the five-point stencil [14], and on the nine-point stencil [6, 10]. Methods for systems of two differential equations on these stencils also exists [5, 17].

In this paper, ILU(p) and SIP are combined to give a more general SIP factorisation (GSIP). The new algorithm is applicable to a general matrix. It also accepts any level of fill-in in the factors L and U, as does the ILU(p) method. However, some knowledge of the location of the grid nodes must be available to the factorisation algorithm.

With CFD applications in mind, the Poisson equation and a convection-diffusion equation has been chosen for testing purposes. These kinds of equations are common in various strategies for solving the Navier-Stokes equations. However, as yet only a structured five-point discretisation has been implemented.

To study the effects of different fill-in levels on GSIP, and to see whether the new algorithm give factorisations that is sufficiently better than the ILU(p) factorisation to warrant a move to more general problems, the preconditioned Krylov-subspace methods BiCGSTAB, GMRES(m), and CGS has been run on the test problems for a wide range of parameters. BiCGSTAB, stabilised further as suggested by Sleijpen and Van der Vorst [11], was found to be the most efficient on almost all runs.

Optimal fill-in level is always larger than zero. For the Poisson equation a fill-in level of 2 is the best choice for all but the smallest matrix where $p = 1$ is best. For the convection-diffusion equation the fill-in levels 1 and 2 alternate on being optimal.

Generalised SIP factorisation show considerable benefits over ILU(p) in several test runs, particularly on the Poisson equation. GSIP, however, has serious problems with the test matrices with a weak main diagonal.

2 A generalised family of factorisations – GSIP($\alpha, p, \mathtt{G}$)

Before the generalised SIP algorithm is developed, the two methods from which it is motivated are briefly reviewed.

The ILU(p) factorisation. To obtain the factors L and U, Watts [16] proposed to truncate a Gaussian elimination process. The elimination induces new elements, denoted as fill-in elements, in the L and U factors. That is, elements with indices not in NZ(A) (the set of indices for the non-zero elements of A). Truncation is done by ignoring some of these new elements, and the tool used for selecting which elements to discard is the notion of the *level* for an elements indices.

In Gaussian elimination, an element a_{ik} in the lower triangular part is zeroed by the product $l_{ik}u_{kk}$. This elimination will create fill-ins of the form $l_{ik}u_{kj}$ (see algorithm 1 below). Let indices in NZ(A) be of level zero and define the *level of fill-in* as

$$\mathrm{lev}(i,j) = \mathrm{lev}(i,k) + \mathrm{lev}(k,j) + 1. \tag{3}$$

when a new fill-in element is created or the level is previously set to a larger value.

Typically, for diagonal dominant matrices, the higher the level of fill-in of an element the smaller its magnitude and this suggests dropping any fill-in element whose level is higher than a certain integer p. However, for more general matrices the size of an element is not necessarily related to its level of fill-in. Stated below is an algorithm for truncated Gaussian elimination that is discarding elements with indices of level larger than p.

ALGORITHM 1 *Incomplete LU factorisation* – ILU(p)

Define $u_{ij} = a_{ij}$ and for all non-zero elements a_{ij} set $\mathrm{lev}(i,j) = 0$.
For $i = 2, \ldots, N$ do
  For each $k = 1, \ldots, i-1$ and if $u_{ik} \neq 0$ do
    Compute $l_{ik} = u_{ik}/u_{kk}$.
    Set $u_{ik} \leftarrow u_{ik} - l_{ik}u_{kk} \equiv 0$.
    For $j = k+1, \ldots, n$ and if $u_{kj} \neq 0$ do
      Compute $\mathrm{lev}(i,j)$ using (3).

if lev$(i,j) \leq p$ Compute $u_{ij} \leftarrow u_{ij} - l_{ik}u_{kj}$.
End For
End For
End For

One drawback of this algorithm is that the computational work for $p > 0$ is generally not predictable. But, most notable is the problems inherited from the Gaussian elimination for non diagonal-dominant matrices, with the possible growth of elements with p.

The SIP factorisation. SIP is developed for matrices appearing from the discretisation on a five point stencil (figure 1a) of a two-dimensional partial differential equation. The matrix A, as in equation (1), resulting from this discretisation, using natural ordering of the nodes, will be a block tridiagonal matrix (figure 1b). The elements of a diagonal is referenced by its matrix row number – that is, B_k denotes the element of the B-diagonal that is located on the k-th row of the matrix.

Firstly, look at the result of a multiplication of a lower and an upper triangular matrix containing only diagonals also present in A. The product will contain an additional two diagonals, C and G, placed just above and below diagonals B and H respectively. We may then talk about an altered matrix $A + S$ that is exactly factorisible into this sparse LU form,

$$A + S = LU, \tag{4}$$

where S is non-zero only on the new diagonals C and G. This altered matrix represent a set of equations where the left hand sides acquire the form

$$B_i\, x_{i-m} + D_i\, x_{i-1} + E_i\, x_i + F_i\, x_{i+1} + H_i\, x_{i+m} + C_i\, x_{i-m+1} + G_i\, x_{i+m-1}. \tag{5}$$

In order to reduce the influence of the new diagonals, and get an altered equation that is hopefully closer to the original equation, Stone [14] introduces the concept of *partial cancellation*. That is, the unknowns x_{i-m+1} and x_{i+m-1} are estimated by Taylor-series expansions in the variables appearing originally in the stencil (figure 1a),

$$x_{i+m-1} = -x_i + x_{i-1} + x_{i+m} \tag{6a}$$
$$x_{i-m+1} = -x_i + x_{i+1} + x_{i-m} \tag{6b}$$

and the result, weighted with a factor $\alpha \in [0, 1]$, is subtracted to give

$$\begin{aligned} &B_i\, x_{i-m} + D_i\, x_{i-1} + E_i\, x_i + F_i\, x_{i+1} + H_i\, x_{i+m} \\ &\quad + C_i\,[x_{i-m+1} - \alpha(x_{i-1} + x_{i+m} - x_i)] + G_i\,[x_{i+m-1} - \alpha(x_{i+1} + x_{i-m} - x_i)] \end{aligned} \tag{7}$$

for the left hand side of the equations.

It is straightforward, albeit somewhat tedious, to perform the matrix LU multiplication by hand and to compare the result with equation (7). However, doing this we convince ourselves that the elements of the L and U matrices must obey the following recursion rules.

$$b_i = B_i\,/(1 + \alpha\, f_{i-m})\,, \qquad C_i = b_i\, f_{i-m} \tag{8a}$$
$$d_i = D_i\,/(1 + \alpha\, h_{i-1})\,, \qquad G_i = d_i\, h_{i-1} \tag{8b}$$
$$e_i = E_i + \alpha\, C_i + \alpha\, G_i - b_i\, h_{i-m} - d_i\, f_{i-1} \tag{8c}$$
$$f_i = (F_i - \alpha\, C_i)\,/e_i \tag{8d}$$
$$h_i = (H_i - \alpha\, G_i)\,/e_i \tag{8e}$$

Again the indices refer to matrix row number, and the lower case letters refer to diagonals in the L and U factors with labelling corresponding to the upper case letters for A (figure 1b).

A further scrutination of the LU product will reveal that coefficients with index such that they fall outside the matrix (e.g. b_1) may be set to zero. Further, it is noted that due to the block structure of the matrix

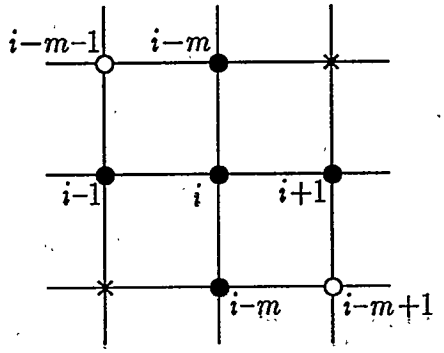


Figure 1a: •: Nodes in the five-point stencil. ○: Additional nodes introduced by the SIP factorisation.

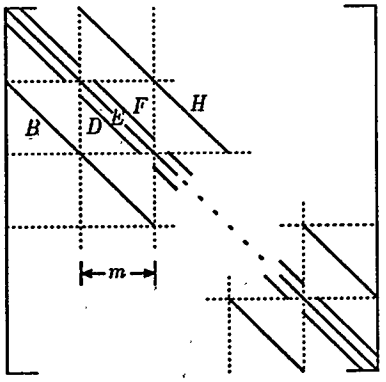


Figure 1b: Matrix representing the five-point stencil on a rectangular domain with $m \times n$. It has $n \times n$ blocks, each of size $m \times m$.

A some c_i, e_i, D_i, and F_i will be identical to zero. Having thus defined the matrix $M = LU$, the strongly implicit procedure uses M as the splitting matrix in the iteration $M\,\delta_k = b - A\,x_k, x_{k+1} = x_k + \delta_k$.

The generalised SIP-factorisation. To acquire an algorithm applicable to more general problems, and with arbitrary levels of fill-in, start out with ILU(p). For the LU factors produced from this algorithm, set up an altered matrix $A + S = LU$ in the same manner as for SIP in equations (4) and (5). The i-th altered equation will be on the form

$$(LU\,x)_i = \sum_{j \in \mathrm{NZ}(a_{i,*})} a_{ij}\,x_j + \sum_{\substack{j \in \mathrm{NZ}((LU)_{i,*}) \\ j \notin \mathrm{NZ}(a_{i,*})}} s_{ij}\,x_j. \tag{9}$$

In order to use equation (9) in an algorithm we need a means of locating the non-zero elements of S. This may be done by considering the fill-in levels calculated by the ILU algorithm 1. The process of finding the levels of the elements is often called a *symbolic cancellation* since it is not concerned with the values of the elements, but only with the placement and level of possibly non-zero elements. The following lemma is what we need.

LEMMA 1 *Apply the symbolic cancellation on the matrix $A \in \mathbb{R}^{n \times n}$, and assume the process not to break down. Then, for the matrix S as defined by equation (9), $s_{ij} \neq 0$ only if* $\mathrm{lev}(i,j)$ *is defined and found to be greater than p.*

Proof. If $\mathrm{lev}(i,j)$ is undefined this means that $l_{ik}u_{kj} = 0$ for any k and $(LU)_{ij} = l_{i,*} \cdot u_{*,j} = 0$. Further, if $\mathrm{lev}(i,j) \leq p$, $(LU)_{ij} = a_{ij}$ by the elimination. This leaves elements with index of level greater than p to be non-zero in S. □

Partial cancellation is now introduced into equation (9). That is, make an estimate $\hat{x}_j$ of all the x_j multiplying S, and subtract a weighted form of this estimate from x_j. This gives us

$$(LU\,x)_i = \sum_{j \in \mathrm{NZ}((a_{i,*}))} a_{ij}\,x_j + \sum_{\substack{j \in \mathrm{NZ}((LU)_{i,*}) \\ j \notin \mathrm{NZ}(a_{i,*})}} s_{ij}\,(x_j - \alpha_{ij}\,\hat{x}_j) \tag{10}$$

as the altered equations. In order to leave the sparsity patterns in L and U unchanged by the partial cancellation, the estimate $\hat{x}$ must be of the form

$$\hat{x}_j = \sum_h w_{jh} x_h \tag{11}$$

where the sum is over some or all indices in $\mathrm{NZ}((LU)_{i,*})$. That is, the indices j such that $\mathrm{lev}(i,j)$ is defined.

Most SIP-like methods use some form of Taylor expansion to calculate $\hat{x}$. In these methods α_{ij} is set to a constant α because there is seen no significant gain for the cost of complexity of a varying α. We will adopt this approach.

To construct our generalised algorithm we start out with algorithm 1. But, when the symbolic cancellation finds an element in S, we will correct the incomplete LU factorisation according to equation (10). However, due to equation (11), this may include elements of A that is already eliminated. This prevents us from just correcting the factorisation, and forces us to put up a set of equations to solve for the elements on row i of L and U.

For the elements of L and U, we see from the Gaussian elimination that if nothing else happen, they will satisfy the equations

$$u_{kk} l_{ik} = a_{ik} \tag{12}$$

and

$$u_{ij} = a_{ij}. \tag{13}$$

There will be one such equation for each l_{ik} or u_{ij} to calculate, and we number them by the second index on the right hand side. Since $k < i$, u_{kk} is already found by the algorithm.

The Gaussian elimination will introduce new terms into the equations. This is a result of the operation $u_{ij} \leftarrow u_{ij} - l_{ik} u_{kj}$, which will add $u_{kj} l_{ik}$ into the left hand side of equation (12) number j for $j < i$. For $j \geq i$, it will add the same into equation (13) number j.

As the process comes to an element with level greater than p, we halt the elimination and tend to inserting equation (11). We have $s_{ij} = l_{ik} u_{kj}$, and together with equation (11) we reformulate the first term in equations (10) to the form $\sum (a_{ih} - \alpha w_{jh} l_{ik} u_{kj})\, x_h$. That is, we add $\alpha w_{jh} l_{ik} u_{kj}$ into the left hand side of either equation (12) or (13) number h, depending on whether j belongs to the indices of non-zero L or non-zero U.

As the symbolic cancellation of row i is done, we end up with a system of equations on the form

$$\begin{bmatrix} P & 0 \\ Q & I \end{bmatrix} \begin{bmatrix} \tilde{l} \\ \tilde{u} \end{bmatrix} = \begin{bmatrix} \tilde{a}_l \\ \tilde{a}_u \end{bmatrix} \tag{14}$$

where $\tilde{l}$ and $\tilde{u}$ are vectors containing the unknown elements of L and U respectively. $\tilde{a}_l$ and $\tilde{a}_u$ are vectors containing the corresponding elements of A. P will be a square matrix. Both P and Q may be empty matrices. This happens if A contains no elements in the lower triangle on row i. Further, Q is non-empty only if P is non-empty. We may break system (14) into two parts,

$$P\tilde{l} = \tilde{a}_l \tag{15}$$

$$\tilde{u} = \tilde{a}_u - Q\tilde{l}. \tag{16}$$

P and Q are typically very small matrices for sparse A and small p. Gaussian elimination is therefore efficient to solve equation (15). If we set $\alpha = 0$ we get the system ILU(p) would produce, and this gives a lower triangular matrix P. For $\alpha \neq 0$ all elements of P may be non-zero. More typically, we then get a few additional elements, and may expect more zeroes in the upper triangle than in the lower triangle of P. A less costly solver for equation (15) can therefore be constructed by eliminating the upper triangle of P, contrary to the conventional elimination of the lower triangle.

Given that the algorithm does not break down, ILU(p) is applicable to any matrix with no regards to the geometry of the underlying problem. In GSIP it is just this knowledge of geometrical properties we try to exploit. When the matrix A is generated by some discretisation of a differential equation, there exists a grid

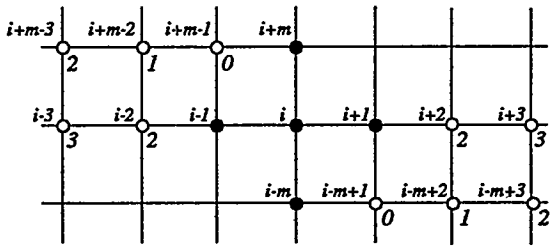


Figure 2: Fill-in elements for the five-point stencil on a uniform grid. The label up to the left of a node is it's index, and the number down to the right is the fill-in level for which it's coefficient in the altered equation is non-zero.

with information on how the points relate to each other. This we may use to calculate the approximation $\hat{x}$ by Taylor's theorem, or we may even use the differential equation for this purpose. We must however restrict ourselves to the form of equation (11) for $\hat{x}$.

For very general problems it may not be easy find a good estimate $\hat{x}$, but for the purpose of formulating a general procedure, assume we have managed to do this, and that we have a subroutine G to return the coefficients and indices for equation (11). We are now able to formulate a strongly implicit procedure for a general matrix.

ALGORITHM 2 *Generalised Strongly Implicit Procedure* – GSIP($p, \alpha,$ G)

Define $u_{ij} = a_{ij}$ and for all non-zero elements a_{ij} set $\text{lev}(i,j) = 0$.
For $i = 2, \ldots, N$ do
  For each $k = 1, \ldots, i-1$ and if $u_{ik} \neq 0$ do
    For each $j = k, \ldots, n$ and if $u_{kj} \neq 0$ do
      Compute $\text{lev}(i,j)$ by formula (3).
      If $\text{lev}(i,j) > p$ Call G to find equation (11).
      Else is this an ordinary Gaussian elimination step.
      Update equations (15) and (16).
    End For
  End For
  Solve equations (15) and (16) and update row i of L and U.
End For

With an appropriate choice of G on the five-point stencil, for $p = 0$ and $p = 1$, this algorithm produces the same preconditioning matrix $M = LU$ as SIP and modified strongly implicit procedure (MSI) [10] respectively. Further, with $\alpha = 0$ it gives the same result as ILU(p).

The difficulties discussed for ILU(p) is also present for GSIP($p, \alpha,$ G), since the lack of diagonal dominance still may give large elements in S. Moreover, the lack of diagonal dominance is often particularly pronounced for convection dominated flows where Taylor expansions may be poor.

As p increases the factors obtained by GSIP($p, \alpha,$ G) and ILU(p) will approach each other. The cost of calculating the factors is larger for GSIP($p, \alpha,$ G) than for ILU(p), and we might expect any superiority of GSIP to appear for low or moderate p.

Cancellation expressions for the five-point stencil. For a scheme based on the five-point stencil on a rectangular domain, the nodes to cancel will be positioned as depicted in figure 2. If we apply a first order Taylor expansion at node i, and estimate the differentials with forward differences, we end up with the following expressions for the cancellation (11),

$$x_{i+m-(1+p)} = -(p+1)\,x_i + (p+1)\,x_{i-1} + x_{i+m} \tag{17a}$$

$$x_{i-m+(1+p)} = -(p+1)\,x_i + (p+1)\,x_{i+1} + x_{i-m} \tag{17b}$$

$$x_{i-p} = (1-p)\,x_i + p\,x_{i-1}, \qquad p \geq 2 \tag{17c}$$

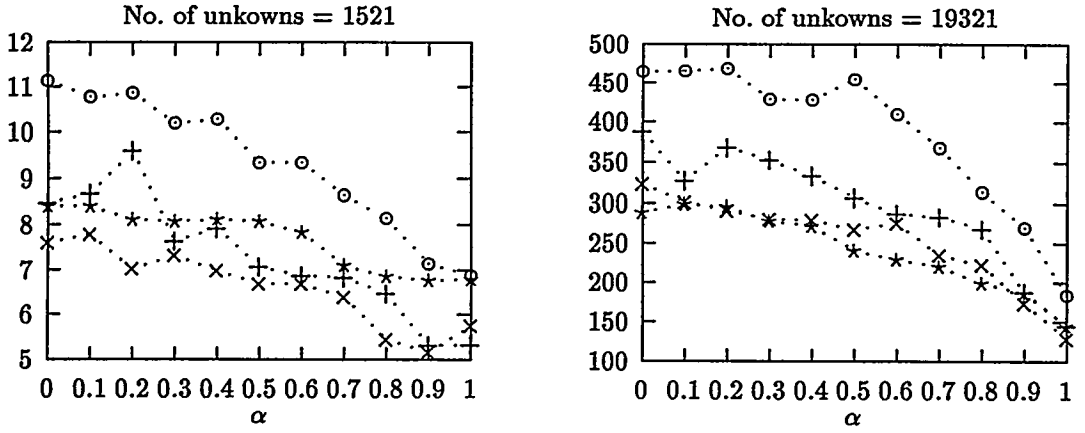


Figure 3: Run time in CPU-seconds for the Poisson equation in example 1, plotted as a function of the cancellation parameter α. Fill-in levels used was; $\circ$: $p = 0$, $+$: $p = 1$, $\times$: $p = 2$, and $\star$: $p = 3$.

Table 1: Run time in CPU-seconds for the Poisson equation in example 1. Optimal p values are marked with an asterisk.

	$N = 1521$		$N = 3481$		$N = 6241$		$N = 9801$		$N = 19321$	
$p \setminus \alpha$	0	1	0	1	0	1	0	1	0	1
0	11.1	6.9	34.5	20.4	88.7	49.1	171.8	72.8	464.7	184.3
1	8.5	5.3*	28.4	15.8	68.9	33.5	133.2	60.3	388.2	145.7
2	7.6*	5.8	24.9*	14.9*	57.9*	32.2*	117.5*	56.9*	323.7*	129.2*
3	8.4	6.8	26.1	18.4	59.0	37.8	139.1	64.0	289.0	144.9
4	9.9	8.9	29.5	23.4	62.7	45.8	113.2	77.3	288.5	178.1
5	11.8	11.1	32.6	28.3	70.8	56.5	121.0	95.1	294.4	206.4

N denotes number of unknowns in the equations.

$$x_{i+p} = (1-p)\,x_i + p\,x_{i+1}, \qquad p \geq 2 \tag{17d}$$

These expressions coincide with those used in SIP and MSI for $p = 0$ and $p = 1$ respectively. The Taylor-expansions (17) are valid for a rectangular grid. For classical SIP, the resulting factorisation (8) is however used with success on more general grids. We may however expect the convergence to deteriorate as the deviation from rectangularity gets large. A further weakness of (17) is that the approximation may be expected to be accurate for smooth solutions x only.

3 Numerical experiments

Three Krylov-subspace methods with right preconditioning was tested, GMRES(m) [9], CGS [13], and BiCGSTAB [15]. Of these, BiCGSTAB was the most effective in most runs, and the results presented here are calculated with BiCGSTAB. BiCGSTAB is known to have problems with matrices derived from convection dominated flow problems. It was therefore slightly modified, as suggested by Sleijpen and Van der Vorst [11].

The algorithms were implemented in Fortran 90, and the programs compiled and run on a DEC-station RS5000/250 with the NAGWare f90 compiler Version 2.0a(264). For all runs, the iteration was stopped when the residual of the unpreconditioned system was less that 10^{-9}, or the number of iterations exceeded 1000.

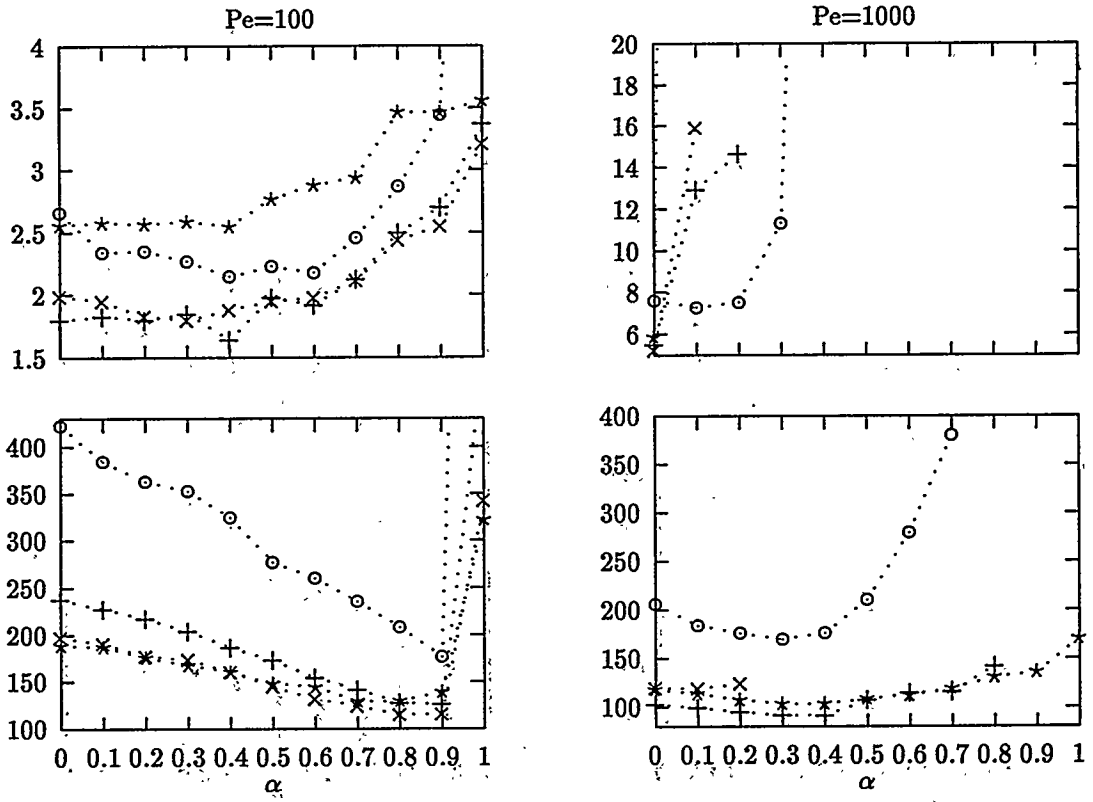


Figure 4: Run time in CPU-seconds for the convection-diffusion equation in example 2, plotted as a function of the cancellation parameter α. The top two boxes are for a matrix with 780 rows, and the bottom two boxes for a matrix of 19900 rows. If no value is plotted for an α, the iteration for this value either stagnated, or converged very slowly. Fill-in levels used was; o: $p = 0$, +: $p = 1$, ×: $p = 2$, ⋆: $p = 3$.

Example 1. This test case was proposed by Axelson and Polman [1]. The Poisson equation

$$-\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} = f(x, y)$$

is solved on the domain $(x, y) \in [0, 1] \times [0, 1]$, and the function f is selected for the solution to become $u(x, y) = e^{x\,y}\,(1 - x)\,x\,(1 - y)\,y$. Timings for this case are presented in table 1 and figure 3.

Example 2. This test case was suggested by Smith and Hutton [12] as a benchmark problem for numerical methods on convection-diffusion problems. The equation

$$u\frac{\partial \theta}{\partial x} + v\frac{\partial \theta}{\partial y} = \frac{1}{\mathrm{Pe}}\left(\frac{\partial^2 \theta}{\partial x^2} + \frac{\partial^2 \theta}{\partial y^2}\right)$$

is solved on the domain $(x, y) \in [-1, 1] \times [0, 1]$. The velocities are given as $u = 2\,y\,(1 - x^2)$ and $v = 2\,x\,(1 - y^2)$. That is, the boundary condition at the inlet $(-1 \le x \le 0, y = 0)$ is convected 180° to the outlet $(0 < x \le 1, y = 0)$.

At the inlet a Dirichlets boundary condition is defined by $\theta = 1 + \tanh[10\,(2\,x + 1)]$. This is profile is horizontal near $x = -1$ and $x = 0$, and with a steep gradient placed at $x = 0.5$. At the outlet, the the von Neumann condition $d\theta/dy = 0$ is used. On the remaining boundaries θ is set to $1 - \tanh 10$.

For high Pe, convection is the dominating force, and the inlet profile should be nearly unchanged (but mirrored about origo) at the outlet. For low Pe the profile is smeared out by diffusion. Timings for this case are presented in table 2 and figure 4.

Table 2: Run time in CPU-seconds for the convection-diffusion equation in example 2. Optimal α for each case is listed in parenthesis. Optimal p values are marked with an asterisk.

		$N = 780$		$N = 1\,770$		$N = 9\,730$		$N = 19\,900$	
Pe	p	$\alpha = 0$	opt. α	$\alpha = 0$	opt. α	$\alpha = 0$	opt. α	$\alpha = 0$	opt. α
100	0	2.7	2.1 (0.4)	7.8	5.6 (0.7)	124.0	62.5 (0.9)	422.1	176.1 (0.9)
	1	1.8*	1.6 (0.4)*	5.5	4.2 (0.6)*	74.8	44.3 (0.9)	237.3	126.0 (0.9)
	2	2.0	1.8 (0.3)	5.3*	4.6 (0.5)	66.0*	44.0 (0.8)*	197.3*	114.8 (0.8)*
	3	2.6	2.5 (0.4)	7.3	6.7 (0.4)	68.0	52.1 (0.7)	188.5	129.7 (0.7)
	4	3.4	3.4 (0.1)	9.7	9.1 (0.2)	81.0	64.9 (0.6)	199.4	155.3 (0.7)
	5	4.8	4.7 (0.2)	11.8	11.7 (0.2)	93.9	80.6 (0.5)	218.5	181.5 (0.6)
1000	0	7.6	7.3 (0.1)	19.3	14.8 (0.3)	91.8	70.0 (0.2)	205.8	169.3 (0.3)
	1	5.5	5.5 (0.0)	10.7*	10.7 (0.0)*	48.4*	44.5 (0.1)*	102.7*	90.9 (0.4)*
	2	5.2*	5.2 (0.0)*	12.6	12.6 (0.0)	56.4	56.4 (0.0)	119.8	118.9 (0.1)
	3	5.9	5.9 (0.0)	12.6	12.6 (0.0)	58.8	56.5 (0.3)	118.4	103.1 (0.4)
	4	5.9	5.9 (0.0)	13.8	13.8 (0.0)	72.7	72.7 (0.0)	150.2	149.6 (0.1)
	5	7.4	7.4 (0.0)	16.7	16.7 (0.0)	87.4	82.2 (0.3)	175.9	165.5 (0.4)

N denotes number of unknowns in the equations.

4 Discussion

In the calculations with the poisson equation, $\alpha = 1$ gave optimal run times for nearly all test runs (all exceptions are seen in left hand figure 3). For the convection-diffusion equation the situation is more complicated. In this case values of α near 1 always give very poor convergence. Poor convergence with α near 1 is also reported by others [6, 10, 14].

The convergence difficulties are particularly pronounced for the strong convection case (Pe = 1000). For the small matrices, and hence coarse grids, $\alpha = 0$ is the only sensible choice. Both the discretisation on a finer grid (larger matrix) and the lowering of the Peclet number Pe, make the matrix approach the Poisson equation matrix, and we see that the convergence bettered for GSIP in both cases (figure 4). For fine grid solutions and weak convection (Pe = 100) considerable gain in run time was achieved, however, optimal α is not easy to predict.

The three time consuming parts of the calculations, are the construction of the factors L and U, the forward and backward substitution, and the matrix–vector multiplication. The factorisation is performed only once, but each iteration of BiCGSTAB perform two calls to the substitution routine, and two calls to the multiplication routine.

The cost of factorisation and substitution increases with p, but the cost of multiplication stays the same. Even though, in all cases observed, an increase in p will reduce the number of iterations, the total run time may increase. An optimal fill-in level p is where the increased cost of preconditioning balances the reduced number of iterations.

Optimal p values for the weak convection equation and the Poisson equation follow the same pattern for matrices of comparable size. The strong convection case, however, behave differently. On large matrices, odd p seem to perform better than even p. E.g., for the largest matrix, $\alpha = 0.1$ and $\alpha = 0.4$ alternate on being optimal for even and odd $p > 0$ respectively. This made $p = 1$ 24% faster than $p = 2$. For the smaller matrices without this alternation, the difference between $p = 1$ and $p = 2$ was never this large.

To sum up the results, typically optimal parameters p and α, are 2 and 1.0 for the Poisson equation, 2 and 0.4–0.9 for the weak convection equation, and 1 and 0.0–0.4 for the strong convection equation. In a similar study, Joly and Eymard [4] found $p = 2$ to be optimal, using ILU(p) to precondition GMRES in reservoir simulations.

It remains to study other possible cancellation expressions, effects of more general grids, and effects of matrix ordering. Further testing on other commonly used discretisation than the central differences is also

needed. However, even though the ILU(p) ($\alpha = 0$) factorisation are more robust in terms of an all-round preconditioner, considerable speedup has been demonstrated with GSIP ($\alpha > 0$) on well behaved problems. This make GSIP an interesting alternative for preconditioning.

References

[1] O. AXELSSON AND B. POLMAN, *A robust preconditioner based on algebraic substructuring and two-level grids*, in Notes in Numerical Fluid Mechanics, Vieweg, 1989.

[2] K. B. BERG, *Numerical solution of the incompressible Navier-Stokes equations*, Dr. Ing. thesis, University of Trondheim, The Norwegian Institute of Technology, Department of Applied Mechanics, Thermo- and Fluiddynamics, 1994.

[3] D. A. H. JACOBS, *Preconditioned conjugate gradient algorithms for solving finite difference systems*, in Preconditioning methods : analysis and applications, D. J. Evans, ed., Gordon & Breach, New York, 1983.

[4] P. JOLY AND R. EYMARD, *Preconditioned biconjugate gradient methods for numerical reservoir simulation*, J. Comput. Phys., 91 (1990), pp. 298–309.

[5] P. K. KHOSLA AND S. G. RUBIN, *A conjugate gradient iterative method*, Comput. Fluids, 9 (1981), pp. 109–121.

[6] M. PERIC, *Efficient semi-implicit solving algorithm for nine-diagonal coefficient matrix*, Numer. Heat Transfer, 11 (1987), pp. 251–279.

[7] Y. SAAD, *Preconditioning techniques for nonsymmetric and indefinite linear systems*, J. Comput. Appl. Math., 24 (1988), pp. 89–105.

[8] ———, *Krylov subspace techniques, conjugate gradients, preconditioning and sparse matrix solver*, Lecture Series 1994–05, von Karman Institute for Fluid Dynamics, Mar. 1994.

[9] Y. SAAD AND M. H. SCHULTZ, *GMRES: a generarized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

[10] G. E. SCHNEIDER AND M. ZEDAN, *A modified strongly implicit procedure for the numerical solution of field problems*, Numer. Heat Transfer, 4 (1981), pp. 1–19.

[11] G. L. G. SLEIJPEN AND H. VAN DER VORST, *Maintaining convergence properties of BiCGstab methods in finite precision arithmetic.* Preprint nr. 861, Dept. Math., Universiteit Utrecht, 1994.

[12] R. M. SMITH AND A. G. HUTTON, *The numerical treatment of advection: a performance comparison of current methods*, Numer. Heat Transfer, 5 (1982), pp. 439–461.

[13] P. SONNEVELD, *a fast Lanczos-type solver for nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 36–52.

[14] H. L. STONE, *Iterative solution of implicit approximations of multidimensional partial differential equations*, SIAM J. Numer. Anal., 5 (1968), pp. 530–558.

[15] H. A. VAN DER VORST, *BI-CGSTAB: A fast and smoothly converging variant of BI-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 631–644.

[16] J. W. WATTS III, *A conjugate gradient-truncated direct method for the iterative solution of the reservoir simulation pressure equation*, Soc. Pet. Eng. J., 21 (1981), pp. 345–353.

[17] M. ZEDAN AND G. E. SCHNEIDER, *A coupled strongly implicit procedure for velocity and pressure computation in fluid flow problems*, Numer. Heat Transfer, 8 (1985), pp. 537–557.

Room TBA	***Workshop Chair***	
7:30 p.m.	***I. Duff***	Sparse Matrix Test Collections

Tuesday Evening's Workshop

Sparse Matrix Test Collections

Organizer: Iain Duff

20 February 1996

Abstract

This workshop will discuss plans for coordinating and developing sets of test matrices for the comparison and testing of sparse linear algebra software. We will talk of plans for the next release (Release 2) of the Harwell-Boeing Collection and recent work on improving the accessibility of this Collection and others through the World Wide Web. There will only be three talks of about 15 to 20 minutes followed by a discussion from the floor.

Speakers

- Iain Duff, Rutherford Appleton Laboratory
  Release II Plans for the Harwell-Boeing Collection

  We discuss the design and scope of Release 2 of the Harwell-Boeing Collection. In particular, we describe how we have extended the format to include more information on matrices and on systems being solved. These include starting guesses, exact solutions, eigenvalues and vectors, singular values and vectors, permutations and geometric data. We also discuss some of the data that will be added to the collection at this release and describe our proposed strategy for matrix generation subprograms.

- Roldan Pozo, NIST, Washington DC
  The Matrix Market: a visual web database for numerical matrix data

  The Matrix Market is introduced as a resource providing access to a repository of matrix test data for use in comparative studies of algorithms. The Matrix Market includes a variety of tools for browsing through the collection or for searching for matrices with special properties. Each matrix (and matrix set) has its own "home page" which details the matrix properties, provides a visualization of matrix structure, and permits downloading of the matrix in a choice of text file formats. Currently, some 300 matrices (including the entire Harwell-Boeing Sparse Matrix Collection, Release I), are available through the Matrix Market, and contributions to the database are being sought.

- Roldan Pozo, NIST, Washington DC
  The Matrix Market evolves: current directions

  As the Matrix Market moves beyond its prototype stage, we are examining several issues regarding its development: How can we facilitate/automate community contributions to the database? How can we structure the database to include matrix generator software in addition to actual matrix data? Should we provide a storage format conversion utility (perhaps using Java applets)? Can we reasonably provide more detailed information about each matrix, and if so, what information would be valuable to users? We seek feedback from participants on how the Matrix Market should evolve.

WEDNESDAY, APRIL 10TH

Topic: *FOSLS*	*Session Chair:* *Steve McCormick*	*Room A*
8:00 - 8:30	T. Manteuffel	First-Order System Least Squares (FOSLS) an Overview
8:30 - 9:00	S. McCormick	First-Order System Least Squares (FOSLS) for the Pure Traction Problem in Linear Elasticity
9:00 - 9:30	P. Bochev	Iterative Least-Squares Solvers for the Navier-Stokes Equations
9:30 - 10:00	G. Starke	Least-Squares Finite Element Discretizations of Neutron Transport Equations in 3 Dimensions

FOSLS
(First-Order Systems Least Squares)
An Overview

Thomas A. Manteuffel
Program in Applied Mathematics
University of Colorado at Boulder
tmanteuf@boulder.colorado.edu
http://amath-www.colorado.edu/appm/faculty/tmanteuf

Abstract

The process of modeling a physical system involves creating a mathematical model, forming a discrete approximation, and solving the resulting linear or nonlinear system. The mathematical model may take many forms. The particular form chosen may greatly influence the ease and accuracy with which it may be discretized as well as the properties of the resulting linear or nonlinear system. If a model is chosen incorrectly it may yield linear systems with undesirable properties such as nonsymmetry or indefiniteness. On the other hand, if the model is designed with the discretization process and numerical solution in mind, it may be possible to avoid these undesirable properties.

This talk will discuss a methodology for solving systems of partial differential equations. The methodology involves expanding the original system as a system of first-order equations by introducing new variables, adding extra constraints, and constructing a least-squares functional. If it can be shown that the least-squares functional is V-elliptic in a convenient norm, then Lax-Milgram theory guarantees that the minimization problem associated with the functional has a unique solution. In other words, the minimization problem is well posed in the V-norm.

In this context, discrete approximations to the minimum of the functional can be easily addressed through restricting the minimization to a finite element space in V. Cea's Lemma and interpolation theory now yield discretization error estimates.

Any basis for the finite element space leads to a symmetric positive definite linear system for the solution of the discrete minimization problem. If the basis has local support, then the condition of the system will be $O(h^{-2})$ because the functional involves only first-order differential operators. Moreover, if the functional can be shown to be V-elliptic in an H^1 product norm, then optimal multigrid performance is guaranteed.

This methodology has been applied to a variety of applications including advection-diffusion equations, transport of neutral particles, Helmholtz equations, Stokes and Navier-Stokes equations and linear elasticity. This talk will attempt to describe the basic methodology and results specific to some of the applications listed above.

FIRST-ORDER SYSTEM LEAST SQUARES FOR THE PURE TRACTION PROBLEM IN PLANAR LINEAR ELASTICITY

Z. CAI, T. MANTEUFFEL, S. MCCORMICK, AND S. PARTER

Abstract. This talk will develop two first-order system least squares (FOSLS) approaches for the solution of the pure traction problem in planar linear elasticity. Both are *two-stage* algorithms that first solve for the gradients of displacement, then for the displacement itself. One approach, which uses L^2 norms to define the FOSLS functional, is shown under certain H^2 regularity assumptions to admit *optimal H^1-like performance* for standard finite element discretization and standard multigrid solution methods that is *uniform in the Poisson ratio for all variables.* The second approach, which is based on H^{-1} norms, is shown under general assumptions to admit optimal uniform performance for displacement flux in an L^2 norm and for displacement in an H^1 norm. These methods do not degrade as other methods generally do when the material properties approach the incompressible limit.

Iterative Least-Squares Solvers for the Navier-Stokes equations.

P. Bochev

Department of Mathematics
University of Texas at Arlington
Box 19408, Arlington, TX 76019-0408
e-mail: bochev@utamat.uta.edu

1 Introduction

In the recent years finite element methods of least-squares type have attracted considerable attention from both mathematicians and engineers. This interest has been motivated, to a large extent, by several valuable analytic and computational properties of least-squares variational principles. In particular, finite element methods based on such principles circumvent Ladyzhenskaya-Babuska-Brezzi condition; see, e.g., [6], and lead to symmetric and positive definite algebraic systems. Thus, it is not surprising that numerical solution of fluid flow problems has been among the most promising and successful applications of least-squares methods; see, e.g., [1]-[4], and [8]-[10]. In this context least-squares methods offer significant theoretical and practical advantages in the algorithmic design, which makes resulting methods suitable, among other things, for large-scale numerical simulations; see, e.g., [8].

Least-squares approach represents a general methodology that can result in a variety of algorithms. To illustrate the main ingredients of this approach consider a boundary value problem of the form

$$\mathcal{L}(U) = F \quad \text{in } \Omega; \qquad \mathcal{R}(U) = \mathbf{0} \quad \text{on } \Gamma, \tag{1}$$

where $\mathcal{L}$ is linear, elliptic differential operator. An abstract least-squares principle for (1) is given by

$$\textit{Seek } U \in \mathbf{X}_q \textit{ such that } J(U) \leq J(V) \textit{ for all } V \in \mathbf{X}_q, \tag{2}$$

where J is a quadratic functional of the form

$$J(U) = \frac{1}{2}\|\mathcal{L}U - F\|^2_{\mathbf{Y}_q}, \tag{3}$$

and $\mathbf{X}_q$, $\mathbf{Y}_q$ are Hilbert spaces. Minimizers of $J(U)$ are subject to the Euler-Lagrange equation

$$\textit{Seek } U \in \mathbf{X}_q \textit{ such that } \mathcal{B}(U,V) = \mathcal{F}(V) \textit{ for all } V \in \mathbf{X}_q, \tag{4}$$

where $\mathcal{B}(U,V) = (\mathcal{L}U, \mathcal{L}V)_{\mathbf{Y}_q}$, and $\mathcal{F}(V) = (F, \mathcal{L}V)_{\mathbf{Y}_q}$. A least-squares finite element method can now be defined by choosing a discrete subspace $\mathbf{X}^h$ of $\mathbf{X}_q$, and minimizing the functional (3) over $\mathbf{X}^h$. The corresponding discrete variational problem is given by

$$\textit{Seek } U \in \mathbf{X}^h \textit{ such that } \mathcal{B}(U^h, V^h) = \mathcal{F}(V) \textit{ for all } V^h \in \mathbf{X}^h. \tag{5}$$

It is not difficult to see that (5) is an algebraic problem with symmetric and positive definite matrix. If, in addition, one can show an a priori estimate of the form

$$\|\mathcal{L}U\|_{\mathbf{Y}_q} \geq C\|U\|_{\mathbf{X}_q} \tag{6}$$

it follows that the bilinear form $\mathcal{B}(\cdot,\cdot)$ is coercive on $\mathbf{X}_q \times \mathbf{X}_q$, and by virtue of the inclusion $\mathbf{X}^h \subset \mathbf{X}_q$ it is also coercive on $\mathbf{X}^h \times \mathbf{X}^h$. As a result, existence and uniqueness of minimizers of both (4) and (5) follows by Lax-Milgramm lemma, and optimal error estimates can be established in a completely standard manner. Unfortunately, this straightforward approach does not always result in methods which are both optimal and easy to implement. It is now well-understood that the form of the operator $\mathcal{L}$, and the choice of the spaces $\mathbf{X}_q$ and $\mathbf{Y}_q$ are critical for the success of the least-squares methods. For example, if $\mathcal{L}$ is second (or higher) order elliptic operator, the method formally retains the theoretical virtues of the least-squares principles such as symmetry and positive definiteness of the discrete problems. However, the need to discretize a second (or higher) order differential operator prevents one from the use of standard C^0 finite element spaces in the method, and effectively renders the method impractical. A substantial progress in overcoming these difficulties has been achieved by transforming higher order elliptic problems to equivalent first-order systems. Thus, our discussion will focus on methods based on the first-order velocity flux-velocity-pressure form of the Navier-Stokes equations; see, e.g., [5], and [4].

2 First-order Navier-Stokes equations

Let Ω denote a bounded domain in $\mathbb{R}^n$, $n = 2, 3$, with Lipschitz continuous boundary Γ. The dimensionless equations governing the steady incompressible flow of a viscous fluid may be written in the form

$$\begin{aligned} -\nu \triangle \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u} + \nabla p &= \mathbf{f} \text{ in } \Omega & (7) \\ \nabla \cdot \mathbf{u} &= 0 \text{ in } \Omega & (8) \end{aligned}$$

where $\mathbf{u}$, p and $\mathbf{f}$ denote velocity, pressure and given body force, respectively, and ν is the inverse of the Reynolds number Re. To transform equations (7)-(8) into an equivalent first order form we introduce the *velocity flux* tensor

$$\underline{\mathbf{U}} = \nabla \mathbf{u}^t, \tag{9}$$

as a new dependent variable; see [5]. Then we consider the following system:

$$\begin{aligned} -\nu(\nabla^t \underline{\mathbf{U}})^t + \underline{\mathbf{U}}^t \mathbf{u} + \nabla p &= \mathbf{f} \text{ in } \Omega & (10) \\ \nabla^t \mathbf{u} &= 0 \text{ in } \Omega & (11) \\ \underline{\mathbf{U}} - \nabla \mathbf{u}^t &= \underline{\mathbf{0}} \text{ in } \Omega & (12) \\ \nabla(tr\,\underline{\mathbf{U}}) &= \mathbf{0} \text{ in } \Omega & (13) \\ \nabla \times \underline{\mathbf{U}} &= \underline{\mathbf{0}} \text{ in } \Omega & (14) \\ \mathbf{n} \times \underline{\mathbf{U}} &= \underline{\mathbf{0}} \text{ on } \Gamma & (15) \end{aligned}$$

Note that (13)-(14) and the boundary condition (15) are satisfied by virtue of definition (9).

3 Least-squares method

We consider minimization of the quadratic functional

$$\begin{aligned} J(\underline{\mathbf{U}}, \mathbf{u}, p) &= \| -(\nabla^t \underline{\mathbf{U}})^t + \frac{1}{\nu}(\underline{\mathbf{U}}^t \mathbf{u}) + \nabla p - \mathbf{f} \|_0^2 \\ &+ \| \nabla^t \mathbf{u} \|_0^2 + \| \underline{\mathbf{U}} - \nabla \mathbf{u}^t \|_0^2 + \| \nabla(tr\,\underline{\mathbf{U}}) \|_0^2 + \| \nabla \times \underline{\mathbf{U}} \|_0^2 \end{aligned} \tag{16}$$

over the space

$$\mathbf{X} = \{(\underline{\mathbf{U}}, \mathbf{u}, p) \in H^1(\Omega)^{n^2} \times H^1(\Omega)^n \times H^1(\Omega) \cap L_0^2(\Omega) \,\Big|\, \mathbf{u} = \mathbf{0}\,, \mathbf{n} \times \underline{\mathbf{U}} = \underline{\mathbf{0}} \;\text{ on } \Gamma\}\,. \tag{17}$$

Corresponding variational problem (Euler-Lagrange equation) is then given by

Find $(\underline{\mathbf{U}}, \mathbf{u}, p) \in \mathbf{X}$ *such that*

$$\begin{aligned}
&\mathcal{B}((\underline{\mathbf{U}}, \mathbf{u}, p), (\underline{\mathbf{V}}, \mathbf{v}, p)) \equiv \\
&\quad \left(-(\nabla^t \underline{\mathbf{U}})^t + \frac{1}{\nu}(\underline{\mathbf{U}}^t \mathbf{u}) + \nabla p - \mathbf{f}, -(\nabla^t \underline{\mathbf{V}})^t + \frac{1}{\nu}\left(\underline{\mathbf{U}}^t \mathbf{v} + \underline{\mathbf{V}}^t \mathbf{u}\right) + \nabla q\right)_0 + \\
&\quad \left(\nabla^t \mathbf{u}, \nabla^t \mathbf{v}\right)_0 + (\nabla(tr\underline{\mathbf{U}}), \nabla(tr\underline{\mathbf{V}}))_0 + \\
&\quad \left(\underline{\mathbf{U}} - \nabla \mathbf{u}^t, \underline{\mathbf{V}} - \nabla \mathbf{v}^t\right)_0 + (\nabla \times \underline{\mathbf{U}}, \nabla \times \underline{\mathbf{V}})_0 = 0
\end{aligned} \tag{18}$$

for all $(\underline{\mathbf{V}}, \mathbf{v}, q) \in \mathbf{X}$.

To discretize (18) we consider the following finite element space

$$\mathbf{X}_h = \{(\underline{\mathbf{U}}^h, \mathbf{u}^h, p^h) \in S_h^{n^2} \times S_h^n \times S_h \cap L_0^2(\Omega) \,\Big|\, \mathbf{u}^h = \mathbf{0}\,, \mathbf{n} \times \underline{\mathbf{U}}^h = \underline{\mathbf{0}} \;\text{ on } \Gamma\,,\}$$

where, for a given triangulation $\mathcal{T}_h$ of the domain Ω, S_h denotes the space of biquadratic finite element functions

$$S_h = \{u^h \in C^0(\Omega) \,|\, u^h|_\square \in Q_2(\square), \quad \square \in \mathcal{T}_h\}\,.$$

It is well-known, that the space S_h has the following property [7]: for $u \in H^3(\Omega)$ there exists an element $u^h \in S_h$ such that

$$\|u - u^h\|_r \leq C\, h^{3-r}\, \|u\|_3\,,\; r = 0, 1$$

Thus, according to the error estimates established in [4] we expect that for all sufficiently smooth solutions of the Navier-Stokes equations

$$\|\underline{\mathbf{U}} - \underline{\mathbf{U}}^h\|_1 + \|\mathbf{u} - \mathbf{u}^h\|_1 + \|p - p^h\|_1 \leq Ch^2\left(\|\underline{\mathbf{U}}\|_3 + \|\mathbf{u}\|_3 + \|p\|_3\right)\,. \tag{19}$$

4 Implementation

Let us formally write the nonlinear variational problem (18) as

$$F(Re, U) \equiv U + T \cdot G(Re, U) = 0\,,$$

where $U = (\underline{\mathbf{U}}, \mathbf{u}, p)$, T denotes a least-squares solution operator for the Stokes problem in velocity flux-velocity-pressure form, and $G(Re, U)$ denotes the nonlinear terms in (18); see [4]. Then, the discrete least-squares problem can be stated as

$$F^h(Re, U^h) \equiv U^h + T^h \cdot G(Re, U^h) = 0\,, \tag{20}$$

where T^h denotes a finite element approximation of T. Equation (20) is a nonlinear system of algebraic equations which must be solved in an iterative manner. For this purpose here we consider combined Newton-continuation method. The need to incorporate continuation in the method stems

from the following observations. Linearization of (20) at a given point U^h yields a linear algebraic system of the form

$$(I + T^h \cdot D_U G(Re, U^h)) \cdot \Delta U^h = -(U^h + T^h \cdot G(Re, U^h))$$

for the Newton increment ΔU^h. The matrix $(I + T^h \cdot D_U G(Re, U^h))$ in the above system is the Hessian matrix for the least-squares functional (16), and therefore, in a neighborhood of a minimizer it is necessarily symmetric and positive definite. Thus, the system for Newton's increment can be solved using Conjugate Gradients method *without* assembly of the discretization matrix. However, it is known that as the Reynolds number increases the size of the attraction ball for Newton's method decreases; see, e.g., [7]. As a result, for an arbitrary initial guess U^h the matrix $(I+T^h \cdot D_U G(Re, U^h))$ may fail to be positive definite, or the Newton's method may fail to converge. In order to guarantee that the initial approximation U^h is in the attraction ball we consider the following algorithm.

Let Re_M denote a target value of the Reynolds number, and let Re_0 denote a starting value, such that Newton's method will converge if initial approximation is taken to be a least-squares solution of the velocity flux-velocity-pressure Stokes problem. For example, we can take $Re_0 = 1$. Thus, we set

$$U^h_{0,0} = T^h \mathbf{f}\,.$$

Then, a sequence of Newton iterates $U^h_{k,m-1}$, for the current value of the Reynolds number Re_{m-1} is generated by solving recursively the linear system

$$(I + T^h \cdot D_U G(Re_{m-1}, U^h_{k-1,m-1})) \cdot \Delta U^h_{k,m-1} = -(U^h_{k-1,m-1} + T^h \cdot G(Re_{m-1}, U^h_{k-1,m-1}))\,.$$

After the approximate solution $U^h_{K,m-1}$ for Re_{m-1} is computed, an initial approximation $U^h_{0,m}$ for the next value Re_m of the Reynolds number can be obtained either by choosing $U^h_{0,m} = U^h_{K,m-1}$ (continuation along the constant), or by solving the system

$$D_U F(U^h_{K,m-1}; Re_{m-1}) \cdot \left(U^h_{0,m} - U_{K,m-1}\right) = -(Re_m - Re_{m-1}) D_{Re} F(U^h_{K,m-1}; Re_{m-1})\,;$$

(continuation along the tangent). This algorithm can be implemented in self-correcting manner, by noting that if $U^h_{0,m}$ is not in the attraction ball, then either the conjugate gradient method, or the Newton iteration will not converge. As a result, the above algorithm for the approximate numerical solution of the incompressible Navier-Stokes equations will encounter only symmetric and positive definite algebraic systems in the solution process.

5 Numerical examples

In this section we report computational experiments with the least-squares method. Our main goal is to investigate the validity of error estimates (19). Although analysis of [4] does not include error estimates in the norm of L^2, below we have also included the computed L^2 rates. In addition, we compare our results with computations performed using a least-squares method based on *velocity-vorticity-pressure* form of the Navier-Stokes equations; see [1], [3], and [8]-[9]. We consider two examples of artificial planar flows in the unit square, that is we start with known velocity and pressure fields and then compute the data by evaluating equations (10)-(14) at the exact solution.

Our examples are as follows:

$$\begin{aligned}
\textbf{Example 1.} \quad \mathbf{u} &= \exp(x)\cos(y) + \sin(y) \\
\mathbf{u}_2 &= -\exp(x)\sin(y) + (1 - x^3) \\
p &= \sin(y)\cos(x) + xy^2 - 1/6 - \sin(1)(1 - \cos(1)) \\
\textbf{Example 2.} \quad \mathbf{u}_1 &= \exp(\pi y)\left(x\cos(\pi x) - \sin(\pi x)/\pi\right)/2 \\
\mathbf{u}_2 &= \exp(\pi y) x \sin(\pi x)/2 \\
p &= \exp(\pi y) x \cos(\pi x)
\end{aligned}$$

Convergence rates for the velocity flux least-squares method, and Examples 1-2 are summarized in Tables 1-2, respectively. Convergence rates for a method based on first-order velocity-vorticity-pressure Navier-Stokes equations, and Example 2, are summarized in Table 3. We note that convergence rates in Tables 1-2 are in very good agreement with the optimal theoretical rates in (19). We also note that computed L^2 rates for both examples are optimal. Furthermore, we see that the rates for velocity-vorticity-pressure least-squares method appear to be suboptimal. This observation is in agreement with the theoretical results of [2], which indicate that for this first-order form an optimal least-squares method requires the use of weights in the functional.

Finally, on Figure 1 we compare number of iterations of diagonally preconditioned conjugate gradient method for Examples 1 and 2. Using this data we can estimate approximately condition numbers $O(h^{-\alpha})$ of the linearized problems according to the formula $\alpha \approx 2\log\left(\frac{m(i)}{m(j)}\right)/\log\left(\frac{i}{j}\right)$, where $m(i)$, $m(j)$ denote number of iterations for grid sizes i and j, respectively. In particular, we have found that for Examples 1 and 2 conditioning is $O(h^{-2.34})$, and $O(h^{-2.38})$, respectively.

Table 1. Convergence rates for Example 1 - *velocity flux-velocity-pressure* least-squares method

Error	L^2		H^1	
Variable	Computed	Optimal	Computed	Optimal
$\mathbf{u}$	3.00	**3.00**	1.99	**2.00**
$\underline{\mathbf{U}}$	3.10	**3.00**	2.01	**2.00**
p	3.51	**3.00**	2.25	**2.00**

Table 2. Convergence rates for Example 2 - *velocity flux-velocity-pressure* least-squares method

Error	L^2		H^1	
Variable	Computed	Optimal	Computed	Optimal
$\mathbf{u}$	3.05	**3.00**	1.99	**2.00**
$\underline{\mathbf{U}}$	3.10	**3.00**	2.02	**2.00**
p	3.50	**3.00**	2.25	**2.00**

Table 3. Convergence rates for Example 2 - *velocity-vorticity-pressure* least-squares method

Error	L^2		H^1	
Variable	Computed	Optimal	Computed	Optimal
$\mathbf{u}$	2.81	**3.00**	2.07	**2.00**
ω	2.42	**3.00**	1.75	**2.00**
p	2.30	**3.00**	1.75	**2.00**

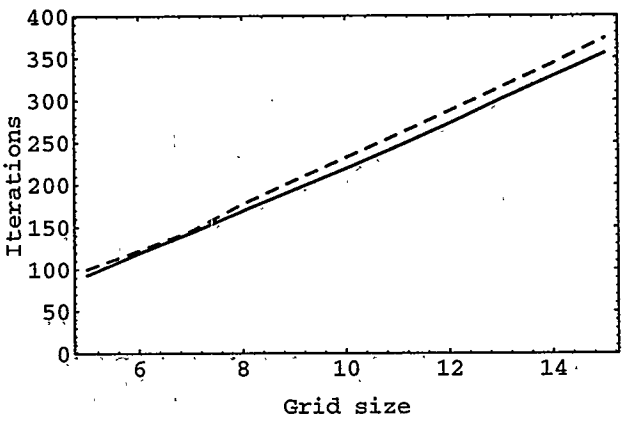


Figure 1: Average number of conjugate gradients iterations per Newton step for Example 1 (solid line) vs. Example 2 (dashed line)

References

[1] P. Bochev; Analysis of least-squares finite element methods for the Navier-Stokes equations, to appear in *SIAM J. Num. Anal.*

[2] P. Bochev and M.D. Gunzburger; Analysis of least squares finite element methods for the Stokes equations. *Math. Comp.* 63/108, 1994, pp. 479-506.

[3] P. Bochev and M. Gunzburger; Accuracy of least-squares methods for the Navier-Stokes equations, *Comput. & Fluids*, **22** 4/5, pp. 549-563, 1993.

[4] P. Bochev, Z.Cai, T.A. Manteuffel and S. F. McCormick; Analysis of velocity-flux least-squares principles for Navier-Stokes equations, submitted to *Math. Comp.*

[5] Z.Cai, T.A. Manteuffel and S. F. McCormick; First-order system least squares for the Stokes equations, to appear in *SIAM J. Num. Anal.*

[6] V. Girault and P.-A. Raviart; *Finite Element Methods for Navier-Stokes Equations*, Springer, Berlin, 1986.

[7] M. Gunzburger; *Finite Element Methods for Viscous Incompressible Flows*, Academic, Boston, 1989.

[8] B.-N. Jiang, T. Lin, and L. Povinelli; Large-scale computation of incompressible viscous flow by least-squares finite element method, *Comput. Methods Appl. Mech. Engrg.*, **114**, pp. 213-231, 1994.

[9] B.-N. Jiang and L. Povinelli; Least-squares finite element method for fluid dynamics, *Comput. Meth. Appl. Mech. Engrg.* **81**, pp. 13-37, 1990.

[10] D. Lefebvre, J. Peraire, and K. Morgan; Least-squares finite element solution of compressible and incompressible flows, *Int. J. Num. Meth. Heat Fluid Flow* **2**, pp. 99-113, 1992.

LEAST-SQUARES FINITE ELEMENT DISCRETIZATIONS OF NEUTRON TRANSPORT EQUATIONS IN 3 DIMENSIONS*

THOMAS A. MANTEUFFEL†, KLAUS J. RESSEL‡ AND GERHARD STARKE§

Abstract. The least-squares finite element framework to the neutron transport equation introduced in [2] is based on the minimization of a least-squares functional applied to the properly scaled neutron transport equation. Here we report on some practical aspects of this approach for neutron transport calculations in three space dimensions. The systems of partial differential equations resulting from a P_1 and P_2 approximation of the angular dependence are derived. In the diffusive limit, the system is essentially a Poisson equation for zeroth moment and has a divergence structure for the set of moments of order 1. One of the key features of the least-squares approach is that it produces a posteriori error bounds. We report on the numerical results obtained for the minimum of the least-squares functional augmented by an additional boundary term using trilinear finite elements on a uniform tesselation into cubes.

1. Introduction. We study least-squares finite element approximations of neutron transport equations in three dimensions. This approach was introduced and studied analytically in [2]. The least-squares finite element approach constitutes a general framework for constructing discretizations of transport problems that are accurate in diffusive regimes. This approach is based on a least-squares variational principle in combination with a scaling transformation. An important practical feature of the least-squares framework is that it automatically provides a posteriori bounds on the discretization error.

For simplicity, we consider the single group, steady state, isotropic form of the neutron transport equation given by

$$\begin{aligned} \Omega \cdot \nabla\psi + \sigma_t\psi - \sigma_s P\psi &= q \text{ for } (\mathbf{x},\Omega) \in \mathcal{R} \times S^2 , \\ \psi &= g \text{ for } \mathbf{x} \in \partial\mathcal{R} \text{ with } \mathbf{n}\cdot\Omega < 0 , \end{aligned} \tag{1}$$

where σ_t is the total cross section, σ_s is the scattering cross section and $\psi(\mathbf{x},\Omega)$ is the angular flux to be determined for all points $\mathbf{x} \in \mathcal{R} \subset \mathbf{R}^3$ and all possible travel directions Ω on the unit sphere S^2.

$$(P\psi)(\mathbf{x}) = \int_{S^2} \psi(\mathbf{x},\Omega)\, d\Omega .$$

For simplicity, we will assume vacuum boundary conditions $g = 0$. The angular dependence in the above equation is approximated by spherical harmonics up to order N, so-called P_N discretization. The boundary conditions will be imposed by adding a boundary functional. The P_N approximation to the neutron transport equation leads to a system of partial differential equations with special structure. The numerical results reported in this paper were obtained with the conjugate gradient method using only diagonal preconditioning. The design of an efficient multilevel approach for these special systems will be reported elsewhere.

* This work was sponsored by the National Science Foundation under grant number DMS-8704169 and Department of Energy, Applied Math Program Grant DE-FG03-94ER25217.

† Program in Applied Mathematics, Campus Box 526, University of Colorado at Boulder, Boulder, CO 80309-0526 (tmanteuf@boulder.colorado.edu)

‡ Interdisciplinary Project Center for Supercomputing (IPS), Clausiusstra"se 59, RZ F-11, ETHZ, CH-8092 Z"urich, Switzerland (kjr@scsc.ethz.ch)

§ Institut für Praktische Mathematik, Universität Karlsruhe, Englerstrasse 2, 76128 Karlsruhe, Germany (starke@math.uni-karlsruhe.de)

The purpose of this paper is to present the practical aspects of the least-squares finite element approach. The next section presents the basic aspects of the least-squares strategy. In Section 3, we discuss the use of the minimum of the least-squares functional as an appropriate error measure. Finally, in Section 4, some numerical results are presented.

2. The Least-Squares Finite Element Approach. It is convenient to derive the least-squares approach for (1) from the standard P_N approximation in the following way. Define the inner product

$$\langle \psi, \chi \rangle := \int_{\mathcal{R}} \int_{S^2} \psi(\mathbf{x}, \Omega) \overline{\chi(\mathbf{x}, \Omega)} \, d\Omega d\mathbf{x}$$

and the associated norm $||\cdot||$. If we represent angular dependence of $\psi(\mathbf{x}, \Omega)$ in terms of a series in spherical harmonics, the so-called P_N discretization, we obtain a system of partial differential equations coupling certain moments. A Galerkin approximation by spherical harmonics of order 1 using

$$\psi(\mathbf{x}, \Omega) = \Phi_{0,0}(\mathbf{x}) + \Phi_{1,0}(\mathbf{x}) Y_1^0(\Omega) + \Phi_{1,1}(\mathbf{x}) Y_1^1(\Omega) + \Phi_{1,-1}(\mathbf{x}) Y_1^{-1}(\Omega) \tag{2}$$

gives

$$\begin{bmatrix} \sigma_a & \frac{1}{\sqrt{3}}\partial_3 & -\frac{\partial_1 + i\partial_2}{\sqrt{6}} & \frac{\partial_1 - i\partial_2}{\sqrt{6}} \\ \frac{1}{\sqrt{3}}\partial_3 & \sigma_t & 0 & 0 \\ \frac{-\partial_1 + i\partial_2}{\sqrt{6}} & 0 & \sigma_t & 0 \\ \frac{\partial_1 + i\partial_2}{\sqrt{6}} & 0 & 0 & \sigma_t \end{bmatrix} \begin{bmatrix} \Phi_{0,0} \\ \Phi_{1,0} \\ \Phi_{1,1} \\ \Phi_{1,-1} \end{bmatrix} = \begin{bmatrix} q_{0,0} \\ q_{1,0} \\ q_{1,1} \\ q_{1,-1} \end{bmatrix}.$$

(We normalize the spherical harmonics as in [1, Appendix A].) In order to avoid complex arithmetic, it is convenient to rewrite this system in terms of odd and even spherical harmonics

$$\tilde{Y}_1^1 = \frac{1}{\sqrt{2}}(Y_1^{-1} - Y_1^1), \; \tilde{Y}_1^{-1} = \frac{i}{\sqrt{2}}(Y_1^1 + Y_1^{-1})$$

which implies

$$\tilde{\Phi}_{1,1} = \frac{1}{\sqrt{2}}(\Phi_{1,-1} + \Phi_{1,1}), \; \tilde{\Phi}_{1,-1} = -\frac{i}{\sqrt{2}}(\Phi_{1,1} - \Phi_{1,-1})$$

leading to

$$\begin{bmatrix} \sigma_a & \frac{1}{\sqrt{3}}\partial_3 & \frac{1}{\sqrt{3}}\partial_1 & \frac{1}{\sqrt{3}}\partial_2 \\ \frac{1}{\sqrt{3}}\partial_3 & \sigma_t & 0 & 0 \\ \frac{1}{\sqrt{3}}\partial_1 & 0 & \sigma_t & 0 \\ \frac{1}{\sqrt{3}}\partial_2 & 0 & 0 & \sigma_t \end{bmatrix} \begin{bmatrix} \Phi_{0,0} \\ \Phi_{1,0} \\ \tilde{\Phi}_{1,1} \\ \tilde{\Phi}_{1,-1} \end{bmatrix} = \begin{bmatrix} q_{0,0} \\ q_{1,0} \\ \tilde{q}_{1,1} \\ \tilde{q}_{1,-1} \end{bmatrix}. \tag{3}$$

Note that eliminating the moments of order 1 from (3) leads to the neutron diffusion equation

$$-\frac{1}{3\sigma_t\sigma_a}(\partial_1^2 + \partial_2^2 + \partial_3^2)\Phi_{0,0} + \Phi_{0,0} = \frac{1}{\sigma_a} q_{0,0} + \frac{1}{\sigma_a \sigma_t}(q_{1,0} + \tilde{q}_{1,1} + \tilde{q}_{1,-1}) \tag{4}$$

for $\Phi_{0,0}$. In the diffusive limit, i.e., for $\sigma_t = \frac{1}{\varepsilon}, \sigma_a = \varepsilon\alpha, q_{0,0} = O(\varepsilon), q_{1,m} = O(\varepsilon^2)$, an asymptotic expansion shows that $\psi(\mathbf{x}, \Omega) = \Phi_{0,0}(\mathbf{x}) + O(\varepsilon)$, where $\Phi_{0,0}$ satisfies (4)

with boundary conditions $\Phi_{0,0} = 0$ on $\partial\mathcal{R}$ (cf. [3]). It is important that the discrete solutions of the neutron transport equations also tend to an approximation of the neutron diffusion equation in the diffusive limit.

For the derivation of the least-squares approach, let us truncate the expansion of ψ after spherical harmonics of order 1 and test (1) against *all* spherical harmonics (of arbitrary degree). The basic recurrence relations for the spherical harmonics and their orthogonality imply

$$\langle \psi, Y_l^m \rangle = 0 \text{ for } l \geq 3 \text{ (and } -l \leq m \leq l) .$$

It is therefore sufficient to test against spherical harmonics of degree 2. As above, we use a basis of odd and even spherical harmonics,

$$\tilde{Y}_l^m = \frac{i^{m-1}}{\sqrt{2}}(Y_l^{-m} - Y_l^m) , \; \tilde{Y}_l^{-m} = \frac{i^m}{\sqrt{2}}(Y_l^m + Y_l^{-m}) , m = 1, \ldots, m$$

and end up with

$$\begin{bmatrix} \sigma_a I & \frac{1}{\sqrt{3}}\partial_3 & \frac{1}{\sqrt{3}}\partial_1 & \frac{1}{\sqrt{3}}\partial_2 \\ \frac{1}{\sqrt{3}}\partial_3 & \sigma_t I & 0 & 0 \\ \frac{1}{\sqrt{3}}\partial_1 & 0 & \sigma_t I & 0 \\ \frac{1}{\sqrt{3}}\partial_2 & 0 & 0 & \sigma_t I \\ 0 & \frac{2}{\sqrt{15}}\partial_3 & -\frac{1}{\sqrt{15}}\partial_1 & -\frac{1}{\sqrt{15}}\partial_2 \\ 0 & \frac{1}{\sqrt{5}}\partial_1 & \frac{1}{\sqrt{5}}\partial_3 & 0 \\ 0 & \frac{1}{\sqrt{5}}\partial_2 & 0 & \frac{1}{\sqrt{5}}\partial_3 \\ 0 & 0 & \frac{1}{\sqrt{5}}\partial_2 & \frac{1}{\sqrt{5}}\partial_1 \\ 0 & 0 & -\frac{1}{\sqrt{5}}\partial_1 & \frac{1}{\sqrt{5}}\partial_2 \end{bmatrix} \begin{bmatrix} \Phi_{0,0} \\ \Phi_{1,0} \\ \tilde{\Phi}_{1,1} \\ \tilde{\Phi}_{1,-1} \end{bmatrix} = \begin{bmatrix} q_{0,0} \\ q_{1,0} \\ \tilde{q}_{1,1} \\ \tilde{q}_{1,-1} \\ q_{2,0} \\ \tilde{q}_{2,1} \\ \tilde{q}_{2,-1} \\ \tilde{q}_{2,2} \\ \tilde{q}_{2,-2} \end{bmatrix} .$$

Appropriate scaling of this system gives

$$\begin{bmatrix} I & \frac{1}{\sqrt{3\sigma_a}}\partial_3\frac{1}{\sqrt{\sigma_t}} & \frac{1}{\sqrt{3\sigma_a}}\partial_1\frac{1}{\sqrt{\sigma_t}} & \frac{1}{\sqrt{3\sigma_a}}\partial_2\frac{1}{\sqrt{\sigma_t}} \\ \frac{1}{\sqrt{3\sigma_a}}\partial_3\frac{1}{\sqrt{\sigma_t}} & I & 0 & 0 \\ \frac{1}{\sqrt{3\sigma_a}}\partial_1\frac{1}{\sqrt{\sigma_t}} & 0 & I & 0 \\ \frac{1}{\sqrt{3\sigma_a}}\partial_2\frac{1}{\sqrt{\sigma_t}} & 0 & 0 & I \\ 0 & \frac{2}{\sqrt{15\sigma_t}}\partial_3\frac{1}{\sqrt{\sigma_t}} & -\frac{1}{\sqrt{15\sigma_t}}\partial_1\frac{1}{\sqrt{\sigma_t}} & -\frac{1}{\sqrt{15\sigma_t}}\partial_2\frac{1}{\sqrt{\sigma_t}} \\ 0 & \frac{1}{\sqrt{5\sigma_t}}\partial_1\frac{1}{\sqrt{\sigma_t}} & \frac{1}{\sqrt{5\sigma_t}}\partial_3\frac{1}{\sqrt{\sigma_t}} & 0 \\ 0 & \frac{1}{\sqrt{5\sigma_t}}\partial_2\frac{1}{\sqrt{\sigma_t}} & 0 & \frac{1}{\sqrt{5\sigma_t}}\partial_3\frac{1}{\sqrt{\sigma_t}} \\ 0 & 0 & \frac{1}{\sqrt{5\sigma_t}}\partial_2\frac{1}{\sqrt{\sigma_t}} & \frac{1}{\sqrt{5\sigma_t}}\partial_1\frac{1}{\sqrt{\sigma_t}} \\ 0 & 0 & -\frac{1}{\sqrt{5\sigma_t}}\partial_1\frac{1}{\sqrt{\sigma_t}} & \frac{1}{\sqrt{5\sigma_t}}\partial_2\frac{1}{\sqrt{\sigma_t}} \end{bmatrix} \begin{bmatrix} \hat{\Phi}_{0,0} \\ \hat{\Phi}_{1,0} \\ \hat{\Phi}_{1,1} \\ \hat{\Phi}_{1,-1} \end{bmatrix} = \begin{bmatrix} \hat{q}_{0,0} \\ \hat{q}_{1,0} \\ \hat{q}_{1,1} \\ \hat{q}_{1,-1} \\ \hat{q}_{2,0} \\ \hat{q}_{2,1} \\ \hat{q}_{2,-1} \\ \hat{q}_{2,2} \\ \hat{q}_{2,-2} \end{bmatrix}$$

with

$$\begin{bmatrix} \hat{\Phi}_{0,0} \\ \hat{\Phi}_{1,0} \\ \hat{\Phi}_{1,1} \\ \hat{\Phi}_{1,-1} \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{\sigma_a}}\Phi_{0,0} \\ \frac{1}{\sqrt{\sigma_t}}\Phi_{1,0} \\ \frac{1}{\sqrt{\sigma_t}}\tilde{\Phi}_{1,1} \\ \frac{1}{\sqrt{\sigma_t}}\tilde{\Phi}_{1,-1} \end{bmatrix} , \begin{bmatrix} \hat{q}_{0,0} \\ \hat{q}_{1,0} \\ \hat{q}_{1,1} \\ \hat{q}_{1,-1} \\ \hat{q}_{2,0} \\ \hat{q}_{2,1} \\ \hat{q}_{2,-1} \\ \hat{q}_{2,2} \\ \hat{q}_{2,-2} \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{\sigma_a}}q_{0,0} \\ \frac{1}{\sqrt{\sigma_t}}q_{1,0} \\ \frac{1}{\sqrt{\sigma_t}}\tilde{q}_{1,1} \\ \frac{1}{\sqrt{\sigma_t}}\tilde{q}_{1,-1} \\ \frac{1}{\sqrt{\sigma_t}}q_{2,0} \\ \frac{1}{\sqrt{\sigma_t}}\tilde{q}_{2,1} \\ \frac{1}{\sqrt{\sigma_t}}\tilde{q}_{2,-1} \\ \frac{1}{\sqrt{\sigma_t}}\tilde{q}_{2,2} \\ \frac{1}{\sqrt{\sigma_t}}\tilde{q}_{2,-2} \end{bmatrix} .$$

If we denote this system by $\mathcal{L}_1\Phi = \mathbf{q}$, then the resulting least-squares system is $\mathcal{L}_1^*\mathcal{L}_1\Phi = \mathcal{L}_1^*\mathbf{q}$. Note that the approach outlined above is equivalent to minimizing

$$||\mathcal{S}^{-1/2}(\Omega\cdot\nabla\psi + \sigma_t\psi - \sigma_s P\psi - q)||^2$$

among all ψ of the form (2) and setting $\psi = \mathcal{S}^{-1/2}\hat{\psi}$, where $\mathcal{S} = \sigma_a P + \sigma_t(I-P)$ (cf. [2]). In the constant coefficient case, for the nodes in the interior of the domain, the operator $\mathcal{L}_1^*\mathcal{L}_1$ has the following form:

$$\begin{bmatrix} I - \frac{\Delta}{3\sigma_a\sigma_t} & 0 & 0 & 0 \\ 0 & I - \frac{\Delta}{5\sigma_t^2} - (\eta+\hat{\eta})\partial_3^2 & -\eta\partial_3\partial_1 - \hat{\eta}\partial_1\partial_3 & -\eta\partial_3\partial_2 - \hat{\eta}\partial_2\partial_3 \\ 0 & -\eta\partial_1\partial_3 - \hat{\eta}\partial_3\partial_1 & I - \frac{\Delta}{5\sigma_t^2} - (\eta+\hat{\eta})\partial_1^2 & -\eta\partial_1\partial_2 - \hat{\eta}\partial_2\partial_1 \\ 0 & -\eta\partial_2\partial_3 - \hat{\eta}\partial_3\partial_2 & -\eta\partial_2\partial_1 - \hat{\eta}\partial_1\partial_2 & I - \frac{\Delta}{5\sigma_t^2} - (\eta+\hat{\eta})\partial_2^2 \end{bmatrix},$$

where

$$\eta = \frac{1}{3\sigma_a\sigma_t} - \frac{2}{15\sigma_t^2}, \quad \hat{\eta} = \frac{1}{5\sigma_t^2}$$

and $\Delta = \partial_1^2 + \partial_2^2 + \partial_3^2$. Note that $\eta = O(1)$ and $\hat{\eta} = O(\varepsilon^2)$ in the diffusive limit leading to the system

$$\begin{bmatrix} I & 0 & 0 & 0 \\ 0 & I & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \end{bmatrix} - \frac{1}{3\sigma_a\sigma_t}\begin{bmatrix} \Delta & 0 & 0 & 0 \\ 0 & \partial_3^2 & \partial_3\partial_1 & \partial_3\partial_2 \\ 0 & \partial_1\partial_3 & \partial_1^2 & \partial_1\partial_2 \\ 0 & \partial_2\partial_3 & \partial_2\partial_1 & \partial_2^2 \end{bmatrix}. \tag{5}$$

Similarly, using a truncated expansion up to spherical harmonics of order 2 leads to a system with the operator

$$\mathcal{L}_2 = \left[\begin{array}{c|ccccc} & 0 & 0 & 0 & 0 & 0 \\ & \frac{2}{\sqrt{15}\sigma_t}\partial_3 & \frac{1}{\sqrt{5}\sigma_t}\partial_1 & \frac{1}{\sqrt{5}\sigma_t}\partial_2 & 0 & 0 \\ & -\frac{1}{\sqrt{15}\sigma_t}\partial_1 & \frac{1}{\sqrt{5}\sigma_t}\partial_3 & 0 & \frac{1}{\sqrt{5}\sigma_t}\partial_2 & -\frac{1}{\sqrt{5}\sigma_t}\partial_1 \\ & -\frac{1}{\sqrt{15}\sigma_t}\partial_2 & 0 & \frac{1}{\sqrt{5}\sigma_t}\partial_3 & \frac{1}{\sqrt{5}\sigma_t}\partial_1 & \frac{1}{\sqrt{5}\sigma_t}\partial_2 \\ \mathcal{L}_1 & I & 0 & 0 & 0 & 0 \\ & 0 & I & 0 & 0 & 0 \\ & 0 & 0 & I & 0 & 0 \\ & 0 & 0 & 0 & I & 0 \\ & 0 & 0 & 0 & 0 & I \\ \hline & \frac{3}{\sqrt{35}\sigma_t}\partial_3 & -\frac{\sqrt{3}}{\sqrt{35}\sigma_t}\partial_1 & -\frac{\sqrt{3}}{\sqrt{35}\sigma_t}\partial_2 & 0 & 0 \\ & \frac{\sqrt{6}}{\sqrt{35}\sigma_t}\partial_1 & \frac{4}{\sqrt{70}\sigma_t}\partial_3 & 0 & -\frac{1}{\sqrt{70}\sigma_t}\partial_2 & \frac{1}{\sqrt{70}\sigma_t}\partial_1 \\ & \frac{\sqrt{6}}{\sqrt{35}\sigma_t}\partial_2 & 0 & \frac{4}{\sqrt{70}\sigma_t}\partial_3 & -\frac{1}{\sqrt{70}\sigma_t}\partial_1 & -\frac{1}{\sqrt{70}\sigma_t}\partial_2 \\ 0 & 0 & \frac{1}{\sqrt{7}\sigma_t}\partial_2 & \frac{1}{\sqrt{7}\sigma_t}\partial_1 & \frac{1}{\sqrt{7}\sigma_t}\partial_3 & 0 \\ & 0 & -\frac{1}{\sqrt{7}\sigma_t}\partial_1 & \frac{1}{\sqrt{7}\sigma_t}\partial_2 & 0 & \frac{1}{\sqrt{7}\sigma_t}\partial_3 \\ & 0 & 0 & 0 & \frac{\sqrt{3}}{\sqrt{14}\sigma_t}\partial_2 & \frac{\sqrt{3}}{\sqrt{14}\sigma_t}\partial_1 \\ & 0 & 0 & 0 & -\frac{\sqrt{3}}{\sqrt{14}\sigma_t}\partial_1 & \frac{\sqrt{3}}{\sqrt{14}\sigma_t}\partial_2 \end{array}\right].$$

The operator associated with a P_2 least-squares discretization can be obtained formally as $\mathcal{L}_2^*\mathcal{L}_2$.

An inherent problem of any P_N approach to the neutron transport equation is the handling of the boundary conditions in (1). To satisfy the vacuum boundary conditions exactly by the finite element spaces for the moments would imply homogeneous Dirichlet boundary conditions for all the moments on the entire boundary. Clearly, this is not physically meaningful, in general.

One way to treat the boundary conditions is by introducing an additional least-squares functional leading to the problem of minimizing

$$G(\psi;q) = \|\mathcal{S}^{-1/2}(\Omega\cdot\nabla\psi + \sigma_t\psi - \sigma_s P\psi - q)\|^2 \tag{6}$$

$$+2\int_{\partial\mathcal{R}}\int_{\{\Omega\in S^2:\mathbf{n}\cdot\Omega<0\}} |\mathbf{n}\cdot\Omega|(\psi(\mathbf{x},\Omega))^2 d\Omega ds\,.$$

3. A Posteriori Error Measure. One of the key features of the least-squares approach is that it automatically provides a bound on the discretization error in the course of approximately minimizing the functional. This feature depends on the ellipticity of the associated bilinear form with respect to a certain norm which we derive in the sequel.

For any fixed $\Omega\in S^2$, the divergence theorem gives

$$(\Omega\cdot\nabla\psi,\psi)_{L^2(\mathcal{R})} = (\nabla\cdot(\Omega\psi),\psi)_{L^2(\mathcal{R})} = -(\Omega\cdot\nabla\psi,\psi)_{L^2(\mathcal{R})} + \int_{\partial\mathcal{R}}(\mathbf{n}\cdot\Omega)\psi^2\,ds$$

and therefore

$$2\langle\Omega\cdot\nabla\psi\rangle = \int_{\partial R}\int_{S^2}(\mathbf{n}\cdot\Omega)\psi^2\,d\Omega ds\,.$$

For the functional in (6), this implies

$$G(\psi^h;q) = \|\mathcal{S}^{-1/2}\Omega\cdot\nabla(\psi^h-\psi) + \mathcal{S}^{1/2}(\psi^h-\psi)\|^2$$

$$+2\int_{\partial\mathcal{R}}\int_{\{\Omega\in S^2:\mathbf{n}\cdot\Omega<0\}} |\mathbf{n}\cdot\Omega|(\psi^h-\psi)^2 d\Omega ds$$

$$\geq \|\mathcal{S}^{-1/2}\Omega\cdot\nabla(\psi^h-\psi)\|^2 + \|\mathcal{S}^{1/2}(\psi^h-\psi)\|^2 + \int_{\partial\mathcal{R}}\int_{S^2}|\mathbf{n}\cdot\Omega|(\psi^h-\psi)^2\,d\Omega ds\,.$$

This means that $G(\psi^h;q)$, where ψ^h is any finite element approximation, serves as an upper bound for the error measured in the norm given by

$$|||\chi||| = \left(\|\mathcal{S}^{-1/2}\Omega\cdot\nabla\chi\|^2 + \|\mathcal{S}^{1/2}\chi\|^2 + \int_{\partial\mathcal{R}}\int_{S^2}|\mathbf{n}\cdot\Omega|\chi^2\,d\Omega ds\right)^{1/2}.$$

4. Computational Experiments. We solve the neutron transport problem (1) on $\mathcal{R}=(0,1)^3$ with an isotropic source function $q(\mathbf{x})$ which is trilinear on a tesselation into squares of side length 0.25 such that q is one at $(\frac{1}{2},\frac{1}{2},\frac{1}{2})$ and zero at the other nodes. Tables 1 and 2 show the relative error, measured in terms of the least-squares functional (6), with $\sigma_a=\varepsilon$ and $\sigma_t=1/\varepsilon$ for $\varepsilon=0.1$ and 0.01, respectively. In brackets, we have listed the number of conjugate gradient iterations using diagonal

TABLE 1
Minimum values and iteration counts for the least-squares functional: $\epsilon = 0.1$

h	1/4	1/8	1/16
P_1	$1.49 \cdot 10^{-1}$ (15)	$5.63 \cdot 10^{-2}$ (23)	$2.32 \cdot 10^{-2}$ (37)
P_2	$1.45 \cdot 10^{-1}$ (19)	$5.11 \cdot 10^{-2}$ (25)	$1.76 \cdot 10^{-2}$ (40)

TABLE 2
Minimum values and iteration counts for the least-squares functional: $\epsilon = 0.01$

h	1/4	1/8	1/16
P_1	$1.16 \cdot 10^{-1}$ (25)	$4.23 \cdot 10^{-2}$ (48)	$1.40 \cdot 10^{-2}$ (57)
P_2	$1.16 \cdot 10^{-1}$ (36)	$4.23 \cdot 10^{-2}$ (52)	$1.39 \cdot 10^{-2}$ (59)

scaling required to reduce the initial residual associated with the discrete system by a factor of 10^{-3}.

Obviously, the number of conjugate gradient iterations grows quite dramatically for decreasing h and is also dependent on σ_t. The use of a standard multigrid approach to this system does not give convergence rates independent of h for large σ_t due to the special structure of (5). The design of an efficient multigrid approach is in progress.

Figure 1 shows cross-sections at $x_3 = 0.5$ of the moments $\hat{\Phi}_{lm}$ obtained with the least-squares approach using a P_2 approximation in angle and trilinear finite elements on a 17 by 17 by 17 mesh. The moments which are not shown, namely, $\hat{\Phi}_{10}$, $\hat{\Phi}_{2,1}$, $\hat{\Phi}_{2,-1}$, vanish for $x_3 = 0.5$.

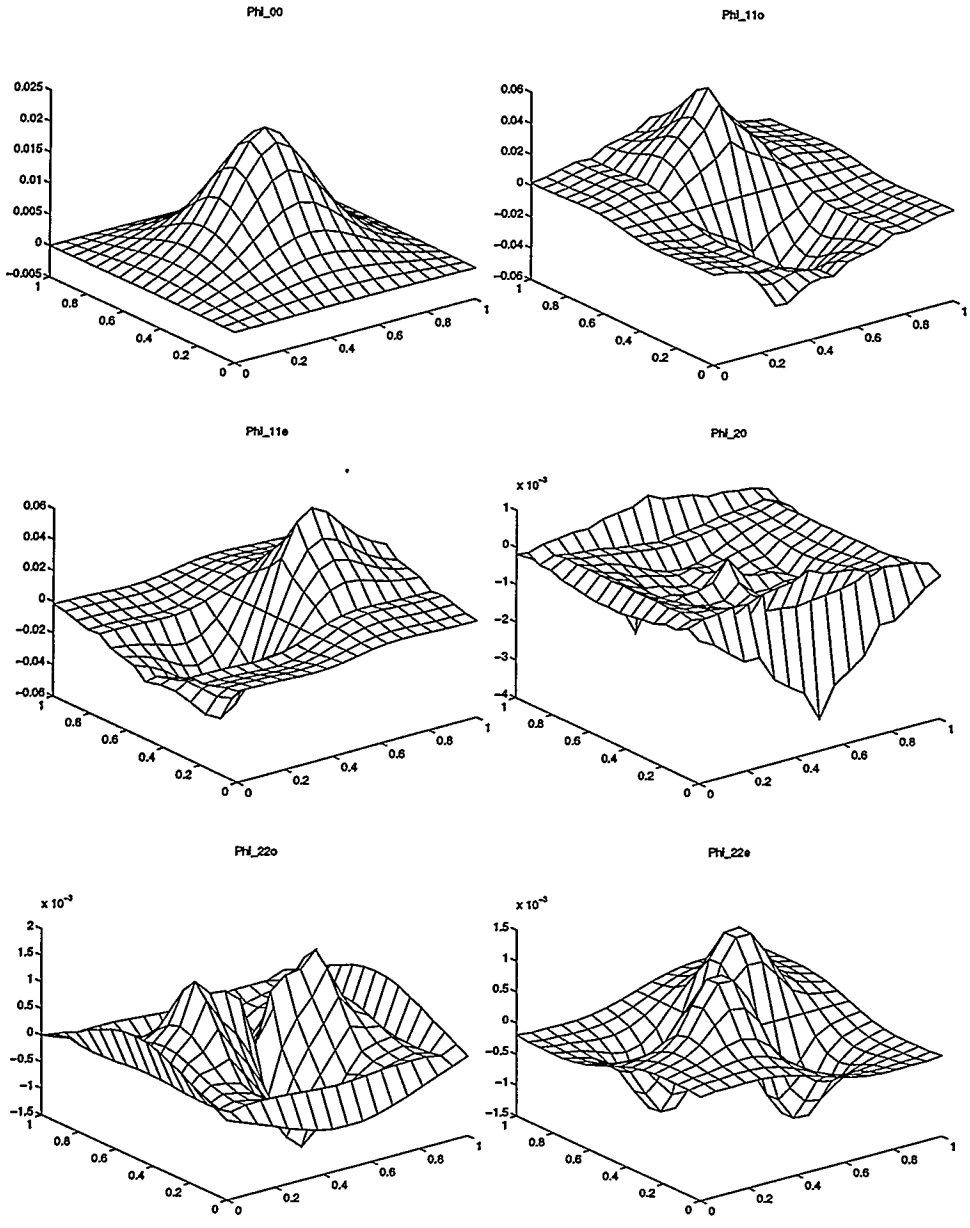
REFERENCES

[1] E. E. Lewis and J. W. F. Miller. *Computational Methods of Neutron Transport.* American Nuclear Society, La Grange Park, IL, 1993.

[2] T. A. Manteuffel and K. J. Ressel. Least-squares finite-element solution of the neutron transport equation in diffusive regimes. *SIAM J. Numer. Anal.*, 1996. Submitted.

[3] G. C. Pomraning. Diffusive limits for linear transport equations. *Nucl. Sci. Eng.*, 112:239–255, 1992.

FIG. 1. *2-D cross-section of the moments obtained from P_2 approximation: $\varepsilon = 0.01$*



Topic: *Parallel*	*Session Chair:* *Paul Saylor*	*Room B*
8:00 - 8:30	K. Maschhoff	A Portable Implementation of ARPACK for Distributed Memory Parallel Archetectures
8:30 - 9:00	P. Gray	Iteration Schemes for Parallelizing Models of Superconductivity
9:00 - 9:30	S. Veroneses	Scalable Implicit Methods for Reaction-Diffusion Equations in Two & Three Space Dimensions
9:30 - 10:00	W. Joubert	Evaluation of Parallel Linear Solvers for Oil Reservoir Simulation Problems

A Portable Implementation of ARPACK for Distributed Memory Parallel Architectures

K. J. Maschhoff and D. C. Sorensen

March 12, 1996

Abstract

ARPACK is a package of Fortran 77 subroutines which implement the Implicitly Restarted Arnoldi Method used for solving large sparse eigenvalue problems. A parallel implementation of ARPACK is presented which is portable across a wide range of distributed memory platforms and requires minimal changes to the serial code. The communication layers used for message passing are the Basic Linear Algebra Communication Subprograms (BLACS) developed for the ScaLAPACK project and Message Passing Interface(MPI).

1 Introduction

One objective for the development and maintenance of a parallel version of the ARPACK [3] package was to construct a parallelization strategy whose implementation required as few changes as possible to the current serial version. The basis for this requirement was not only to maintain a level of numerical and algorithmic consistency between the parallel and serial implementations, but also to investigate the possibility of maintaining the parallel and serial libraries as a single entity.

On many shared memory MIMD architectures, a level of parallelization can be accomplished via compiler options alone without requiring any modifications to the source code. This is rather ideal for the software developer.

For example, on the SGI Power Challenge architecture the MIPSpro F77 compiler uses a POWER FORTRAN Accelerator (PFA) preprocessor to automatically uncover the parallelism in the source code. PFA is an optimizing Fortran preprocessor that discovers parallelism in Fortran code and converts those programs to parallel code. A brief discussion of implementation details for ARPACK using pfa preprocessing may be found in [6]. The effectiveness of this preprocessing step is still dependent on how suitable the source code is for parallelization. Since most of the vector and matrix operations for ARPACK are accomplished via BLAS and LAPACK routines, access to efficient parallel versions of these libraries alone will provide a reasonable level of parallelization.

Unfortunately, for distributed memory architectures the software developer is required to do more work. For distributed memory implementations, message passing between processes must be explicitly addressed within the source code and numerical computations must take into account the distribution of data. In addition, for the parallel code to be portable, the communication interface used for message passing must be supported on a wide range of parallel machines and platforms. The Basic Linear Algebra Communication Subprograms (BLACS) [7], developed for the ScaLAPACK project provide a natural level of message passing for linear algebra computations and were found to be very suitable for the communication requirements of Parallel ARPACK (PARPACK). We have found BLACS to be easy to use and its simplification of message passing to be convenient.

A Message Passing Interface (MPI) [9] implementation of PARPACK is also available. Although an alpha release for MPI-BLACS is currently available, we have experienced some difficulty in portability for non MPICH [10] implementations and decided to provide an alternative implementation to accommodate vendor supplied MPI implementations until portability issues with MPI-BLACS have been resolved.

2 Parallel ARPACK

The parallelization paradigm found to be most effective for ARPACK on distributed memory machines was to provide the user with a Single Program Multiple Data (SPMD) template. The reverse communication interface is one of the most important aspects in the design of ARPACK and this feature

lends itself to a simplified SPMD parallelization strategy. This approach was used for previous implementations of ARPACK [2] and is simple for the the user to implement. The reverse communication interface feature of ARPACK allows the PARPACK codes to be parallelized internally without imposing a fixed parallel decomposition on the matrix or the user supplied matrix-vector product. Memory and communication management for the matrix-vector product can be optimized independent of PARPACK. This feature enables the use of various matrix storage formats as well as calculation of the matrix elements on the fly.

The calling sequence to ARPACK remains unchanged except for the addition of the BLACS context (or MPI communicator). Inclusion of the context (or communicator) is necessary for global communication as well as managing I/O. The addition of the context is new to this implementation and reflects the improvements and standardizations being made in message passing [9, 7].

2.1 Data Distribution of the Arnoldi Factorization

The numerically stable generation of the Arnoldi factorization

$$AV_k = V_k H_k + f_k e_k^T$$

where

A, $n \times n$ matrix
H_k, $k \times k$ - projection matrix (Upper Hessenberg)
V_k, $n \times k$ matrix, $k \ll n$ - Set of Arnoldi vectors
f_k, residual vector, $f_k V_k = 0$, length n

coupled with an implicit restarting mechanism [1] is the basis of the ARPACK codes. The simple parallelization scheme used for PARPACK is as follows.

- H_k replicated on every processor
- V_k is distributed across a 1-D processor grid. (Blocked by rows)
- f_k and workspace distributed accordingly

The SPMD code looks essentially like the serial code except that the local block of the set of Arnoldi vectors, V_{loc}, is passed in place of V, and n_{loc}, the dimension of the local block, is passed instead of n.

With this approach there are only two communication points within the construction of the Arnoldi factorization inside PARPACK: computation of the 2-norm of the distributed vector f_k and the orthogonalization of f_k to V_k using Classical Gram Schmidt with DGKS correction [5]. Additional communication will typically occur in the user supplied matrix-vector product operation as well. Ideally, this product will only require nearest neighbor communication among the processes. Typically the blocking of V is commensurate with the parallel decomposition of the matrix A. The user is free, however, to select an appropriate blocking of V such that an optimal balance between the parallel performance of PARPACK and the user supplied matrix-vector product is achieved.

An additional concern which must be addressed if this simple parallelization strategy is to be effective is scalability. Because H_k is replicated on every processor, operations on H due to the implicit restart mechanism within ARPACK are also replicated. This redundant work may present a "serial bottleneck" and could possibly prevent scalability if k grows with n as the problem size increases. The potential for such a bottleneck will hopefully be diminished when *locking* of converged vectors, as presented in [4], is implemented in the ARPACK codes. *Locking* , which is equivalent to deflation for the QR algorithm, is used to decouple converged Ritz values and associated vectors from the active part of the factorization. It is anticipated that locking may be used to keep the active portion of H_k small as well as to aid the convergence of eigenvalues with multiplicity greater than one. This feature is planned for the next release of ARPACK.

Another strategy which was tested was to use Parallel BLAS (PBLAS) [8] software developed for the ScaLAPACK project to achieve parallelization. The function of the PBLAS is to simplify the parallelization of serial codes implemented on top of the BLAS. The ARPACK package is very well suited for testing this method of parallelization since most of the vector and matrix operations are accomplished via BLAS and LAPACK routines.

Unfortunately this approach required additional parameters to be added to the calling sequence (the distributed matrix descriptors) as well as redefining the workspace data structure. Although there is no significant degradation in performance, the additional code modifications, along with the data

decomposition requirements, make this approach less favorable. As our parallelization is only across a one dimensional grid, the functionality provided by the PBLAS was more sophisticated than we required. The current implementation of the PBLAS (ScaLAPACK version 1.1) assumes the matrix operands to be distributed in a block-cyclic decomposition scheme.

2.2 Parallel Performance

To illustrate the potential scalability of Parallel ARPACK on distributed memory architectures some example problems have been run on the Maui HPCC SP2. The results shown in Table 1 attempt to illustrate the potential internal performance of the of the PARPACK routines independent of the users implementation of the matrix vector product.

In order to isolate the performance of the ARPACK routines from the performance of the user's matrix-vector product and also to eliminate the effects of a changing problem characteristic as the problem size increases, a test was comprised of replicating the same matrix repeatedly to obtain a block diagonal matrix. This completely contrived situation allows the workload to increase linearly with the number of processors. Since each diagonal block of the matrix is identical, the algorithm should behave as if *nproc* identical problems are being solved simultaneously (provided an appropriate starting vector is used). For this example we use a starting vector of all "1's". The only obstacles which prevent ideal speedup are the communication costs involved in the global operations and the "serial bottleneck" associated with the replicated operations on the projected matrix H. If neither of these were present then one would expect the execution time to remain constant as the problem size and the number of processors increase.

The matrix used for testing is a diagonal matrix of dimension $100,000$ with uniform random elements between 0 and 1 with four of the diagonal elements separated from the rest of the spectrum by adding an additional 1.01 to these elements. The problem size is then increased linearly with the number of processors by adjoining an additional diagonal block for each additional processor. For these timings we used the non-symmetric PARPACK code pdnaupd with the following parameter selections: mode is set to 1, number of Ritz values requested is 4, portion of the spectrum is "LM", and the maximum number of columns of V is 20.

Number of Nodes	Problem Size	Total Time (s)	Efficiency
1	100,000 * 1	40.53	
4	100,000 * 4	40.97	0.98
8	100,000 * 8	42.48	0.95
12	100,000 * 12	42.53	0.95
16	100,000 * 16	42.13	0.96
32	100,000 * 32	46.59	0.87
64	100,000 * 64	54.47	0.74
128	100,000 * 128	57.69	0.70

Table 1: Internal Scalability of Parallel ARPACK

3 Summary

We have presented a parallel implementation of the ARPACK library which is portable across a wide range of distributed memory platforms. The portability of PARPACK is achieve by utilization of the BLACS. We have found BLACS to be easy to use and have been quite satisfied with how little effort it takes to port PARPACK to a wide variety of parallel platforms. So far we have tested PARPACK on a SGI Power Challenge cluster using PVM-BLACS, on a CRAY T3D using Cray's implementation of the BLACS, on an IBM SP2 using MPL-BLACS, and on a network of Sun stations using PVM-BLACS. The MPI version of PARPACK has been tested on these platforms as well.

References

[1] D. C. Sorensen, *Implicit application of polynomial filters in a k-step Arnoldi method,* SIAM Journal on Matrix Analysis and Applications, 13(1):357-385, January 1992.

[2] D. C. Sorensen, *Implicitly-Restarted Arnoldi/Lanczos Methods for Large Scale Eigenvalue Calculations,* (invited paper), in Parallel Numerical Algorithms: Proceedings of an ICASE/LaRC Workshop, May 23-25, 1994, Hampton, VA, D. E. Keyes, A. Sameh, and V. Venkatakrishnan, eds., Kluwer, 1995 (to appear).

[3] R.B. Lehoucq, D.C. Sorensen, P.A. Vu, and C. Yang, *ARPACK: Fortran subroutines for solving large scale eigenvalue problems,* Release 2.1

[4] R. B. Lehoucq and D.C. Sorensen *Deflation Techniques for an Implicitly Re-started Arnoldi Iteration* , To appear in SIAM Journal of Matrix Analysis

[5] J. Daniel, W.B. Gragg, L. Kaufman, and G.W. Stewart *Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization* , Mathematics of Computation, 30:772-795, 1976

[6] M.P. Debicki, P. Jedrzejewski, J. Mielewski, P. Przybyszewski, and M. Mrozowski *Application of the Arnoldi Method to the Solution of Electromagnetic Eigenproblems on the Multiprocessor Power Challenge Architecture,* Technical Report Number 19/95, Department of Electronics, Technical University of Gdansk, Poland.

[7] J. J. Dongarra and R. C. Whaley *LAPACK Working Note 94, A User's Guide to the BLACS v1.0,* , June 7, 1995

[8] J. Choi, J. Dongarra, S. Ostrouchov, A. Petitet, D. Walker, and R. C. Whaley *LAPACK Working Note 100, A Proposal for a Set of Parallel Basic Linear Algebra Subprograms* , May 1995

[9] Message Passing Interface Forum, *MPI: A Message-Passing Interface Standard* , International Journal of Supercomputer Applications and High Performance Computing, 8(3/4), 1994

[10] MPICH, *A Portable Implementation of MPI,* available from info.mcs.anl.gov/pub/mpi

SUBMITTED TO THE 1996 COPPER MOUNTAIN CONFERENCE STUDENT PAPER COMPETITION. PRINTED DECEMBER 4, 1995

ITERATION SCHEMES FOR PARALLELIZING MODELS OF SUPERCONDUCTIVITY*

PAUL A. GRAY†

Abstract. The time dependent Lawrence–Doniach model, valid for high fields and high values of the Ginzburg–Landau parameter, is often used for studying vortex dynamics in layered high-T_c superconductors. When solving these equations numerically, the added degrees of complexity due to the coupling and nonlinearity of the model often warrant the use of high-performance computers for their solution. However, the interdependence between the layers can be manipulated so as to allow parallelization of the computations at an individual layer level. The reduced parallel tasks may then be solved independently using a heterogeneous cluster of networked workstations connected together with *Parallel Virtual Machine* (PVM) software. Here, this parallelization of the model is discussed and several computational implementations of varying degrees of parallelism are presented. Computational results are also given which contrast properties of convergence speed, stability, and consistency of these implementations. Included in these results are models involving the motion of vortices due to an applied current and pinning effects due to various material properties.

Key words. superconductivity, Lawrence Doniach model, layered superconductors, parallel computing, parallel virtual machine

AMS subject classifications. 65N30,65Y05,65C20,35J65,81Q05,81-08

1. Introduction. One of the features of high-T_c superconductors is their layered structure consisting alternating layers of superconducting and non- (or weakly) superconducting materials. In planes parallel to the layers, the material is isotropic. However, there is a strong anisotropy present when one compares material properties parallel and perpendicular to the layers. One may consult [10] for a survey providing a lucid discussion of layered superconductors.

One of the models for layered superconductors is the *Lawrence–Doniach model* introduced in [12]; see also [2] and [11]. When the *coherence length* $\xi_\perp$ in the direction perpendicular to the superconducting layer is of the order of the layer spacing, then it is necessary to account, in some way, for the discrete nature of the layered structure. In this model, the material is treated as a stack of superconducting planes separated by a vacuum or insulating material, and the coupling between the superconducting planes is similar to that which occurs in a Josephson junction. Again, one may consult [10] and the references cited therein for a complete discussion of these models and the physical circumstances necessary for their validity.

The relation between the Lawrence–Doniach (LD) model and the anisotropic Ginzburg–Landau model has been explored in [3]. Discretization of the steady state LD model using a gauge invariant difference approximation has been studied previously in [8], its parallel implementation on the 512 node Intel Touchtone Delta computer has won the Gordon Bell prize in computer science. Our interest lies in the study of the vortex motion and pinning mechanisms exhibited by the various variants of the LD models. These phenomena cannot be dealt with using the steady state model.

The difficulties involved in solving the Lawrence–Doniach model, numerically or otherwise, include the nonlinearity of the equations in addition to the coupling of the variables between the layers. In this paper, various methods for overcoming these difficulties are addressed. In order to simplify the equations themselves, the LD model is examined in the setting where the superconducting material possesses a high Ginzburg–Landau coefficient, κ, which is the radio of the in-plane *penetration depth*, λ, and the parallel coherence length $\xi_\parallel$. These terms yield a measure of distances over which the order parameter and magnetic field may undergo appreciable change ($\xi_\perp$ gives a measure of the change allowed of the order parameter perpendicular to the layers; $\xi_\parallel$ measures change allowed within the layer). This setting is significant inasmuch as materials made today which maintain their superconducting properties at high temperatures exhibit a large value of κ. Additional discussions on simplifications in the high-κ regime may be found in [4]. With the simplifications brought about by the high-κ setting, simplifications of the numerical implementation are sought. Specifically, simplifications are sought which address the complexity introduced by the coupling of the variables between the layers. This interdependence of the variables between layers may be solved for iteratively. The results of

* Supported in part by NSF grant MS-9500718

† Department of Mathematics, Michigan State University, East Lansing, MI 48824-1027.

this paper show that with a judiciously chosen iterative scheme, the coupling of the variables between the layers may be broken which allows for a straight-forward parallelization of the solution.

The remaining portion of the paper is outlined as follows: In §2, notational conventions are given which will be used in subsequent sections. Also given in §2 are the time dependent Lawrence–Doniach equations and their simplifications which are valid for the high-κ, high-field setting. In §3, the details of the discretization of the equations are discussed. §4 presents the algorithms posed to decouple and parallelize the model. Numerical comparisons of the methods given in §4 and a discussion of the physical description of the model being solved are given in §5. In §6, concluding remarks are given along with a discussion on the *Parallel Virtual Machine* software which makes it possible to connect a cluster of workstations together as a single parallel machine.

2. The time dependent Lawrence–Doniach equations. To describe the Lawrence–Doniach model, the layers are assumed to be perpendicular to the z-axis. $\mathcal{D}$ will correspond to a layered material sample such that $\mathcal{D} = \Omega \times [0, S]$, where $\Omega \subset \mathbb{R}^2$ is a planar domain and S is the z-thickness of the material sample. There are $N+1$ superconducting planes, each having projection Ω on the (x,y)-plane, and each separated from its neighbors by a distance s; thus, $Ns = S$. The boundary of Ω will be denoted by Γ. The region exterior to $\mathcal{D}$ will be denoted by $\mathcal{D}_e$, i.e., $\mathcal{D}_e = \mathbb{R}^3/\overline{\mathcal{D}}$. The interface between $\mathcal{D}$ and $\mathcal{D}_e$ will be denoted by $\partial\mathcal{D}$.

Throughout, three-vectors will be denoted by $\vec{(\cdot)}$ and two-vectors by bold face notation. Thus, A, $\mathbf{A}$, and $\vec{A}$ denote a scalar, a two-vector, and a three-vector, respectively. There will be occasions where it is convenient to partition a three-vector $\vec{A}$ into the form

$$\vec{A} = \begin{pmatrix} \mathbf{A} \\ A_z \end{pmatrix}$$

so that here A_z denotes the third component of $\vec{A}$. The same notational convention will be used for differential operators, for example, $\vec{\text{grad}} = (\mathbf{grad}, \frac{\partial}{\partial z})$.

The LD model uses three primary variables: the order parameter $\psi = \psi(x,y,z)$, whose magnitude is related to the density of superconducting carriers; the magnetic vector potential $\vec{A} = \vec{A}(x,y,z) = (\mathbf{A}, A_z)^T$; and $\Phi = \Phi(x,y,z)$ be the electric potential. We let A_z be the component of $\vec{A}$ in z direction, $\mathbf{A_n} = \mathbf{A_n}(x,y) = \mathbf{A}(x,y,ns)$, $n = 0,1,\ldots,N$, is the restriction of $\mathbf{A}$ to the n-th superconducting plane and similarly $\Phi_n = \Phi_n(x,y) = \Phi(x,y,ns)$ and $\psi_n = \psi_n(x,y) = \psi(x,y,ns)$. The same notation applies to the auxiliary variables that are defined by

$$\vec{\text{curl}}\, \vec{A}^a = \vec{H}$$

and

$$-\sigma\kappa\nabla\Phi^a = \vec{\text{curl}}\, \vec{H}\ .$$

After proper nodimensionalization, the time dependent *Lawrence–Doniach equations* are given by

$$\begin{aligned} &\frac{\partial\psi_n}{\partial t} + i(\Phi_n + \Phi_n^a)\psi_n - \psi_n + |\psi_n|^2\psi_n + (\frac{i}{\kappa}\mathbf{grad} + \mathbf{A_n} + \mathbf{A_n^a})^2\psi_n \\ &\quad + \rho\Big[2\psi_n - \psi_{n+1}\exp(-i\phi_n^{n+1}) - \psi_{n-1}\exp(i\phi_{n-1}^n)\Big] = 0 \\ &\qquad\qquad \text{in } \Omega \text{ and for } n = 0,1,\ldots,N\,, \end{aligned} \tag{2.1}$$

where

$$\phi_n^{n+1} = \kappa\int_{ns}^{(n+1)s} A_z + A_z^a\, dz \quad \text{for } n = 0,\ldots,N-1\,, \tag{2.2}$$

and the terms in the bracket describe the Josephson coupling. The characterization of $\psi_{-1}, \psi_{N+1}, \phi_{-1}^0$ and ϕ_N^{N+1} depend on the boundary conditions imposed and/or on the periodicity assumptions of the model. In addition,

$$\sigma(\frac{\partial\vec{A}}{\partial t} + \kappa\nabla\Phi) + \vec{\text{curl}}\,\vec{\text{curl}}\,\vec{A} = \begin{pmatrix} \mathbf{j} \\ j_z \end{pmatrix} \quad \text{in } \mathbb{R}^3\,, \tag{2.3}$$

where σ is a material constant representing a driving force. Further, the components of the current are given by

$$\mathbf{j} = s\sum_{n=0}^{N}\left(\frac{i}{2\kappa}(\psi_n \mathbf{grad}\,\psi_n{}^* - \psi_n{}^*\mathbf{grad}\,\psi_n) - |\psi_n|^2(\mathbf{A_n} + \mathbf{A_n^a})\right)\delta(z-ns) \quad \text{in } \mathbb{R}^3\ , \tag{2.4}$$

$$j_z = i\kappa\rho s\Big[\psi_n \exp(i\phi_n^{n+1})\psi_{n+1}^* - \psi_n^* \exp(-i\phi_n^{n+1})\psi_{n+1}\Big] \quad \text{in } \Omega\times[ns,(n+1)s],\ n=0,\dots,N-1, \tag{2.5}$$

and

$$j_z = 0 \quad \text{in } \mathcal{D}_e = \mathbb{R}^3/\overline{\mathcal{D}}\ . \tag{2.6}$$

One also obtains the boundary conditions

$$(\frac{i}{\kappa}\mathbf{grad}\,\psi_n + \mathbf{A_n^a}\psi_n)\cdot\mathbf{n} = 0 \quad \text{on } \Gamma \text{ and for } n=0,1,\dots,N, \tag{2.7}$$

$$[\vec{A}\times\vec{n}] = \mathbf{0} \quad \text{and} \quad [\vec{\mathrm{curl}}\,\vec{A}\times\vec{n}] = \mathbf{0} \quad \text{on the boundary } \partial\mathcal{D} \text{ of } \Omega\times[0,S], \tag{2.8}$$

and

$$\vec{\mathrm{curl}}\,\vec{A}\to\vec{0} \quad \text{as } |\vec{x}|\to\infty. \tag{2.9}$$

In the high-κ, high field setting, the applied field is assumed to be of the form $\vec{H} = \kappa\vec{H}_0$ and $\vec{A}_0^a$ is chosen to be a magnetic potential such that $\vec{\mathrm{curl}}\times\vec{A}_0^a = \vec{H}_0$. Then, it can be shown that there exists a solution ψ_n, $n=0,1,\dots,N$, and $\vec{A}$ of the Lawrence–Doniach equations satisfying

$$\vec{A} = \kappa\left(\vec{A}_0 + \mathcal{O}\left(\kappa^{-2}\right)\right) \quad \text{and} \quad \psi_n = \psi_{0n} + \mathcal{O}\left(\kappa^{-2}\right),\ n=0,1,\dots,N,$$

where ψ_{0n}, $n=0,1,\dots,N$, satisfies

$$\frac{\partial\psi_{0n}}{\partial t} + i\Phi_{0n}^a\psi_{0n} - \psi_{0n} + |\psi_{0n}|^2\psi_{0n} + (i\mathbf{grad} + \mathbf{A}_{0n}^a)^2\psi_{0n} + \rho\Big[2\psi_{0n} - \psi_{0(n+1)}\exp(-i\phi_{0n}^{n+1}) - \psi_{0(n-1)}\exp(i\phi_{0(n-1)}^{n})\Big] = 0 \quad \text{in } \Omega \text{ and for } n=0,1,\dots,N,$$

where

$$\phi_{0n}^{n+1} = \int_{ns}^{(n+1)s} A_{0z}^a\,dz \quad \text{for } n=0,\dots,N-1,$$

and

$$(i\mathbf{grad}\,\psi_{0n} + \mathbf{A}_{0n}^a\psi_{0n})\cdot\mathbf{n} = 0 \quad \text{on } \Gamma \text{ and for } n=0,1,\dots,N.$$

3. The discretization scheme. The numerical scheme is intended to model long-term behavior of a system. For this reason, the implicit Euler scheme is used for the discretization in time to maintain stability. Letting $\delta_{\Delta t}^-$ denote the backward difference operator, the discretization in time of the leading order equations leads to the following equation:

$$\delta_{\Delta t}^-(\psi_{0n}) + i\Phi_{0n}^a\psi_{0n} - \psi_{0n} + |\psi_{0n}|^2\psi_{0n} + (i\mathbf{grad} + \mathbf{A}_{0n}^a)^2\psi_{0n} + \rho\Big[2\psi_{0n} - \psi_{0(n+1)}\exp(-i\phi_{0n}^{n+1}) - \psi_{0(n-1)}\exp(i\phi_{0(n-1)}^{n})\Big] = 0 \quad \text{in } \Omega \text{ and for } n=0,1,\dots,N. \tag{3.1}$$

Furthermore, the spatial discretization is done by Galerkin finite element methods, namely, we take the weak form of the equation (3.1) and use finite element spaces as the test and trial spaces. Let S^h be a finite element subspace corresponding to a mesh, with mesh parameter h. Let $\{\psi_n^h, n = -1, \ldots, N+1\}$ be the finite element solution in S^h. Then, we have

$$\begin{aligned} &\delta_{\Delta t}^{-}\left(\psi_n^h, v^h\right) + \left(i\Phi_{0n}^a \psi_n^h, v^h\right) + \left(-\psi_n^h + |\psi_n^h|^2\psi_n^h, v^h\right) + \left((i\mathbf{grad} + \mathbf{A}_{0n}^a)^2\psi_n^h, v^h\right) \\ &\quad + \rho\left(\left[2\psi_n^h - \psi_{(n+1)}^h \exp(-i\phi_{0n}^{n+1}) - \psi_{(n-1)}^h \exp(i\phi_{0(n-1)}^n)\right], v^h\right) = 0 \\ &\qquad \forall \quad v^h \in S^h \text{ and for } n = 0, 1, \ldots, N \end{aligned} \tag{3.2}$$

with appropriate conditions on ψ_{-1}^h and ψ_{N+1}^h.

(3.2) may be rewritten as:

$$\begin{aligned} &\frac{1}{\Delta t}\left(\psi_n^h, v^h\right) + \left(i\Phi_{0n}^a \psi_n^h, v^h\right) + \left((2\rho - 1 + |\psi_n^h|^2)\psi_n^h, v^h\right) + \left((i\mathbf{grad} + \mathbf{A}_{0n}^a)^2\psi_n^h, v^h\right) \\ &\quad - \rho\left(\psi_{(n+1)}^h \exp(-i\phi_{0n}^{n+1}), v^h\right) - \rho\left(\psi_{(n-1)}^h \exp(i\phi_{0(n-1)}^n), v^h\right) = \frac{1}{\Delta t}\left(\psi_{n,old}^h, v^h\right) \\ &\qquad \forall \quad v^h \in S^h \text{ and for } n = 0, 1, \ldots, N \end{aligned} \tag{3.3}$$

where $\psi_{n,old}^h$ is the solution at the previous time step. In this form, the above nonlinear system of equations is easily cast into matrix terms as

$$\mathbf{M}(\mathbf{P}_n)\mathbf{P}_n + \mathbf{C}_n\mathbf{P}_{n+1} + \mathbf{C}_{n-1}^*\mathbf{P}_{n-1} = \mathbf{F}_n \qquad \text{for } n = 0, 1, \ldots, N,$$

where the vector $\mathbf{P}_n$ represents the n-th layer unknowns, $\mathbf{M}(\mathbf{P}_n)$ and $\mathbf{C}_n$ represent the coefficient matrices, $\mathbf{C}_n^*$ is the adjoint matrix of $\mathbf{C}_n$, and $\mathbf{F}_n$ is associated with the computations involving the previous time step. Denoting $\mathbf{M}_n = \mathbf{M}(\mathbf{P}_n)$, one can group all of these variables together to form the system:

$$\begin{pmatrix} \mathbf{M}_0 & \mathbf{C}_0 & & & & \\ \mathbf{C}_0^* & \mathbf{M}_1 & \mathbf{C}_1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & \mathbf{C}_{N-2}^* & \mathbf{M}_{N-1} & \mathbf{C}_{N-1} & \\ & & & \mathbf{C}_{N-1}^* & \mathbf{M}_N & \end{pmatrix} \mathbf{P} = \mathbf{F}. \tag{3.4}$$

4. Parallelization of the computational solution. The algorithms for parallelization presented here focus on decoupling the dependence of ψ_n^h on $\psi_{(n+1)}^h$ (from the layer above) and $\psi_{(n-1)}^h$ (from the layer beneath). Methods for decoupling ψ_n^h from $\psi_{(n+1)}^h$ and $\psi_{(n-1)}^h$ focus on modification of the bracketed portion of equation (3.2) and thus, the corresponding off-diagonal blocks in (3.4).

There are a number of ways to solve system (3.3) or (3.4). The focus of this paper is the iterative solution of ψ_n^h, or equivalently, the iterative solution of the vector $\mathbf{P}$ in (3.4). Taking a six-layer model as an example, the physical interpretation of iterative schemes which may be used to solve the above system is generalized in Figure 1. The mathematical interpretation of the schemes represented in Figure 1 is given in the subsequent section.

4.1. The sequential update model. This model sequentially updates a single layer at one time. Referring to equation (3.3), ψ_n^j will be used to denote the j-th iterative solution of ψ_n. In the sequential model presented below, ψ_n^j is computed using the updated value of the layer beneath it, $\psi_{0(n-1)}^j$, and the previous value of the variables in the layer above it, $\psi_{0(n+1)}^{j-1}$. Initialization of the method is satisfied by solving the expression:

$$\begin{aligned} &\delta_{\Delta t}^{-}\left(\psi_{01}^1\right) + i\Phi_{01}^a\psi_{01}^1 - \psi_{01}^1 + |\psi_{01}^1|^2\psi_{01}^1 \\ &\qquad + (i\mathbf{grad} + \mathbf{A}_{01}^a)^2\psi_{01}^1 + \rho\left[\psi_{01}^1 - \psi_{02}^0 \exp(-i\phi_{01}^2)\right] = 0 \qquad \text{in } \Omega. \end{aligned}$$

This yields $\psi_{01}^1(t)$, which is then available for the computation of ψ_{02}^1. Note that computation of ψ_{02}^1 will require information from the third layer. Since the third layer has yet to be updated, ψ_{03}^0 is used for the update of ψ_{02}^1. In general, the method reverts solving the equation;

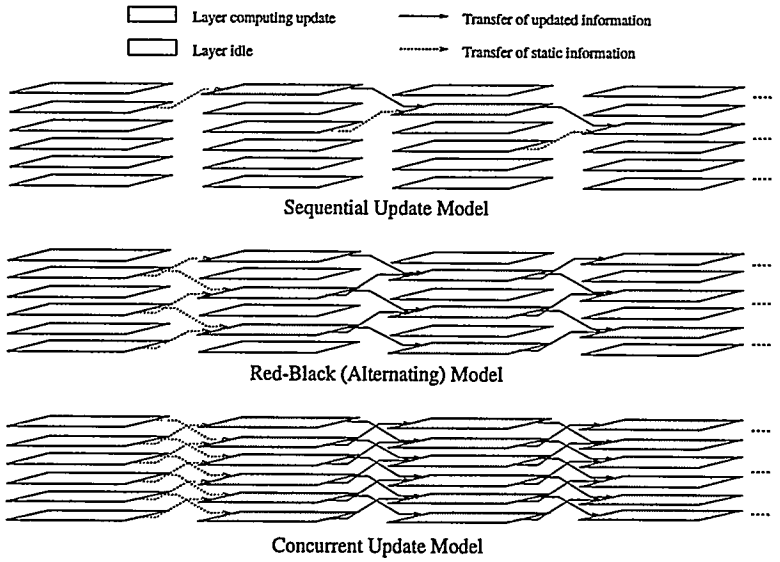


FIGURE 1. *A pictorial description of iterative schemes used to solve the system* (3.3), *or equivalently,* (3.4).

$$\delta_{\Delta t}^{-}\left(\psi_{0n}{}^{j}\right) + i\Phi_{0n}^{a}\psi_{0n}{}^{j} - \psi_{0n}{}^{j} + |\psi_{0n}{}^{j}|^{2}\psi_{0n}{}^{j} + (i\mathbf{grad} + \mathbf{A}_{0n}^{a})^{2}\psi_{0n}{}^{j}$$
$$+ \rho\Big[2\psi_{0n}{}^{j} - \psi_{0(n+1)}{}^{j-1}\exp(-i\phi_{0n}^{n+1}) - \psi_{0(n-1)}{}^{j}\exp(-i\phi_{0(n-1)}^{n})\Big] = 0 \qquad \text{in } \Omega.$$

Since each layer must wait until the layer beneath it has completed its update, this model is best suited for a single-processor setting. Note that initialization using the lowest layer is somewhat arbitrary. The method is easily generalized to an arbitrary layer's initialization and a more general sequence of subsequent layer updates. For example, the pictorial representation of the sequential method in Figure 1 shows a top-layer-down update approach.

In matrix terms, this algorithm is similar to the block Gauss-Seidel iteration for the equation (3.4):

$$\begin{pmatrix} \mathbf{M}_0^j & 0 & & & \\ \mathbf{C}_0^* & \mathbf{M}_1^j & 0 & & \\ 0 & \ddots & \ddots & \ddots & \\ & \ddots & \mathbf{C}_{N-2}^* & \mathbf{M}_{N-1}^j & 0 \\ & & 0 & \mathbf{C}_{N-1}^* & \mathbf{M}_N^j \end{pmatrix} \mathbf{P}^j = \mathbf{F} + \begin{pmatrix} 0 & \mathbf{C}_0 & & & \\ & 0 & \mathbf{C}_1 & & \\ & & \ddots & \ddots & \\ & & & 0 & \mathbf{C}_{N-1} \\ & & & & 0 \end{pmatrix} \mathbf{P}^{j-1}. \tag{4.1}$$

4.2. The Red-Black model. This model alternative computes updates to every other layer. The model presented below can be loosely described as a computing of the j-th iteration of the red layers (the odd layers, for example) with information from the $(j-1)$-th iteration of the black (even) layers. Next, the j-th update of the black layers are updated using the updated j-th iteration values of the red layers. The method is given as follows:

$$\delta_{\Delta t}^{-}\left(\psi_{0n}{}^{j}\right) + i\Phi_{0n}^{a}\psi_{0n}{}^{j} - \psi_{0n}{}^{j} + |\psi_{0n}{}^{j}|^{2}\psi_{0n}{}^{j} + (i\mathbf{grad} + \mathbf{A}_{0n}^{a})^{2}\psi_{0n}{}^{j}$$
$$+ \rho\Big[2\psi_{0n}{}^{j} - \psi_{0(n+1)}{}^{j-1}\exp(-i\phi_{0n}^{n+1}) - \psi_{0(n-1)}{}^{j-1}\exp(-i\phi_{0(n-1)}^{n})\Big] = 0 \qquad \text{in } \Omega \texttt{ if } n \texttt{ red.}$$

$$\delta_{\Delta t}^{-}\left(\psi_{0n}{}^{j}\right) + i\Phi_{0n}^{a}\psi_{0n}{}^{j} - \psi_{0n}{}^{j} + |\psi_{0n}{}^{j}|^{2}\psi_{0n}{}^{j} + (i\mathbf{grad} + \mathbf{A}_{0n}^{a})^{2}\psi_{0n}{}^{j}$$
$$+ \rho\Big[2\psi_{0n}{}^{j} - \psi_{0(n+1)}{}^{j}\exp(-i\phi_{0n}^{n+1}) - \psi_{0(n-1)}{}^{j}\exp(-i\phi_{0(n-1)}^{n})\Big] = 0 \qquad \text{in } \Omega \texttt{ if } n \texttt{ black.}$$

In this model, only the black layers must wait for updated information from the red layers. Further, the red-layer and black-layer updates, respectively, may be computed independently. This makes the alternating update model well suited for the setting where the material being modeling is characterized by an even number of layers and there is one processor available per two layers of the material. Each processor would be assigned computation of a red layer's update and the subsequent computation of a black layer's update.

Again, in matrix terms, this translates to the two-stage linear system:

$$\begin{pmatrix} \mathbf{M}_0^j & 0 & & \\ 0 & \mathbf{M}_2^j & 0 & \\ & 0 & \mathbf{M}_4^j & \ddots \\ & & \ddots & \ddots \end{pmatrix} \mathbf{P}_{\mathtt{red}}^j = \mathbf{F}_{\mathtt{red}} + \begin{pmatrix} \mathbf{C}_0 & & & \\ \mathbf{C}_1^* & \mathbf{C}_2 & & \\ & \mathbf{C}_3^* & \mathbf{C}_4 & \\ & & \ddots & \ddots \end{pmatrix} \mathbf{P}_{\mathtt{black}}^{j-1}, \tag{4.2.a}$$

$$\begin{pmatrix} \mathbf{M}_1^j & 0 & & \\ 0 & \mathbf{M}_3^j & 0 & \\ & 0 & \mathbf{M}_5^j & \ddots \\ & & \ddots & \ddots \end{pmatrix} \mathbf{P}_{\mathtt{black}}^j = \mathbf{F}_{\mathtt{black}} + \begin{pmatrix} \mathbf{C}_1 & & & \\ \mathbf{C}_2^* & \mathbf{C}_3 & & \\ & \mathbf{C}_4^* & \mathbf{C}_5 & \\ & & \ddots & \ddots \end{pmatrix} \mathbf{P}_{\mathtt{red}}^j. \tag{4.2.b}$$

4.3. Concurrent update model. In this model, the j-th update of each layer is computed using the $(j-1)$-th value of its neighboring layers. The computational algorithm is as follows:

$$\delta_{\Delta t}^{-}\left(\psi_{0n}{}^{j}\right) + i\Phi_{0n}^{a}\psi_{0n}{}^{j} - \psi_{0n}{}^{j} + |\psi_{0n}{}^{j}|^2\psi_{0n}{}^{j} + (i\mathbf{grad} + \mathbf{A}_{0n}^{a})^2\psi_{0n}{}^{j} + \rho\Big[2\psi_{0n}{}^{j} - \psi_{0(n+1)}{}^{j-1}\exp(-i\phi_{0n}^{n+1}) - \psi_{0(n-1)}{}^{j-1}\exp(-i\phi_{0(n-1)}^{n})\Big] = 0 \qquad \text{in } \Omega.$$

Since each layer's update depends only upon the previous updates of its neighboring layers, each layer update may be computed independently. This makes the simultaneous model best suited to the setting where there is one processor available per layer of the model.

In matrix terms, this method is similar to a block Gauss-Jacobi approach to solving (3.4).

$$\begin{pmatrix} \mathbf{M}_0^j & & & \\ & \mathbf{M}_1^j & & \\ & & \ddots & \\ & & & \mathbf{M}_{N+1}^j \end{pmatrix} \mathbf{P}^j = \mathbf{F} + \begin{pmatrix} 0 & \mathbf{C}_0 & & & \\ \mathbf{C}_0^* & 0 & \mathbf{C}_1 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \mathbf{C}_{N-1} \\ & & & \mathbf{C}_{N-1}^* & 0 \end{pmatrix} \mathbf{P}^{j-1}. \tag{4.3}$$

5. Numerical comparisons. In this section, numerical results are given for four implementations of the algorithms given in §4, namely

i. *Non-parallelized sequential model.* (NPSM) This method will serve as the baseline in the comparisons. Basically, this implementation is what one would possibly revert to in the absence of means for parallel computations. The NPSM is intended to run on a single processor.

ii. *Parallelized sequential model.* (PSM) In order to determine the overhead of the parallelized implementations, the sequential model is parallelized in the crudest of fashions, parallelizing only the initialization routines. The PSM is able to run in a minimal setting of a single processor but may be run on any number of processors. In the fullest setting of the PSM, there is one processor for each layer in the model. In such a setting, where there are K layers in the simulation, the nature of the sequential algorithm would dictate that $K-1$ of the processors would be idle, waiting for the update information from the layer beneath.

iii. *Red-Black model.* (RBM) The implementation of the red-black algorithm requires a minimum of $(K-1)/2+1$ (as an integer) processors for the simulation of a K-layer material, where each processor

would be designated the task of calculating a "red" layer update and a subsequent "black" layer update. E.g., the ideal implementation would require a minimum of 2 processors for the simulation of a 4-layer model and 3 processors for the simulation of a 5-layer model. The maximal or full implementation would again be the setting where there is one processor for each layer of the model. In such an implementation, the red-designated processors would be idle while the black-designated processors were calculating updates and vice versa.

iv. *Concurrent model.* (CM) The full parallelization of the Lawrence–Doniach model requires a processor for each layer of the model. Each processor simultaneously computes the update of its respective layer.

5.1. The physical model of the simulation. The computational results given below were derived from the simulation of a material consisting of six layers. The superconducting layers are square, measuring $25\xi_{\|} \times 25\xi_{\|}$, the spacing between the layers is taken to be $\xi_{\|}$ (i.e. $s = \xi_{\|}$), and the initial state of the system is taken to be the steady state corresponding to the external applied field

$$\vec{H}_{\text{ext}} = (0.07, 0.07, 0.125)\kappa. \tag{5.1.1}$$

The sample is then subjected to a variable field which induces a current across the sample:

$$\vec{H}_{\text{var}} = (0.07, 0.07, 0.125 - 0.2\left(\frac{x - l/2}{l}\right))\kappa, \tag{5.1.2}$$

where l is the length of the sample.

5.2. Computational results. The numerical solution of the respective algorithms is computed by a finite element implementation using biquadratic basis functions over rectangular elements. The nonlinearity is solved using Newton's method. This computational implementation used was modeled after the code used in [6,7]. Each layer was decomposed into a 21×21 rectangular grid and the computations were carried out on a cluster of six PVM-connected DEC Alpha workstations.

The tables below yield some indication of the *speed* of the respective algorithms. The data in Table 1–Table 3 below was obtained by letting the computer models run over a fixed block of time in order to examine how "far" each algorithm was able to go. The first row of each table shows the number of time steps completed by the algorithm within the fixed block of time. The second entry in each table shows the total number of layer iterations required of the algorithm within the same block of time. The last entry in each table shows the percentage of time steps that the parallelized algorithm was able to calculate compared to the nonparallelized sequential method.

In short, one sees that the parallelized sequential model (PSM) performs just slightly better that the nonparallelized (NPSM) implementation. This is perhaps due to the fact that the initializations of the PSM are performed in parallel. These tasks include initialization of the order parameter, the generation of the vector magnetic potential, and the characterization of the geometry. Otherwise, the methods are identical. This does, however, show that the overhead introduced by the parallelization is somewhat insignificant in light of the small gains enjoyed by parallelizing the initialization routines.

Note also that the Red-Black method (RBM) was able to realize more than a three-fold gain over the sequential method(s); this, contrary to what one might initially expect. On the other hand, the concurrent method (CM) showed a more moderate gain of slightly over four-fold.

Parallelization aside, the above data also gives an indication of the computational *efficiency* of the respective algorithms. Examination of the number of layer iterations computed for each method shows that the number of layer iterations required per time step varies for each algorithm and for each fixed time step. In short, by looking at the number of layer iterations required for each time step, one sees that the CM generally required more layer iterations to reach tolerance at a given time step whereas the RBM generally required the least. This yields a partial explanation of the remarks made above in that the RBM was able to achieve a more than three-fold gain due to an increase in the algorithm's efficiency. (Another contributing factor was that Newton's method required slightly fewer iterations on average in the Red-Black scheme.) A comparison of the required number of layer iterations for convergence is given in Table 4. Table 4 shows the number of layer iterations required to meet tolerance per time step as a function of the step size taken in time for each of the respective algorithms. In other words, Table 4 shows the number of iterations required

dt=0.2	NPSM	PSM	RBM	CM
# Time Steps	26	26	75	107
# Layer Iterations	156	157	451	964
% Performance	Baseline	100%	288%	412%

TABLE 1. *Benchmarks obtained for a fixed time step of* dt= 0.2.

dt=0.1	NPSM	PSM	RBM	CM
# Time Steps	36	36	113	148
# Layer Iterations	181	184	569	1039
% Performance	Baseline	100%	314%	411%

TABLE 2. *Benchmarks obtained for a fixed time step of* dt= 0.1.

dt=0.025	NPSM	PSM	RBM	CM
# Time Steps	54	55	168	263
# Layer Iterations	216	220	504	1052
% Performance	Baseline	102%	311%	487%

TABLE 3. *Benchmarks obtained for a fixed time step of* dt= 0.025.

to reach convergence in equations (4.1), (4.2.a) – (4.2.b) and (4.3), respectively, based on the value of $\triangle t$ in (3.3). Note that the two sequential methods, the NPSM and PSM, yield equivalent solutions, therefore they need not be considered independently. From the table, one sees that the RBM required the least number of layer iterations to converge to the specified tolerance while the CM required the most layer iterations to converge.

dt	0.025	0.05	0.1	0.2	0.3	0.4	0.5	0.75
Sequential	4	4	5	6	7	8.5	10	12
Red-Black	3	4	5	6	6	8	9	12
Concurrent	4	5	7	9	10	14	16	21

TABLE 4

A comparison of the number of iterations required to meet tolerance based on step size in time.

5.4. Physical comparisons and results. A succinct indication of how the results of the respective methods compare with each other is seen in the energy of the respective systems. Figure 2 below shows the energies associated with the computational solutions of the extreme cases: the simultaneous algorithm and the concurrent algorithm. The close resemblance indicates that the two algorithms are indeed consistent in their calculation of the solution of the model.

Turning attention to the the physical interpretation of the energy plot itself, the reason for the variations in the energy is best explained by examining the physical dynamics of the system. Initially, the system is at the steady-state associated with a fixed external field strength given by (5.1.1), which corresponds to the setting where there are eight vortex "tubes" initially present in the sample. Given in the first row of Figure 3 are contour plots representing this initial state of the six-layer material. At the center of the circular contours is the core of the vortex, where the field is able to penetrate the sample. The material is then subjected to a variable field as given by (5.1.2), which induces current across the material. As a result, the vortices are carried across the material. The subsequent contour plots of Figure 3 show the state of the system at the time indicated above each row, as the current moves the vortices across the material.

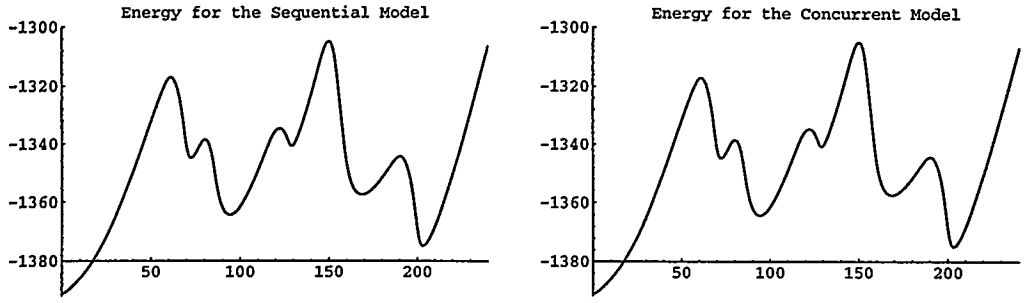


FIGURE 2. *Energy vs. time for the Sequential and Concurrent schemes. The close resemblance substantiates the solutions of the two algorithms yield consistent information.*

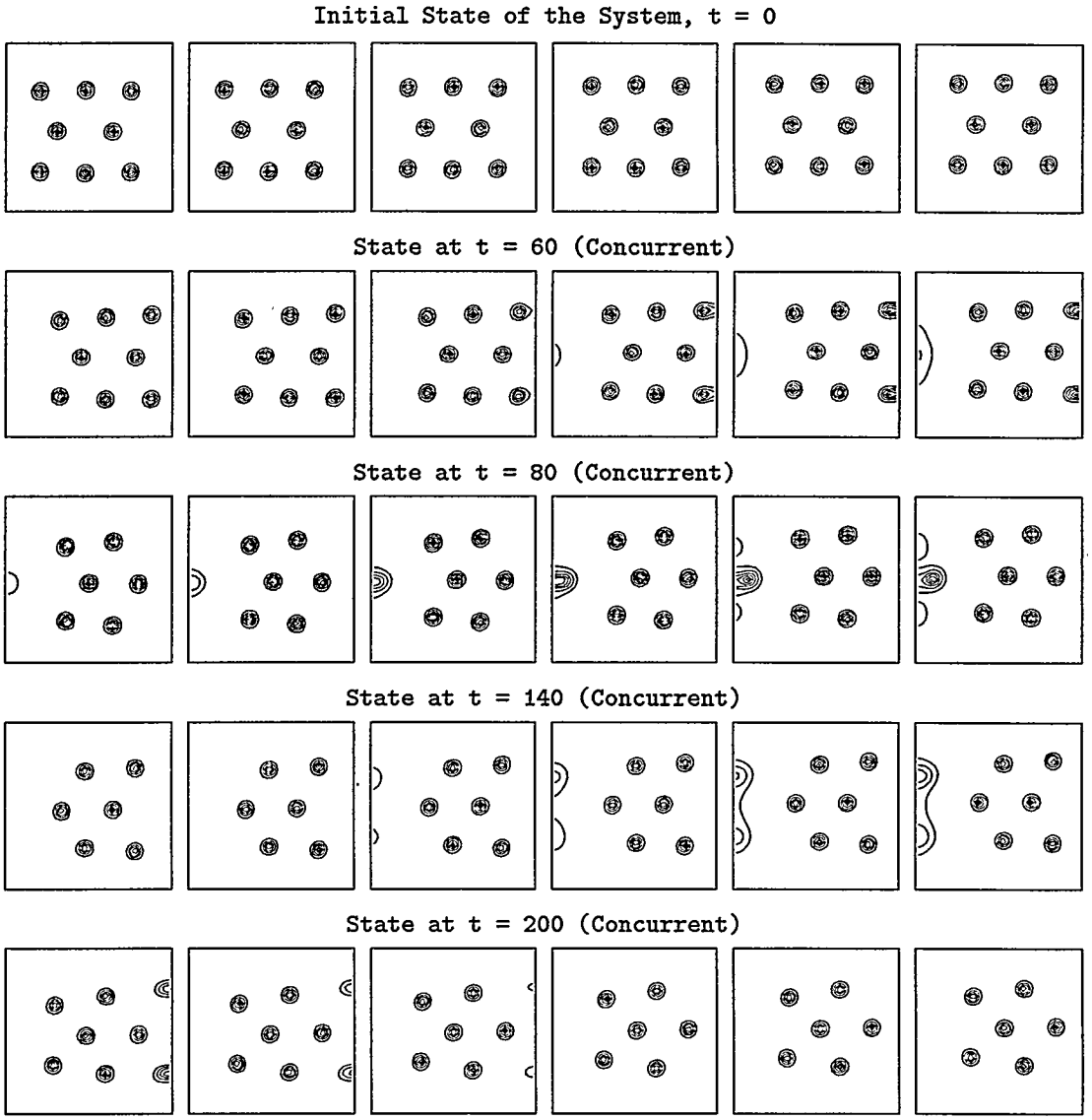


FIGURE 3 *Contour Plots of the magnitude of the order parameter at indicated times as computed by the concurrent model. With the application of current, the vortices move (to the right) across the sample. The vortices drop off the right edge and make room for the generation of vortices on the opposite edge. In each row, the left-most contour plot represents the bottom layer of the material; the right-most plot represents the top layer.*

Eventually, vortices exit the one end of the sample (the right end) while more vortices are generated on the other side. This loss and gain of vortices corresponds to the variations in the energy seen in Figure 2. To

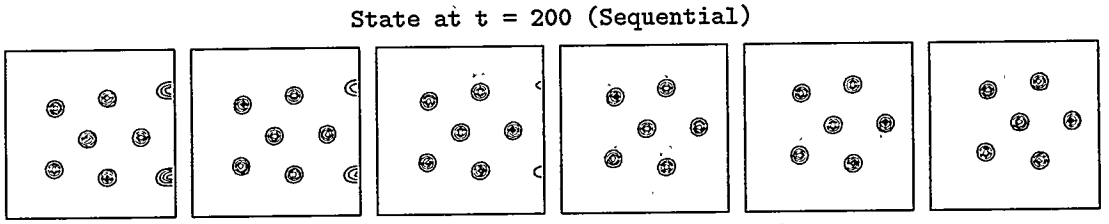


FIGURE 4 *The corresponding state of the system as computed by the sequential algorithm. Compare with the final row of* FIGURE 3 *above.*

further demonstrate the compatibility of the respective solutions, Figure 4 shows the same physical state of the system as computed by the sequential method at $t = 200$. Note the similarity of the contour plots of Figure 4 with the corresponding plots of Figure 3.

6. Remarks on the Parallel Virtual Machine (PVM) software and conclusion. In general terms, PVM software is a medium for *distributed computations.* Distributed computing is described as the process of connecting together, a set of networked computers for the purpose of collectively solving a single problem. PVM software provides the communication link between the set of computers. Thus, the collection of computers may be viewed as a single large *virtual* parallel computer, hence its name.

The PVM software runs on a wide variety of platforms — including serial, parallel and vector computers. Moreover, PVM is able to connect these varying platforms, together into one *heterogeneous* virtual machine. For a general introduction, installation guide, instructions for obtaining the software or sample programs, see [1,9]. Our configuration consisted of six DEC Alpha workstations and a SPARC Multiprocessor connected together via a fiber optic network.

In conclusion, the parallel implementation of the finite element Lawrence–Doniach code using PVM has proved to be a powerful tool for the study of the validity and limitations of the Lawrence–Doniach models for layered superconductors. We are currently conducting numerical simulations of various phenomena involving the vortex motion and vortex pinning [5]. Further development in the convergence theory of the algorithms presented in this paper will also be studied there, which may help us to improve the performance of the code by employing proper preconditioners and linearization schemes. It is hoped that useful tools will be developed not only for the use of physicists and material scientists interested in studying superconducting phenomena but also for those involved in the design of superconducting devices.

Acknowledgment. The author would like to extend thanks to Prof. Qiang Du for his generous support and suggestions and to the Advanced Computing Systems laboratory at Michigan State University for the use of their equipment and support.

References:

[1] A. Beguelin, J. Dongarra, A. Geist, W. Jiang, R. Manchek, and V. Sunderam, *PVM: Parallel virtual machine. A user's guide and tutorial for networked parallel computing*, 1994 MIT Press.

[2] L. Bulaevskii, *Magnetic properties of layered superconductors with weak interaction between the layers*, Zh. Eksperim. i Teor. Fiz., 64 (1973), 2241–2247. [English translation: Soviet Phys.-JETP, 37 (1973), 1133–1136.]

[3] S. J. Chapman, Q. Du, M. Gunzburger, *On the Lawrence–Doniach and anisotropic Ginzburg–Landau models for layered superconductors*, to appear in SIAM J. Appl. Math.

[4] Q. Du and P. Gray, *High-kappa limits of the time-dependent Ginzburg–Landau model*, to appear in SIAM J. Appl. Math.

[5] Q. Du and P. Gray, *Analysis and parallelization of the Lawrence–Doniach model of superconductivity using PVM*, in preparation.

[6] Q. Du, M. Gunzburger, and J. Peterson, *Analysis and approximation of Ginzburg-Landau models for superconductivity*, SIAM Review, 34 (1992), 54–81.

[7] Q. Du, M. Gunzburger, and J. Peterson, *Computational simulation of type-II superconductivity including pinning phenomena*, Phy. REv. B., 1995.

[8] J. Garner, M. Jones and P. Plassmann, *Parallel algorithms for modeling superconductors*, in *Computing at the leading edge*, edited by A. Mirin, publication UCRL-TB-111084, National Energy Research Supercomputer Center, 1993.

[9] G. A. Geist and V. S. Sunderam, *The PVM system: Supercomputer level concurrent computation on a heterogeneous network of workstations*, Proceedings of the Sixth Distributed Memory Computing Conference, IEEE, April 1991, 258-261.

[10] Y. Iye, *How anisotropic are the cuprate high Tc superconductors?*, Comments Cond. Mat. Phys., 16 (1992), 89–111.

[11] R. Klemm, A. Luther, and M. Beasley, *Theory of the upper critical field in layered superconductors*, Phys. Rev. B, 12 (1975), 877–891.

[12] W. Lawrence and S. Doniach; Theory of layer structure superconductors, *Proc. 12th Inter. Conf. on Low Temperature Physics*, Academic Press of Japan, Kyoto, 1971, 361-362.

SCALABLE IMPLICIT METHODS FOR REACTION-DIFFUSION EQUATIONS IN TWO AND THREE SPACE DIMENSIONS

SILVIA V. VERONESE* AND HANS G. OTHMER †

Abstract. This paper describes the implementation of a solver for systems of semi-linear parabolic partial differential equations in two and three space dimensions. The solver is based on a parallel implementation of a non-linear Alternating Direction Implicit (ADI) scheme which uses a Cartesian grid in space and an implicit time-stepping algorithm. Various reordering strategies for the linearized equations are used to reduce the stride and improve the overall effectiveness of the parallel implementation. We have successfully used this solver for large-scale reaction-diffusion problems in computational biology and medicine in which the desired solution is a traveling wave that may contain rapid transitions. A number of examples that illustrate the efficiency and accuracy of the method are given here; the theoretical analysis will be presented in [7].

1. Introduction. A large class of models in computational biology and medicine gives rise to systems of non-linear parabolic partial differential equations which may be coupled with ordinary differential equations. Let Ω be a bounded domain in $\Re^m$, $m = 2, 3$, with outward normal n and boundary Γ, and let $\mathbf{u}(\mathbf{x}, t) = (u_1(\mathbf{x}, t), \cdots, u_s(\mathbf{x}, t))$ be the s-component state vector, which is the solution of

$$
\begin{aligned}
\frac{\partial \mathbf{u}}{\partial t} &= \nabla \cdot \mathbf{D} \nabla \mathbf{u} + \mathbf{f}(\mathbf{u}) && \text{for } \mathbf{x} \text{ in } \Omega \\
u_i &= h_i(\mathbf{x}) && \text{for } \mathbf{x} \text{ in } \Gamma_{1i},\ i = 1, \cdots, s \\
\frac{\partial u_i}{\partial n} &= k_i(\mathbf{x}) && \text{for } \mathbf{x} \text{ in } \Gamma_{2i},\ i = 1, \cdots, s \\
\mathbf{u}(\mathbf{x}, 0) &= \mathbf{u_0}(\mathbf{x}) && \text{for } \mathbf{x} \text{ in } \Omega.
\end{aligned} \tag{1}
$$

Here $\Gamma_{1i} \cup \Gamma_{2i} = \Gamma$ for each $i = 1, \cdots, s$. When different components satisfy different types of boundary conditions we have what are called mixed boundary conditions [4]. The reaction term $\mathbf{f}$ is a specified nonlinear function of $\mathbf{u}$, and $\mathbf{D} = diag(d_{pq})$ for $p, q = (1,1), \cdots (s, m)$ is a diagonal diffusion tensor whose first s diagonal components are the diffusion coefficients in the x direction, etc. This allows for the incorporation of anisotropic diffusion, but for simplicity in this paper we only consider diffusion coefficients that are constant in a given direction; the general case will be treated elsewhere. One or more of the diffusion coefficients may vanish identically, in which case the boundary conditions are modified appropriately.

Systems like (1) are often used to model nonlinear wave propagation in excitable media [9]. For example, the Hodgkin-Huxley system, which consists of a single parabolic equation and a system of four ordinary differential equations, models the propagation of a depolarization wave (a nerve impulse) along a nerve axon. Waves of permanent form exist for one-dimensional propagation, but not in two or three space dimensions. Extensive computations in two and three space dimensions have been done on systems of this type, but heretofore large-scale computations have employed

* DEPARTMENT OF MATHEMATICS, UNIVERSITY OF UTAH, SALT LAKE CITY, UT, 84112, USA
E-MAIL: SILVIA@OSIRIS.USI.UTAH.EDU
† DEPARTMENT OF MATHEMATICS, UNIVERSITY OF UTAH, SALT LAKE CITY, UT, 84112, USA
E-MAIL: OTHMER@MATH.UTAH.EDU

explicit time-stepping algorithms. However, it has been shown that explicit methods may require time steps that are several orders of magnitude smaller than those which produce comparable accuracy in an implicit scheme [5], and that explicit methods may produce spurious solutions in excitable systems [6].

In the problems of interest here there may be moving interface regions in which spatial variation of the solution is rapid (compared to the size of the domain). In general the position of the moving interface is not known *a-priori* and is difficult to track, especially when there are several fronts or rotating structures like spirals and rotors in the domain. In our numerical experiments, we found that spatial adaptivity is mainly useful in simple cases where the moving interface is localized to a small portion of the domain. In presence of multiple waves, the computational effort involved in tracking the fronts and regenerating the spatial grid is considerable.

In the following section we present ADI schemes for two and three dimensional problems. In section 3, we present a summary of the test results obtained with sample problems. Section 4 addresses the parallel implementation of the method.

2. An Outline of the Numerical Methods.

2.1. Two space dimensions. In two space dimensions the following split-step method provides a second order scheme for a suitable choice of the weights of the nonlinearity. We assume that the domain is rectangular and rescaled to the unit square. We also assume that $s = 1$, since this simplifies the notation and the extension to systems is straightforward.

$$
(2) \qquad \begin{aligned}
\frac{u^*_{ij} - u^n_{ij}}{h_t} &= \frac{D_1}{2}\delta^2_x u^*_{ij} + \frac{D_2}{2}\delta^2_y u^n_{ij} + \alpha f(u^n_{ij}) + \beta f(u^*_{ij}) \\
\frac{u^{n+1}_{ij} - u^*_{ij}}{h_t} &= \frac{D_1}{2}\delta^2_x u^*_{ij} + \frac{D_2}{2}\delta^2_y u^{n+1}_{ij} + \gamma f(u^n_{ij}) + \delta f(u^{n+1}_{ij})
\end{aligned}
$$

Here h_t is the time step, h_x and h_y are the space steps, $u^n_{ij} \equiv u(ih_x, jh_y, nh_t)$, $\delta_x u^n_{ij} \equiv (u^n_{i+1/2,j} - u^n_{i-1/2,j})/h_x$ (and similarly for $\delta_y u^n_{ij}$), α, β, γ and $\delta \in \Re$, D_1 and D_2 are the diffusion coefficients in the x and y directions, respectively. u^* is an intermediate solution of (2), that may not necessarily satisfy (1).

It is known that in the homogeneous case ($f(u) = 0$) the overall truncation error for (2) is $\mathcal{O}(h_t^2 + max(h_x^2, h_y^2))$. In the non-homogeneous case we can combine the equations in (2) to obtain

$$
(3) \qquad \frac{u^{n+1} - u^n}{h_t} = D_1\delta^2_x u^* + \frac{D_2}{2}\delta^2_y(u^n + u^{n+1}) + \alpha f(u^n) + \beta f(u^*) + \gamma f(u^n) + \delta f(u^{n+1})
$$

In order to simplify the notation, here and hereafter we drop the indices that denote the grid point. By expanding around $u^{n+1/2}_{ij}$, one can show that (2) gives a scheme that is second-order both in time and space. The analysis shows that there are several choices of α, β, and γ, δ that lead to a second-order scheme, one of which is $\beta = \gamma = 0$ and $\alpha = \delta = \frac{1}{2}$. In this case the first equation of (2) becomes linear and can therefore be solved very efficiently, and the second equation contains only one non-linear function evaluation. This is advantageous in many applications in which the evaluation of $f(u)$ is expensive, as for instance, when it involves numerous exponentials.

2.2. Three space dimensions. Unfortunately the straightforward extension of the two-dimensional scheme to three space dimensions is not stable when $h_t/max\{h_x^2, h_y^2\}$ is sufficiently large. In three dimensions we introduce the following scheme:

$$
\begin{aligned}
\frac{u^* - u^n}{h_t} &= \frac{D_1}{2}\delta_x^2(u^* + u^n) + D_2\delta_y^2 u^n + D_3\delta_z^2 u^n + \alpha f(u^n) + \beta f(u^*) \\
\frac{u^{**} - u^n}{h_t} &= \frac{D_1}{2}\delta_x^2(u^* + u^n) + \frac{D_2}{2}\delta_y^2(u^{**} + u^n) + D_3\delta_z^2 u^n + \gamma f(u^n) + \delta f(u^{**}) \\
\frac{u^{n+1} - u^n}{h_t} &= \frac{D_1}{2}\delta_x^2(u^* + u^n) + \frac{D_2}{2}\delta_y^2(u^{**} + u^n) + \frac{D_3}{2}\delta_z^2(u^{n+1} + u^n) + \zeta f(u^n) + \eta f(u^{n+1}).
\end{aligned} \tag{4}
$$

It can be shown that for $\alpha = \beta = \gamma = \delta = 0$ and $\zeta = \eta = 1/2$ the combined equations are a perturbation of the three-dimensional Crank-Nicholson method and therefore second-order correct both in time and space. The analytical properties of the method are discussed in detail in [7].

3. Computational Results. In this section we present some numerical results for problems that demonstrate the performance of the method both for linear and nonlinear systems.

Problem 3.1 This example uses a scalar parabolic equation with anisotropic diffusion on the unit cube Ω. In order to demonstrate the convergence properties of the three dimensional scheme we first solve the following linear problem

$$u_t(\mathbf{x}, t) = \nabla \cdot D\nabla u(\mathbf{x}, t) + f(\mathbf{x}, t) \quad \text{on} \quad \Omega \times \{t > 0\} \tag{5}$$

where $f(u(\mathbf{x}, t)) = u(\mathbf{x}, t)(-\lambda + 4.2\pi^2)$, D is a diagonal tensor with $(d_{11}, d_{21}, d_{31}) = (1.0, 1.4, 1.8)$ and $\lambda = 1$. Initial and homogeneous Newman boundary conditions were chosen so that the exact solution is

$$u(\mathbf{x}, t) = e^{-\lambda t} cos(\pi x) cos(\pi y) cos(\pi z) \tag{6}$$

Thus the solution converges to zero exponentially at a rate λh_t, *i.e.*,

$$u(\mathbf{x}, t + h_t)/u(\mathbf{x}, t) = e^{-\lambda h_t} \tag{7}$$

The problem was solved for different values of the time step h_t ranging from 0.16 to 0.005, and space step $h = h_x = h_y = h_z$ from 0.04 to 0.01. Table 1 reports the behavior of the errors and residual computed at a fixed time according to the following definitions. Let u be the exact solution and u' the approximate solution, then we define the absolute maximum error $= ||u - u'||_\infty$ and the l_2 relative error $= ||u - u'||_2/||u||_2$.

The results of Table 1 confirm that the scheme is second order correct both in time and space. After an initial decrease, the global residual of (5) settles to approximately 10^{-7} as the time and space steps are refined. This is also reflected in the values of the maximum absolute error and relative error. This indicates that the choice of time and space steps is crucial in order to obtain maximum performance. We conclude from these results that time and space steps of approximately the same magnitude are the optimal choice for this second order scheme. Indeed as the time step is decreased, for a fixed space step, the errors and residual do not decrease accordingly, and in some cases even increase. We attribute this to accumulation of roundoff errors. As a further test the approximate value of λ was derived from the decay ratio (7), obtained with the computed solution, and found to agree up to the 8th decimal digit with the exact value.

TABLE 1

The maximum absolute error, relative error and residual for a 3D non-homogeneous linear parabolic problem with anisotropic diffusion coefficients.

	absolute maximum error			l_2 relative error			residual		
h_t/h	4.0e-02	2.0e-02	1.0e-2	4.0e-02	2.0e-02	1.0e-2	4.0e-02	2.0e-02	1.0e-2
1.6e-1	2.71e-01	2.71e-01	2.71e-01	9.95e-01	9.95e-01	9.95e-01	1.68e-01	1.60e-01	1.55e-01
8.0e-2	9.66e-03	9.67e-03	9.68e-03	8.80e-01	8.80e-01	8.80e-01	8.76e-05	8.30e-05	8.06e-05
4.0e-2	2.80e-04	2.69e-04	2.68e-04	1.75e-01	1.70e-01	1.69e-01	1.07e-06	9.98e-07	9.66e-07
2.0e-2	2.65e-05	1.70e-05	1.47e-05	1.98e-02	1.27e-02	1.10e-02	4.91e-07	4.57e-07	4.42e-07
1.0e-2	1.21e-05	2.59e-06	3.26e-07	9.11e-03	1.96e-03	2.47e-04	3.57e-07	3.32e-07	3.22e-07
5.0e-3	1.20e-05	2.49e-06	2.33e-07	9.04e-03	1.89e-03	1.77e-04	2.94e-07	2.80e-07	2.71e-07

Problem 3.2 To test the effects of the non-linear weights we solved problem (5) using the reaction term

$$f(u(\mathbf{x},t)) = -(e^{-\lambda t}cos(\pi x)cos(\pi z)cos(\pi z))^3 + u(\mathbf{x},t)^3 + u(\mathbf{x},t)(-1+4.2\pi^2) \tag{8}$$

In this case the non-linear weights were chosen to be $\alpha = \gamma = 1.0, \beta = \delta = 0.0, \zeta = \eta = 0.5$ This choice generates a second order scheme in which the nonlinearity is solved implicitly only in the third equation of (4). The non-linear systems of equations were solved both by using a direct Newton's method as well as using the *nksol* [3] package. The problem implemented with *nksol* was solved by Newton's method as the basic nonlinear iteration, where the Newton equations were solved only approximately by a linear Krylov iteration. As the Krylov iteration technique we chose the generalized minimum residual method (GMRES) with the linesearch strategy without preconditioning. In both cases the tolerance for the residual was set to 10^{-7}.

In Table 2 we show the maximum absolute error and the relative error of the solution obtained with the Newton's method. As in the previous example, the order of the scheme is maintained. However for large time steps both solvers fail to converge within the limit of 200 non-linear iterations. That there is an optimal relationship between temporal and spatial step sizes is even more apparent in this problem. Indeed, in the numerical experiments we found that $h_t = h/2$ is optimal. Comparing the performance of the two non-linear solvers we found no relevant differences in the accuracy of the solution. However the CPU time spent to solve the problem in serial mode with Newton's method was approximately half the time required for the solution with *nksol*.

TABLE 2

The maximum absolute error and relative error for a 3D non-linear parabolic problem with anisotropic coefficients.

	absolute maximum error			l_2 relative error		
h_t/h	4.0e-02	2.0e-02	1.0e-2	4.0e-02	2.0e-02	1.0e-2
8.0e-2	-	-	-	-	-	-
4.0e-2	2.35e-02	3.17e-02	3.36e-02	2.81e-02	3.83e-02	4.07e-02
2.0e-2	2.51e-03	4.56e-03	6.23e-03	2.91e-03	5.32e-03	7.29e-03
1.0e-2	7.42e-03	5.74e-04	6.23e-03	8.55e-03	6.67e-04	7.29e-03
5.0e-3	8.55e-03	1.75e-03	1.05e-03	9.84e-03	2.03e-03	1.22e-03

Problem 3.3 This example shows that the scheme can be used effectively to compute traveling waves. We extend Fisher's equation, which describes the nonlinear evolution of a population in a

one-dimensional habitat, to three space dimensions. The equation is given by

$$u_t = \nabla \cdot D\nabla u + u(1-u) \tag{9}$$

where D is a tensor of diffusion coefficients. In this test we consider three particular choices of D, namely $D = (d_{11}, d_{21}, d_{31}) = (1,0,0), (0,1,0)$ or $(0,0,1)$. The initial data is chosen to satisfy

$$0 \leq u \leq 1. \tag{10}$$

The general problem, of which (9) is a particular case, has been analyzed theoretically by several authors [1], in studies which show the existence of a minimum wave speed. In [2] the authors have found that an exact traveling solution for (9) is given by

$$u(\mathbf{k}\cdot\mathbf{x} - St) = (1 + e^{(\mathbf{k}\cdot\mathbf{x} - St)\sqrt{6})})^{-2} \tag{11}$$

where $\mathbf{k}$ is the direction of propagation determined by D and S is the wave speed. This speed corresponds to a class of stable minimal wave speed solutions, with exponential decay for $\mathbf{k}\cdot\mathbf{x} \to \infty$. For D chosen as indicated above, it can be shown [2] that the wave speed is $S = 5/\sqrt{(6)}$. In our numerical experiments we set the domain to $-10 \leq x, y, z \leq 40$ to avoid boundary effects that would interfere and modify the profile and speed of the wave. The initial and Neumann boundary condition were given by the exact solution (11). This generates a planar wave that propagates throughout the domain (Fig.1a) in the direction $\mathbf{k}$ parallel to the Cartesian planes. The one-dimensional profile shown in Fig.1a was extracted from the solution at various time steps up to time $t = 20$. The profile was taken in the mid plane parallel to the $z = 0$ plane for a wave propagating in the x direction (X-wave). Fig.1a indicates that the wave propagates from the left to the right of the domain with its structure preserved. Both non-linear solvers were tested and found to perform with comparable accuracy. However due to large memory requirements we were unable to run the problem with *nksol* for grids larger that (150,150,150) points on a DEC Alpha 2100 with 2 Gbytes of memory. In order to test the effects of the nonlinear weights we computed the wave speed from the solution. Fig.1b shows the computed wave speed versus time for the three simulations where the wave was propagating respectively in the x, y and z directions. The weights of the non-linearities were the same as in problem 3.2. The simulations of waves propagating in direction with explicit non-linearity, x and y, showed a minimal increase in the speed of propagation. However after an initial stage the speed was found to converge to a constant value.

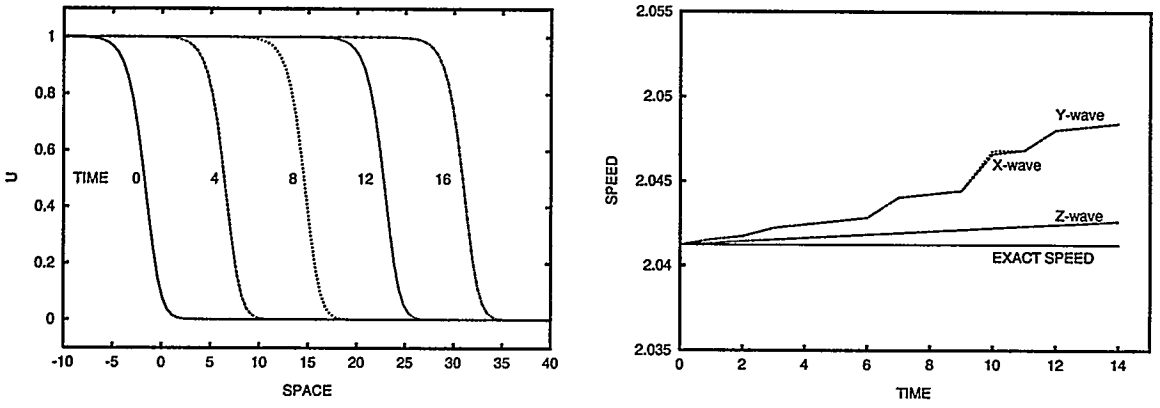


Fig. 1 (a) *A one-dimensional profile extracted from the numerical solution of (9).* (b) *The initial variation of speed for the three wave propagation simulations.*

4. Discussion on parallel implementation. The ADI method consists of solving alternatively in the x,y and z direction at each time step. The particular discretization approach adopted in the code and the ADI implementation permit to point to independent tasks even in the sequential code.

At the most elementary level ADI scheme performs a line by line sweep in all the directions on the entire domain.

In two dimensions the domain sweep is performed by two do-loops (X-sweep and Y-sweep) as the following :

```
do I=1,ny
   solve for row
enddo
```

```
do J=1,nx
   solve for col
enddo
```

In three dimensions the sweep is performed by three nested do-loops as:

```
do J=1,nz
   do I=1,ny
      solve for row
   enddo
enddo
```

```
do J=1,nz
   do I=1,ny
      solve for col
   enddo
enddo
```

```
do J=1,ny
   do I=1,nx
      solve for ver
   enddo
enddo
```

where 'row,col,ver' should be interpreted as one-dimensional problems in the x, y and z direction.

If the storage of the mesh points is done by lines in the x direction then the solution of the X-sweep is equivalent to solving N_y in two dimensions (and $N_y * N_z$ in three dimensions) independent systems. It must be noticed that because of the splitting of the ADI, adjacent 'rows' only require updated values from points laying along the same direction.

On a parallel computer with n processors, the load can then be equidistributed among all the processors since there are no dependencies. Running preliminary tests in a shared memory environment we were able to obtain speed-ups close to peak performance. In a distributed memory environment the amount of communication involved in passing data among different sweeps of the ADI procedure has greatly penalized the performances, even when the load was perfectly balanced. This factor is often cited as the main drawback of ADI in regards to its implementation on distributed memory machines [8].

The Y- and Z- sweeps constitute a system whose stride is greater than nx. However, if we reorder the grid points by lines in the y (and z) direction, we can treat the system in a similar way as before and expect to obtain the maximum speed up.

The code has been implemented on a 12 processor Silicon Graphics Power Challenge and a 3 processor DEC 2100 Alpha. Preliminary results for the parallel implementation of the three-dimensional ADI have shown that, by proper reordering of the unknowns and by applying the coarse grain parallelization described above, we can obtain satisfactory global speedup for several wave propagation problems. Further studies in this direction will be devoted to improve storage

requirement and spatial splitting techniques for large realistic reaction-diffusion models.

5. Conclusions . We have presented a two and three dimensional implementation of a fully-implicit Alternating Direction Implicit method for nonlinear parabolic systems of partial differential equations. In the two dimensional case of dimension $nx*ny$, the solution is reduced to the solution of $nx*ny$ one-dimensional problems. In the three-dimensional case, a problem of dimension $nx*ny*nz$ is solved by $nx * ny + nz * nx + nz * ny$ one-dimensional problems. Results from the examples demonstrate the performance of the method. The first example was used to compare various spatial and time steps, confirming that for the correct choices of the weights of the non-linearities, the scheme is second order accurate both in space and time. The third example indicates that our scheme can be used to solve realistic problems of reaction-diffusion type. The results show the effects of unidirectional propagation in excitable media. In our implementation, the auxiliary storage in the Newton-Krylov iteration accounted for almost half of the total size of the problems. This limitation indicates that Newton-Krylov methods may not be suitable for three-dimensional large problems where resources are limited. Finally our tests show that for wave propagation problems the performance of the Newton method is on the average two to two and a half times faster in solving problems with the same tolerance.

6. Acknowledgements. This work was supported in part by NIH Grant # GM29123. We thank the Utah Supercomputing Institute for the use of the Silicon Graphics Power Challenge.

REFERENCES

[1] I. Petrovsky A. Kolomogorov and N. Piscounoff. Etude de l'equation de la diffusion avec croissance de la quantite de la matiere et son application a un problem biologique. *Moscow University Bull. Math.*, 1:1–25, 1937.

[2] M.J. Ablowitz and A. Zeppetella. Explicit solutions of fisher's equation for a special wave speed. *Dept. Mathematics Clarkson College,Potsdam, NY*, 1979.

[3] P. N. Brown and Y. Saad. Hybrid Krylov methods for nonlinear systems of equations. *SIAM J. Sci. Stat. Comput.*, 11(3):450–481, 1990.

[4] R. Dillon, P. K. Maini, and H. G. Othmer. Pattern formation in generalized Turing systems I. Steady-state patterns in systems with mixed boundary conditions. *J. Math. Biol.*, pages 345–393, 1994.

[5] P. B. Monk. Two dimensional wave propagation in a model of *Dictyostelium discoideum*. In A. V. Holden, M. Markus, and Hans G. Othmer, editors, *Nonlinear Wave Processes in Excitable Media*, chapter 23, pages 245–258. Plenum Press, New York, 1991.

[6] H. G. Othmer and S. Veronese. Current flow and wave propagation in a two-dimensional anisotropic bidomain. *in preparation*, 1996.

[7] H.G. Othmer and S. V. Veronese. An implicit solver for reaction-diffusion equations in three space dimensions. *in preparation*, 1996.

[8] M. H. Schultz S. L. Hohnsson, Y. Saad. Alternating direction methods on multiprocessors. *SIAM J. Sci. Stat. Comput.*, 8(5):686–700, 1987.

[9] S. Veronese and H.G. Othmer. A computational study of wave propagation in a model for anisotropic cardiac ventricular tissue. *Lecture Notes in Computer Science*, 919:248–253, 1995.

Experiences with Linear Solvers for Oil Reservoir Simulation Problems

Wayne Joubert, Los Alamos National Laboratory
Deepankar Biswas, University of Texas
Graham Carey, University of Texas
Raghunandan Janardhan, Los Alamos National Laboratory

This talk will focus on practical experiences with iterative linear solver algorithms used in conjunction with Amoco Production Company's Falcon oil reservoir simulation code. The goal of this study is to determine the best linear solver algorithms for these types of problems. The results of numerical experiments will be presented.

Topic:	*Session Chair:*	*Room C*
Preconditioning	*TBA*	

8:00 - 8:30	X. Wang	A Necessary and Sufficient Symbolic Condition for the Existence of Incomplete Cholesky Factorization
8:30 - 9:00	V. Il'in	Advanced Incomplete Factorization Algorithms for Stiltijes Matrices
9:00 - 9:30	S. Holmgren	Convergence Acceleration for Time-Independent First-Order PDE Using Optimal PNB-Approximations
9:30 - 10:00	T. Huckle	Iterative Methods for Symmetric Ill-Conditioned Toeplitz Matrices

ABSTRACT: A NECESSARY AND SUFFICIENT SYMBOLIC CONDITION FOR THE EXISTENCE OF INCOMPLETE CHOLESKY FACTORIZATION*

XIAOGE WANG AND RANDALL BRAMLEY
DEPARTMENT OF COMPUTER SCIENCE
INDIANA UNIVERSITY - BLOOMINGTON
KYLE A. GALLIVAN
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Incomplete Cholesky factorization (IC) is a widely known and effective method of accelerating the convergence of conjugate gradient (CG) iterative methods for solving symmetric positive definite (SPD) linear systems. A major weakness of IC is that it may break down due to nonpositive pivots. Methods of overcoming this problem can be divided into two classes: numerical and structural strategies. A numerical strategy uses numerical values generated during the factorization process to modify the factorization, as in the work by [Jennings and Malik, 1977, Manteuffel, 1979, Munksgaard, 1980, Wittum and Liebau, 1989]. A structural strategy, as in the work by [Coleman, 1988], selects the sparsity pattern to insure the completion of the IC process. Structural strategies are important for applications where a sequence of linear systems must be solved, each coefficient matrix with the same non-zero pattern. This occurs when solving linear programming problems using interior point methods, or when solving discretized nonlinear partial differential equations with a fixed mesh. Although the values can change from one step to another, the sparsity pattern is fixed.

This talk does not give any specific algorithm for modifying the sparsity pattern to assure the existence of IC. Instead, we present a sufficient and necessary condition on the sparsity pattern of the incomplete Cholesky factor for the existence of IC, for SPD matrices with a given sparsity pattern. This condition, called property C_+, is more general than that in [Coleman, 1988], and includes Coleman's sparsity pattern condition as a special case. The necessity condition says that for a given sparsity pattern Q for a SPD matrix, if the target sparsity pattern P of the incomplete Cholesky factor does not satisfy property C_+, then there is some SPD matrix B with sparsity pattern Q, for which incomplete Cholesky factorization will break down.

REFERENCES

[Coleman, 1988] Coleman, T. (1988). A chordal preconditioner for large-scale optimization. *Mathematical Programming*, 40:265–287.

[Jennings and Malik, 1977] Jennings, A. and Malik, G. M. (1977). Partial elimination. *Journal of the Institute of Mathematics and Its Applications*, 20:307–316.

[Manteuffel, 1979] Manteuffel, T. A. (1979). Shifted incomplete Choleski factorization. In Duff, I. S. and Stewart, G., editors, *Sparse Matrix Proceedings 1978*, Philadelphia,PA. SIAM Publications.

[Munksgaard, 1980] Munksgaard, N. (1980). Solving sparse symmetric sets of linear equations by preconditioned conjugate gradients. *ACM Transactions on Mathematical Software*, 6:206–219.

[Wittum and Liebau, 1989] Wittum, G. and Liebau, F. (1989). On truncated incomplete decompositions. *BIT*, 29:719–740.

* THIS WORK WAS SUPPORTED BY THE NATIONAL SCIENCE FOUNDATION UNDER GRANT NOS. CDA-9309746 AND CCR-9120105

Advanced Incomplete Factorization Algorithms for Stiltijes Matrices

V.P.Il'in
Computing Center, Siberian Division RAS, Novosibirsk, Russia

The modern numerical methods for solving the linear algebraic systems Au = f with high order sparse matrices A, which arise in grid approximations of multidimensional boundary value problems, are based mainly on accelerated iterative processes with easily invertible preconditioning matrices presented in the form of approximate (incomplete) factorization of the original matrix A, see [1], [2] for example. We consider some recent algorithmic approaches, theoretical foundations, experimental data and open questions for incomplete factorization of Stiltijes matrices which are "the best" ones in the sense that they have the most advanced results. Special attention is given to solving the elliptic differential equations with strongly variable coefficients, singular perturbated diffusion-convection and parabolic equations.

In general, the block tridiagonal Stiltijes matrix A=D-L-U with block diagonal or diagonal matrix D, low- and upper-triangular matrices L, U = L' can be approximated by the factorized matrix B=(G - L)G^(G - U), where G is block-diagonal matrix defined by some approximations of Shur complement matrices of A and G^ is inverse of G. We describe two sets of explicit and implicit methods with closure in the sense that in limit B is the exact factorization of A.

There are presented several modifications of the algorithms with various iterative parameters which provide the generalization of wellknown methods and application of extended row sum criteria

$$B\,y_k = A\,y_k, \quad k=1,2,\ldots,$$

the symmetrized alternating direction methods with nonsymmetric preconditioners, adapted "flow-directed" algorithms, some versions of algebraic domain decomposition and hierarhical multigrid approaches. Both theoretical estimates and computational results are included.

References

1. V.P.Il'in. Iterative incomplete factorization methods, Wold Sci. Pub. Co., 1992, 192 pp.
2. V.P.Il'in. Incomplete factorization methods for solution of the algebraic systems (in Russian), Moscow: Nauka, 1995, 288 pp.

Convergence Acceleration for Time-independent First-order PDE using Optimal PNB-approximations

S. Holmgren[‡] and H. Brandén

Department of Scientific Computing, Uppsala University
Box 120, S-751 04 Uppsala
Sweden

January, 1996

Abstract

We consider solving time-independent (steady-state) flow problems in 2D or 3D governed by hyperbolic or "almost hyperbolic" systems of partial differential equations (PDE). Examples of such PDE are the Euler and the Navier-Stokes equations. The PDE is discretized using a finite difference or finite volume scheme with arbitrary order of accuracy. If the matrix B describes the discretized differential operator and u denotes the approximate solution, the discrete problem is given by a large system of equations

$$Bu = g. \tag{1}$$

In many cases, the system is non-linear, and $B = B(u)$. A standard solution procedure, often used in application codes, is defined by solving the *time-dependent* problem corresponding to (1):

$$\frac{\mathrm{d}u}{\mathrm{d}t} = r(u) = g - Bu. \tag{2}$$

The problem (2) is solved using an explicit stationary iterative method in time, e.g., a three- or five-stage Runge-Kutta iteration. The iteration is pursued until steady-state is reached, i.e., until $r(u) \approx 0$.

Normally, some form of convergence acceleration is applied to attain an acceptable rate of convergence. This implies that instead of (2), the following problem is solved

$$\frac{\mathrm{d}u}{\mathrm{d}t} = M^{-1}r(u) = M^{-1}(g - Bu). \tag{3}$$

Here, the matrix M corresponds to a preconditioner. Standard convergence acceleration procedures are, e.g., local time-marching (M is diagonal) and implicit residual smoothing

[‡] This work was supported by the Swedish National Board for Industrial and Technical Development (NUTEK)

(M ia a Kronecker product of tridiagonal matrices, one for each dimension in space). A common property for these methods is that the maximal step k in pseudo-time is determined by a CFL-criterion of the type $k = \mathcal{O}(h_{min})$, where h_{min} is the minimal space-step.

We describe a new convergence acceleration technique, where M is chosen as a *PNB-approximation* [2] [3] of B, or possibly a matrix closely related to B. The matrix M corresponds to a separable PDE-problem, where the discretization in one space direction is constructed so that the corresponding matrix is diagonalized by a unitary transformation. If this transformation is computable using a fast $\mathcal{O}(n \log_2 n)$ algorithm, the resulting convergence acceleration step is of the same complexity. Also, since the solves are based on a dimensional splitting, the intrinsic parallelism is good.

Different choices of the unitary transformation are considered, e.g., the discrete Fourier transform, sine transform, and modified sine transform. The preconditioners fully exploit the structure of the original problem, and in earlier work [2] it is shown how to compute the parameters describing them subject to different optimality constraints.

Numerical experiments for model problems in 2D are presented, where the PNB convergence acceleration procedure is compared to other techniques. The results are very encouraging. When the PNB approximations are utilized, the step k in pseudo-time is not limited by a CFL-criterion of the type described above, and $k = \mathcal{O}(1)$ may be used for all sizes of the spatial grid. In fact, the number of iterations required for convergence actually *decreases* slightly as the number of gridpoints in space increases. Also, the results show that, when exploiting the PNB approximations, the three- or five-stage Runge-Kutta iteration may be exchanged by the one-stage Euler forward iteration without any decrease in convergence rate. Comparing the arithmetic work required for convergence, the conclusion is that for large problems the solution is computed more than 20 times faster than if the standard implicit residual smoothing procedure is used.

The numerical results are related to theoretical results obtained for related problems in earlier work [1], and finally the implementation of the PNB convergence acceleration procedure in existing codes for non-linear application problems is discussed.

References:

[1] S. Holmgren and K. Otto, *Semicirculant preconditioners for first-order partial differential equations*, SIAM J. Sci. Comput., 15 (1994), pp. 385-407.

[2] S. Holmgren and K. Otto, *A framework for polynomial preconditioners based on fast transforms I: Theory*, Report No. 167, Department of Scientific Computing, Uppsala University, Uppsala, Sweden, 1995.

[3] S. Holmgren and K. Otto, *A framework for polynomial preconditioners based on fast transforms II: PDE applications*, Report No. 168, Department of Scientific Computing, Uppsala University, Uppsala, Sweden, 1995.

Iterative methods for symmetric ill-conditioned Toeplitz matrices

Thomas Huckle
Institut für Informatik
TU München
D-80290 München, F.R.G.

Abstract. We consider ill-conditioned symmetric positive definite Toeplitz systems $T_n x = b$. If we want to solve such a system iteratively with the conjugate gradient method, we can use band-Toeplitz-preconditioners or Sine-Transform-peconditioners $M = S_n \Lambda S_n$, S_n the Sine-Transform-matrix and Λ a diagonal matrix.

A Toeplitz matrix $T_n = (t_{i-j})_{i,j=1}^n$ is often related to an underlying function f defined by the coefficients t_j, $j = -\infty, .., -1, , 0, 1, .., \infty$. There are four cases, for which we want to determine a preconditioner M:

- T_n is related to an underlying function which is given explicitly;
- T_n is related to an underlying function that is given by its Fourier coefficients;
- T_n is related to an underlying function that is unknown;
- T_n is not related to an underlying function.

Especially for the first three cases we show how positive definite and effective preconditioners based on the Sine-Transform can be defined for general nonnegative underlying function f. To define M, we evaluate or estimate the values of f at certain positions, and build a Sine-transform matrix with these values as eigenvalues. Then, the spectrum of the preconditioned system is bounded from above and away from zero.

Key Words. Toeplitz matrix, preconditioning, Sine transform.

AMS(MOS) Subject Classifications. 65F10,65F35

Topic:	*Session Chair:*	*Room A*
FOSLS	*Steve McCormick*	

10:30 - 11:00	G. Fix	Application of Least-Squares Principles in Optimal Shape Design Problems
11:00 - 11:30	B. Bloechle	First-Order Least-Squares for Second-Order Elliptic Problems with Discontinuous Coeeficients: Further Results
11:30 - 12:00	J. Pasciak	Least-Squares Methods Involving the H^{-1} Inner Product

Application of Least-Squares Principles in Optimal Shape Design Problems

G. Fix

abstract not available

First-Order System Least-Squares for Second-Order Elliptic Problems with Discontinuous Coefficients: Further Results

B. Bloechle, T. Manteuffel, S. McCormick, and G. Starke

April 9, 1996

Abstract

Many physical phenomena are modeled as scalar second-order elliptic boundary value problems with discontinuous coefficients. The first-order system least-squares (FOSLS) methodology is an alternative to standard mixed finite element methods for such problems. The occurrence of singularities at interface corners and cross-points requires that care be taken when implementing the least-squares finite element method in the FOSLS context. We introduce two methods of handling the challenges resulting from singularities. The first method is based on a weighted least-squares functional and results in non-conforming finite elements. The second method is based on the use of singular basis functions and results in conforming finite elements. We also share numerical results comparing the two approaches.

"Least-squares methods involving the H^{-1} inner product."

author:
Joe Pasciak

Abstract:

Least-squares methods are being shown to be an effective technique for the solution of elliptic boundary value problems. However, the methods differ depending on the norms in which they are formulated. For certain problems, it is much more natural to consider least-squares functionals involving the H^{-1} norm. Such norms give rise to improved convergence estimates and better approximation to problems with low regularity solutions. In addition, fewer new variables need to be added and less stringent boundary conditions need to be imposed. In this talk, I will describe some recent developments involving least-squares methods utilizing the H^{-1} inner product.

Topic: *Parallel*	*Session Chair:* *Paul Saylor*	*Room B*
10:30 - 11:00	S. Hutchinson	AZTEC: A Parallel Iterative Package for the Solving Linear Systems
11:00 - 11:30		
11:30 - 12:00	D. Young	On the Implementation of Parallel Implicit USSOR Methods for Solving Discrete Periodic Problems

AZTEC: A parallel iterative package for the solving linear systems

Scott A. Hutchinson, John N. Shadid, Ray S. Tuminaro
Speaker: Scott A. Hutchinson

MS 1110
Sandia National Labs
PO Box 5800
Albuquerque, NM 87185

We describe a parallel linear system package, AZTEC. The package incorporates a number of parallel iterative methods (e.g. GMRES, biCGSTAB, CGS, TFQMR) and preconditioners (e.g. Jacobi, Gauss-Seidel, polynomial, domain decomposition with LU or ILU within subdomains). Additionally, AZTEC allows for the reuse of previous preconditioning factorizations within Newton schemes for nonlinear methods. Currently, a number of different users are using this package to solve a variety of PDE applications.

In this talk we emphasize the parallel programming ease and the overall efficiency of AZTEC. Ease-of-use is attained using the notion of a global distributed matrix. The global distributed matrix allows a user to specify pieces (different rows for different processors) of an application matrix exactly as in the serial setting. Efficiency is achieved by using a transformation function which rewrites the user supplied matrix into one more convenient for efficient distributed memory computing (locally numbered entries, ghost variables, grouped messages). Additional performance is attained using efficient dense matrix algorithms for block sparse matrices.

AZTEC can be used on the Intel Paragon, nCUBE 2, IBM SP2, individual workstations (e.g. SUN and SGI) and uses the MPI message passing system.

On the Implementation of Parallel Implicit USSOR Methods for Solving Discrete Periodic Problems

David M. Young*, David R. Kincaid, and Wan Chen
Center for Numerical Analysis
The University of Texas at Austin

In this paper, we consider certain iterative methods for solving discrete periodic problems based on the standard 5-point finite difference representation of the Poisson equation in two dimensions. For a given positive integer m, our procedure involves the implementation of m**2 single iterations of the implicit USSOR (unsymmetric SOR) method based on the use of m**2 pairs of relaxation factors, followed by the construction of a linear combination of the results of those iterations. It has previously been shown by the authors that the result obtained by this process is the same as one would obtain by using m iterations of the SSOR methods with varying relaxation factor in sequence. (Thus, there is a potential saving in the wall-clock time of a factor of m.) To carry out the implicit USSOR method, we consider various procedures including the (non-implicit) SOR method. The choice of the optimum relaxation factor for the SOR method and the estimation of the corresponding convergence properties is greatly simplified since a related matrix is p-cyclic.

Topic: *Arnoldi's Method*	*Session Chair:* *TBA*	*Room C*
10:30 - 11:00	F. Raeven	A New Arnoldi Approach for Polynomial Eigenproblems
11:00 - 11:30	R.B. Lehoucq	Re-Starting an Arnoldi Iteration
11:30 - 12:00	I. Elfadel	Stable Reduced-Order Models of Generalized Dynamical Systems Using Coordinate-Transformed Arnoldi Algorithms

A new Arnoldi approach for polynomial eigenproblems

Femke A. Raeven

Abstract

In this paper we introduce a new generalisation of the method of Arnoldi for matrix polynomials. The new approach is compared with the approach of rewriting the polynomial problem into a linear eigenproblem and applying the standard method of Arnoldi to the linearised problem. The algorithm that can be applied directly to the polynomial eigenproblem turns out to be more efficient, both in storage and in computation.

1 Introduction

Large polynomial eigenproblems are usually rewritten into a larger linear eigenproblem in actual computations. The linear problem can be solved with, e.g. the method of Arnoldi [1] or Lanczos [4]. For quadratic problems the Lanczos method is studied in [5]. A disadvantage of these approaches is that the linearised problem is of larger size and the basis vectors of the subspaces involved in these subspace methods are large as well. We will see that the method of Arnoldi can be generalised, such that the basis vectors have the same length as the dimension of the matrices in the matrix polynomial.

In Section 2 we discuss how polynomial problems can be rewritten into linear eigenproblems and how Arnoldi's method is applied to these problems. The generalisation of the Arnoldi method is presented in Section 3. It also becomes clear why less storage is required for this new approach. In Section 4 we explain that it is also more efficient in computation. We illustrate these observations in Section 5 with an experiment.

2 The method of Arnoldi on the linearised form

Suppose we want to solve a polynomial eigenproblem:

$$P(\lambda)u = \sum_{i=0}^{m} \lambda^i A_i u = 0. \tag{1}$$

In this equation the A_i represent $n \times n$ matrices. A solution of this problem is a pair (λ, u) of a scalar eigenvalue λ and an eigenvector u, for which (1) holds. For

simplicity, we assume the polynomial $P(\lambda)$ to be monic: $A_m = I$. Every non-monic problem with non-singular A_m can be transformed into a mathematically equivalent monic one by a multiplication with A_m^{-1}.

In computations, the usual approach is to rewrite (1) as a linear eigenproblem $Au = \lambda u$. If $A_m = I$, then (1) is equivalent to

$$\begin{pmatrix} -A_{m-1} & \cdots & \cdots & -A_1 & -A_0 \\ I & & & & \\ & \ddots & & & \\ & & \ddots & & \\ & & & I & \end{pmatrix} \begin{pmatrix} \lambda^{m-1}u \\ \lambda^{m-2}u \\ \vdots \\ \vdots \\ u \end{pmatrix} = \lambda \begin{pmatrix} \lambda^{m-1}u \\ \lambda^{m-2}u \\ \vdots \\ \vdots \\ u \end{pmatrix}. \quad (2)$$

For this linear problem, we can use the method of Arnoldi.

The method of Arnoldi for general linear eigenproblems $Au = \lambda u$ constructs the Krylov subspace

$$K_k(A; y_0) = span\{y_0, Ay_0, A^2y_0, ..., A^{k-1}y_0\}.$$

The basis vectors $y_j = A^j y_0$ are orthonormalised and collected as the columns of a projection matrix V_k, which projects a vector in $\mathbf{C}^n$ onto the Krylov subspace. After a number of iteration steps, approximations for the eigenvalues and eigenvectors are calculated as the solutions of the projected problem

$$V_k^* A V_k z = \theta z.$$

The scalars θ are called the Ritz values; the vectors z are called the Ritz vectors.

As the Krylov subspace expands, the exterior Ritz values converge to the exterior eigenvalues. More details about the method of Arnoldi can be found in [1] and [7].

3 Using the information more efficiently

A disadvantage of the linearisation approach in Section 2 is that the dimension of the linear eigenproblem is m times as large as the dimension of the original problem, where m is the degree of the matrix polynomial. The amount of memory required to store the projection matrix V_k is $n \cdot m \cdot k$. If n is large, then this amount may be huge. We will see that much of the information in V_k is stored double.

Suppose we start the method of Arnoldi on (2) with the starting vector

$$y_0 = \begin{pmatrix} x_0 \\ \vdots \\ x_{-m+1} \end{pmatrix},$$

where $x_0, ..., x_{-m+1}$ are vectors of length n. Then, the Krylov subspace $K_k(C; y_0)$ with C the matrix from (2) is

$$K_k(C; y_0) = span\{y_0, Cy_0, C^2y_0, C^3y_0, ..., C^{k-1}y_0\} =$$

$$= span\{\begin{pmatrix} x_0 \\ \cdot \\ \cdot \\ x_{-m+1} \end{pmatrix}, \begin{pmatrix} x_1 \\ \cdot \\ \cdot \\ x_{-m+2} \end{pmatrix}, \begin{pmatrix} x_2 \\ \cdot \\ \cdot \\ x_{-m+3} \end{pmatrix}, ..., \begin{pmatrix} x_{k-1} \\ \cdot \\ \cdot \\ x_{-m+k} \end{pmatrix}\}$$

where

$$x_{i+1} = -A_{m-1}x_i - A_{m-2}x_{i-1} - ... - A_0 x_{-m+1+i} = -\sum_{j=0}^{m-1} A_j x_{-m+1+i+j}, \quad (3)$$

for all $i \geq 0$.

Instead of storing y_i, it also suffices to store x_i. We propose the following approach for matrix polynomials:

Generalised Arnoldi for matrix polynomials

- Choose orthonormal starting vectors $x_0, ..., x_{-m+1}$.
- Put $W = [x_0, ..., x_{-m+1}]$.
- iterate:
  - Compute x_{i+1} with the recursion in (3).
  - Orthonormalise x_{i+1} against previous iterates and add this vector to W.
- Solve the projected system $W^* P(\theta) W z = 0$.

Because the x_i's are of the same dimension as the original problem, and because they are approximations for eigenvectors of the original problem, they can be used to project the polynomial. This method will be referred to as the generalised Arnoldi method for matrix polynomials.

We compare the storage requirements for both approaches. The linearisation of a non-monic polynomial problem into a linear problem $Au = \lambda Bu$ is of dimension $m \cdot n$. In the Arnoldi method we need to store V_k, $A \cdot V_k$, and $B \cdot V_k$. The total amount of storage is therefor $3 \cdot k \cdot n \cdot m$. For the generalised Arnoldi method on the same matrix polynomial, we need W_k and $A_i \cdot W_k$ for all $0 \leq i \leq m$. The number of starting vectors is m, so for $k-1$ iterations W is a $n \times (k+m-1)$ matrix. The total amount of storage is $(m+2)(k+m-1)n$, which is, for polynomial problems, smaller than $3knm$ if $k \geq m$.

4 Comparing subspaces

We have seen that the storage requirements can be reduced by the generalised Arnoldi method. We also want to investigate how the approximations compare with the approximations of the standard Arnoldi method on the linearised

eigenproblem. In other words, we want to compare the solutions of

$$W^*P(\theta_g)Wz_g = 0, \tag{4}$$

with the solutions of

$$V^* \begin{pmatrix} -A_{m-1} & \cdots & \cdots & -A_1 & -A_0 \\ I & & & & \\ & \cdot & & & \\ & & \cdot & & \\ & & & I & \end{pmatrix} Vz_s = \theta_s z_s. \tag{5}$$

Since the dimensions of the two problems are not equal, they are not comparable. Equation (4) can be rewritten into the equivalent formulation

$$Q \begin{pmatrix} \lambda^{m-1}y \\ \lambda^{m-2}y \\ \cdot \\ \cdot \\ y \end{pmatrix} = \lambda \begin{pmatrix} \lambda^{m-1}y \\ \lambda^{m-2}y \\ \cdot \\ \cdot \\ y \end{pmatrix}, \tag{6}$$

with

$$Q = \begin{pmatrix} -W^*A_{m-1}W & \cdots & \cdots & -W^*A_1W & -W^*A_0W \\ I & & & & \\ & \cdot & & & \\ & & \cdot & & \\ & & & I & \end{pmatrix}.$$

In the matrix Q, the projections can be taken out to obtain:

$$Q = \begin{pmatrix} W & & & \\ & \cdot & & \\ & & \cdot & \\ & & & W \end{pmatrix}^* C \begin{pmatrix} W & & & \\ & \cdot & & \\ & & \cdot & \\ & & & W \end{pmatrix}; \tag{7}$$

where C is the matrix in (5).

We conclude that problem (4) can be viewed as a projected linear problem with a special projection matrix.

Comparing the two solution methods comes down to a comparison of $span\{V\}$ with m times the direct sum of $span\{W\}$, $\oplus_{i=0}^{m} span\{W\}$. The number of columns of the projection matrix in (7) is m times as large as the number of columns in V. Thus, $\oplus_{i=0}^{m} span\{W\}$ is much larger than $span\{V\}$. In fact $span\{V\} \subset \oplus_{i=1}^{m} span\{W\}$, because each basis vector $(x_i, x_{i-1}, ..., x_{i-m+1})^T \in span\{V\}$ is contained in $\oplus_{i=0}^{m} span\{W\}$, since each $x_i \in span\{W\}$.

This observation shows, that in each iteration step the subspace of the generalised Arnoldi method contains an approximation for the eigenvectors, that is at least as good, but probably better than the approximation by standard Arnoldi method for linearised problem.

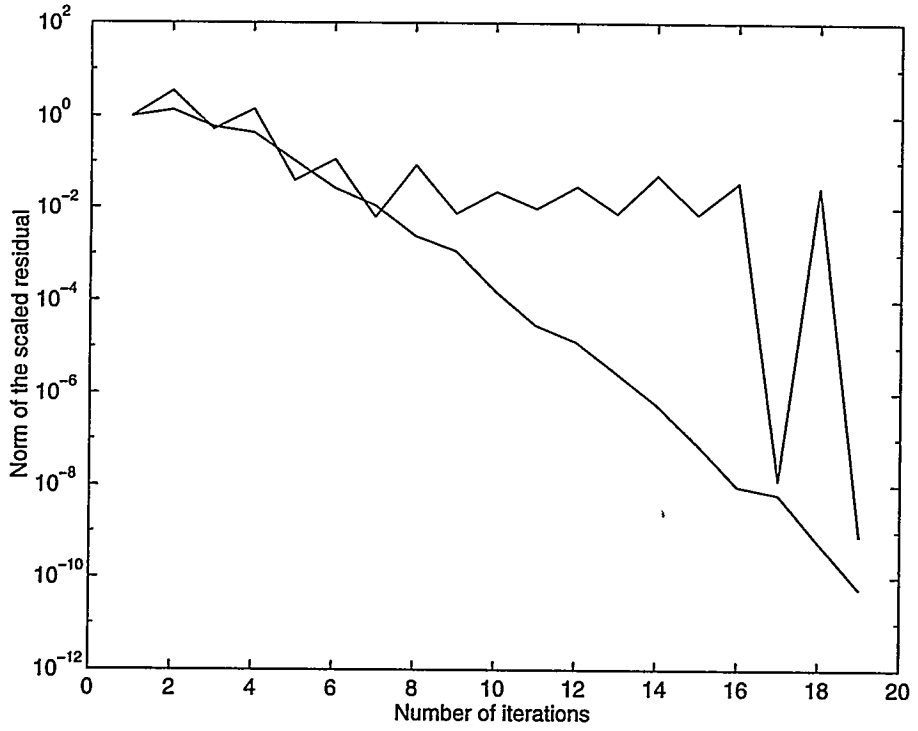


Figure 1: Arnoldi versus Generalised Arnoldi

5 An experiment

We have tested the generalised Arnoldi method on a quadratic problem from mechanics of the form

$$\lambda^2 Mu + \lambda Iu + (M + c \cdot F)u = 0.$$

In this problem M is a diagonal matrix with diagonal elements $M_i = i/n$. The matrix F is a tridiagonal matrix with 2 on its main diagonal and -1 on the co-diagonals.

We solve the equivalent monic problem

$$\lambda^2 Iu + \lambda M^{-1}u + (I + c \cdot M^{-1}F)u = 0.$$

The dimension of the matrices was taken as 5,000. If the scalar c is large, then the exterior eigenvalues are clustered. We have chosen $c = 10,000$.

Figure 1 shows the norm of the scaled residual for both methods. The convergence behaviour of the standard Arnoldi method is rather wild. From Table 1 we see why. Table 1 shows the Ritz value with the largest absolute value in each step. This is expected to be the best approximation for the dominant eigenvalue The generalised Arnoldi method selects a good approximation for the largest

	standard	*generalised*
1	$-1.089 \cdot 10^2 - 9.0965 \cdot 10^1 i$	$-1.4821 \cdot 10^3 + 4.4523 \cdot 10^3 i$
2	$9.9754 \cdot 10^3$	$-2.2499 \cdot 10^3 + 8.6237 \cdot 10^3 i$
3	$-1.5796 \cdot 10^4$	$-2.2954 \cdot 10^3 - 1.0335 \cdot 10^4 i$
4	$2.4515 \cdot 10^4$	$-2.2623 \cdot 10^3 - 1.0463 \cdot 10^4 i$
5	$-1.1796 \cdot 10^4$	$-2.1858 \cdot 10^3 - 1.0608 \cdot 10^4 i$
6	$4.4340 \cdot 10^4$	$-2.1754 \cdot 10^3 - 1.0614 \cdot 10^4 i$
7	$-1.4771 \cdot 10^4$	$-2.1753 \cdot 10^3 + 1.0614 \cdot 10^4 i$
8	$4.4691 \cdot 10^4$	$-2.1753 \cdot 10^3 + 1.0614 \cdot 10^4 i$
9	$-1.2729 \cdot 10^4$	$-2.1753 \cdot 10^3 + 1.0614 \cdot 10^4 i$
10	$3.9214 \cdot 10^4$	$-2.1753 \cdot 10^3 + 1.0614 \cdot 10^4 i$
11	$-1.1769 \cdot 10^4$	$-2.1753 \cdot 10^3 - 1.0614 \cdot 10^4 i$
12	$3.4910 \cdot 10^4$	$-2.1753 \cdot 10^3 - 1.0614 \cdot 10^4 i$
13	$-1.1236 \cdot 10^4$	$-2.1753 \cdot 10^3 - 1.0614 \cdot 10^4 i$
14	$2.9369 \cdot 10^4$	$-2.1753 \cdot 10^3 + 1.0614 \cdot 10^4 i$
15	$-1.1238 \cdot 10^4$	$-2.1753 \cdot 10^3 - 1.0614 \cdot 10^4 i$
16	$2.4815 \cdot 10^4$	$-2.1753 \cdot 10^3 - 1.0614 \cdot 10^4 i$
17	$-2.1753 \cdot 10^3 - 1.0614 \cdot 10^4 i$	$-2.1753 \cdot 10^3 + 1.0614 \cdot 10^4 i$
18	$2.3576 \cdot 10^4$	$-2.1753 \cdot 10^3 - 1.0614 \cdot 10^4 i$
19	$-2.1753 \cdot 10^4 - 1.0614 \cdot 10^4 i$	$-2.1753 \cdot 10^3 - 1.0614 \cdot 10^4 i$

Table 1: Selected Ritz values

eigenvalue from the second iteration. The method of Arnoldi on the linearised form computes Ritz values that are bad approximations for eigenvalues up to iteration 17. The number of iterations to find an approximation for which the norm of the residual is smaller than 10^{-8}, is about the same: 16 versus 19, but the convergence of the generalised Arnoldi method is monotone. This property can be of crucial importance if restarting is used. If we restart after 10 iterations with the best Ritz vector so far as a starting vector, then the standard Arnoldi method needs 61 iterations. The generalised Arnoldi method maintains largely its convergence behaviour after restarting, and needs only 28 iterations.

6 Concluding remarks

We have proposed a generalisation of the method of Arnoldi for polynomial algorithms. We have shown that our approach saves memory and also leads to approximations that are at least as good as for the standard Arnoldi method for the linearised form of the polynomial problem.

Many details about the method still have to be examined. The method, as presented here, computes only exterior eigenvalues for monic problems. A non-monic implementation is required, since most problems are non-monic. We also have to develop a shift-and-invert approach for matrix polynomials for the

computation of interior eigenvalues. On restarting more research has to be done as well.

We have seen that the rather wild convergence behaviour of the Arnoldi method on the linearised form may be smoothened by the generalised Arnoldi method. Probably, this is caused by the fact that the subspace for the generalised Arnoldi method projects the original problem, while the standard Arnoldi method projects the linearised problem, which may be very ill-conditioned. To understand these results in detail, more research is needed.

References

[1] W.E. Arnoldi. The principle of minimized iteration in the solution of the matrix eigenvalue problem. *Quart. Appl. Math.*, 9:17–29, 1951.

[2] I. Gohberg, P. Lancaster, and L. Rodman. *Matrix Polynomials.* Academic Press Inc., New York, 1982.

[3] G.H. Golub and C. van Loan. *Matrix Computations.* North Oxford Academic, Oxford, 1984.

[4] C. Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential an integral operators. *J. Res. Nat. Bur. Standards*, 45:255-282, 1950.

[5] B.N. Parlett and H.C. Chen. Use of indefinite pencils for computing damped natural modes. *Lin. Alg. Appl.*, 140:53-88, 1990.

[6] F.A. Raeven. *Quadratic eigenproblems.* Masters thesis, Utrecht University, June 1995.

[7] Y. Saad. *Numerical methods for large eigenvalue problems.* Manchester University Press, Manchester, 1992

Re-starting an Arnoldi Iteration

R. B. Lehoucq
MCS Division
Argonne National Laboratory
lehoucq@mcs.anl.gov

January 10, 1996

Abstract

The Arnoldi iteration is an efficient procedure for approximating a subset of the eigensystem of a large sparse $n \times n$ matrix A. The iteration produces a partial orthogonal reduction of A into an upper Hessenberg matrix H_m of order m. The eigenvalues of this small matrix H_m are used to approximate a subset of the eigenvalues of the large matrix A. The eigenvalues of H_m improve as estimates to those of A as m increases. Unfortunately, so does the cost and storage of the reduction. The idea of re-starting the Arnoldi iteration is motivated by the prohibitive cost associated with building a large factorization.

This talk considers the various approaches used to re-start the iteration.

In exact arithmetic, all schemes should generate almost the same orthogonal and upper Hessenberg matrices. However, numerical examples are presented that serve to illustrate how the variants may drastically differ in practice. Analysis is presented that explains the generally superior properties of the Sorensen's implicitly re-started Arnoldi iteration, IRA, over all the others. However, Sorensen's approach, as originally proposed, also suffers some numerical difficulties.

Recent work explains these difficulties. The analysis exploits the fact that the IRA iteration is mathematically equivalent to a truncated QR iteration. Thus, general purpose software may be constructed for the numerical solution of the large scale eigenvalue problem.

Stable Reduced-Order Models of Generalized Dynamical Systems Using Coordinate-Transformed Arnoldi Algorithms *

L. Miguel Silveira Mattan Kamon Ibrahim Elfadel Jacob White
Research Laboratory of Electronics
Massachusetts Institute of Technology
Cambridge, MA 02139
e-mail: {lms,matt,elfadel,white@rle-vlsi.mit.edu}

Abstract

Model order reduction based on Krylov subspace iterative methods has recently emerged as a major tool for compressing the number of states in linear models used for simulating very large physical systems (VLSI circuits, electromagnetic interactions). There are currently two main methods for accomplishing such a compression: one is based on the nonsymmetric look-ahead Lanczos algorithm that gives a numerically stable procedure for finding Padé approximations, while the other is based on a less well characterized Arnoldi algorithm. In this paper, we show that for certain classes of generalized state-space systems, the reduced-order models produced by a coordinate-transformed Arnoldi algorithm inherit the stability of the original system. Complete Proofs of our results will be given in the final paper.

1 Introduction

Consider the generalized state space system described by

$$\begin{aligned} \mathcal{L}\dot{\boldsymbol{x}} &= -\mathcal{R}\boldsymbol{x} + \boldsymbol{b}u \\ y &= \boldsymbol{c}^T\boldsymbol{x} \end{aligned} \tag{1}$$

where $\boldsymbol{x}, \boldsymbol{b}, \boldsymbol{C} \in \mathbb{R}^n, y, u \in \mathbb{R}$, $\mathcal{L}, \mathcal{R} \in \mathbb{R}^{n\times n}$, and $\mathcal{R}$ is assumed invertable. This is a single-input/single-output system whose transfer function is given by

$$G(s) = \boldsymbol{c}^T(\boldsymbol{I} - s\boldsymbol{A})^{-1}\boldsymbol{b}_m, \tag{2}$$

where the $n \times n$ matrix $\boldsymbol{A} \equiv -\mathcal{R}^{-1}\mathcal{L}$, and the vector $\boldsymbol{b}_m \equiv \mathcal{R}^{-1}\boldsymbol{b} \in \mathbb{R}^n$. If the complex eigenvalues of $\boldsymbol{A}$ have strictly negative real parts, then the system (1) and the transfer function (2) are said to be stable. Stable systems such as (1) arise in a variety of engineering contexts, particularly circuit analysis and simulation [1, 2, 3, 4, 5, 6].

*This work was supported by ARPA under contracts DABT63-94-C-0053 and N00174-93-C-0035, NSF under contract 9117724-MIP, and the Semiconductor Research Corporation under contract SJ-558.

In a wide variety of applications, the dimension of the model given by (1) or (2) can be tens of thousands, and is part of a large nonlinear dynamical simulation. Thus, the nonlinear simulation can be made much more efficient if a reduced-order model is used in place of (1). The most common and simplest way to compute a reduced-order model for $G(s)$ is to use a Padé approximation. Expanding (2) into a McLaurin series we get

$$G(s) = \boldsymbol{c}^T (I - s\boldsymbol{A})^{-1} \boldsymbol{b}_m = \sum_{k=0}^{\infty} m_k s^k. \tag{3}$$

where

$$m_k = \boldsymbol{c}^T \boldsymbol{A}^k \boldsymbol{b}_m \tag{4}$$

is known as the k^{th} moment of the transfer function. A Padé approximation of q^{th} order is then defined as the rational function

$$G_q^P(s) = \frac{b_{q-1}s^{q-1} + \cdots + b_1 s + b_0}{a_q s^q + a_{q-1}s^{q-1} + \cdots + a_1 s + 1} \tag{5}$$

whose coefficients are selected to match the first $2q-1$ moments of the transfer function.

Padé approximates can be computed using direct evaluation of the moments, followed by a moment-matching procedure [1]. Unfortunately this approach is known to be ill-conditioned as obtaining the moments involves the computation of the quantities $\boldsymbol{A}^k \boldsymbol{b}_m$, which corresponds to vector iterations with the matrix $\boldsymbol{A}$. Instead, nonsymmetric Lanczos-style algorithms can be used [7, 5]. Such algorithms avoid the power iterations by recovering information about more than one eigenvalue of the matrix, thus allowing higher-order approximations to be obtained and higher accuracy to be achieved. They generate a sequence of bi-orthogonal vectors thereby reducing the matrix $\boldsymbol{A}$ to a sequence of tridiagonal matrices which are successively better approximations to the original system. It can be shown that these are in fact exactly the Padé approximations and that the pole-residue representation of the q^{th} order Padé approximant, (5), can be obtained by running q iterations of the non-symmetric Lanczos algorithm and by computing an eigendecomposition of the tridiagonal matrix produced.

An alternative approach to using the nonsymmetric Lanczos algorithm, which robustly generates an approximation somewhat different from Padé's, can be derived using an Arnoldi process [8, 2]. Herein, the idea is to select an orthonormal basis for the Krylov subspace $\mathcal{K}_k(\boldsymbol{A}, \boldsymbol{b}_m) = span\{\boldsymbol{b}_m, \boldsymbol{A}\boldsymbol{b}_m, \boldsymbol{A}^2\boldsymbol{b}_m, \cdots, \boldsymbol{A}^{k-1}\boldsymbol{b}_m\}$. Similar to the Lanczos process, the Arnoldi algorithm is a better conditioned process than direct evaluation of the moments because it generates an orthogonal set of vectors which span $\boldsymbol{A}^k\boldsymbol{b}_m, k = 0, \ldots, 2q-1$.

After q steps, the Arnoldi algorithm returns a set of q orthonormal vectors, as the columns of the matrix $\boldsymbol{V}_q \in \mathbb{R}^{m \times q}$, and a $q \times q$ upper Hessenberg matrix $\boldsymbol{H}_q$. These two matrices satisfy the following relationship

$$\boldsymbol{A}\boldsymbol{V}_q = \boldsymbol{V}_q \boldsymbol{H}_q + h_{q+1,q}\boldsymbol{v}_{q+1}\boldsymbol{e}_q^T \tag{6}$$

where $\boldsymbol{e}_q$ is the q^{th} unit vector in $\mathbb{R}^{m \times m}$, and $\boldsymbol{v}_{q+1} \in \mathbb{R}^n$ is a vector of unit norm orthogonal to the q columns of $\boldsymbol{V}_q$.

From (6), it can easily be seen that after q steps of an Arnoldi process, for $k < q$,

$$\boldsymbol{A}^k\boldsymbol{b}_m = \|\boldsymbol{b}_m\|_2 \boldsymbol{A}^k \boldsymbol{V}_q \boldsymbol{e}_1 = \|\boldsymbol{b}_m\|_2 \boldsymbol{V}_q \boldsymbol{H}_q^k \boldsymbol{e}_1. \tag{7}$$

With this relation, the moments (4) can be related to $\boldsymbol{H}_q$ by

$$m_k = \boldsymbol{c}^T \boldsymbol{A}^k \boldsymbol{b}_m = \underbrace{\|\boldsymbol{b}_m\| \boldsymbol{c}^T \boldsymbol{V}_q}_{\boldsymbol{c}_r} \underbrace{\boldsymbol{H}_q^k}_{\boldsymbol{A}_r^k} \underbrace{\boldsymbol{e}_1}_{\boldsymbol{b}_r} \tag{8}$$

for $k < q$. So, by analogy with (4), the q^{th} order Arnoldi-based approximation to $G(s)$ can be written as

$$\boldsymbol{G}_q^A(s) = \|\boldsymbol{b}_m\|_2 \boldsymbol{c}^T \boldsymbol{V}_q (\boldsymbol{I} - s\boldsymbol{H}_q)^{-1} \boldsymbol{e}_1. \tag{9}$$

where $\boldsymbol{G}_q^A(s)$ matches roughly half the number of moments of the q^{th} order Padé approximation, $\boldsymbol{G}_q^P(s)$. As we will show in the subsequent sections, there is a reward for this factor of two loss in efficiency, $\boldsymbol{G}_q^A(s)$ inherits certain stability properties of A, where as $\boldsymbol{G}_q^P(s)$ does not. This is a direct result of a corollary derived from (6), that

$$\boldsymbol{V}_q^T \boldsymbol{A} \boldsymbol{V}_q = \boldsymbol{H}_q, \tag{10}$$

which means that the matrices $\boldsymbol{A}$ and $\boldsymbol{H}_q$ are related by a congruence transform [6].

2 Main Results

As observed in [6], an important consequence of (10) is that if the matrix $\boldsymbol{A}$ is negative definite then so is the matrix $\boldsymbol{H}_q$. In general, negative definiteness implies stability, but for normal matrices the properties are equivalent. This implies that if $\boldsymbol{A}$ is either negative definite, or normal and stable, then $\boldsymbol{H}_q$ is stable regardless of the model order. However, although the matrices $\mathcal{L}$ and $\mathcal{R}$ are frequently symmetric and positive definite, the matrix system $\boldsymbol{A} = -\mathcal{R}^{-1}\mathcal{L}$ is not necessarily normal or negative definite. Since matrix stability is a property of the spectrum, it is basis independent, but negative definiteness and normality depend on the basis chosen for the state space $\mathbb{R}^n$. A natural question then is whether there is a change of coordinates in the state space such that the resulting system matrix is negative definite. If $\mathcal{L}$ and $\mathcal{R}$ are symmetric and positive definite, the answer is indeed affirmative as can be seen from the change of variable $\boldsymbol{z} = \mathcal{R}^{\frac{1}{2}}\boldsymbol{x}$ where $\mathcal{R}^{\frac{1}{2}}$ is the unique symmetric, positive, definite square root of the symmetric, positive, definite matrix $\mathcal{R}$. From this it follows that (1) can be written as

$$(\mathcal{R}^{-\frac{1}{2}}\mathcal{L}\mathcal{R}^{-\frac{1}{2}})\dot{\boldsymbol{z}} = -\boldsymbol{z} + \mathcal{R}^{-\frac{1}{2}}\boldsymbol{b}u, \tag{11}$$

and that the output equation becomes $y = \boldsymbol{c}^T\mathcal{R}^{-\frac{1}{2}}\boldsymbol{z}$. The modified system matrices are now given by

$$\tilde{\boldsymbol{A}} = -\mathcal{R}^{-\frac{1}{2}}\mathcal{L}\mathcal{R}^{-\frac{1}{2}}, \quad \tilde{\boldsymbol{b}} = \mathcal{R}^{-\frac{1}{2}}\boldsymbol{b}, \quad \tilde{\boldsymbol{c}}^T = \boldsymbol{c}^T\mathcal{R}^{-\frac{1}{2}}. \tag{12}$$

As it can be easily verified, moments are invariant under a change of coordinates in the state space. Therefore, a reduced order model that matches the moments of (12) will also match the moments of the original system.

Note that in the modified model, the matrix $\tilde{\boldsymbol{A}}$ has the interesting properties of being both symmetric and negative definite. Applying the Arnoldi algorithm to the Krylov subsapce $\mathcal{K}_q(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}})$ will result in matrices $\tilde{\boldsymbol{V}}_q$ and $\tilde{\boldsymbol{H}}_q$ such that

$$\tilde{\boldsymbol{V}}_q^T \tilde{\boldsymbol{A}} \tilde{\boldsymbol{V}}_q = \tilde{\boldsymbol{H}}_q \tag{13}$$

where now the matrix $\tilde{\boldsymbol{H}}_q$ shares with $\tilde{\boldsymbol{A}}$ the properties of symmetry and negative definiteness. Moreover, $\tilde{\boldsymbol{H}}$ is tridiagonal because it is upper Hessenberg and symmetric. We assemble these results in the following

Theorem 2.1 *Under the assumption that the state space model is given by (1) where both $\mathcal{L}$ and $\mathcal{R}$ are symmetric postive definite, and for any integer $q \geq 1$, there exists a stable reduced-order model of order q with a tridiagonal, symmetric, negative definite, system matrix $\tilde{\boldsymbol{H}}_q$. The transfer function of the reduced model is given by*

$$\boldsymbol{G}_q^{\tilde{A}}(s) = \|\mathcal{R}^{-\frac{1}{2}}\boldsymbol{b}\|_2 \boldsymbol{c}^T \mathcal{R}^{-\frac{1}{2}} \tilde{\boldsymbol{V}}_q \left(\boldsymbol{I} - s\tilde{\boldsymbol{H}}_q\right)^{-1} \boldsymbol{e}_1, \tag{14}$$

where $\mathcal{R}^{\frac{1}{2}}$ is the unique symmetric, positive definite, square root of the matrix $\mathcal{R}$.

In summary, model-order reduction for these systems using the Arnoldi algorithm is guaranteed stable at any order. The form of the above transfer function might lead to the belief that the computation of the matrix $\mathcal{R}^{\frac{1}{2}}$, potentially a costly operation, is needed. But just as in efficient preconditioned Conjugate-Gradient algorithms [9], the explicit computation of the matrix square-root can be hidden in the algorithm's inner products. In the next paragraph, we describe this technique.

Square-root-free Arnoldi iteration In order to obtain the stable transfer function (14), the Arnoldi algorithm was applied to the Krylov subspace $\mathcal{K}_q(-\mathcal{R}^{-\frac{1}{2}}\mathcal{L}\mathcal{R}^{-\frac{1}{2}}, \mathcal{R}^{-\frac{1}{2}}\boldsymbol{b})$. In fact, the stable transfer function can be obtained *without* the explicit computation of $\mathcal{R}^{-\frac{1}{2}}$. More precisely, we can show that the Arnoldi iteration on $\mathcal{K}_q(-\mathcal{R}^{-\frac{1}{2}}\mathcal{L}\mathcal{R}^{-\frac{1}{2}}, \mathcal{R}^{-\frac{1}{2}}\boldsymbol{b})$ is related through an invertible linear transformation to an Arnoldi iteration on the Krylov subspace $\mathcal{K}_q(-\mathcal{L}\mathcal{R}^{-1}, \boldsymbol{b})$.

In addition to being "square-root-free", an interesting feature of the Arnoldi process on the latter Krylov subspace is that it uses as a dot product the one induced by the matrix $\boldsymbol{P} \equiv \mathcal{R}^{-1}$ rather than the canonical dot product of $\mathbb{R}^n$. The basis constructed by the Arnoldi iteration will be orthonormal with respect to the matrix $\boldsymbol{P}$. Finally, we will show that the stable transfer function (14) can be entirely determined using the Arnoldi iteration in $\mathcal{K}_q(-\mathcal{L}\mathcal{R}^{-1}, \boldsymbol{b})$.

Correspondence: We now give explicitly the correspondence between the Arnoldi iteration for the Krylov subspace $\mathcal{K}_q(-\mathcal{R}^{-\frac{1}{2}}\mathcal{L}\mathcal{R}^{-\frac{1}{2}}, \mathcal{R}^{-\frac{1}{2}}\boldsymbol{b})$ and the $\boldsymbol{P}$-orthogonal Arnoldi iteration for the Krylov subspace $\mathcal{K}_q(-\mathcal{L}\mathcal{R}^{-1}, \boldsymbol{b})$. Let q be an integer $\leq n$. For $1 \leq k \leq q$, denote by $\boldsymbol{V}_k$ and $\boldsymbol{H}_k$ the output matrices of the Arnoldi algorithm and by $\boldsymbol{U}_k$ and $\boldsymbol{K}_k$ the output matrices of the $\boldsymbol{P}$-orthogonal Arnoldi algorithm at the k-th iteration. Recall that these matrices satisfy the equalities

$$-\mathcal{R}^{-\frac{1}{2}}\mathcal{L}\mathcal{R}^{-\frac{1}{2}}\boldsymbol{V}_k = \boldsymbol{V}_k\boldsymbol{H}_k + h_{k+1,k}\boldsymbol{v}_{k+1}\boldsymbol{e}_k^T \qquad \boldsymbol{V}_k^T\boldsymbol{V}_k = \boldsymbol{I}_k \tag{15}$$

$$-\mathcal{L}\mathcal{R}^{-1}\boldsymbol{U}_k = \boldsymbol{U}_k\boldsymbol{K}_k + k_{k+1,k}\boldsymbol{u}_{k+1}\boldsymbol{e}_k^T \qquad \boldsymbol{U}_k^T\boldsymbol{P}\boldsymbol{U}_k = \boldsymbol{I}_k \tag{16}$$

Proposition 2.2 *The two Arnoldi iterations on the Krylov subspaces $\mathcal{K}_q(-\mathcal{R}^{-\frac{1}{2}}\mathcal{L}\mathcal{R}^{-\frac{1}{2}}, \mathcal{R}^{-\frac{1}{2}}\boldsymbol{b})$ and $\mathcal{K}_q(-\mathcal{L}\mathcal{R}^{-1}, \boldsymbol{b})$, respectively, are related by the equations $\boldsymbol{V}_k = \mathcal{R}^{-\frac{1}{2}}\boldsymbol{U}_k$ and $\boldsymbol{H}_k = \boldsymbol{K}_k$.*

Transfer Function: We would like now to show that the state-space representation and the transfer function of the reduced-order model could be entirely determined from the outputs of the $\boldsymbol{P}$-orthogonal Arnoldi process on the Krylov subspace $\mathcal{K}_q(-\mathcal{L}\mathcal{R}^{-1}, \boldsymbol{b})$.

After q steps, the $\boldsymbol{P}$-orthogonal Arnoldi algorithm applied to the Krylov subspace $\mathcal{K}_q(-\mathcal{L}\mathcal{R}^{-1}, \boldsymbol{b})$ returns a set of q $\boldsymbol{P}$-orthonormal vectors, as the columns of the matrix $\boldsymbol{U}_q \in \mathbb{R}^{m\times q}$, and a $q\times q$ tridiagonal matrix $\tilde{\boldsymbol{H}}_q$. These two matrices satisfy the following relationship

$$-\mathcal{L}\mathcal{R}^{-1}\boldsymbol{U}_q = \boldsymbol{U}_q\tilde{\boldsymbol{H}}_q + \tilde{h}_{q+1,q}\boldsymbol{u}_{q+1}\boldsymbol{e}_q^T \tag{17}$$

where $\boldsymbol{e}_q$ is the q^{th} unit vector in $\mathbb{R}^{m\times m}$.

From (17) and the fact that $\boldsymbol{U}_q\boldsymbol{e}_1 = \boldsymbol{b}/\|\boldsymbol{b}\|_{\mathrm{P}}$ ($\|.\|_{\mathrm{P}}$ is the norm induced by the matrix $\boldsymbol{P}$), it can be easily seen that

$$-\mathcal{L}\mathcal{R}^{-1}\boldsymbol{b} = \|\boldsymbol{b}\|_{\mathrm{P}}(-\mathcal{L}\mathcal{R}^{-1})\boldsymbol{U}_q\boldsymbol{e}_1 = \|\boldsymbol{b}\|_{\mathrm{P}}\boldsymbol{U}_q\tilde{\boldsymbol{H}}_q\boldsymbol{e}_1.$$

Multiplying through by $\mathcal{R}^{-1}$, we get $\boldsymbol{A}\left(\mathcal{R}^{-1}\boldsymbol{b}\right) = \|\boldsymbol{b}\|_{\mathrm{P}}\mathcal{R}^{-1}\boldsymbol{U}_q\tilde{\boldsymbol{H}}_q\boldsymbol{e}_1$. Note that the vector $\mathcal{R}^{-1}\boldsymbol{b}$ corresponds to the input vector of a transfer function associated with the state-space representation (1). Therefore, the first moment m_1 is given by

$$m_1 = \boldsymbol{c}^T\boldsymbol{A}\mathcal{R}^{-1}\boldsymbol{b} = \|\boldsymbol{b}\|_{\mathrm{P}}\boldsymbol{c}^T\mathcal{R}^{-1}\boldsymbol{U}_q\tilde{\boldsymbol{H}}_q\boldsymbol{e}_1.$$

Since $\tilde{\boldsymbol{H}}_q$ is an upper Hessenber matrix, $\boldsymbol{e}_q^T\tilde{\boldsymbol{H}}_q^k\boldsymbol{e}_1 = 0,\ 1 \le k < q$. Therefore, the power iterates of $\boldsymbol{A}$ and $\tilde{\boldsymbol{H}}_q$ are related by

$$\boldsymbol{A}^k\mathcal{R}^{-1}\boldsymbol{b} = \|\boldsymbol{b}\|_{\mathrm{P}}\mathcal{R}^{-1}\boldsymbol{U}_q\tilde{\boldsymbol{H}}_q^k\boldsymbol{e}_1, \quad 0 \le k < q.$$

In other words, if we choose the triplet $\left(\tilde{\boldsymbol{H}}_q, \boldsymbol{e}_1, \|\boldsymbol{b}\|_2\boldsymbol{U}_q^T\mathcal{R}^{-1}\boldsymbol{c}\right)$ for the reduced-order model, we could match q moments of the original model.

With this choice of the reduced state-space representation, the reduced transfer function an be written as

$$\tilde{\boldsymbol{G}}_q^A(s) = \|\boldsymbol{b}\|_{\mathrm{P}}\boldsymbol{c}^T\mathcal{R}^{-1}\boldsymbol{U}_q\left(\boldsymbol{I}_q - s\tilde{\boldsymbol{H}}_q\right)^{-1}\boldsymbol{e}_1 \tag{18}$$

The transfer functions (9) and (18) differ in that the upper Hessenberg matrix in (18) is now guaranteed symmetric, therefore tridiagonal, and stable. The presence of the matrix $\mathcal{R}^{-1}(=\boldsymbol{P})$ can be construed a result of endowing the state space $\mathbb{R}^n$ with the $\boldsymbol{P}$-induced dot product. Note also that the $\boldsymbol{U}_q$ matrix is $\boldsymbol{P}$-orthonormal rather than orthonormal like the $\boldsymbol{V}_q$ matrix. We finally state the following result about the nature of the coefficients of the tridiagonal matrix $\boldsymbol{H}_q$.

Corollary 2.3 *The output matrix $\boldsymbol{H}_q$ of the $\boldsymbol{P}$-orthogonal Arnoldi algorithm applied to the Krylov subspace $\mathcal{K}_q(-\mathcal{L}\mathcal{R}^{-1}, \boldsymbol{b})$ has negative coefficients on the diagonal and positive coefficients on the subdiagonals.*

This corollary could for instance be used to check up the correctness of the algorithm implementation.

3 Conclusions and Acknowledgments

In this paper, we demonstrated that for certain types of generalized state-space systems, the Arnoldi-based model-order reduction algorithms have the numerically and physically important feature of generating reduced order models that inherit the stability properties of the original system at any order. This result was obtained via a coordinate transformation in the state space of the original system and the application of the Arnoldi algorithm to the transformed Krylov subspaces. It is worth noting that our stability result holds when block versions of the Arnoldi algorithm are used. This is typically the case when the original system is multi-input/multi-output.

The authors wish to acknowledge the very helpful discussions with Dr. Peter Feldmann and Dr. Roland Freund of the A.T. & T research center.

References

[1] Lawrence T. Pillage and Ronald A. Rohrer. Asymptotic Waveform Evaluation for Timing Analysis. *IEEE Transactions on Computer-Aided Design*, 9(4):352–366, April 1990.

[2] L. M Silveira, M. Kamon, and J. K. White. Direct computation of reduced-order models for circuit simulation of 3-d interconnect structures. In *Proceedings of the 3nd Topical Meeting on Electrical Performance of Electronic Packaging*, pages 254–248, Monterey, California, November 1994.

[3] K. Gallivan, E. Grimme, and P. Van Dooren. Multi-point padé approximants of large-scale systems via a two-sided rational krylov algorithm. In 33rd *IEEE Conference on Decision and Control*, December 1994.

[4] L. M. Silveira, M. Kamon, I. M. Elfadel, and J. White. Coupled circuit-interconnect analysis using Arnoldi-based model order reduction. *IEEE Trans. on CAD*, 1995. Submitted.

[5] P. Feldmann and R. W. Freund. Efficient linear circuit analysis by Pad´e approximation via the Lanczos process. *IEEE Iransactions on Computer-Aided Design of Integrated Circuits and Systems*, 14:639–649, 1995.

[6] K. J. Kerns, I. L. Wemple, and A. T. Yang. Stable and efficient reduction of substrate model networks using congruence transforms. In *IEEE/ACM International Conference on Computer Aided Design*, pages 207 – 214, San Jose, CA, November 1995.

[7] W. B. Gragg. Matrix interpretations and applications of the continued fraction algorithm. *Rocky Mountain Journal of Mathematics*, 4:213 – 225, 1974.

[8] Y. Saad. Overview of krylov subspace methods with application to control problems. In *Interntaional Symposium on the Mathematical Theory of Networks and Systems*, pages 401–410, Boston, MA, 1990. Birkhauser.

[9] G. H. Golub and C. F. Van Loan. *Matrix Computation.* The Johns Hopkins University Press, second edition, 1989.

Topic:	*Session Chair:*	*Room A*
FOSLS	*Steve McCormick*	

4:45 - 5:15	B. Lee	First-Order System Least-Squares for the Hemholtz Equation
5:15 - 5:45	C-Y Lai	Multilevel Solvers of First-Order System Least Squares for Stokes Equations
5:45 - 6:15	W.T. Mahavier	A Numerical Method for Solving Singular DE's
6:15 - 6:45	M.A. Noor	Generalized Quasi Variational Inequalities

First-Order System Least-Squares for the Helmholtz Equation

B. Lee, T. Manteuffel, S. McCormick, and J. Ruge

January 15, 1996

Abstract. We apply the FOSLS methodology to the exterior Helmholtz equation $\Delta p + k^2 p = 0$. Several least-squares functionals, some of which include both $H^{-1}(\Omega)$ and $L^2(\Omega)$ terms, are examined. We show that in a special subspace of $[\mathbf{H}(\mathrm{div};\Omega) \cap \mathbf{H}(\mathbf{curl};\Omega)] \times H^1(\Omega)$, each of these functionals are equivalent independent of k to a scaled $H^1(\Omega)$ norm of p and $\mathbf{u} = \nabla p$. This special subspace does not include the oscillatory near-nullspace components $\mathbf{c}e^{\imath k(\alpha x+\beta y)}$, where $\mathbf{c}$ is a complex vector and where $\alpha^2 + \beta^2 = 1$. These components are eliminated by applying a non-standard coarsening scheme. We achieve this scheme by introducing "ray" basis functions which depend on the parameter pair (α, β), and which approximate $\mathbf{c}e^{\imath k(\alpha x+\beta y)}$ well on the coarser levels where bilinears cannot. We use several pairs of these parameters on each of these coarser levels so that several coarse grid problems are spun off from the finer levels.

Some extensions of this theory to the transverse electric wave solution for Maxwell's equations will also be presented.

Multilevel Solvers of First-Order System Least-Squares for Stokes Equations

Chen-Yao G. Lai
email: cylai@math.ccu.edu.tw.

Department of Mathematics,
National Chung Cheng University,
No. 160, San-Hsing Village,
Ming-Hsiung, Chia-Yi 621,
Taiwan.

Recently, The use of first-order system least squares principle for the approximate solution of Stokes problems has been extensively studied by Cai, Manteuffel, and McCormick [1], [2]. In this paper, we study multilevel solvers of first-order system least-squares method for the generalized Stokes equations based on the velocity-vorticity-pressure formulation in three dimensions. The least-squares functionals is defined to be the sum of the L^2-norms of the residuals, which is weighted appropriately by the Reynolds number. We develop convergence analysis for additive and multiplicative multilevel methods applied to the resulting discrete equations.

REFERENCES

[1] Z. Cai, T. Manteuffel, and S. McCormick. First–order system least squares for the stokes equations, with application to linear elasticity. *The Seventh Copper Mountain Conference on Multigrid Methods.*

[2] Z. Cai, T. Manteuffel, and S. McCormick. First–order system least squares for velocity-vorticity-pressure form of the stokes equations, with application to linear elasticity. *The Seventh Copper Mountain Conference on Multigrid Methods.*

A Numerical Method for Solving Singular De's

William Ted Mahaver

A numerical method is developed for solving singular differential equations using steepest descent based on weighted Sobolev gradients. The method is demonstrated on a variety of first and second order problems, including linear constrained, unconstrained, and partially constrained first order problems, a nonlinear first order problem with irregular singularity, and two second order variational problems.

The method is an extension of steepest descent in Sobolev spaces which is a variation of descent based on the Euclidean gradient. The differential equation is cast as a least-squares problem yielding a functional representing the equation. A weighted Sobolev space for the problem is chosen where the weights are based on the functional. The gradients associated with the functional take into account both the weights and the boundary conditions for the given equation.

Results are presented which demonstrate the improvements obtained by computing based on weighted Sobolev gradients rather than computing based on either unweighted Sobolev gradients or on the Euclidean gradient.

Generalized Quasi Variational Inequalities

Muhammad Aslam Noor
Mathematics Department, College of Science,
P.O. Box 2455, King Saud University, Riyadh 11451, Saudi Arabia
E-mail: F40M040@SAKSU00.BITNET

Abstract. In this paper, we establish the equivalence between the generalized quasi variational inequalities and the generalized implicit Wiener-Hopf equations using essentially the projection technique. This equivalence is used to suggest and analyze a number of new iterative algorithms for solving generalized quasi variational inequalities and the related complementarity problems. The convergence criteria is also considered. The results proved in this paper represent a significant improvement and refinement of the previously known results.

1. Introduction

Variational inequality theory has emerged as a novel and powerful technique for studying a wide class of unrelated linear and nonlinear problems arising in various fields of pure and applied sciences in a unified framework. This theory enables us to develop highly efficient and powerful new numerical methods to solve, for example, free and moving boundary value problems and the general equilibrium problems. It has been show that the variational inequality theory provides the most natural, direct, simple and efficient framework for a unified treatment of all the equilibrium type problems. For recent state of the art in this field, see Harker and Pang [6] and Noor-Noor and Rassias [12,13]. In recent years, various extensions and generalizations of variational inequalities have been considered and studied. A useful and important generalization is known as the quasi variational inequality originally considered and studied by Bensoussan and Lions [2] in impulse control. Fang and Peterson [4] introduced another useful generalization known as generalized variational inequality. Chan and Pang [3] considered the generalized quasi variational inequality problem, which unifies these both concepts, see [1-17] and the references therein.

One of the most interesting and difficult problems in the variational inequality theory is the development of an efficient iterative algorithm. There is a substantial number of algorithms for finding the numerical solution of variational inequalities. Among the most effective numerical algorithms are projection methods and its variant forms. On the other hand, there is no such method for solving generalized quasi variational inequalities except those of Chan and Pang [3] and Noor [9]. In this paper, we establish the equivalence between generalized quasi variational inequalities and generalized implicit Wiener-Hopf

equations by using the projection technique. This equivalence plays an important and significant part in suggesting a number of new iterative algorithms for solving numerically generalized quasi variational inequalities and related quasi complementarity problems.

2. Formulation and Basic Results

Let H be a real Hilbert space whose inner product and norm are denoted by $< .,. >$ and $\|.\|$ respectively. Let K be a closed convex set in H. Let $T : H \to C(H)$ be a multi-valued nonlinear operator, where $C(H)$ is the family of all nonempty compact subsets of H. Given a point-to-set mapping $K : u \to K(u)$, which associates a closed convex set $K(u)$ with any element u of H, we consider a problem of finding $u \in K(u)$ such that $\omega \in T(u)$ and

$$< \omega, v - u > \geq 0 \quad \text{for all } v \in K(u). \tag{2.1}$$

The problem (2.1) is known as the generalized quasi variational inequality problem, which was introduced by Chan and Pang [3]. For recent applications and existence results, see [1,6,8,10,12,13]. For a suitable choice of the operator T and convex set K, one can obtain a large number of variational inequality and complementarity problems as special case of the problems (2.1) and (2.2). This shows that the problems (2.1) is more general and unifying ones and has many significant and important applications in regional, physical, mathematical and engineering science.

Lemma 2.1 [1]. Let K be a closed convex set in H. Then, for a given $z \in H$, $u \in K$ satisfies the inequality

$$< u - z, v - u > \geq 0 \quad \text{for all } v \in K,$$

if and only if $u = P_K z$, where P_K is the projection of H into the convex set K.

Definition 2.1. For all $u_1, u_2 \in H$, the operator $T : H \to C(H)$ is said to be

(i) **strongly monotone** if there exists a constant $\alpha > 0$ such that

$$< \omega_1 - \omega_2, u_1 - u_2 > \geq \alpha \|u_1 - u_2\|^2, \quad \text{for } \omega_1 \in T(u_1), \omega_2 \in T(u_2).$$

(ii) M-**Lipschitz continuous** if there exists a constant $\beta > 0$ such that

$$M(T(u_1), T(u_2)) \leq \beta \|u_1 - u_2\|,$$

where $M(\cdot, \cdot)$ is Hausdorff metric on $C(H)$.

3. Main Results

In this section, we first prove that the generalized quasi variational inequality (2.1) is equivalent to the fixed point problem and to the generalized implicit Wiener-Hopf equations. For this purpose, we need the following result, which can be proved by invoking Lemma 2.1.

Lemma 3.1. Let $K(u)$ be a closed convex-valued set in H. Then (u,ω) is a solution of (2.1) if and only if (u,ω) satisfies the relation

$$u = P_{K(u)}[u - \rho\omega],$$

where $\rho > 0$ is a constant and $P_{K(u)}$ is the projection of H onto the convex valued set $K(u)$.

From Lemma 3.1, it follows that the generalized quasi variational inequality (2.1) is equivalent to the fixed point problem

$$u = F(u) \equiv P_{K(u)}[u - \rho\omega].$$

This alternate fixed point formulation is very important from the numerical and approximation point of views. This formulations enables us to suggest and analyze the following general and unified iterative algorithm.

Algorithm 3.1. For a given $u_0 \in K(u_0)$ such that $\omega_0 \in T(u_0)$, compute $\{u_n\}$ and $\{\omega_n\}$ by the iterative schemes

$$\omega_n \in T(u_n) : \|\omega_n - \omega_{n+1}\| \leq M(T(u_n), T(u_{n+1})).$$
$$u_{n+1} = P_{K(u_n)}[u_n - \rho\omega_n], \quad n = 0, 1, 2, \cdots$$

If $K(u) \equiv K$, then Algorithm 3.1 reduces to:

Algorithm 3.2. For a given $u_0 \in K$ such that $\omega_0 \in T(u_0)$, compute the sequences $\{u_n\}$ and $\{\omega_n\}$

$$\omega_n \in T(u_n) : \|\omega_n - \omega_{n+1}\| \leq M(T(u_n)), T(u_{n+1}).$$

$$u_{n+1} = P_K[u_n - \rho\omega_n], \quad n = 0, 1, 2, \cdots.$$

Let $P_{K(u)}$ be the projection of H onto the convex valued set $K(u)$ and $Q_{K(u)} \equiv I - P_{K(u)}$, where I is the identity operator. For a given multivalued operator $T : H \to C(H)$, find $z \in H$, $u \in K(u)$ such that $\omega \in T(u)$ and

$$\omega + \rho^{-1} Q_{K(u)} z = 0, \tag{3.1}$$

where $\rho > 0$ is a constant. The equations (3.1) are called the generalized implicit Wiener-Hopf equations. If $T : H \to H$ is a single-valued operator, then problem (3.1) is equivalent to finding $z \in H$ such that

$$TP_{K(u)}z + \rho^{-1}Q_{K(u)}z = 0,$$

which are known as the implicit Wiener-Hopf equations, see Noor [10]. For the general treatment, applications and formulations, see [12,13,15,16].

We can establish the equivalence between generalized quasi variational inequality (2.1) and generalized implicit Wiener-Hopf equations (3.1) using Lemma 2.1 and techniques of Shi [15] and Noor [10,11].

Theorem 3.1. The function $u \in K(u)$ such that $\omega \in T(u)$ is a solution of generalized quasi variational inequality (2.1) if and only if $z \in H$, $u \in K(u)$ such that $\omega \in T(u)$ satisfies the generalized implicit Wiener-Hopf equation (3.1), where

$$u = P_{K(u)}z \tag{3.2}$$

$$z = u - \rho\omega. \tag{3.3}$$

Theorem 3.1 establishes the equivalence between the problems (2.1) and (3.1). This equivalence enables us to suggest a number of new iterative algorithms for solving generalized quasi variational inequalities and related quasi complementarity problems. To convey an idea, we discuss only the following cases.

I. The generalized implicit Wiener-Hopf equation (3.1) can be written as

$$Q_{K(u)}z = -\rho\omega,$$

from which it follows that

$$z = P_{K(u)}z - \rho\omega = u - \rho\omega, \quad \text{using (3.2).} \tag{3.4}$$

This fixed point formulation enables us to suggest the following:

Algorithm 3.3. For given $z_0 \in H$, $u_0 \in K(u_0), \omega_0 \in T(u_0)$, compute $\{z_n\}, \{u_n\}$ and $\{\omega_n\}$ by the iterative schemes.

$$u_n = P_{K(u_n)}z_n, \tag{3.5}$$

$$\omega_n \in T(u_n) : \|\omega_n - \omega_{n+1}\| \leq M(T(u_n), T(u_{n+1})) \tag{3.6}$$

$$z_{n+1} = u_n - \rho\omega_n, \quad n = 0, 1, 2, \cdots \tag{3.7}$$

II. The generalized Wiener-Hopf equation (3.1) may be written as

$$Q_{K(u)}z = -\omega + (I - \rho^{-1})Q_{K(u)}z,$$

which implies that

$$\begin{aligned} z &= P_{K(u)}z - \omega + (I-\rho^{-1})Q_{K(u)}z \\ &= u - \omega + (I-\rho^{-1})Q_{K(u)}z, \quad \text{using (3.2).} \end{aligned}$$

Using this fixed point formulation, we can suggest the following:

Algorithm 3.4. For given $z_0 \in H$, $u_0 \in K(u_0)$, $\omega_0 \in T(u_0)$, compute $\{z_n\}, \{u_n\}$ and $\{\omega_n\}$ by the iterative schemes

$$\begin{aligned} &u_n = P_{K(u_n)}z_n \\ &\omega_n \in T(u_n) : \|\omega_n - \omega_{n+1}\| \leq M(T(u_n), T(u_{n+1})) \\ &z_{n+1} = u_n - \omega_n + (I-\rho^{-1})Q_{K(u_n)}z_n, \quad n = 0, 1, 2, \cdots \end{aligned}$$

We now study the convergence criteria of Algorithms 3.1-3.4. For this we need the following hypothesis, which plays a significant part in the convergence analysis of the Algorithms

Assumption 3.1. For all $u, v, z \in H$, and a constant $\eta > 0$,

$$\|P_{K(u)}z - P_{K(v)}z\| \leq \eta\|u - v\|.$$

Theorem 3.2. Let the multivalued operator $T : H \to C(H)$ be strongly monotone, with constant $\alpha > 0$ and M-Lipschitz continuous with constant $\beta > 0$. If Assumption 3.1 holds and

$$\left|\rho - \frac{\alpha}{\beta^2}\right| < \frac{\sqrt{\alpha^2 - \beta^2(2-\eta)}}{\beta^2} \tag{3.8}$$

$$\alpha > \beta\sqrt{\eta(2-\eta)}, \tag{3.9}$$

then there exist $z \in H$, $u \in K(u)$, $\omega \in T(u)$ satisfying the Wiener- Hopf equation (3.1) and the sequences $\{z_n\}, \{u_n\}$ and $\{\omega_n\}$ generated by Algorithm 3.3 converge to z, u and ω strongly in H respectively.

Proof: From Algorithm 3.3, we have

$$\begin{aligned} \|z_{n+1} - z_n\|^2 &= \|u_n - u_{n-1} - \rho(\omega_n - \omega_{n-1})\|^2 \\ &= \|u_n - u_{n-1}\|^2 - 2\rho < \omega_n - \omega_{n-1}, u_n - u_{n-1} > \\ &\quad +\rho^2\|\omega_n - \omega_{n-1}\|^2 \\ &= \|u_n - u_{n-1}\|^2 - 2\rho < \omega_n - \omega_{n-1}, u_n - u_{n-1} > \\ &\quad +\rho^2\{M(T(u_n), T(u_{n-1}))\}^2 \\ &\leq (1 - 2\rho\alpha + \rho^2\beta^2)\|u_n - u_{n-1}\|^2. \end{aligned} \tag{3.10}$$

From (3.5) and Assumption 3.1, we have

$$\begin{aligned}\|u_n - u_{n-1}\|^2 &= \|P_{K(u_n)}z_n - P_{K(u_n)}z_{n-1}\| + P_{K(u_n)}z_{n-1} - P_{K(u_{n-1})}z_{n-1}\| \\ &\leq \|z_n - z_{n-1}\| + \eta\|u_n - u_{n-1}\|,\end{aligned}$$

from which it follows that

$$\|u_n - u_{n-1}\| \leq \frac{1}{1-\eta}\|z_n - z_{n-1}\|. \tag{3.11}$$

Combining (3.10) and (3.11), we obtain

$$\|z_{n+1} - z_n\| \leq \left\{\frac{\sqrt{1-2\rho\alpha+\rho^2\beta^2}}{1-\eta}\right\}\|z_n - z_{n-1}\| = \theta\|z_n - z_{n-1}\|, \tag{3.12}$$

where $\theta = \frac{\sqrt{1-2\rho\alpha+\rho^2\beta^2}}{1-\eta}$.

From (3.8) and (3.9), we see that $\theta < 1$, and consequently from (3.12), it follows that the sequence $\{z_n\|$ is a Cauchy sequence in H, that is, $z_{n+1} \to z \in H$ or $n \to \infty$. From (3.6), we have

$$\|\omega_n - \omega_{n-1}\| \leq M(T(u_n), T(u_{n-1})) \leq \beta\|u_n - u_{n-1}\|,$$

from which it follows that the sequence $\{\omega_n\}$ is also a Cauchy sequence in H, that is, there exists $\in H$ such that $\omega_n \to \omega$ as $n \to \infty$. Now by using the continuity of the operators, P_K and Theorem 3.1, we have

$$z = u - \rho\omega = P_{K(u)}z - \rho\omega \in H.$$

Now we shall show that $\omega \in T(u)$. In fact,

$$\begin{aligned}d(\omega, T(u)) &\leq \|\omega - \omega_n\| + M(\omega_n, T(u)) \leq \|\omega - \omega_n\| + M(T(u_n), T(u)) \\ &\leq \|\omega_1 - \omega_n\| + \eta\|u_n - u\| \to 0 \quad \text{as } n \to \infty,\end{aligned}$$

where $d(\omega, T(u)) = \inf\{\|\omega - v\| : v \in T(u)\}$. Consequently we have $d(\omega, T(u)) = 0$, from which it follows that $\omega \in T(u)$, since $T(u) \in C(H)$. By invoking Theorem 3.1, it follows that $z \in H$, $u \in K(u)$, $\omega \in T(u)$ are solutions of (3.1) and consequently $z_n \to z$, $u_n \to u$ and $\omega_n \to \omega$ strongly in H.

References

1. C. Baiocchi and A. Capelo, Variational and Quasi Variational Inequalities, J. Wiley and Sons, New York , 1984.
2. A. Bensoussan and J.L. Lions, Applications des Inequations Variationelles en Control et en Stochastiques, Dunod, Paris, 1978.
3. D. Chan and J.S. Pang, The generalized quasi variational inequality problems, Math. Oper. Research **7** (1982), 211-222.
4. S.C. Fang and E.L. Peterson, Generalized variational inequalities, J. Optim. Theory Appl. **38**(1982), 363-383.
5. R. Glowinski, J.L. Lions and R. Tremolieres, Numerical Analysis of Variational Inequalities, North-Holland, Amsterdam, 1981.
6. P.T. Harker and J.S. Pang, Finite dimensional variatonal inequality and nonlinear complementarity problems: a survey of theory, algorithms and applications, Math. Program. **48**(1990), 181-220.
7. N. Kikuchi and J.T. Oden, Contact Problems in Elasticity, SIAM Publications Co., Philadelphia, 1988.
8. U. Mosco, Implicit variational problems and quasi variational inequalities, Lect. Notes Math. 543, Springer-Verlag, Berlin, (1976), 83-126.
9. M. Aslam Noor, An iterative scheme for a class of quasi variational inequalities, J. Math. Anal. Appl. **110** (1985), 463-468.
10. M. Aslam Noor, Iterative algorithms for quasi variational inequalities, PanAmer. Math. J. **2** (2) (1992), 17-26.
11. M. Aslam Noor, Wiener-Hopf equations and variational inequalities, J. Optim. Theory Appl. **79** (1993), 197-206.
12. M. Aslam Noor, K. Inayat Noor and Th.M. Rassias, Some aspects of variational inequalities, J. Comput. Appl. Math., **47** (1993), 285-312.
13. M. Aslam Noor, K. Inayat Noor and Th.M. Rassias, in: Analysis, Geometry, Groups: A Riemann Legacy Volume (Eds. H.M. Srivastava and Th.M. Rassias), Hadronic Press, (1993), 373-448.
14. M. Aslam Noor, Computational techniques for variational inequalities, J. Natural Geometry (1996), in press.
15. P. Shi, Equivalence of variational inequalities with Wiener-Hopf equations, Proc. Amer. Math. Soc. **111** (1991), 339-346.
16. F.O. Speck, General Wiener-Hopf Factorization Methods, Pitman, London, 1985.
17. G. Stampacchia, Formes bilineaires coercitives sur les ensembles convexes, C.R. Acad. Sci. Paris, Sér I. Math. **258** (1964), 4413-4416.

4:45 - 5:15		
5:15 - 5:45	J. Tausch	Preconditioning First & Second Kind Integral Formulations of the Capacitance Problem
5:45 - 6:15	B. Singer	Green's Function of Maxwell's Equations and Corresponding Implications for Iterative Methods
6:15 - 6:45	B. Spotz	Iterative Solution of High Order Compact Systems

Preconditioning First and Second Kind Integral Formulations of the Capacitance Problem *

Johannes Tausch and Jacob White

January 15, 1996

1 Introduction

Engineering programs which compute electrostatic capacitances for complicated arrangements of conductors commonly set up the electrostatic potential u as a superposition of surface carges σ

$$u(x) = \int_S G(x,y)\sigma(y)\, dS(y)\,.$$

where $G(x,y) = 1/4\pi|x-y|$ is the Green's function for the Laplacian in the three-space. For a specified potential on the conductor surface(s) S, this approach leads to an integral equation of the first kind on S for the charge density σ. The capacitance is the net-charge on the conductors and is given by the surface integral of σ.

It is standard to discretize this integral equation with a collocation scheme. The resulting linear system is dense and can be large for complex geometries of S. In the recent past, there has been a major progress to sparsify this system with the fast multipole method. As a result the matrix vector product can be carried out in $\mathcal{O}(N)$ operations and thus the solution of the linear system with an iterative scheme like GMRES is feasible [1, 3].

However, discretizations of the first kind integral formultation lead to matrices with condition numbers which increase as the mesh is refined. This behavior makes the iterative solution of the linear system more expensive and sometimes impossible without a good preconditioner. Furthermore, the numerical analysis of the discretization error as well as the design of adaptive mesh refinement stragies are more difficult for first kind formulations. Some of these issues are still unresolved for collocation schemes.

For smooth surfaces, the difficulties associated with the first kind formulation can be entirely avoided using a second kind integral formulations for the capacitance problem, and we investigate two different types. Typically, the arising

*This work was supported by ARPA under contracts DABT63-94-C-0053 and J-FBI-92-196, and the Semiconductor Research Corporation under contract SJ-558.

operators are compact on smooth surfaces and thus the Riesz-Fredholm theory provides a framework for analyzing the second kind formulation. Of particular importance is the result that the condition number of a discretized second-kind formulation is bounded independent of discretization mesh, and so requires no preconditioning.

The issues are quite different when the surfaces contain edges and corners, as is quite common in engineering problems. The integral operators in the second kind formulations are no longer compact, and the discretizations generate somewhat more ill-conditioned matrices. In addition, for problems with corners, empirical evidence suggests that first kind formulations produce significantly more accurate results for a given discretization. Since the corners introduce localized nonsmoothness, we investigate local inversion preconditioners for first and second kind formulations for the nonsmooth case.

2 Second Kind Formulations

The first formulation we consider was suggested by Mikhlin [2] and is based on writing the electrostatic potential as a superposition of a surface dipole potential and the potential due to one point charge in each conductor

$$u(x) = K\mu + \sum_j q_j G(x, x_j) = \int_S \frac{\partial}{\partial n_y} G(x,y)\mu(y)\, dS(y) + \sum_j q_j G(x, x_j)\,.$$

The point charges in this approach are necessary to ensure that the potential decays like $1/r$. It turns out that the scalars q_j are the desired capacitances, if the net-dipole density for each conductor vanishes. Taking into account the jump relations of the dipole operator, we obtain the integral equation

$$\begin{aligned} (1/2 + K + P)\mu + \sum_j q_j G(x, x_j) &= f(x)\,, \quad x \in S \\ \int_{S_j} \mu &= 0\,, \quad j = 1, \ldots, n\,. \end{aligned}$$

The operator P is defined by

$$P\mu(x) = \sum_j \int_{S_j} \mu\, dS\, \chi_j(x)\,,$$

where χ_j denotes the characteristic function of conductor j.

Once the dipole density μ has been determined, the charge denstity σ can be calculated by an application of the hypersingular integral operator

$$\sigma(x) = -\int_S \frac{\partial^2}{\partial n_x \partial n_y} G(x,y)\mu(y)\, dS(y) - \sum_j q_j \frac{\partial}{\partial n_x} G(x, x_j)\,.$$

We also investigate an alternative approach which allows the direct calculation of the charge density without resorting to the dipole density. To obtain this

formulation we remark that in the capacitance problem the potential assumes a constant value p_i within each conductor. Thus the normal derivative of the potential on the conductor surfaces when approached from the inside vanishes. Taking into account the jump relation of the adjoint operator, we obtain

$$(1/2 + K^*)\sigma(x) = 1/2\,\sigma(x) + \int_S \frac{\partial}{\partial n_x} G(x,y)\sigma(y)\,dS(y) = 0\,, \quad x \in S\,.$$

The integral equation above is singular. This is because the orthogonal complement of the range of the operator $(1/2 + K^*)$ consists of the functions which are constant on each conductor surface. Thus the equation

$$(1/2 + K^* + P)\,\sigma - \sum_j q_j \chi_j = 0$$

has a unique solution for any choice of the scalars q_j. We can choose these scalars so that the potential of σ satisfies the given conductor potentials. This leads to the integral equation

$$\begin{aligned} (1/2 + K^* + P)\sigma + \sum_j q_j \chi_j &= 0\,, \quad x \in S \\ \int_{S_j} G(x_j, y)\sigma(y)\,dS(y) &= p_j\,, \quad j = 1, \ldots, n\,, \end{aligned}$$

Note that this integral equation is adjoint to Mikhlin's formulation.

3 Preconditoning

Since the eigenvalues of a compact operator can only accumulate in the origin, discretizations of second kind integral equations will require only a relatively small number of iterations to converge. This situation changes when corners and edges are present. In this case the operators are no longer compact and the iteration may be accelerated with a preconditioner.

The fast multipole algorithm decomposes the problem domain into a hierarchy of cubes, this decomposition can also be used to construct preconditioners [4, 3].

Denoting the intersection of the surface S with cube i by S_i, the part of the operator K from S_j to S_i is given by

$$K_{ij}\sigma_j(x) := \int_{S_j} \frac{\partial}{\partial n_y} G(x,y)\sigma(y)\,dS(y) \quad x \in S_i\,.$$

The idea of the (nonoverlapping) local inversion preconditioner is to solve the integral equation for each surface S_i neglecting the interaction of the other pieces. Thus the preconditioner factors the second kind integral equation $(1/2 + K)\sigma = f$ in the form

$$\tilde{\sigma}_i + \sum_{i \neq j} (1/2 + K_{ii})^{-1} K_{ij} \tilde{\sigma}_j = f_i$$

where

$$\tilde{\sigma}_i = (1/2 + K_{ii})\sigma_i \,.$$

This approach is conceivable for isolated corners like the tip of a cone, because in this case the operators K_{ij} are compact for $i \neq j$. Moreover, if the integral operator is weakly singular in the corner, then the size of the cubes can be made small enough such that the norm of the operators K_{ii} is less than $1/2$ and thus the operator $1/2 + K_{ii}$ has a continuous inverse. Hence the transformed system is of second kind with a compact operator.

In the case of edges, the operator K_{ij} is not compact if the cubes i and j are adjacent and intersect the same edge. Still, the integral equation can be transformed into the form "identity plus compact" by an overlapping preconditioner. Denoting all neighbors of cube i by $N(i)$ and the operator on the cube i and its neighbors by $K_{N(i)}$ we can factor the integral equation in the form

$$\tilde{\sigma}_i + \sum_{j \notin N(i)} (1/2 + K_{N(i)})^{-1} K_{ij} \tilde{\sigma}_j = f_i \,.$$

4 Preliminary Numerical Results

Numerical experiments were carried out for two different domains, namely the unit sphere and an L-shaped domain. . The sphere gives rise to compact integral operators, whereas the L-block has corner and edge singularities. For the sphere the discretization is almost uniform, wereas the mesh of the L-block was refined towards the edges.

All equations were discretized with piecewise constant collocation, the arising linear systems were solved with multipole accelerated GMRES, where the expansion order in all experiments was set to three. Increasing this value did not result in significant changes.

The discretization errors of the capacitance and the numbers of iterations for the sphere are dispayed in Table 4. The results there suggest that the discretization errors for all three integral formulations decay like $\mathcal{O}(h^2)$. The number of GMRES iterations varies between the formulations, they remain constant for both second kind formulations, but grow for the first kind formulation without preconditioner as the mesh is refined. The increase can be avoided by preconditioning, but only if the same cube hierarchy is used for all mesh refinements. If the cubesize is reduced to avoid expensive preconditioners, then the number iterations grows as well.

Table 2 displays the iteration results for the L-block. The capacitances suggest that the first kind formulation converges faster to the true value than the second kind formulations. As expected, the number of iterations for the first kind formulation increases when refining the mesh and the increase can be slowed down by the use of the overlapping preconditioner. The second kind formulations require a comparatively small number of iterations to converge, even for small panel sizes. Preconditioners can further accelerate the iteration.

number Panels		192		768		3072	
First Kind	no PC	0.321	(4)	0.085	(9)	0.022	(11)
	OL 1		(6)		(6)		(6)
	OL 2		(5)		(7)		(9)
Dipole	no PC	0.286	(5)	0.074	(6)	0.019	(6)
Adjoint	no PC	0.382	(5)	0.100	(4)	0.025	(5)

Table 1: Comparison of the discretization errors and iterations obtained for the sphere. The number of iterations required to reduce the residual to 10^{-6} are shown in brackets. OL 1 refers to the overlapping preconditioner with constant cube size, OL 2 refers to the overlapping preconditioner with decreasing cube size.

number Panels		350		1400		5600	
First Kind	no PC	12.626	(17)	12.658	(27)	12.663	(40)
	OL 1		(12)		(7)		(8)
	OL 2		(12)		(18)		(24)
Dipole	no PC	12.467	(13)	12.602	(14)	12.648	(15)
	OL 1		(9)		(9)		(10)
	OL 2		(7)		(8)		(10)
Adjoint	no PC	12.310	(10)	12.485	(12)	12.582	(14)
	OL 1		(7)		(8)		(8)
	OL 2		(8)		(9)		(11)

Table 2: Comparison of the calculated capacitances and iterations obtained for the L-shaped domain. The number of iterations required to reduce the residual to 10^{-6} are shown in brackets.

5 Conclusion

For problems with smooth surfaces discretizations of the second kind formulations result in better conditioned linear systems than the first kind formulation by maintaining the accuracy of the approximation. In addition, the adjoint formulation directly produces surface densities which are more useful in subsequent application than the dipole layer.

The conditioning of all formulations worsens in the case of non-smooth domains, although the ill-conditioning appears to be milder for the second kind formulations. Local inversion preconditioners are effective at removing this ill-conditioning, though the spatial extent required for the preconditioner is still under investigation. Finally, the convergence of discretization error is slower for the second kind formulation, and this may be an artifact of the piecewise constant collocation used for the experiments. We will investigate higher order approximation schemes in our future research.

References

[1] Lesslie Greengard. *The Rapid Evaluation of Potential Fields in Particle Systems.* M.I.T. Press, Cambridge, Massachusetts, 1988.

[2] S. G. Mikhlin. *Integral Equations.* Pergamon Press, London, 1957.

[3] K. Nabors, F. T. Korsmeyer, F. T. Leighton, and J. White. Preconditioned, adaptive, multipole-accelerated iterative methods for three-dimensional first-kind integral equations of potential theory. *SIAM J. Sci. Comput.*, 15(3):713–735, 1994.

[4] Stephen Vavasis. Preconditioning for boundary integral equations. *SIAM J. Matrix Anal.*, 13(3):905–925, 1992.

Green's function of Maxwell's equations and corresponding implications for iterative methods

B. Sh. Singer [1] and E.B.Fainberg [2]

[1] CRCAMET, E5A, Macquarie University, North Ryde, Sydney, NSW 2109, Australia
[2] Institute of Physics of the Earth, 142092 Troitsk, Moscow Region, Russia

SUMMARY

Energy conservation law imposes constraints on the norm and direction of the Hilbert space vector representing a solution of Maxwell's equations. In this paper, we derive these constrains and discuss the corresponding implications for the Green's function of Maxwell's equations in a dissipative medium.

It is shown that Maxwell's equations can be reduced to an integral equation with a contracting kernel. The equation can be solved using simple iterations. Software based on this algorithm have successfully been applied to a wide range of problems dealing with high contrast models.

The matrix corresponding to the integral equation has a well defined spectrum. The equation can be symmetrized and solved using different approaches, for instance one of the conjugate gradient methods.

1. ENERGY INEQUALITIES

The model under consideration consists of a conductor occupying some volume v, and an isolator surrounding the volume. The resistivity distribution inside the volume is specified by a restricted function $\rho_0(\mathbf{r})$ $(0<\rho_m<\rho_0(\mathbf{r})<\rho_M)$. In this section we neglect the displacement currents. The corresponding generalization is straightforward and will be discussed in section 4. We assume also that the magnetic permeability coincides with that of a vacuum, and that the time factor is $\exp(-\imath\omega t)$.

Maxwell's equations for the electric field $\mathbf{E}$ and magnetic field $\mathbf{H}$ generated by an external current $\mathbf{j}^e$ flowing inside volume v can be written in the form

$$\nabla\times\boldsymbol{H} = \boldsymbol{j}\ , \qquad \nabla\times(\rho_0\boldsymbol{j}) = \imath\omega\mu_0\boldsymbol{H} + \nabla\times(\rho_0\boldsymbol{j}^e)\ , \tag{1}$$

where the current density is given by the equation

$$\boldsymbol{j} = \rho_0^{-1}\boldsymbol{E} + \boldsymbol{j}^e\ . \tag{2}$$

Using standard manipulations one can easily derive the equation expressing the energy conservation law, that is

$$\int_v \rho_0\{|\boldsymbol{j}|^2 - \tilde{\boldsymbol{j}}\boldsymbol{j}^e\}dv = \imath\omega\mu_0\int_{\mathbb{R}^3}|\boldsymbol{H}|^2dv\ . \tag{3}$$

It is convenient to consider current distributions as vectors of a Hilbert space defining the scalar product as

$$(U, V) = \int_V \rho_0 U \cdot \tilde{V} dv , \tag{4}$$

Using this definition, eq.(3) is reduced to

$$Re(j, j^e) = \|j\|^2 , \tag{5}$$

where $\|.\|$ stands for the Hilbert space norm. Application of Cauchy's inequality to eq.(5) immediately results in the condition

$$\|j\| \le \|j^e\| . \tag{6}$$

Thus, the total current flowing in the conductor is smaller in the L_2-norm than the external current generating the field. Another condition that also follows from eq.(5) is

$$\|j - \frac{1}{2} j^e\| = \frac{1}{2} \|j^e\| . \tag{7}$$

Geometrically, this condition means that the end of the vector j lies in the Hilbert space on the sphere having j^e as a diameter (Fig.1).

2. GREEN'S FUNCTION

Recalling that solution to Maxwell's equations can be expressed via convolution

$$j = \mathfrak{G}_0 * j^e \tag{8}$$

of the Green's function $\mathfrak{G}_0$ with the distribution of the external current, one can immediately conclude that convolution with the Green's function does not increase the norm of a current distribution U, i.e.

$$\|\mathfrak{G}_0 * U\| \le \|U\| . \tag{9}$$

Defining also a modified (or renormalized) Green's function as

$$\mathfrak{Z}_0 = 2 \cdot \mathfrak{G}_0 - \delta \mathbf{I} , \tag{10}$$

where $\mathbf{I}$ is a unit tensor and δ is the Dirac's δ-function, we obtain from eq.(8) a more restrictive (unitary) condition

$$\|\mathfrak{Z}_0 * U\| = \|U\| . \tag{11}$$

In particular, this condition means that eigenvalues of convolution with the Green's function $\mathfrak{Z}_0$ lie on a unit circle $|\varsigma|=1$ in the complex ς-plane.

3. INTEGRAL EQUATION

Up to this point, we did not impose any restrictions on the function $\rho_0(r)$. From now on we will imply that it belongs to a class of distributions allowing for an efficient analytical or numerical solution of Maxwell's equations. For instance, it is relatively easy to solve Maxwell's equations in a 1-D spherical or plane earth model. Suppose, now we have to find a solution for the model with the resistivity distribution inside volume v given by $\rho(r)$. Then Maxwell's equations can be reduced to

$$\nabla\times H = j\,, \quad \nabla\times(\rho_0 j) = \imath\omega\mu_0 H + \nabla\times(\rho_0 j')\,, \quad j' = \frac{\rho}{\rho_0} j^e - \frac{\rho-\rho_0}{\rho_0} j\,. \qquad (12)$$

Using the Green's function of eq.(1), we will find that eq.(12) is equivalent to the integral equation

$$j = j_0 - \mathbb{G}_0 * \left(\frac{\rho-\rho_0}{\rho_0} j\right)\,, \qquad j_0 = \mathbb{G}_0 * \left(\frac{\rho}{\rho_0} j^e\right)\,. \qquad (13)$$

From section 2, eq.(13) has a contractive kernel provided $max|\rho/\rho_0-1|<1$. It is always possible to find the (reference) distribution $\rho_0(r)$ satisfying this condition. Having found such distribution and calculated the Green's function $\mathbb{G}_0$, one can solve eq.(13) using simple iterations. The convergence can be optimized by the choice of the reference resistivity model. The corresponding algorithm was proposed by the authors in (Fainberg & Singer, 1980). It was later named as Iterative Dissipative Method or IDM and used for modeling of electromagnetic fields in high contrast media.

The convergence of the method can be greatly improved (Singer,1995) after substitution of eq.(10) in eq.(13). The integral equation is then transformed into

$$\xi = \xi_0 - \mathfrak{Z}_0 * \left(\frac{\rho-\rho_0}{\rho+\rho_0}\xi\right)\,, \qquad \xi = \frac{\rho+\rho_0}{\rho_0} j\,,$$
$$\xi_0 = \frac{\rho}{\rho_0} j^e + \mathfrak{Z}_0 * \left(\frac{\rho}{\rho_0} j^e\right)\,. \qquad (14)$$

The last equation has a contractive kernel for an arbitrary reference distribution $\rho_0(r)$. The fastest convergence of iterations is achieved if the reference distribution is chosen so that at each depth z

$$\rho_0^2(z) = min_x\,\rho(x_\tau, z)\cdot max_x\,\rho(x_\tau, z)\,, \qquad (15)$$

where $min_x\{\rho(x_\tau, z)\}$ and $max_x\{\rho(x_\tau, z)\}$ are minimum and maximum resistivity values at the same depth level. With this choice of the reference resistivity model, the number of operations necessary to achieve the given accuracy ε_0 is controlled by the factor

$$\sqrt{K} \cdot N_\tau \cdot N_z^2 \cdot \log N_\tau \cdot \log \varepsilon_0^{-1} \qquad (16)$$

where N_τ is the number of nodes in the surface numerical grid, N_z is the number of nodes covering the inhomogeneity in the vertical direction, and K is the maximum contrast of the resistivity variations in the lateral direction. A similar expression is applicable to eq.(13) but with factor $\sqrt{K}$ being replaced by K.

4. GENERALIZATIONS

The discussed approach does not impose any restrictions on the frequency of field variations or distributions of resistivity and external sources; it can used in the frequency or time domain. The approach can be generalized to include displacement currents, frequency dependence of the resistivity or dielectric permittivity distributions, and their anisotropy. Avoiding further discussions in this direction, we will notice only that for a model with the electrical resistivity and dielectric permittivity distributions $\rho(r)$ and $\epsilon(r)$, respectively, the basic equations (4)–(11) and the integral equation stay unchanged (Singer & Fainberg, 1995). The only difference is that ρ in this equation should be replaced with the function

$$\chi^{-1} = \rho^{-1} - \imath\omega(\epsilon - \epsilon_0) \ , \qquad (17)$$

where $\epsilon_0(r)$ is the dielectric permittivity of the reference model. Of course, the Green's function $\mathfrak{Z}_0$ should be calculated for the reference model with the resistivity and permittivity distributions given by the functions $\rho_0(r)$ and $\epsilon_0(r)$, respectively.

5. EXAMPLE

As an example, we have calculated electromagnetic field induced in a model spanning the western coast of the Northern America and the adjacent part of the Pacific Ocean by a six hour variation of the uniform magnetic field. The model consists of an inhomogeneous subsurface layer and a 1-D underlying structure. The subsurface layer simulates the ocean and the continental sedimentary cover. Its conductance and conductivity of the underlying formation are shown in Fig. 2. Plots of the in-phase and out-of-phase components of the normalized vertical magnetic fields are shown in Fig.3 and Fig.4. The computation were carried out on a 128*128 surface grid using a 486/33MHz PC.

ACKNOWLEDGEMENTS

The authors express their gratitude to V. Druskin who was first to notice that the method cab be applied for preconditioning.

REFERENCES

Fainberg, E.B. & Singer, B.Sh., 1980. Electromagnetic induction in a nonuniform spherical model of the Earth, *Ann. Geophys.*, 36, 127-134.

Singer, B.Sh., 1995. Method for solution of Maxwell's equations in non-uniform media, *Geophys.J.Int.*, 120, 590-598.

Singer, B.Sh. & Fainberg, E.B., 1996. Generalization of the iterative dissipative method for modeling electromagnetic fields in nonuniform media with displacement currents, *J.Appl.Geophys.*, 34, 41-46.

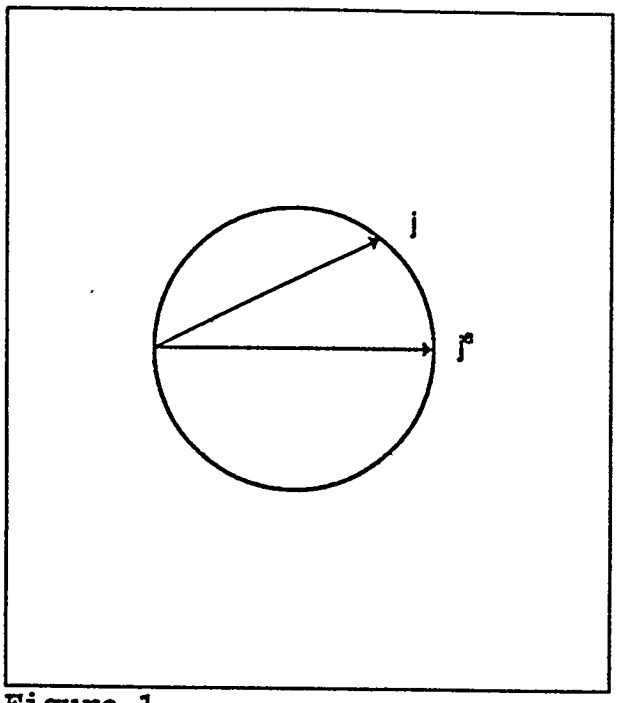


Figure 1.

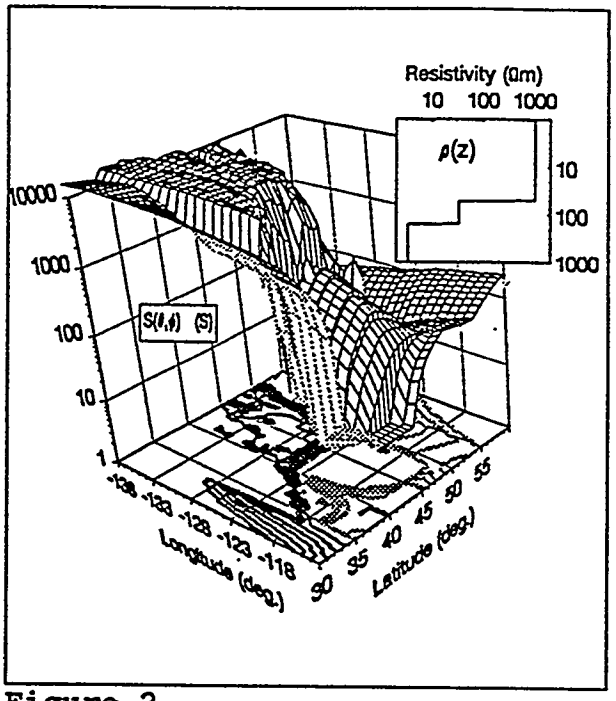


Figure 2.

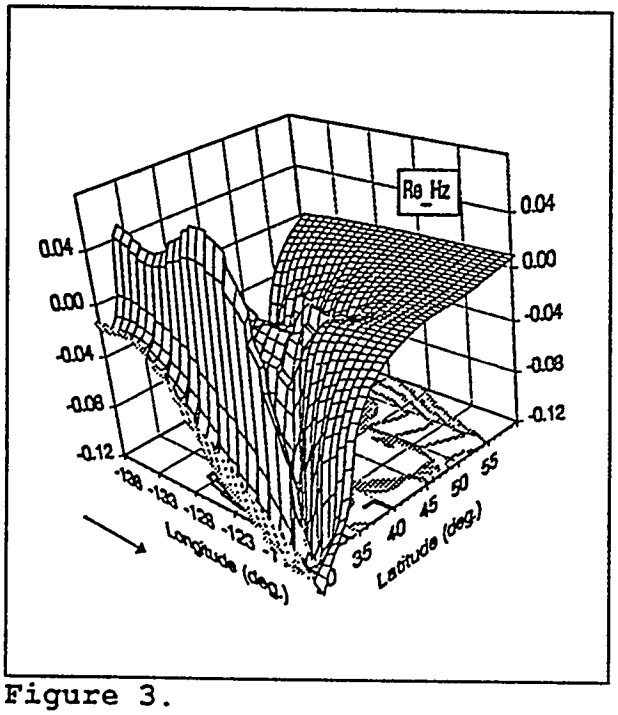


Figure 3.

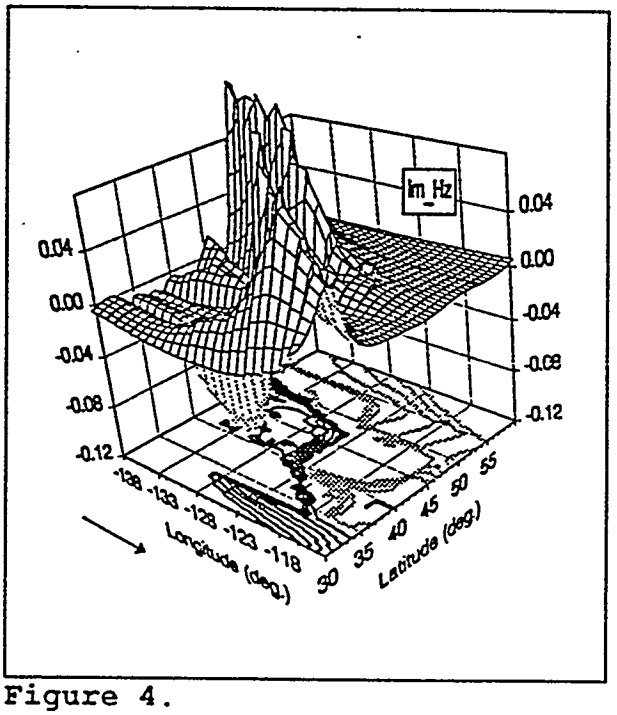


Figure 4.

Iterative Solution of High Order Compact Systems

W. F. Spotz and G. F. Carey
Computational Fluid Dynamics Laboratory
The University of Texas at Austin

We have recently developed a class of finite difference methods which provide higher accuracy and greater stability than standard central or upwind difference methods, but still reside on a compact patch of grid cells. In the present study we investigate the performance of several gradient-type iterative methods for solving the associated sparse systems. Both serial and parallel performance studies have been made. Representative examples are taken from elliptic PDE's for diffusion, convection-diffusion, and viscous flow applications.

Topic: *Software*	*Session Chair:* *Iain Duff*	*Room C*
4:45 - 5:15	R. Pozo	Library Designs for Generic C++ Sparse Matrix Computations of Iterative Methods
5:15 - 5:45	F. Saied	MGLab3D: An Interactive Environment for Iterative Solvers for Elliptic PDEs in Two and Three Dimensions
5:45 - 6:15	S. Salvini	New Iterative Solvers for the NAG Libraries

Roldan Pozo
Research Computer Scientist
Computing and Applied Mathematics Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899 USA

Tel: (301) 975-4317
Fax: (301) 990-4127
Email: pozo@nist.gov
http://math.nist.gov/pozo

Library Designs for Generic C++ Sparse Matrix Computations of Iterative Methods

Roldan Pozo
National Institute of Standards and Technology

A new library design is presented for generic sparse matrix C++ objects for use in iterative algorithms and preconditioners. This design extends previous work on C++ numerical libraries (SparseLib++[3], IML++[4], Lapack++[5]) by providing a framework in which efficient algorithms can be written *independent* of the matrix layout or format. That is, rather than supporting different codes for each (element type) / (matrix format) combination, only one version of the algorithm need be maintained. This not only reduces the effort for library developers, but also simplifies the calling interface seen by library users. Furthermore, the underlying matrix library can be naturally extended to support user-defined objects, such as hierarchical block-structured matrices, or application-specific preconditioners. Utilizing optimized kernels[1,7] whenever possible, the resulting performance of such framework can be shown to be competitive with optimized Fortran programs.

1] S. Carney, M. A. Heroux, G. Li, K. Wu, "A Revised Proposal for a Sparse LAS Toolkit", Army High Performance Computing Research Center Technical eport 94-034, June 1994.

2] Barrett, et. al, "Templates for the Solution of Linear Systems: Building locks for Iterative Methods," SIAM Press, 1994.

3] J. Dongarra, A. Lumsdaine, R. Pozo, K. Remington, "A Sparse atrix Library in C++ for High Performance Linear Algebra," Proceedings f the Object Oriented Numerics Conference, 1994, pp. 214-218.

4] J. Dongarra, A. Lumsdaine, R. Pozo, K. Remginton, "IML++: Iterative ethods Library Reference Guide", 1995, available from ttp://math.nist.gov/pozo/iml++.html.

5] J. Dongarra, R. Pozo, D. Walker, "Lapack++: A Design Overview f Object-Oriented Extensions for High Performance Linear Algebra", roceedings of Supercomputing '93.

5] R. Pozo, K. Remington, "C Sparse BLAS", source code available from ttp://math.nist.gov/pozo/sparse_blas/sparse_blas.html

Assistant Professor
Computer Science Department
University of Illinois at Urbana-Champaign
1304 W Springfield Ave
Urbana, IL 61801

(217) 244-1621 office
(217) 333-3501 fax

saied@cs.uiuc.edu

MGLab3D: An interactive environment for iterative solvers for elliptic PDEs in two and three dimensions

James Bordner and Faisal Saied
Department of Computer Science
University of Illinois at Urbana-Champaign

MGLab3D is an enhancement of an interactive environment (MGLab) for experimenting with iterative solvers and multigrid algorithms. It is implemented in MATLAB.

The new version has built-in 3D elliptic pde's and several iterative methods and preconditioners that were not available in the original version. A sparse direct solver option has also been included. The multigrid solvers have also been extended to 3D.

The discretization and pde domains are restricted to standard finite differences on the unit square/cube. The power of this software lies in the fact that no programming is needed to solve, for example, the convection-diffusion equation in 3D with TFQMR and a customized V-cycle preconditioner, for a variety of problem sizes and mesh Reynolds' numbers.

In addition to the graphical user interface, some sample drivers are included to show how experiments can be composed using the underlying suite of problems and solvers.

New Iterative Solvers for the NAG Libraries

Stefano Salvini and Gareth Shaw

The Numerical Algorithms Group Ltd,
Wilkinson House, Jordan Hill Road, Oxford OX2 8DR, United Kingdom

E-mail: stef@nag.co.uk, gareths@nag.co.uk

1 Introduction

The purpose of this paper is to introduce the work which has been carried out at NAG Ltd to update the iterative solvers for sparse systems of linear equations, both symmetric and unsymmetric, in the NAG Fortran 77 Library [7]. Our current plans to extend this work and include it in our other numerical libraries (Fortran 90 [8], C [9], Numerical PVM [10]) in our range are also briefly mentioned.

We have added to the Library the new Chapter F11, entirely dedicated to sparse linear algebra. At Mark 17 (released in early 1996), the F11 Chapter includes sparse iterative solvers, preconditioners, utilities and black-box routines for sparse symmetric (both positive-definite and indefinite) linear systems [13]. Mark 18 (due for realease in early 1997) will add solvers, preconditioners, utilities and black-boxes for sparse unsymmetric systems [14]: the development of these has already been completed.

The new routines aim to address the needs of more advanced users as well as of 'general' users: for the former, 'basic' solver routines are provided which make no assumptions on either the sparsity pattern of the coefficient matrix, or on the storage format employed; for the latter we provide black-box routines for particular storage formats which offer less flexibility but are easier to use. Currently, black-boxes are only provided for the Coordinate Storage (CS) format, in which a real array holds the non-zero matrix elements, while two integer arrays store the corresponding row and column indices. For specific, high-performance applications, our new library software probably cannot compete with bespoke, application-specific algorithms, but it does provide the tools for quick and efficient *prototyping*, hence allowing the more rapid development of a package.

Other important points, for both symmetric and unsymmetric systems are:

- *reverse communication* is used for the basic solvers: the Library routine returns repeatedly to the calling program requesting some computation to be carried out. This allows maximum flexibility in the representation and storage of sparse matrices: all matrix operations, including the application of preconditioners, are performed outside the solver routine. Also, the user can interact effectively with the solution process: the progress of the solution can be closely monitored and tidy or immediate terminations can be requested. This is useful, for example, when alternative termination criteria are

to be employed or in case of failure of the external routines used to perform matrix operations.

- for both symmetric and unsymmetric systems, the suite of basic routines consists of:
  - an initialization routine, where the user specifies the method of solution to be employed, termination criterion, tolerances, etc.;
  - the iterative solver, which uses reverse communication;
  - a monitoring routine which can be used to return the latest available information about the convergence of the solution during or after the computation.

  This allows users to employ different algorithms without modifying their programs, and allows us to make more methods available in future with minimum impact on existing Library and software;
- all solvers will automatically restart the solution process, if this is of benefit to convergence, trivially so for RGMRES;
- a termination criterion based on the backward error for the original, unpreconditioned, system is available for all solvers: $\|r\| \leq \epsilon f(n)(\|A\|\|x\| + \|b\|)$, where $r = b - Ax$ is the residual, ϵ is the machine accuracy, $f(n)$ is a slowly varying function of the matrix size n, typically $f(n) \sim \sqrt{n}$. Other termination criteria are also available.
- if $\|A\|$ is not known, the solvers can provide an estimate of its value using Higham's method;
- for matrices stored in CS format, we provide a full range of black-box routines, preconditioners (Incomplete Cholesky and LU, SSOR, Jacobi) and utilities (solution of the preconditioning equations when incomplete factorizations are used, matrix-vector and matrix transpose-vector multiplication).

Much of the new software for iterative solvers has already been included in our Numerical PVM Library [10], a parallel library for distributed memory machines. However, neither preconditioners nor utility routines are provided there yet: we felt that the requirements of distributed memory parallelism implied that storage schemes were strongly problem-dependent and no particular one was likely to satisfy the majority of users.

The CPU times listed in the tables in the following sections were obtained on an Indigo SGI workstation.

2 Iterative Solvers for Sparse Symmetric Linear Systems

The software introduced at Mark 17 [13] includes new versions of iterative algorithms already available, though in a less flexible form, in earlier Marks of the NAG Library:

- preconditioned Conjugate Gradient method [5], for positive-definite systems;
- a Lanczos method, based on the algorithm SYMMLQ [11], for indefinite systems.

Both algorithms are provided in a suite of basic routines and in black-box routines. In the latter case, the matrix must be stored in CS format.

A new incomplete Cholesky preconditioner is available for matrices in CS storage. The amount of fill-in occurring in the factorization can vary from zero to complete fill and can be controlled either by specifying the maximum level of fill allowed or by a drop tolerance. Optionally, the factorization can be modified in order to preserve row-sums, and diagonal pivoting can be employed, either with a user-specified ordering or using the Markowitz strategy [4] which aims to minimize fill-in. The diagonal elements may require perturbation to avoid the breakdown of the factorization.

The utilities provided are applicable only to matrices in CS storage and include routines to compute matrix-vector products, to solve the preconditioning equations, SSOR and Jacobi preconditioners.

We have carried out a range of numerical experiments to assess the performance and robustness of the new routines. These are fully reported in [13]; here we just give two examples. Below, we have included the results from the new routines using the Conjugate Gradient method without and with preconditioning (incomplete Cholesky factorization with zero level of fill), from the the preconditioned Conjugate Gradient method previously available in the Library (F01MAF/F04MAF, based on Harwell Library routine MA31 [6]) and the timings obtained using the new incomplete Cholesky preconditioner as a direct solver (i.e. with complete fill).

The first test problem is a discontinuous diffusion equation due to Stone [17].

$$\nabla \cdot (D \nabla \phi(x,y)) = f(x,y), \quad (x,y) \in \Omega = [0,32]^2$$

where

$$D = \begin{pmatrix} D_{xx} & 0 \\ 0 & D_{yy} \end{pmatrix}$$

and

$$D_{xx} = \begin{cases} 100 & (x,y) \in [5,12] \times [5,12] \\ 0 & (x,y) \in [12,19] \times [21,28] \\ 1 & \text{elsewhere} \end{cases} \qquad D_{yy} = \begin{cases} 100 & (x,y) \in [14,32] \times [0,16] \\ 0 & (x,y) \in [12,19] \times [21,28] \\ 1 & \text{elsewhere} \end{cases} .$$

Sources of strength 1.0, 0.5 and 0.6 are placed at $(3,3)$, $(3,27)$ and $(23,4)$ respectively. Sinks of strength -1.83 and -0.27 are placed at $(14,15)$ and $(27,27)$ respectively. Zero flux boundary conditions are implemented on the boundary of Ω.

h^{-1}	8	16	32	64	128
n	64	256	1024	4096	16384
non-zeros	162	694	2894	11738	47270
CG	93 (0.12)	338 (1.58)	967 (19.98)	2246 (221.31)	4638 (2406.50)
PCG	18 (0.05)	28 (0.29)	41 (1.72)	63 (11.62)	96 (81.30)
F01MAF/F04MAF	18 (0.04)	40 (0.25)	40 (1.53) [2]	26 (7.59) [3]	27 (46.24) [4]
Direct Method	(0.03)	(0.17)	(1.73)	(17.23)	(235.24)

Table 1: Results for Stone's test problem: *Number of iterations* (*CPU time*) [*number of restarts*].

Table 1 gives the number of iterations and the CPU times required by each method. The numbers in square brackets show the number of unsuccesful runs with F01MAF/F04MAF before an adequate drop tolerance could be found: their CPU time is *not* included in the table. The two ICCG methods, i.e. the new PCG routines and the older F01MAF/F04MAF, are clearly the best for this test problem, although the direct method competes very well on the coarser meshes. Note that the numbers of iterations required by the new PCG routine suggest that the condition number has been reduced to $O(h^{-1})$. For this problem CG without preconditioning is very inefficient. F01MAF/F04MAF proved more efficient than the new PCG routines, but we should comment that a number of unsuccesful runs were carried out before a successful drop tolerance parameter could be defined: the new PCG routines appear more robust than F01MAF/F04MAF.

n	100	200	400	800	1600	3200
non-zeros	554	1110	2239	4403	8806	17610
CG	1070 (2.51)	2225 (10.97)	3833 (39.37)	5053 (107.32)	5806 (255.54)	6283 (583.39)
PCG	17 (0.11)	26 (0.34)	38 (0.91)	17 (1.10)	27 (3.16)	43 (9.55)
F01MAF/ F04MAF	64 (0.24)	55 (0.49) [2]	45 (1.08) [3]	100 (5.38) [4]	67 (7.80) [4]	60 (32.11) [6]
Direct Method	(0.09)	(0.36)	(1.95)	(10.79)	(61.75)	(520.59)

Table 2: Results for random matrices with $\nu = 10$: *Number of iterations* (*CPU time*) [*number of restarts*].

Table 2 reports the results for random matrices with average $\nu = 10$ non-zero elements per row. Random test matrices were generated from tridiagonal matrices of known spectrum by performing a sequence of random elementary plane rotations, in CS format, until the average number of non-zeros per row exceeded the specified value ν. All random matrices were chosen to have condition number $\kappa_2(A) = 10^6$. The unmodified zero-fill incomplete Cholesky preconditioner was used for the new routines. The new PCG routine is the most efficient, although for small values of n there is little difference between the ICCG methods and the direct method. Preconditioning significantly improves the efficiency of the unpreconditioned CG algorithm which suffers from the lack of clustering in the spectrum. Once again, the CPU time required by experimentation with different values of the drop tolerance was omitted from the results for F01MAF/F04MAF.

3 Iterative Solvers for Sparse Unsymmetric Linear Systems

We will introduce at Mark 18 new software [14], already completed, which includes the following solvers:

- the restarted generalized minimum residual (RGMRES) method [12];
- the conjugate-gradient squared (CGS) method [16];
- the (polynomial) bi-conjugate gradient stabilized (ℓ) (Bi-CGSTAB (ℓ)) method [15]. This was extensively modified to increase its robustness: in particular, an order less than the specified value of ℓ may be used or the computation automatically restarted

when stability considerations require it. When $\ell = 1$, the method is identical to Van der Vorst's Bi-CGSTAB method [18].

A routine for incomplete LU factorization, black-boxes for ILU, SSOR and Jacobi preconditioned methods, and utility routines are provided for matrices in CS format.

As for the incomplete Cholesky factorization, the new incomplete LU factorization allows arbitrary levels of fill. The strategy used carries out pivoting by rows for sparsity and by columns for stability. The factorization may optionally be modified to preserve the row-sum of the original matrix. A *local restart* scheme is employed for the cases when the incomplete factorization breaks down.

A range of numerical experiments were carried out [14]. Here, we report only the results for random matrices and for some matrices from the Harwell-Boeing collection [2]. In the results below, we have included the direct solvers F01BRF/F04AXF, based on the Harwell Library package MA28 [1] and currently available in the NAG Library, the new Harwell direct solver MA48 [3] and the timings obtained using the new ILU preconditioner as a direct solver (i.e. allowing complete fill).

n	200	400	800	1600
NNZ	2000	4028	8009	16000
PRGMRES	20 (1.14) [2]	10 (2.72) [2]	20 (2.71) [1]	20 (5.78) [1]
PCGS	9 (1.08) [2]	5 (2.70) [2]	8 (2.42) [1]	7 (4.93) [1]
PBICGSTAB	8 (1.13) [2]	4 (2.66) [2]	8 (2.59) [1]	8 (5.22) [1]
F01BRF/F04AXF	(0.70)	(4.03)	(38.06)	(272.16)
Direct Method	(1.62)	(9.64)	(77.63)	(551.06)
MA48	(0.43)	(2.09)	(14.86)	(115.47)

Table 3: Results for random matrices with $\nu = 10$: *Number of iterations* (*CPU time*) [*level of fill*].

Random test matrices were generated by performing randomly indexed plane rotations on a lower bidiagonal matrix of given singular values, in CS format, until the average number of non-zeros per row exceeded the specified value ν. The condition number of the matrix has been arbitrarily chosen to be 10^6. Table 3 shows the results obtained for random test matrices with $\nu = 10$. The unpreconditioned iterative methods all failed to converge within the limit of 1000 iterations. For small values of n the direct methods and the preconditioned iterative methods require similar CPU times, but the direct methods are to be preferred since they require no experimentation with algorithmic control parameters. For larger values of n the preconditioned iterative methods begin to show their advantage; for $n = 1600$ the best iterative method is more than 23 times faster than the best direct method. The preconditioner used as a direct method is considerably slower than the other direct methods shown.

The SHERMAN matrices in the Harwell-Boeing collection are a set of 5 oil reservoir simulation challenge matrices. For these matrices, ILU without pivoting was found to provide the best preconditioning. Table 4 shows that the preconditioned iterative methods compete with the direct methods for the small cases (SHERMAN1 and SHERMAN4), and are clearly superior for larger values of n. The new ILU preconditioner used as a direct method was less efficient than either F01BRF/F04AXF or MA48 for SHERMAN1 and SHERMAN4, but almost matched MA48 for SHERMAN2 AND SHERMAN5. For SHERMAN3 it was at least comparable to F01BRF/F04AXF.

	SHERMAN1	SHERMAN2	SHERMAN3	SHERMAN4	SHERMAN5
n	1000	1080	5005	1104	3312
no. non-zeros	3750	23094	20033	3786	20793
PRGMRES	70 (2.30)	20 (3.72)	130 (21.49)	50 (1.69)	50 (7.66)
PCGS	32 (1.75)	7 (3.03)	75 (19.98)	25 (1.42)	25 (6.61)
PBICGSTAB	32 (1.91)	8 (3.30)	48 (14.36)	24 (1.49)	20 (5.98)
RGMRES	*	*	*	760 (11.89)	*
CGS	274 (7.17)	*	*	94 (2.48)	*
BICGSTAB	312 (8.46)	*	*	92 (2.62)	*
F01BRF/F04AXF	(4.55)	(287.99)	(730.52)	(4.32)	(162.48)
Direct MEthod	(11.21)	(157.30)	(583.28)	(32.20)	(102.56)
MA48	(1.79)	(129.99)	(74.86)	(1.56)	(102.99)

Table 4: Results for the SHERMAN matrices from the Harwell-Boeing collection: *Number of iterations* (*CPU time*)

4 Future Developments

The software has been designed so that it can easily be extended in future Marks of the Library in various ways:

- more iterative solvers: QMR, for both symmetric and unsymmetric systems, and TFQMR, for unsymmetric systems, are prime candidates;
- more preconditioners;
- more storage formats.

We also plan to make these iterative solvers available in our C and Fortran 90 Libraries.

References

[1] I.S. Duff (1977), *MA28 — a set of Fortran subroutines for sparse unsymmetric linear equations.* A.E.R.E. Report R.8730., HMSO, London.

[2] I.S. Duff, R.G. Grimes, and J.G. Lewis (1992), *Users' guide for the Harwell–Boeing sparse matrix collection (Release 1)*, Technical Report RAL-92-086, Rutherford Appleton Laboratory, Chilton, England.

[3] I.S. Duff and J.K. Reid (1995), *The design of MA48, a code for the direct solution of sparse unsymmetric linear systems of equations.* Technical Report RAL-TR-95-039, Rutherford Appleton Laboratory, Chilton, England.

[4] H.M. Markowitz (1957), The elimination form of the inverse and its application to linear programming, *Management Sci.* **3** 255–269.

[5] J. Meijerink and H. Van der Vorst (1977), An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix, *Math. Comput.* **31** 148–162.

[6] Munksgaard N (1980) Solving sparse symmetric sets of linear equations by preconditioned conjugate gradients *ACM TOMS* **6** (**2**) 206–219.

[7] NAG (1995) *NAG Fortran Library Manual, Mark 17*, The Numerical Algorithms Group Ltd, Oxford, UK.

[8] NAG (1995) *NAG Fortran 90 Library Manual, Release 2*, The Numerical Algorithms Group Ltd, Oxford, UK.

[9] NAG (1994) *NAG C Library Manual, Mark 3*, The Numerical Algorithms Group Ltd, Oxford, UK.

[10] NAG (1995) *NAG Numerical PVM Library Manual, Release 1*, The Numerical Algorithms Group Ltd, Oxford, UK.

[11] C.C. Paige and M.A. Saunders (1975), Solution of Sparse Indefinite Systems of Linear Equations, *SIAM J. Numer. Anal.* **12** 617–629.

[12] Y. Saad and M. Schultz (1986), GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems, *SIAM J. Sci. Statist. Comput.* **7** 856–869.

[13] S.Salvini and G.J.Shaw (1995) *An Evaluation of New NAG Library Solvers for Large Sparse Symmetric Linear Systems*, NAG Technical Report TR1/95, The Numerical Algorithms Group Ltd, Oxford, UK.

[14] S.Salvini and G.J.Shaw (1996) *An Evaluation of New NAG Library Solvers for Large Sparse Unsymmetric Linear Systems*, NAG Technical Report (in preparation), The Numerical Algorithms Group Ltd, Oxford, UK.

[15] G.L.G. Sleijpen and D.R. Fokkema (1993), BiCGSTAB(ℓ) for linear equations involving matrices with complex spectrum, *ETNA* **1** 11–32.

[16] P. Sonneveld (1989), CGS, a fast Lanczos-type solver for nonsymmetric linear systems, *SIAM J. Sci. Statist. Comput.* **10** 36–52.

[17] Stone H L (1968) Iterative solution of implicit approximation of multidimensional partial differential equations *SIAM J. Numer. Anal.* **5 (3)** 530–558.

[18] H. Van der Vorst (1989), Bi-CGSTAB, A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems, *SIAM J. Sci. Statist. Comput.* **13** 631–644.

Room TBA	***Workshop Chair***	
7:30 p.m.	***P. Forsyth***	Navier-Stokes & Euler Equations

Wednesday Evening's Workshop

Iterative Methods for Compressible Navier-Stokes and Euler Equations

Organizers: W. P. Tang and P. A. Forsyth

Abstract

This workshop will focus on methods for solution of compressible Navier-Stokes and Euler equations. In particular, attention will be focused on the interaction between the methods used to solve the non-linear algebraic equations (e.g. full Newton or first order Jacobian) and the resulting large sparse systems. Various types of block and incomplete LU factorization will be discussed, as well as stability issues, and the use of Newton-Krylov methods. These techniques will be demonstrated on a variety of model transonic and supersonic airfoil problems. Applications to industrial CFD problems will also be presented. Experience with the use of C++ for solution of large scale problems will also be discussed. The format for this workshop will be four fifteen minute talks, followed by a roundtable discussion.

Speakers

- T. Barth, NASA; T. Chan, UCLA; W.P. Tang, University of Waterloo
  Implicit Parallel Preconditioning Techniques for Computational Fluid Dynamics
- E. Chow and Y. Saad, University of Minnesota
  Block Preconditioning for the Compressible Flow Calculations
- D. P. Young, Boeing
  Newton Krylov Methods in Nonlinear Aerodynamics
- P. Forsyth, University of Waterloo; H. Jiang, Lucent Technologies
  Iterative Methods for Full Newton Jacobians for Compressible Navier Stokes Equations

THURSDAY, APRIL 11TH

Topic: *Navier-Stokes*	*Session Chair:* *Howard Elman*	*Room A*
8:00 - 8:30	V. Sarin	An Efficient Iterative Method for the Generalized Stokes Problem
8:30 - 9:00	P. Fischer	A Deflation based Parallel Algorithm for Spectral Element Solution of the Incompressible Navier-Stokes Equations
9:00 - 9:30	A. Wathen	An Iteration for Indefinite and Non-Symmetric Systems and its Application to the Navier-Stokes Equations
9:30 - 10:00	H. Elman	Perturbation of Eigenvalues of Preconditioned Navier-Stokes Operators

An Efficient Iterative Method for the Generalized Stokes Problem*

Ahmed Sameh† and Vivek Sarin‡

Abstract

This paper presents an efficient iterative scheme for the generalized Stokes problem, which arises frequently in the simulation of time-dependent Navier-Stokes equations for incompressible fluid flow. The general form of the linear system is

$$\begin{pmatrix} A & B \\ B^T & 0 \end{pmatrix} \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} f \\ 0 \end{pmatrix} \tag{1}$$

where $A = \alpha M + \nu T$ is an $n \times n$ symmetric positive definite matrix, in which M is the mass matrix, T is the discrete Laplace operator, α and ν are positive constants proportional to the inverses of the time-step Δt and the Reynolds number Re respectively, and B is the discrete gradient operator of size $n \times k$ $(k < n)$. Even though the matrix A is symmetric and positive definite, the system is indefinite due to the incompressibility constraint $(B^T u = 0)$. This causes difficulties both for iterative methods and commonly used preconditioners. Moreover, depending on the ratio α/ν, A behaves like the mass matrix M at one extreme and the Laplace operator T at the other, thus complicating the issue of preconditioning.

The generalized Stokes problem is one of the most time-consuming steps in the large scale simulation of incompressible fluid flows. Iterative methods, which are indispensable for solving this problem in realistic situations, rely heavily on effective preconditioners that can be efficiently implemented on multiprocessors. Therefore, the issues of efficient iterative algorithms and robust, effective and parallelizable preconditioners for the generalized Stokes problem must be resolved satisfactorily.

Previous efforts to solve the linear system in Eq. 1 can be broadly classified as follows:

1. **Uzawa-type methods.** These methods solve the linear system:

$$B^T A^{-1} B p = B^T A^{-1} f \tag{2}$$

*This research has been supported in part by the NSF under the grant NSF/CDA 9396332-001.

†Dept. of Computer Science, Univ. of Minnesota, Twin Cities

‡Dept. of Computer Science, Univ. of Illinois, Urbana-Champaign

Each iteration requires the solution of the linear system $Ax = b$. If an iterative method is used to solve $Ax = b$, we obtain a two-level solver with inner and outer iterations. The inner system may be solved in many ways including multigrid, spectral methods and preconditioned CG algorithm. Convergence may be considerably improved by replacing A with $A_\epsilon = A + \frac{1}{\epsilon}BB^T$ or by accelerating the linear system by the CG method. Single-level methods which approximate the generalized Stokes problem by a *nearly* incompressible problem have also been developed. However, effective preconditioners for this approach often rely on expensive linear system solves using an inner iterative method. The reader is referred to [4, 1, 3, 7, 10, 9] for a variety of Uzawa-type methods.

2. **Projection methods.** Projection methods (see [8, 2]) require the solution of the linear system:

$$PAPu = Pf \tag{3}$$

where P is the orthogonal projection onto the null space of B^T. In each iteration, one computes the action of the projector $P = I - B(B^TB)^{-1}B^T$ on a vector u, which requires solving the system $B^TBx = b$, where B^TB is analogous to a Laplace operator on the domain. For a large three dimensional problem, computing the projection of u onto $Null(B^T)$ is almost as difficult as solving the system $Au = b$, and may require the use of an inner iterative method.

3. **Augmentation methods.** Augmentation methods proposed in [5, 6] reformulate the system by augmenting the matrix B with a suitable matrix F, and use the CG method to solve the resulting symmetric and positive definite system. Here, one needs to solve systems of the type $(B, F)x = b$ and $(B, F)^Tx = b$. At present however, the choice of F for which (B, F) is easy to invert and the system matrix is well-conditioned, has not yet been determined.

Both Uzawa-type and projection methods are expensive two-level nested iterative methods with rapid convergence assured only for certain extreme values of α/ν. Further, these schemes suffer from the lack of good preconditioners, which have been elusive due to the complicated coefficient matrices arising in these methods.

In this paper, we propose a multilevel scheme for the solution of elliptic problems that has the desirable properties of effective preconditioning and efficient implementation on multiprocessors. We also extend this multilevel scheme to the generalized Stokes problem and present numerical experiments for the solution of the driven cavity problem.

Multilevel Scheme for the Generalized Stokes Problem

A multilevel approach is used to compute a basis for $Null(B^T)$, where B is the discrete gradient operator. This null-space-basis, P_2 is expressed as the product of a sequence of

sparse matrices, s.t. it requires only $O(n)$ operations to perform a matrix-vector product with P_2. In fact, we determine matrices P, D and Z such that

$$P^T B Z = \begin{pmatrix} D \\ 0 \end{pmatrix}$$

where P is a nonsingular $n \times n$ matrix, D is a $k \times k$ diagonal matrix and Z is a $k \times k$ orthogonal matrix. Further, $P = [P_1, P_2]$ where P_2 comprises of the last $n-k$ columns of P.

Eq. 1 may be rewritten as

$$\begin{pmatrix} P_1^T A P_1 & P_1^T A P_2 & D \\ P_2^T A P_1 & P_2^T A P_2 & 0 \\ D & 0 & 0 \end{pmatrix} \begin{pmatrix} \hat{u}_1 \\ \hat{u}_2 \\ Z^T p \end{pmatrix} = \begin{pmatrix} P_1^T f \\ P_2^T f \\ 0 \end{pmatrix} \tag{4}$$

where $u = P\hat{u}$. The algorithm for computing the solution of the generalized Stokes problem is as follows:

Algorithm *Multilevel_Generalized_Stokes*

1. Set $\hat{u}_1 = 0$.
2. Solve using the CG method
$$P_2^T A P_2 \hat{u}_2 = P_2^T f \tag{5}$$
3. Compute $\hat{p} = D^{-1} P_1^T (f - A P_2 \hat{u}_2)$.
4. Set $u = P\hat{u}$ and $p = Z\hat{p}$.

To solve the generalized Stokes problem efficiently, it is imperative that the matrix $P_2^T A P_2$, in which $A = \alpha M + \nu T$, is well-conditioned for a wide range of the parameters α and ν. When α/ν is large, A may be approximated by αM, and when α/ν is small, A approaches the Laplace operator νT. It can be shown that for large values of α/ν, standard preprocessing techniques like diagonal scaling of the matrix A yields a well-conditioned system. In cases where α/ν is small, we expect $P_2^T A P_2$ to be well-conditioned due to the "inverse" nature of the matrices A and $P_2^T P_2$.

Numerical Experiments

Numerical experiments for the solution of the generalized Stokes problem for a lid-driven cavity problem were performed on an IBM RS6000 with 66.5 MHz clock and 256 MB memory.

We consider a two-dimensional unit square domain with $\mathbf{u} = (0,0)^T$ on the boundary except the top edge where $\mathbf{u} = (1,0)^T$, and $p = 0$ at the bottom-left corner. The domain is uniformly discretized with the Marker-and-Cell (MAC) scheme. Fig. 1 presents the number of iterations required to solve the linear system in Eq. 5. These results indicate that in the worst case, the condition number of the system matrix $P_2^T A P_2$ is $O(\sqrt{n})$, which indicates effective preconditioning of the system matrix.

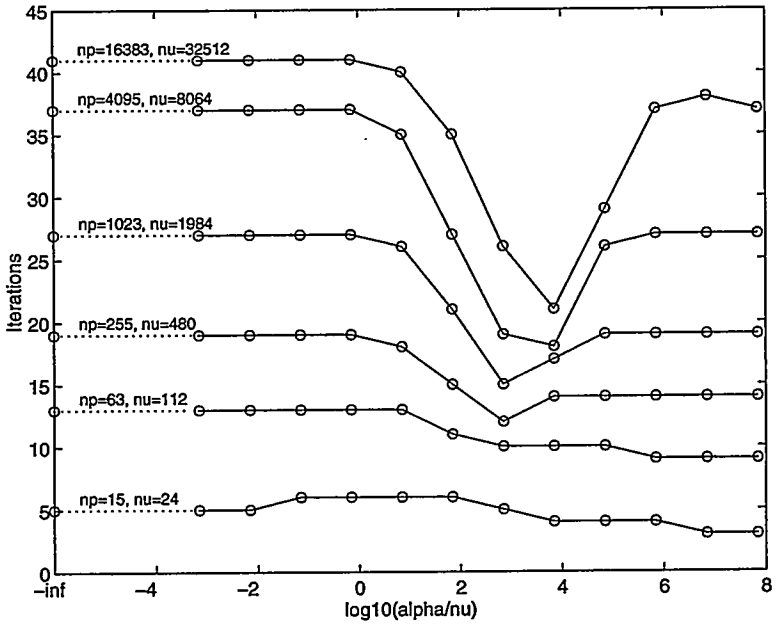


Figure 1: Multilevel scheme for the MAC scheme on a uniform square mesh. *np*= pressure unknowns, *nu*= velocity unknowns.

These results clearly indicate that the multilevel scheme is a robust and effective preconditioning strategy for the solution of the linear system of the generalized Stokes problem for a wide choice of α/ν. This makes the multilevel scheme an attractive approach especially since convergence is relatively independent of $\alpha = 1/\Delta t$, and therefore doesn't constrain the choice of the time-step Δt. Furthermore, it is a single-level scheme with $O(n)$ computation per iteration, unlike the two-level Uzawa-type and projection methods which require expensive linear system solves in each iteration. The matrix-vector product with the system matrix $P_2^T A P_2$ can be efficiently parallelized for multiprocessor implementations. It must also be pointed out that the effort required in generating the basis P_2 can be amortized over a number of system solves since B is invariant over numerous time steps for the nonlinear Navier-Stokes equations. Current work includes numerical experiments with systems ob-

tained using the finite elements method on unstructured grids for two and three-dimensional domains. We expect to present comparative results of numerical experiments for some of these problems.

References

[1] R.E. Bank, B.D. Welfert, and H. Yserentant. A class of iterative methods for solving saddle point problems. *Numer. Math.*, 56:645–666, 1990.

[2] R. Bramley. An orthogonal projection algorithm for generalized stokes problems. Technical Report 1190, CSRD, Univ. of Illinois Urbana-Champaign, 1992.

[3] N. Dyn and Jr. W.E. Ferguson. The numerical solution of equality-constrained quadratic programming problems. *Math. Comp.*, 41(163):165–170, 1983.

[4] M. Fortin and R. Glowinski. *Augmented Lagrangian Methods: Applications to the Numerical Solution of Boundary-Value Problems.* North-Holland, New York, 1983.

[5] F. G. Lou and A. Sameh. An expansion method for solving saddle-point problems. Technical Report 92-083, AHPCRC, Univ. of Minnesota, 1992.

[6] F. G. Lou, A. Sameh, and V. Sarin. An augmentation method for solving saddle-point problems. Technical Report 95/29, MSI, Univ. of Minnesota, 1995.

[7] T. Rusten and R. Winther. A preconditioned iterative method for saddle-point problems. *SIAM J. Matrix Anal. Appl.*, 13(3):887–904, 1992.

[8] A. Sameh and J. A. Wisniewski. A trace minimization algorithm for the generalized eigenvalue problem. *SIAM J. Numer. Anal.*, 19(6):1243–1259, 1982.

[9] D. Silvester and A. Wathen. Fast iterative solution of stabilised stokes systems part 2: Using general block preconditioners. *SIAM J. Numer. Anal.*, 31(5):1352–1367, 1994.

[10] A. Wathen and D. Silvester. Fast iterative solution of stabilised stokes systems part 1: Using simple diagonal preconditioners. *SIAM J. Numer. Anal.*, 30(3):630–649, 1993.

A Deflation based Parallel Algorithm for Spectral Element Solution of the Incompressible Navier-Stokes Equations

Paul F. Fischer
Division of Applied Mathematics
Brown University
Providence, RI 02912

Abstract

Efficient solution of the Navier-Stokes equations in complex domains is dependent upon the availability of fast solvers for sparse linear systems. For unsteady incompressible flows, the pressure operator is the leading contributor to stiffness, as the characteristic propagation speed is infinite. In the context of operator splitting formulations, it is the pressure solve which is the most computationally challenging, despite its elliptic origins. We seek to improve existing spectral element iterative methods for the pressure solve in order to overcome the slow convergence frequently observed in the presence of highly refined grids or high-aspect ratio elements.

Our present algorithm, developed by Rønquist [1], uses a two-level iteration scheme in which the coarse-grid modes are projected out of the governing system via deflation [2]. The coarse-grid system is generated via application of a piecewise constant prolongation operator, which is admissible due to the $\mathcal{L}^2$ approximation space used for the pressure. In one-dimension, this approach leads to unit condition number when block-Jacobi preconditioning is applied to the deflated operator [1]. However, in more complex domains, the conditioning deteriorates when the grid is highly refined, and numerical evidence suggests that this is due to lack of inter-element (subdomain) coupling in the block-Jacobi preconditioner. Unfortunately, the $\mathcal{L}^2$ pressure approximation space based upon the (interior) Gauss quadrature points does not readily lend itself to incorporation of subdomain operators with overlap. However, inter-element coupling can be increased by enriching the coarse-grid space, albeit with increased cost in the coarse-grid solve.

We explore the potential of using an enriched coarse-grid operator (based upon piecewise polynomial approximation) to reduce the condition number of the governing pressure system. The success of the method is largely dependent upon the availability of an efficient direct solver for the coarse-grid system, a problem which is widely recognized as a source of difficulty on distributed memory parallel computers. We present a new parallel algorithm for the repeated coarse-grid solves which is more efficient than standard approaches. Its use offsets the additional overhead incurred by the enlarged coarse-grid system, and the combination leads to a faster solution algorithm. Parallel results for several two- and three-dimensional Navier-Stokes calculations are presented.

[1] E.M. Rønquist, *A Domain Decomposition Method for Elliptic Boundary Value Problems: Application to Unsteady Incompressible Fluid Flow,* in Fifth Conference on Domain Decomposition Methods for Partial Differential Equations T.F. Chan, D.E. Keyes, G.A. Meurant, J.S. Scroggs, and R.G. Voigt, eds., SIAM, Philadelphia, PA, 1992.

[2] R.A. Nicolaides, *Deflation of conjugate gradients with applications to boundary value problems* SIAM J. Numer. Anal., 24(2), (1987) pp. 355-65.

An iteration for Indefinite and Non-symmetric systems and its application to the Navier-Stokes equations

Andy Wathen, Oxford University, UK
Gene Golub, Stanford University, USA

A simple fixed point linearisation of the Navier-Stokes equations leads to the Oseen problem which after appropriate discretisation yields large sparse linear systems with coefficient matrices of the form

$$\begin{pmatrix} A & B^T \\ B & -C \end{pmatrix}.$$

Here A is non-symmetric but its symmetric part is positive definite, and C is symmetric and positive semi-definite. Such systems arise in other situations.

In this talk we will describe and present some analysis for an iteration based on an indefinite and symmetric preconditioner of the form

$$\begin{pmatrix} D & B^T \\ B & -C \end{pmatrix}.$$

In the case of the Oseen problem, this covers the case of preconditioning by the solution of a Stokes system - in recent years several optimal iterative solvers for the Stokes systems have been proposed (see [1]) and can be employed here.

We will present some convergence analysis for a simple iteration based on this preconditioner and for GMRES acceleration of this underlying process. Numerical results for both these situations as well as QMR acceleration will be presented.

[1] Elman, H.C., 1995, 'Multigrid and Krylov subspace methods for the discrete Stokes equations', preprint (University of Maryland, Dept of Computer Science)

Perturbation of Eigenvalues of Preconditioned Navier-Stokes Operators

Howard C. Elman
Department of Computer Science and
Institute for Advanced Computer Studies
University of Maryland, College Park, MD 20742
e-mail: elman@cs.umd.edu

Abstract. We study the sensitivity of algebraic eigenvalue problems associated with matrices arising from linearization and discretization of the steady-state Navier-Stokes equations. In particular, for several choices of preconditioners applied to the system of discrete equations, we derive upper bounds on perturbations of eigenvalues as functions of the viscosity and discretization mesh size. The bounds suggest that the sensitivity of the eigenvalues is at worst linear in the inverse of the viscosity and quadratic in the inverse of the mesh size, and that scaling can be used to decrease the sensitivity in some cases. Experimental results supplement these results and confirm the relatively mild dependence on viscosity. They also indicate a dependence on the mesh size of magnitude smaller than the analysis suggests.

Topic:	*Session Chair:*	*Room B*
Krylov Methods	*M. Gutknecht*	

8:00 - 8:30	T. DeLillo	Numerical Conformal Mapping Methods for Exterior and Doubly Connected Regions
8:30 - 9:00	M. Sosonkina	A New Adaptive GMRES Algorithm for Achieving High Accuracy
9:00 - 9:30	E. de Sturler	Truncation Strategies for (Nested) Krylov Methods
9:30 - 10:00	K. Ressel	Hybrid Lanczos-Type Product Methods

Numerical Conformal Mapping Methods for Exterior and Doubly Connected Regions

Thomas K. DeLillo†
Department of Mathematics and Statistics, Wichita State University, Wichita, KS 67260-0033, email: delillo@twsuvm.uc.twsu.edu

John A. Pfaltzgraff
Department of Mathematics, The University of North Carolina at Chapel Hill, CB 3250, Phillips Hall, Chapel Hill, NC 27599-3250, email: jap@math.unc.edu

Abstract. Methods are presented and analyzed for approximating the conformal map from the exterior of the disk to the exterior a smooth, simple closed curve and from an annulus to a bounded, doubly connected region with smooth boundaries. The methods are Newton-like methods for computing the boundary correspondences and conformal moduli similar to Fornberg's method for the interior of the disk. We show that the linear systems are discretizations of the identity plus a compact operator and, hence, that the conjugate gradient method converges superlinearly.

1. Introduction. Several methods for numerical conformal mapping are presented in the texts [Ga] and [He]. For simply connected regions with smooth boundaries several methods which map from the unit disk to the region and employ the fast Fourier transform (FFT) are available [De], [DE1], [Fo1], [Weg1], [Weg2], [Weg4]. For doubly connected regions with smooth boundaries, methods which use the annulus as a computational domain and employ FFTs are available [Fo2], [LM], [Weg3]. Several of these methods are Newton-like methods for computing the boundary correspondence. It can be shown for the interior disk case [Fo1], [Weg4] that the matrices for the inner systems are discretizations of the identity plus a compact operator and that the conjugate gradient method therefore converges superlinearly [Weg2], [Wid]. In this paper we extend this analysis to Fornberg's method for the exterior of the disk and the annulus where a Newton-like version of [Fo2] is given.

Currently the most robust and stable methods are based on solving Riemann-Hilbert problems for the Newton updates. In [Weg4], a discrete interpolation problem is solved and in [Weg5] (which can be specialized to the disk) damping of higher order Fourier coefficients is used to avoid the instabilities in [Weg1]; see e.g. [De], [DE1]. In both cases convergence of the numerical method is proved. It is not known to us whether these methods generalize to the doubly connected cases or whether [LM] or [Weg3] suffer from instabilities like [Weg1].

The Fornberg-like methods below generalize to a variety of computational domains, such as ellipses [DE2], [Weg5] and cross shaped regions [DEP], and apparently to multiply connected circle domains. These methods are based on finding analyticity conditions for the computational domains and provide, we believe, an interesting class of problems for conjugate gradient-like methods. In section 2 of this paper, we define certain useful linear

† This author's research was partially supported by U. S. Department of Energy grant DE-FG02-92ER25124 and by National Science Foundation EPSCoR grant OSR-9255223

operators. In section 3, we state the analyticity conditions for the exterior disk and annulus. In sections 4 and 5, we discuss the exterior disk and annulus maps, respectively,

2. Linear operators. We have need of several operators occuring in Fourier analysis; see e.g. [Weg2]. We take the domain of these operators to be the set of 2π-periodic functions h in L^2. Let

$$h(\theta) = \sum_{k=-\infty}^{\infty} a_k e^{ik\theta}.$$

Then the conjugation operator, K, is given by

$$Kh(\theta) = \frac{1}{2\pi} P.V. \int_0^{2\pi} \cot(\frac{\theta-\phi}{2}) h(\phi) d\phi = \sum_{k=-\infty}^{-1} i a_k e^{ik\theta} - \sum_{k=1}^{\infty} i a_k e^{ik\theta}.$$

$$Jh(\theta) = \frac{1}{2\pi} \int_0^{2\pi} h(\theta) d\theta = a_0.$$

$$P_+ h = \frac{1}{2}(I + iK - J)h = \sum_{k=1}^{\infty} a_k e^{ik\theta}$$

$$P_- h = \frac{1}{2}(I - iK + J)h = \sum_{k=-\infty}^{0} a_k e^{ik\theta}$$

We will also make use of the discretizations of the operators above using N-point trigonometric interpolation. If

$$\underline{h} = (h_0, ..., h_{N-1})^T, \; h_k = h(\theta_k), \; \theta_k = 2\pi k/N, \; k = 0, ..., N-1,$$

and

$$\hat{a}_k = \frac{1}{N} \sum_{j=0}^{N-1} h_j w^{-jk}, \; w = e^{2\pi i/N},$$

then with $n = N/2$ the trigonometric polynomial interpolating h is given by

$$T_N h(\theta) = \sum_{k=-n+1}^{n-1} \hat{a}_k e^{ik\theta} + \hat{a}_n \cos(n\theta),$$

and the discrete operators corresponding to those introduced above are

$$K_N h(\theta) = \sum_{k=-n+1}^{-1} i\hat{a}_k e^{ik\theta} - \sum_{k=1}^{n-1} i\hat{a}_k e^{ik\theta},$$

$$J_N h(\theta) = \hat{a}_0 - \hat{a}_n \cos(n\theta),$$

and

$$P_{+,N}h = \frac{1}{2}(T_N + iK_N - J_N)h = \sum_{k=1}^{n-1} \hat{a}_k e^{ik\theta} + \hat{a}_n \cos n\theta$$

$$P_{-,N} = \frac{1}{2}(T_N - iK_N + J_N) = \sum_{k=-n+1}^{0} \hat{a}_k e^{ik\theta}.$$

If the $N \times N$ matrix $F := (w^{-k\nu}), k, \nu, = 0, ..., N-1$, then

$$\frac{1}{N} F\underline{h} = \underline{a} = (\hat{a}_0, ..., \hat{a}_{N-1})^T = (\hat{a}_0, ..., \hat{a}_n, \hat{a}_{-n+1}, ..., \hat{a}_{-1})^T$$

since $\hat{a}_k = \hat{a}_{k-N}$. Note

$$\frac{1}{N} F^H F = \frac{1}{N} F F^H = I_N \quad (H = \text{Hermitian transpose})$$

In matrix form

$$P_{-,N} = \frac{1}{N} F^H I_{-,N} F \quad P_{+,N} = \frac{1}{N} F_N^H I_{+,N} F$$

where $I_{-,N} = \text{diag}(1, 0, \cdots, 0, 1, \cdots, 1)$ and $I_{+,N} = \text{diag}(0, 1, \cdots, 1, 0, \cdots, 0)$ are $N \times N$ diagonal matrices of rank n.

3. Analyticity conditions. We will use the following conditions:

Theorem. *A function $f \in \text{Lip}(C)$ on the boundary of the unit disk C extends to an analytic function on the exterior of C with $h(\infty)$ finite if and only if*

$$P_+ f(e^{i\theta}) = 0.$$

Theorem. *Let E be the annulus $\rho \leq |z| \leq 1$ with boundaries $C = C_1 - C_2$ where $C_1 : e^{i\theta}, C_2 : \rho e^{i\theta}, 0 < \rho < 1$. Then $f \in \text{Lip}(C)$ extends analytically to E if and only if*

$$\int_{C_1} f(z) z^k dz = \int_{C_2} f(z) z^k dz, \ k = 0, \pm 1, \pm 2, \ldots$$

Then for

$$f(e^{i\theta}) = \sum_{k=-\infty}^{\infty} a_k e^{ik\theta},$$

$$f(\rho e^{i\theta}) = \sum_{k=-\infty}^{\infty} b_k e^{ik\theta},$$

we have

$$\rho^k a_k = b_k, \ k = 0, \pm 1, \pm 2, \ldots.$$

4. Fornberg-like method for simply connected regions exterior to a Jordan curve. Here we extend the method [Fo1] for the interior of the disk to the exterior case. We show that the analysis for the interior case [Weg2], [Wid] carries over to the exterior case.

We wish to find the conformal map f from the exterior of the unit disk to the exterior of a smooth Jordan curve $\Gamma : \gamma(S)$ parametrized by, for instance, arclength S with $f(\infty) = \infty$ and $f'(\infty) > 0$ or $f(1)$ fixed. In this case, f extends smoothly to the boundary and $f(e^{i\theta}) = \gamma(S(\theta))$. The numerical problem is to approximate the *boundary correspondence* $S(\theta)$. This will yield an approximation to the Laurent series $f(z) = a_1 z + a_0 + \sum_{k=1}^{\infty} a_{-k} z^{-k}$. Newton-like methods can be used for determining $S(\theta)$. At the kth Newton step a correction $U^{(k)}(\theta)$ real to $S^{(k)}(\theta)$ is computed from the condition that the linearization

$$h(e^{i\theta}) = \xi(\theta) + e^{i(\beta(\theta)-\theta)} U^{(k)}(\theta) \approx f(e^{i\theta}) e^{-i\theta},$$

where $\xi(\theta) = \gamma(S^{(k)}(\theta)) e^{-i\theta}$ and $\beta(\theta) = \arg \gamma'(S^{(k)}(\theta))$, extends analytically to the exterior of the unit disk with h analytic at ∞. From the analyticity conditions in section 3, we have

$$2P_+ h = (I + iK - J)h = 0.$$

This implies (with $U = U^{(k)}$) that

$$(I + iK - J) e^{i(\beta(\theta)-\theta)} U(\theta) = -2P_+ \xi(\theta).$$

Using U real gives

$$(I + R^{ext})U = r$$

where $R^{ext} = \mathrm{Re}(e^{-i(\beta-\theta)}(iK - J)e^{i(\beta-\theta)})$ and $r = -\mathrm{Re}(e^{-i(\beta-\theta)}(I + iK - J)\xi)$. For γ sufficiently smooth R^{ext} is a compact operator on L^2; see [Weg2, section 4] where $R^{ext} = -R_V = \mathrm{Re}(\overline{V}(iK - J)V), V = e^{i(\beta-\theta)}$. With $E = \mathrm{diag}_j(e^{i(\beta_j - \theta_j)}), j = 0, 1, \cdots, N-1$ discretization with $N-$point trigonometric interpolation gives

$$(I_N + R_N^{ext})U_N = r_N$$

The matrix

$$A_N = I_N + R_N^{ext} = \frac{2}{N} E^H F^H I_{+,N} F E$$

is thus symmetric with eigenvalues well-grouped around 1 and the conjugate gradient method converges superlinearly. The FFT is used to perform the matrix-vector multiplications in $O(N \log N)$. The Newton update is given by $S^{k+1} = S^k + U^k$ and we set $U_0 = 0$ to fix a boundary point as in [Fo1].

5. Fornberg-like method for the bounded, doubly connected regions. In this section, we generalize [Fo1] to doubly connected regions. Fornberg himself extended his method to the doubly connected case [Fo2]. He solves a system of equations [Fo2, eq. (6)], which are essentially our analyticity conditions, using a linearly convergent method of successive approximation. Here we show how to linearize these equations to get a

quadratically convergent, Newton-like method. We derive a symmetric linear system which is a discretization of the identity plus a compact operator, and so the conjugate gradient method converges superlinearly with $O(N \log N)$ matrix-vector multiplications using the FFT. Our linearization is that used by Luchini and Manzo [LM], however, they solve Riemann-Hilbert problems for the Newton updates. Wegmann [Weg3] also solves Riemann-Hilbert problems, but uses a slightly different and more expensive linearization. We do not know if these latter methods suffer from any instability, as in the simply connected case.

If the target region Ω is bounded by two smooth Jordan curves $\Gamma_1 : \gamma_1(S_1)$ and $\Gamma_2 : \gamma(S_2)$, we want to find the boundary correspondences $S_1(\theta)$ and $S_2(\theta)$ and the conformal modulus ρ such that $f(z)$ is analytic in the annulus $\rho < |z| < 1$ and $f(e^{i\theta}) = \gamma_1(S_1(\theta))$ and $f(\rho e^{i\theta}) = \gamma_2(S_2(\theta))$. We have programmed a Newton-like method to do this. At each Newton step we want to compute corrections $U_1(\theta)$, $U_2(\theta)$, and $\delta\rho$ to $S_1(\theta)$, $S_2(\theta)$, and ρ. With S_j arclength, $\beta_j(S_j(\theta)) := \arg \gamma_j'(S_j(\theta))$, $\xi_j(\theta) := \gamma_j(S_j(\theta))$, $j = 1, 2$, $\zeta(\theta) := f'(\rho e^{i\theta})e^{i\theta} = -ie^{\beta_2(S_2(\theta))} dS_2(\theta)/d\theta/\rho$, as in [LM] we linearize about S_1, S_2 as follows:

$$f(e^{i\theta}) = \xi_1(\theta) + e^{i\beta_1(S_1(\theta))} U_1(\theta)$$

$$f(\rho e^{i\theta}) = \xi_2(\theta) + e^{i\beta_2(S_2(\theta))} U_2(\theta) - \zeta(\theta)\delta\rho.$$

For the annulus, it is easier to begin with the discrete equations. We discretize the analyticity conditions for the annulus, sec. 3, and the linearizations above with $N-$point trigonometric interpolation to get a discrete approximation to the U_j's at the Fourier points, $\theta_k = 2\pi k/N, k = 0, 1, \ldots, N-1$. Letting a_k and b_k now denote the N discrete Fourier coefficients and using the $N-$periodicity $a_{k+N} = a_k$, we have with $N = 2n$

$$\underline{a} = (a_0, a_1, \ldots, a_n, a_{n+1}, \ldots, a_{N-1})^T = (a_0, a_1, \ldots, a_n, a_{-n+1}, \ldots, a_{-1})^T.$$

$\underline{b}$ is defined similarly. Next define the $N \times N$ matrices $P_1 = \mathrm{diag}(1, \rho, \ldots, \rho^{n-1}, 1, \ldots, 1)$ and $P_2 =$-diag$(1, \ldots, 1, 1, \rho^{n-1}, \ldots, \rho)$. If we set $a_n = b_n$ as in [Fo2, eq. 6], we write the discrete form of our analyticity conditions as

$$P_1\underline{a} + P_2\underline{b} = 0.$$

With $E_j := \mathrm{diag}_{l=0,\ldots,N-1}(e^{i\beta_j(S_j(\theta_l))}), j = 1, 2$, our discrete linearizations become

$$N\underline{a} = F\underline{\xi}_1 + FE_1\underline{U}_1$$

$$N\underline{b} = F\underline{\xi}_2 + FE_2\underline{U}_2 - F\underline{\zeta}\delta\rho.$$

Substituting these linearizations into the discrete analyticity conditions gives our linear system for $\underline{U}_1$, $\underline{U}_2$, and $\delta\rho$,

$$P_1FE_1\underline{U}_1 + P_2FE_2\underline{U}_2 + P_2F\underline{\zeta}\delta\rho = -P_1F\underline{\xi}_1 - P_2F\underline{\xi}_2 =: \underline{g}.$$

This system can be written in the form

$$D\underline{U} = \underline{g}$$

where D is the product of block matrices,

$$D = (P_1\ P_2\ \underline{w}) \begin{pmatrix} FE_1 & 0 & 0 \\ 0 & FE_2 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

and

$$\underline{U} = \begin{pmatrix} \underline{U}_1 \\ \underline{U}_2 \\ \delta\rho \end{pmatrix}$$

with $\underline{w} = P_2 F\underline{\zeta}$. This is a system of N complex equations in $2N+1$ real unknowns, $\underline{U}$. To satisfy the normalization $f(1) = \gamma_1(0)$, we set $U_0 = 0$. Then the equation count is correct.

We solve the "normal" equations

$$A\underline{U} = \frac{2}{N} Re(D^H \underline{g})$$

by the conjugate gradient method (CGNR) using FFTs, where

$$A = \frac{2}{N} Re(D^H D)\underline{U} = \begin{pmatrix} A_{11} & A_{12} & \underline{w}_1 \\ A_{12}^T & A_{22} & \underline{w}_2 \\ \underline{w}_1^T & \underline{w}_2^T & \underline{w}^T\underline{w} \end{pmatrix}$$

and

$$A_{ij} = \frac{2}{N} \mathrm{Re}(E_i^H F^H P_i P_j F E_j)$$

$$\underline{w}_i = \frac{2}{N} \mathrm{Re}(P_i \underline{w}),\ i,j = 1,2$$

Now it is easy to see that A_{11} is a (low rank perturbation of) the discretization of

$$2\mathrm{Re}(e^{-i\beta_1}(P_- + l*)e^{i\beta_1}) = I + R_1 + C_1$$

with N–point trigonometric interpolation where $R_1 = \mathrm{Re}(e^{-i\beta_1}(J - iK)e^{i\beta_1}$ is compact, $*$ is convolution, $l(\theta) = \rho^2 e^{i\theta}/(1 - \rho^2 e^{i\theta}) = \sum_{k=1}^{\infty} \rho^{2k} e^{ik\theta}$ and $C_1 = \mathrm{Re}(e^{-i\beta_1} l * (e^{i\beta_1}))$ is the product of bounded operators and convolution and is, hence, compact. A_{22} is also the discretization of an operator $I + R_2 + C_2$ of similar form and A_{12} is the discretization of an operator of the the form $C_3 + C_4$ involving products with convolutions. Therefore A is a symmetric (low rank perturbation of) the discretization of the identity plus a compact operator, the eigenvalues cluster around 1, and so the conjugate gradient method converges rapidly. The Newton update at the kth Newton step is

$$\underline{S}_1^{(k+1)} = \underline{S}_1^{(k)} + \underline{U}_1^{(k)}$$

$$\underline{S}_2^{(k+1)} = \underline{S}_2^{(k)} + \underline{U}_2^{(k)}$$

$$\rho^{(k+1)} = \rho^{(k)} + \delta\rho^{(k)}.$$

The Newton iterations converge quadratically. The method is sensitive to the initial guess like Fornberg's method for the disk.

References

[De] T. K. DeLillo, *The accuracy of numerical conformal mapping methods: a survey of examples and results,* SIAM J. Numer. Anal., 31 (1994) 788–812.

[DE1] T. K. DeLillo and A. R. Elcrat, *A comparison of some numerical conformal mapping methods for exterior regions,* SIAM J. Sci. Statis. Comput., 12 (1991) 399–422.

[DE2] T. K. DeLillo and A. R. Elcrat, *A Fornberg-like conformal mapping method for slender regions,* J. Comput. Appl. Math., **46** (1993) 49–64.

[DEP] T. K. DeLillo, A. R. Elcrat, and J. A. Pfaltzgraff, *Numerical conformal mapping methods based on Faber series,* submitted for publication.

[Fo1] B. Fornberg, *A numerical method for conformal mappings,* SIAM J. Sci. Statist. Comput., 1 (1980) 386–400.

[Fo2] B. Fornberg, *A numerical method for conformal mapping of doubly connected regions,* SIAM J. Sci. Statis. Comput., 4 (1984) 771–783.

[Ga] D. Gaier, *Konstruktive Methoden der konformen Abbildung* (Springer-Verlag, Berlin, Göttingen, Heidelberg, 1964).

[He] P. Henrici, *Applied and Computational Complex Analysis,* Vol. III, (Wiley, New York, 1986).

[LM] P. Luchini and F. Manzo, *Flow around simply and multiply connected bodies: a new iterative scheme for conformal mapping,* AIAA J., 27 (1989) 345–351.

[Weg1] R. Wegmann, *Convergence proofs and error estimates for an iterative method for conformal mapping,* Numer. Math., 44 (1984) 435–461.

[Weg2] R. Wegmann, *On Fornberg's numerical method for conformal mapping,* SIAM J. Numer. Anal., 23 (1986) 1199–1213.

[Weg3] R. Wegmann, *An iterative method for the conformal mapping of doubly connected regions,* J. Comput. Appl. Math., 14 (1986) 79–98.

[Weg4] R. Wegmann, *Discrete Riemann-Hilbert problems, interpolation of simply closed curves, and numerical conformal mapping,* J. Comput. Appl. Math., 23 (1988) 323–352.

[Weg5] R. Wegmann, *Fast conformal mapping of an ellipse to a simply connected region,* to appear in J. Comput. Appl. Math.

[Wid] O. Widlund, *On a numerical method for conformal mapping due to Fornberg,* unpublished.

A NEW ADAPTIVE GMRES ALGORITHM FOR ACHIEVING HIGH ACCURACY

MARIA SOSONKINA[†], LAYNE T. WATSON[†], HOMER F. WALKER[‡], AND RAKESH K. KAPANIA[§]

Abstract. GMRES(k) is widely used for solving nonsymmetric linear systems. However, it is inadequate either when it converges only for k close to the problem size or when numerical error in the modified Gram-Schmidt process used in the GMRES orthogonalization phase dramatically affects the algorithm performance. An adaptive version of GMRES(k) which tunes the restart value k based on criteria estimating the GMRES convergence rate for the given problem is proposed here.

The essence of the adaptive GMRES strategy is to adapt the parameter k to the problem, similar in spirit to how a variable order ODE algorithm tunes the order k. With FORTRAN 90, which provides pointers and dynamic memory management, dealing with the variable storage requirements implied by varying k is not too difficult. The parameter k can be both increased and decreased—an increase-only strategy is described next followed by pseudocode.

If k in GMRES(k) is not sufficiently large, GMRES(k) can stagnate. A test of stagnation developed in [19] detects an insufficient residual norm reduction in the restart number k of steps by estimating the GMRES behavior on a particular linear system. Precisely, GMRES(k) is declared to have stagnated and the iteration is aborted if at the rate of progress over the last restart cycle of steps, the residual norm tolerance cannot be met in some large multiple (*bgv*) of the remaining number of steps allowed (*itmax* is a bound on the number of steps permitted). Slow progress of GMRES(k) which indicates that an increase in the restart value k may be beneficial [18] can be detected with a similar test. The near-stagnation test uses a different, smaller multiple (*smv*) of the remaining allowed number of steps. If near-stagnation occurs, the restart value k is incremented by some value m and the *same* restart cycle continues. Restarting would mean repeating the nonproductive iterations that previously resulted in stagnation, at least in the case of complete stagnation (no residual reduction at all). Such incrementing is used whenever needed if the restart value k is less than some maximum value *kmax*. When the maximum value for k is reached, adaptive GMRES(k) proceeds as GMRES(*kmax*).

Frequently, a modified version of the Gram-Schmidt process is used in the construction of an orthonormal basis for the Krylov subspace. However, convergence of GMRES may be seriously affected by roundoff error, which is especially noticeable when a high accuracy solution is required. The orthogonalization phase of GMRES is suspectable to numerical instability. Thus, a more robust orthogonalization with

[†] Departments of Computer Science and Mathematics, Virginia Polytechnic Institute and State University, Blacksburg, Virginia 24061-0106. The work of these authors was supported in part by Department of Energy Grant DE-FG05-88ER25068 and Air Force Office of Scientific Research Grant F49620-92-J-0236.

[‡] Department of Mathematics and Statistics, Utah State University, Logan, UT 84322

[§] Department of Aerospace and Ocean Engineering, Virginia Polytechnic Institute and State University, Blacksburg, Virginia 24061-0203.

Householder reflections is implemented as in [20]. In theory, the implementation of GMRES using Householder reflections is about twice as expensive as when modified Gram-Schmidt is used [21]. However, the Householder reflection method produces a more accurate orthogonalization of the Krylov subspace basis when the basis vectors are nearly linearly dependent and the modified Gram-Schmidt method fails to orthogonalize the basis vectors; this can result in fewer GMRES iterations compensating for the higher cost per iteration using Householder reflections. Let e_j be the jth standard basis vector, *avnz* be the average number of nonzeros per row, and all norms the 2-norm. Pseudocode for an adaptive version of GMRES(k) with orthogonalization via Householder reflections implemented as in [20] and [21] follows. Call this algorithm AGMRES(k).

choose $x, itmax, kmax, m$;
$r := b - Ax$; $itno := 0$; $cnmax := 1/(50\mathbf{u})$;
$xtol := \max\{100.0, 1.01avnz\}\mathbf{u}$; $tol := \max\{\|r\|, \|b\|\}xtol$;
while $\|r\| > tol$ **do**
begin
$r^{old} := r$;
determine $P_1 r = \pm\|r\|e_1$
where the Householder transformation matrix P_1 is defined in [8];
for $j := 1$ **step** 1 **until** k **do**
begin
L1: $itno := itno + 1$;
$v := P_j \cdots P_1 A P_1 \cdots P_j e_j$;
determine P_{j+1} such that $P_{j+1}v$ has zero components
after the $(j+1)$st;
update $\|r\|$ as described in [16];
compute incremental condition number ICN as in [1];
if $ICN > cnmax$ **then** abort;
if $\|r\| \leqq tol$ **then goto** L2
end
$test := k \times \log[tol/\|r\|] \,/\, \log\left[\|r\|/\left((1.0 + 10\mathbf{u})\,\|r^{old}\|\right)\right]$;
if $k \leqq kmax - m$ **and** $test \geqq smv \times (itmax - itno)$ **then**
$k := k + m$;
goto L1
end if
L2: $e_1 := (1, 0, \cdots, 0)^T$;
solve $\min_y \left\| \|r\|e_1 - \bar{H}_j y \right\|$ for y_j where $\bar{H}_j$ is described in [16];

```
    q := (y_j, 0);
    x := x + P_1 ··· P_j q;        r := b − Ax;
    if ||r|| ≦ tol then exit;
    if ||r^old|| < ||r|| then
        if ||r|| < tol^{2/3} then
            exit
        else
            abort
        end if
    end if
    test := k × log[tol/||r||] / log[||r||/((1.0 + 10u) ||r^old||)];
    if test ≧ bgv × (itmax − itno) then
        abort
    end if
end
```

High accuracy computation is known to be especially important for some circuit design and simulation problems, which involve nonsymmetric unstructured matrices [14]. These problems have proven to be very difficult for any iterative method, and presently, only specially tailored direct methods are used to solve them in a production environment. The impact of numerical error is shown here to be significant in the postbuckling stability analysis of structures which exhibit snap-back and snap-through phenomena. A postbuckling analysis amounts to tracking what is called an equilibrium curve, involving the solution of linear systems with (tangent stiffness) matrices that vary along the equilibrium curve. Mathematically, the snap-back and snap-through behavior of structures corresponds to symmetric, structured, and strongly indefinite tangent stiffness matrices, which can be difficult for Krylov subspace iterative methods. Along some portions of the curve the tangent stiffness matrices may be well conditioned and positive definite—easy for iterative methods. This wide variation in linear system difficulty clearly suggests an adaptive strategy. With good preconditioning, the details of an iterative solver become less important, and for many real problems good preconditioning strategies (e.g., multigrid) are available. For some classes of problems, like analog DC circuit design and structural postbuckling stability analysis, good preconditioning strategies are not known, and subtle implementation details of iterative algorithms become paramount.

Tables 1–6 show the iteration counts and timing results for adaptive GMRES(k) with modified Gram-Schmidt and Householder reflection orthogonalization procedures (AGS, AGH respectively) and different subspace increment values m, standard GMRES(k) (GS), flexible GMRES(k) (FGS), and QMR (Q). The notation AGH:2, for example, means AGH with $m = 2$. Each of the iterative methods was applied to all of the test problems. An iterative solver is deemed to have converged when the relative

residual norm reduction is less than $\max\{100, avnz\}\mathbf{u}$. The limit on the number of iterations is $30n$. All reported CPU times are for 64-bit IEEE arithmetic on a DEC Alpha 3000/600 workstation. The average, maximum, and minimum number of iterations per linear system solution along the homotopy zero curve are shown in Tables 1–3. Note that in the FGS column the numbers for the x_k *outer* iterations of flexible GMRES(k) are given. Tables 5, 6 report the total CPU time in seconds required for an iterative method to obtain all the linear system solutions needed for HOMPACK [24] to track a homotopy zero curve for a certain arc length specified for each problem. For the circuit problems (Table 4), the reported CPU time is the time necessary to solve a certain number of linear systems arising along a portion of the homotopy zero curve.

TABLE 1

Average, maximum, and minimum number of iterative solver iterations per linear system along homotopy zero curve for circuit design problems (ILU preconditioner).

n	AGH:2	AGH:4	AGH:6	AGS:2	AGS:4	AGS:6
31	23,112,1 (2 → 6)	19,108,1 (2 → 6)	19,108,1 (2 → 6)	27,1,192 (2 → 6)	26,165,1 (2 → 6)	26,165,1 (2 → 6)
59	50,231,14 (4 → 6)	46,231,14 (4 → 8)	46,231,14 (4 → 10)	53,231,14 (4 → 6)	53,231,14 (4 → 8)	*
67	601,1225,322 (5 → 15)	622,1650,333 (5 → 15)	506,948,285 (5 → 15)	553,1117,315 (5 → 15)	493,892,299 (5 → 15)	530,1057,340 (5 → 15)
125	143,194,84 (4 → 6)	121,168,84 (4 → 8)	103,131,82 (4 → 10)	136,221,66 (4 → 6)	135,202,66 (4 → 10)	134,192,66 (4 → 12)
468	585,1650,252 (15 → 15)	585,1650,252 (15 → 15)	585,1650,252 (15 → 15)	603,1740,252 (15 → 15)	603,1740,252 (15 → 15)	603,1740,252 (15 → 15)
1854	2576,3739,1177 (35 → 48)	2591,3955,956 (35 → 47)	2660,3815,873 (35 → 47)	2710,4165,1347 (35 → 47)	2725,3990,1096 (35 → 47)	2523,3947,1060 (35 → 47)

n	GS	FGS	Q
31	(2) * (6)12,48,1	(2) * (3)8,196,1	*
59	(4) * (10)9,10,7	*	*
67	(5) * (15)288,359,217	(3) * (8)20,107,9	*
125	(4) * (12)68,90,36	(4) * (6)62,142,9	*
468	(15)603,1740,252	*	*
1854	(35) * (47)611,838,539	*	*

An asterisk denotes convergence failure, meaning any of the following occurred: (1) desired error tolerance not met after $30n$ iterations, (2) dangerous near singularity detected for the GMRES least squares problem, (3) residual norm increased between restart cycles and was not small, (4) stagnation occurred for GMRES-like methods. In Tables 1–3, the numbers in parentheses (k) show the restart values k whenever applicable. For AGS and AGH, the notation ($a \rightarrow b$) shows the initial restart value a and the maximum restart value b reached by the adaptive strategy anywhere along the homotopy zero curve.

TABLE 2

Average, maximum, and minimum number of iterative solver iterations per linear system along homotopy curve for thin shell problem (Gill-Murray preconditioner).

n	AGH:2	AGH:4	AGH:6	AGS:2	AGS:4	AGS:6	GS	FGS	Q
55	80,1239,1 (8 → 20)	57,888,1 (8 → 20)	44,748,1 (8 → 24)	*	*	*	*	*	*
119	6,134,1 (5 → 5)	6,134,1 (5 → 5)	6,134,1 (5 → 5)	6,140,1 (5 → 5)	6,140,1 (5 → 5)	6,140,1 (5 → 5)	6,140,1 (5)	*	*
1239	3,14,1 (12 → 20)	3,14,1 (12 → 20)	3,14,1 (12 → 20)	*	*	*	*	*	*

Table 3

Average, maximum, and minimum number of iterative solver iterations per linear system along homotopy zero curve for lamella dome problem (Gill-Murray preconditioner).

n	AGH:2	AGH:4	AGH:6	AGS:2	AGS:4	AGS:6	GS	FGS	Q
21	22,452,1 (4 → 10)	19,536,1 (4 → 12)	16,528,1 (4 → 10)	23,508,1 (4 → 10)	18,387,1 (4 → 12)	*	*	*	*
69	38,1374,1 (8 → 18)	34,1347,1 (8 → 18)	35,1576,1 (8 → 18)	40,1331,1 (8 → 18)	39,1442,1 (8 → 18)	*	*	(4) * (9)3,75,1	*

TABLE 4

Iterative solver execution time in seconds for circuit problems (ILU preconditioner).

n	AGH:2	AGH:4	AGH:6	AGS:2	AGS:4	AGS:6	GS	FGS
31	0.67	0.55	0.56	0.43	0.41	0.41	0.18	1.53
59	0.56	0.52	0.51	0.31	0.30	*	0.08	*
67	4.58	4.84	3.87	1.94	1.76	1.76	0.53	11.04
125	1.83	1.70	1.53	0.96	0.86	0.73	0.71	14.56
468	15.03	15.04	15.05	5.33	5.32	5.33	5.32	*
1854	514.62	515.12	530.69	135.37	136.25	125.99	32.98	*

TABLE 5

Iterative solver execution time in seconds for thin shell problem (Gill-Murray preconditioner).

n	AGH:2	AGH:4	AGH:6	AGS:2	AGS:4	AGS:6	GS
55	25.82	19.68	14.53	*	*	*	*
119	2.70	2.71	2.69	2.43	2.42	2.41	2.44
1239	111.40	107.12	100.90	*	*	*	*

TABLE 6

Iterative solver execution time in seconds for lamella dome problem (Gill-Murray preconditioner).

n	AGH:2	AGH:4	AGH:6	AGS:2	AGS:4	AGS:6	FGS
21	2.24	1.89	1.66	1.74	1.39	*	*
69	29.45	25.93	27.27	24.26	23.64	*	38.86

For applications requiring high accuracy linear system solutions, the adaptive GMRES(k) algorithm proposed here, which is based on Householder transformations and monitoring the GMRES convergence rate, outperforms several standard GMRES variants and QMR. The superiority is not in CPU time or memory requirements, but in robustness as a linear system solution "server" for a much larger nonlinear analysis

"client" computation. The extra cost of Householder transformations is well justified by the improved reliability, and the adaptive strategy is well suited to the need to solve a sequence of linear systems of widely varying difficulty. In the context of large scale, multidisciplinary, nonlinear analysis it is hard to imagine not wanting the various components to be adaptive.

The high accuracy requirement causes types of failures in GMRES not usually seen in moderate accuracy uses of GMRES, and monitoring condition number estimates and the onset of stagnation within GMRES becomes crucial. Furthermore, "sanity" checks (e.g., residual norm should be nonincreasing) within a GMRES algorithm must be modified for high accuracy. For instance, the limit on the number of iterations must be increased, and depending on other factors, an increasing residual norm may or may not dictate an abort (such details are in the pseudocode above). Finally, the three test problems are rather different, so the details of AGMRES(k) have not been tuned to one particular class of problems.

Future work should investigate a strategy for both increasing and decreasing the Krylov subspace dimension depending on measures of progress. Ideally the accuracy requested of the linear solver should match the accuracy actually required by the calling nonlinear analysis program, although in large scale computation it may be extremely difficult to estimate this minimum required accuracy.

REFERENCES

[1] C. H. Bischof and P. T. P. Tang, *Robust incremental condition estimation*, Tech. Rep. CS-91-133, LAPACK Working Note 33, Computer Sci. Dept., Univ. of Tennessee, May, 1991.

[2] A. Björck, *Solving linear least squares problems by Gram-Schmidt orthogonalization*, BIT, 7 (1967), pp. 257–278.

[3] P. N. Brown and H. F. Walker, *GMRES on (nearly) singular systems*, SIAM J. Matrix Anal. Appl., submitted.

[4] C. DeSa, K. M. Irani, C. J. Ribbens, L. T. Watson, and H. F. Walker, *Preconditioned iterative methods for homotopy curve tracking*, SIAM J. Sci. Stat. Comput., 13 (1992), pp. 30–46.

[5] R. W. Freund and N. M. Nachtigal, *An implementation of the look-ahead lanczos algorithm for non-Hermitian matrices, part II*, Tech. Rep. 90.46, RIACS, NASA Ames Research Center, November, 1990.

[6] R. W. Freund and N. M. Nachtigal, *QMR: a quasi-minimal residual method for non-Hermitian linear systems*, Numer. Math., 60 (1991), pp. 315–339.

[7] P. E. Gill and W. Murray, *Newton-type methods for unconstrained and linearly constrained optimization*, Math. Programming, 28 (1974), pp. 311–350.

[8] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, MD, 2nd ed., 1989.

[9] S. M. Holzer, R. H. Plaut, A. E. Somers, Jr., and W. S. White, *Stability of lattice structures under combined loads*, J. Engrg. Mech., 106 (1980), pp. 289–305.

[10] K. M. Irani, M. P. Kamat, C. J. Ribbens, H. F. Walker, and L. T. Watson, *Experiments with conjugate gradient algorithms for homotopy curve tracking*, SIAM J. Optim., 1(2) (1991), pp. 222–251.

[11] R. K. Kapania and T. Y. Yang, *Formulation of an imperfect quadrilateral doubly curved shell element for postbuckling analysis*, AIAA J., 24 (1986), pp. 310–311.

[12] W. D. McQuain, R. C. Melville, C. J. Ribbens, and L. T. Watson, *Preconditioned iterative methods for sparse linear algebra problems arising in circuit simulation*, Comput. Math. Appl., 27 (1994), pp. 25–45.

[13] R. C. MELVILLE, S. MOINIAN, P. FELDMANN, AND L. T. WATSON, *Sframe: an efficient system for detailed DC simulation of bipolar analog integrated circuits using continuation methods*, Analog Integrated Circuits Signal Processing, 3 (1993), pp. 163–180.

[14] R. C. MELVILLE, LJ. TRAJKOVIĆ, S.-C. FANG, AND L. T. WATSON, *Artificial parameter homotopy methods for the DC operating point problem*, IEEE Trans. Computer-Aided Design, 12 (1993), pp. 861–877.

[15] Y. SAAD, *A flexible inner-outer preconditioned algorithm*, SIAM J. Sci. Comput., 14 (1993), pp. 461–469.

[16] Y. SAAD AND M. H. SCHULTZ, *GMRES: a generalized minimal residual method for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856 – 869.

[17] H. A. VAN DER VORST AND C. VUIK, *GMRESR: a family of nested GMRES methods*, Numer. Linear Algebra Appl., 1 (1994), pp. 369–386.

[18] H. A. VAN DER VORST AND C. VUIK, *The superlinear convergence behaviour of GMRES*, J. Comp. Appl. Math., 48 (1993) pp. 327–341.

[19] H. F. WALKER, *GMRES on (nearly) singular systems*, Tech. Rep. 2/94/73, Math. Stat. Dept., Utah State Univ., 1994.

[20] H. F. WALKER, *Implementation of the GMRES method using Householder Transformations*, SIAM J. Sci. Stat. Comput., 9 (1988), pp. 152–163.

[21] H. F. WALKER, private communication, 1996.

[22] L. T. WATSON, *Globally convergent homotopy methods: a tutorial*, Appl. Math. Comput., 31BK (1989), pp. 369–396.

[23] L. T. WATSON, S. M. HOLZER, M. C. HANSEN, *Tracking nonlinear equilibrium paths by a homotopy method*, Nonlinear Anal., 7 (1983), pp. 1271–1282.

[24] L. T. WATSON, S. C. BILLUPS, AND A. P. MORGAN, *Algorithm 652: HOMPACK: A suite of codes for globally convergent homotopy algorithms*, ACM Trans. Math. Software., 13 (1987), pp. 281–310.

[25] T. Y. YANG, R. K. KAPANIA, AND S. SAIGAL, *Accurate rigid-body modes representation for a nonlinear curved thin-shell element*, AIAA J., 27 (1989), pp. 211–218.

Eric de Sturler
Interdisciplinary Project Center for Supercomputing (IPS-ETH Zurich)

Swiss Federal Institute of Technology
Clausiusstrasse 59, RZ F-11
ETH-Zentrum, CH-8092 Zurich, Switzerland

phone : +41-1-632 5566
fax : +41-1-632 1104
email : sturler@ips.id.ethz.ch

Truncation Strategies for (Nested) Krylov Methods

Eric de Sturler
<sturler@scsc.ethz.ch>

Swiss Center for Scientific Computing (SCSC-ETHZ)
Swiss Federal Institute of Technology Zurich
ETH Zentrum, RZ F-11
CH-8092 Zurich, Switzerland

optimal methods based on the Arnoldi iteration, if a large number
iterations is necessary, the storage requirements become excessive
. the iterations too expensive. Therefore, in practice one restarts
er a given number of iterations, m, (like GMRES(m)), or one truncates
set of Arnoldi vectors, typically by keeping only the last m vectors
ke GCR(m)).

ever, this often leads to a loss of superlinear convergence or
n to stagnation of the convergence. Nested iterative methods
and/or more general truncation schemes offer the possibility of
r optimal convergence with reasonable memory requirements and low
ration cost, if the right vectors to keep for orthogonalization in
ure iterations can be selected.
ently proposed strategies seem to focus on removing certain parts of the
ctrum. However, these strategies generally assume that the spectrum is in
positive real half plane, and are therefore not generally applicable.
eover, recent papers by Z. Strakos and A. Greembaum show that even
a favourable spectrum the convergence of full GMRES (which is optimal) can
ll be arbitrarily bad (stagnating) depending on the eigenvectors of the
rix; i.e. the non-normality plays a large role as well.

refore, we propose a different strategy.
compute the singular value decomposition of a small matrix (Z)
structed from the iteration parameters, that is, from the
senberg matrix in GMRES(m) or from the product of the matrices
t describe the projections in the nested method. This singular
ue decomposition reveals exactly what role the orthogonalization on
h arbitrary subspace of the space spanned by the Arnoldi vectors has
yed in the convergence so far. Where we use the term Arnoldi vectors
indicate both the search vectors generated by GMRES or GCR and the
rch vectors in the outer iteration if we use a nested method.
be more precise, if we consider one or more vectors defined as the
ducts of the matrix with the Arnoldi vectors as its columns and the
t singular vectors of Z, then a function of the corresponding singular
ues determines how much worse the convergence would have been had this

vector or the space spanned by these vectors not been projected out in the previous iterations. This can be generalized to arbitrary subspaces. So this function of the singular values defines the deterioration from the optimal convergence. This information can be used to select the dimension and the basis of the space to be projected out (orthogonalized upon) in future iterations.
Preliminary numerical tests indicate that this approach leads to a very effect truncation strategy.

[1] E. de Sturler, "Nested Krylov methods based on GCR",
Technical Report 93-50, Faculty of Technical Mathematics and Informatics, Delft University of Technology, Delft, The Netherlands, 1993.
(accepted Journal of Comp. and Appl. Mathematics, North -Holland)

HYBRID LANCZOS-TYPE PRODUCT METHODS

KLAUS J. RESSEL*

Abstract. A general framework is proposed to construct hybrid iterative methods for the solution of large nonsymmetric systems of linear equations. This framework is based on Lanczos-type product methods, whose iteration polynomial consists of the Lanczos polynomial multiplied by some other arbitrary "shadow" polynomial. By using for the shadow polynomial Chebyshev (more general Faber) polynomials or L^2-optimal polynomials, hybrid (Chebyshev-like) methods are incorporated into Lanczos-type product methods. In addition, to acquire spectral information on the system matrix, which is required for such a choice of shadow polynomials, the Lanczos-process can be employed either directly or in an QMR-like approach. The QMR like approach allows the cheap computation of the roots of the B-orthogonal polynomials and the residual polynomials associated with the QMR iteration. These roots can be used as a good approximation for the spectrum of the system matrix. Different choices for the shadow polynomials and their construction are analyzed. The resulting hybrid methods are compared with standard Lanczos-type product methods, like BiOStab, BiOStab(ℓ) and BiOS.

Key words. iterative methods, Lanczos method, product methods, hybrid, sparse linear systems

AMS subject classifications. 65F15, 65F10

1. Introduction. Polynomial iterative methods for the solution of large systems of linear equations $Ax = b$ with nonsymmetric sparse system matrix $A \in \mathbb{C}^{N\times N}$ can be divided into two classes. *Hybrid (Chebyshev-like) methods* acquire in a first phase some spectral information of the system matrix on which then the iteration polynomial is constructed. In a second phase the coefficients of the constructed iteration polynomial are used in a Chebyshev or Richardson iteration. In practice, after performing some iteration steps in the second phase, hybrid methods adaptively loop back to the first phase to update the spectral information and to ensure an adequate convergence rate. Hybrid methods do not require the computation of inner products, which is an advantage for an implementation on vector or parallel computers. Moreover, they are constructed in order to converge independently of the initial guess and the initial residual, so that the spectral information of a first run can be reused in problems with multiple right hand sides of for the solution of successive problems, where the system matrix differs only slightly, as in the solution of nonlinear systems or time-dependent physical models.

On the other hand, *CG-like methods* construct the iteration polynomial during the iteration process based on some orthogonality relations. They do not require a priori information about the system matrix, but they involve the computation of inner products and are susceptible to breakdowns or stagnation.

In the following work we combine these two classes of polynomial iterative methods and analyze how large the synergetic effect is. This combination can also be simply viewed as a polynomial preconditioning of a CG-like method.

To motivate this approach, we list now briefly the milestones in the history of polynomial iterative methods that are of relevance here. The history of hybrid iterative methods goes back at least to the adaptive Chebyshev algorithm of Manteuffel [13], [14], which is based on a modified power iteration in the first phase, followed by a Chebyshev iteration with respect to an ellipse enclosing the approximation of the spectrum. The use of scaled and shifted Chebyshev polynomials, however, restricts

* Swiss Center for Scientific Computing (SCSC), ETH Zürich, ETH-Zentrum, CH-8092 Zürich, Switzerland (kjr@scsc.ethz.ch).

this approach to problems, where the eigenvalues are located in an ellipse not containing the origin. Elman, Saad and Saylor [4] replaced the modified power iteration by an Arnoldi process in phase one. Later, Saylor and Smolarski [20], proposed a hybrid method which also works in situations, where the eigenvalues cannot be enclosed by an ellipse. They construct an L^2-optimal polynomial on a polygon representing the boundary of a region containing the approximation of the spectrum. Moreover, they use in phase one an Arnoldi method and construct already in Phase one an approximate solution with GMRES. In phase two a Richardson iteration is applied. To avoid problems of numerical instability, a variant of this algorithm, based on modified moments, was suggested by Saad [18]. The hybrid GMRES method proposed by Nachtigal, Reichel and Trefethen [17] does not rely on eigenvalue estimates, since the iteration polynomial is implicitly constructed in phase one by the GMRES algorithm. Finally, the hybrid Arnoldi-Faber methods of Starke and Varga [23] and of Manteuffel and Starke [16] use a Richardson iteration in phase two based on Faber polynomials constructed with respect to a polygonal domain in phase one.

The origin of CG-like methods goes back to the conjugate gradient method (CG) due to Hestenes and Stiefel [11] for Hermitian positive definite problems. A straightforward extension to non-Hermitian problems is to apply CG to the normal equations. However, this means also squaring the condition number of the problem. Therefore, a considerable part of research has been devoted to generalizations of CG to non-Hermitian problems. For an excellent overview and a classification we refer the reader to [1]. The most widely used methods of these types is the generalized minimum residual method (GMRES) of Saad and Schultz [19]. In general, these generalizations of CG cannot be based on a short-term recurrence (for necessary and sufficient conditions see [5]), so that their work and storage requirements grow linearly with the iteration number. On the contrary, methods based on the nonsymmetric Lanczos biorthogonalization process [12] can be implemented with low and roughly constant work and storage requirements per iteration. However, these methods lack a minimization property over the generated Krylov space and typically have therefore, a rather irregular convergence behavior with wild oscillations in the residual norm. The QMR method, development by Freund and Nachtigal [7], is based on the Lanczos process and overcomes partly these problems by computing the iterates via a quasi-minimal residual property. Later, Barth and Manteuffel [2], [3] showed that QMR is actually minimizing the residual with respect to a norm which depends on the initial residual. Furthermore, they generalize the approach in [15] to obtain the roots of the B-orthogonal and the residual polynomials by the QMR method. These roots can then be used as a good approximation of the spectrum of the system matrix. An improvement of the Lanczos process in another direction is the development of Lanczos-type product methods (LTPMs), like CGS [22], BiCGStab [25], BiCGStab2 [9], and BiCGStab(ℓ) [21]. These methods either square the Lanczos process or combine it with a local minimization of the residual. They have the advantage of converging roughly twice as fast as the plain Lanczos method and further they do not require a routine for applying the adjoint system matrix A^H to a vector.

In the following work we combine many of these ideas by incorporating hybrid methods into LTPMs.

2. Lanczos-type Product Methods (LTPMs).

2.1. The Lanczos Biorthogonalization (BiO) process. We recall that the BiO algorithm generates a pair of finite sequences, $\{\widetilde{y}_n\}_{n=0}^{\nu}$, $\{y_n\}_{n=0}^{\nu}$, of *left* and *right*

Lanczos vectors, such that

$$
\begin{aligned}
y_n \in \mathcal{K}_{n+1} &:= \text{span}\,\{y_0, Ay_0, \ldots, A^n y_0\}, \\
\widetilde{y}_n \in \mathcal{L}_{n+1} &:= \text{span}\,\{\widetilde{y}_0, A^H\widetilde{y}_0, \ldots, (A^H)^n\widetilde{y}_0\},
\end{aligned} \tag{2.1}
$$

and

$$
\widetilde{y}_n \perp \mathcal{K}_n, \quad y_n \perp \mathcal{L}_n. \tag{2.2}
$$

This sequence of pairs of Lanczos vectors can be constructed by the following three-term recursion

$$
\begin{aligned}
y_{n+1} &= \left(Ay_n - y_n\alpha_n - y_{n-1}\beta_n\right)/\gamma_n \\
\widetilde{y}_{n+1} &= \left(A^H\widetilde{y}_n - \widetilde{y}_n\overline{\alpha}_n - \widetilde{y}_{n-1}\overline{\beta}_n\right)/\overline{\gamma}_n,
\end{aligned} \tag{2.3}
$$

with coefficients α_n and β_n that are determined from the orthogonality condition (2.2) and freely choosable nonvanishing scale factors γ_n. The Lanczos vectors can then be written in the form

$$
y_n = \rho_n(A)y_0, \qquad \widetilde{y}_n = \overline{\rho}_n(A^H)\widetilde{y}_0, \tag{2.4}
$$

where ρ_n denotes the n-th Lanczos polynomial. Since we aim here at LTPMs, we consider for the Krylov spaces $\mathcal{L}_n$ more general basis vectors of the form

$$
\widetilde{z}_n = \overline{\tau}_n(A^H)\widetilde{z}_0, \tag{2.5}
$$

with arbitrarily chosen polynomials τ_n of exact degree n and $\widetilde{z}_0 := \widetilde{y}_0$. In general, $\widetilde{z}_n \perp \mathcal{K}_n$ will no longer hold, but $y_n \perp \mathcal{L}_n$ can be still obtained by choosing the coefficients α_n and β_n in (2.3) in the following way:

$$
\beta_n = \frac{\langle \widetilde{z}_{n-1}, Ay_n\rangle}{\langle \widetilde{z}_{n-1}, y_{n-1}\rangle}, \qquad \alpha_n = \frac{\langle \widetilde{z}_n, Ay_n\rangle - \langle \widetilde{z}_n, y_{n-1}\rangle\,\beta_n}{\langle \widetilde{z}_n, y_n\rangle}. \tag{2.6}
$$

Clearly, the recursive process terminates with $y_\nu = 0$ or $\widetilde{z}_\nu = 0$, or it breaks down with $\delta_\nu^\tau := \langle \widetilde{z}_\nu, y_\nu\rangle = 0$ and $y_\nu \neq 0$, $\widetilde{z}_\nu \neq 0$. The look-ahead Lanczos process [6] overcomes a breakdown if curable. To simplify our presentation, we assume in the following that breakdown appears in the underlying Lanczos process. For a description of look-ahead procedures for Lanczos-type product methods we refer the reader to [10].

2.2. Standard Lanczos-type product methods. By introducing the *product vectors*

$$
w_n^l := \tau_l(A)y_n := \tau_l(A)\rho_n(A)y_0 \tag{2.7}
$$

and by rewriting the inner products in (2.6) in terms of product vectors, i.e.,

$$
\beta_n = \frac{\langle \widetilde{z}_0, Aw_n^{n-1}\rangle}{\langle \widetilde{z}_0, w_{n-1}^{n-1}\rangle}, \qquad \alpha_n = \frac{\langle \widetilde{z}_0, Aw_n^n\rangle - \langle \widetilde{z}_0, w_{n-1}^n\rangle\,\beta_n}{\langle \widetilde{z}_0, w_n^n\rangle}, \tag{2.8}
$$

LTPMs are derived. Different choices of the polynomials $\tau_n(\zeta)$ lead to different LTPMs. In BiOStab, the three-term version of Van der Vorst's BiCGStab [25], the polynomials $\tau_n(\zeta)$ are chosen as a product of polynomials of degree 1:

$$
\tau_0(\zeta) \equiv 1, \qquad \tau_{n+1}(\zeta) = (1 - \chi_n\zeta)\tau_n(\zeta) \quad \text{for } n \geq 0, \tag{2.9}
$$

where χ_n is defined by minimizing the norm of $w_{n+1}^{n+1} = w_{n+1}^n - \chi_n A w_{n+1}^n$. The choice

$$
\begin{array}{lcll}
\tau_0(\zeta) & \equiv & 1, & \\
\tau_{n+1}(\zeta) & = & (1-\chi_n\zeta)\,\tau_n(\zeta) & \text{if } n \text{ is even,} \\
\tau_{n+1}(\zeta) & = & (\xi_n+\eta_n\zeta)\,\tau_n(\zeta) + (1-\xi_n)\,\tau_{n-1}(\zeta) & \text{if } n \text{ is odd,}
\end{array} \tag{2.10}
$$

where χ_n is defined as above, whereas ξ_n and η_n are obtained by minimizing the norm of $w_{n+1}^{n-1} + \left(w_{n+1}^n - w_{n+1}^{n-1}\right)\xi_n + A w_{n+1}^n \eta_n$ leads to BiOStab2, the three-term version of Gutknecht's BiCGStab2 [9]. Generating the polynomials $\tau_n(\zeta)$ also for n even by the three-term recurrence

$$\tau_{n+1}(\zeta) = (\xi_n + \eta_n\zeta)\,\tau_n(\zeta) + (1-\xi_n)\,\tau_{n-1}(\zeta), \tag{2.11}$$

with the above definition of ξ_n and η_n, results in BiOxMR2, where the Lanczos process is combined with a local two dimensional minimization in every step. In BiOS, the three-term version of Sonneveld's (Bi)CGS, the Lanczos polynomial is squared by the choice $\tau_n(\zeta) = \rho_n(\zeta)$.

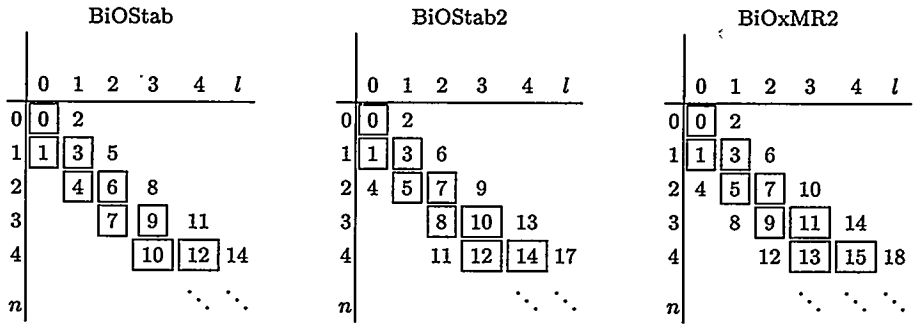


FIG. 2.1. *Movement of BiOStab, BiOStab2 and BiOxMR2 in the w-table. The numbers illustrate in which sequence the entries are computed. A framed entry indicates that the corresponding entry as well as its product with the system matrix A is computed.*

The aim of all introduced algorithms in every iteration step is to obtain an improved new approximation x_{n+1}^{n+1} by computing a new product vector w_{n+1}^{n+1} from previously computed product vectors in a stable way. To visualize the different strategies used by the algorithms, we arrange the product vectors w_n^l in a *w-table*, where the n-axis of the table points downwards and the l-axis to the right. We describe then how every algorithm moves in this table from the upper left corner downwards right. By multiplying the three-term recursion (2.3) for the right Lanczos vectors y_{n+1} with $\tau_l(A)$ we obtain for the product vectors the recursion

$$w_{n+1}^l = \left(A w_n^l - w_n^l \alpha_n - w_{n-1}^l \beta_n\right)/\gamma_n, \tag{2.12}$$

which can be applied to move forward in vertical direction in the w-table. To obtain a recursion formula for moving forward in horizontal direction, the recursion formula for the chosen polynomials $\tau_n(\zeta)$ is capitalized upon. For example, by multiplying y_n with $\tau_{l+1}(A)$ and applying then (2.11) leads to the recurrence

$$w_n^{l+1} = \eta_l A w_n^l + \xi_l w_n^l + (1-\xi_l) w_n^{l-1}. \tag{2.13}$$

Figure 2.1 illustrates the movement of different LTPMs in the w-table.

2.3. Hybrid Lanczos-type product methods. In the standard LTPMs a problem arises, if $|\chi_n|$ and $|\eta_n|$ respectively, is small, since we do not advance then in the Krylov space. For this reason BiCGStab(ℓ) was proposed [21], where the power basis is used to extend the Krylov space for $\ell > 2$ steps and then to solve an ℓ dimensional minimization problem for the residual. Of course, for large ℓ the use of the power basis is not very stable and in addition the solution of the ℓ dimensional minimization problem becomes expensive. This leads to the idea, first indicated in [9], to use as polynomial τ_n an iteration polynomial computed in phase one of an hybrid method. Recall that the Richardson iteration $x_{n+1} = x_n + \chi_n r_n$ results in the two-term recurrence $r_{n+1} = r_n - \chi_n A r_n$ for the residual, so that the iteration polynomial can be generated by the recurrence (2.9), whereas the iteration polynomial for the Chebyshev iteration [13] is given by a recurrence (2.11). Therefore, we can incorporate phase two of an hybrid method directly into an LTPM.

3. Obtaining the solution of $\mathbf{Ax = b}$. So far we have only introduced different algorithms to construct a sequence of product vectors w_n^l, which can be used to form a basis for $\mathcal{K}_m$. However, the overall goal is to solve the linear system $Ax = b$ and we have not yet described how to obtain such a solution. The following section is devoted to this issue. We first describe a direct approach and discuss later how a solution can also be constructed in a quasi-minimal residual (QMR) like way.

3.1. Direct Approach. The following direct approach is based on a generalization of Gutknecht's unnormalized BiORes algorithm in [8]. Let the doubly indexed sequence of scalars $\dot{\rho}_n^l$ be given by

$$\dot{\rho}_n^l := \tau_l(0)\rho_n(0). \tag{3.1}$$

Then we define a doubly indexed sequence of *product iterates* x_n^l in the following way. Starting with an arbitrary initial product iterate $x_0^0 \in \mathbb{C}^N$, we assume that the initial product vector is chosen so that

$$w_0^0 = b\dot{\rho}_0^0 - Ax_0^0 = b - Ax_0^0. \tag{3.2}$$

Here we used that $\dot{\rho}_0^0 = 1$, since $\tau_0(\zeta) = \rho_0(\zeta) \equiv 1$. For $l, n > 0$ we define the product iterates by

$$b\dot{\rho}_n^l - Ax_n^l := w_n^l \qquad \Longleftrightarrow \qquad b - A\frac{x_n^l}{\dot{\rho}_n^l} = \frac{w_n^l}{\dot{\rho}_n^l}. \tag{3.3}$$

It follows that $\frac{x_n^l}{\dot{\rho}_n^l}$ is an approximation for the solution of $Ax = b$ and that the corresponding residual is given by $\frac{w_n^l}{\dot{\rho}_n^l}$. It remains to derive recursions for the scalars $\dot{\rho}_n^l$ and the product iterates x_n^l, which can be easily done by using relation (3.3) and the recurrences for the product vectors w_n^l.

3.2. QMR Like Approach. The computed product vectors w_n^l can also be used to form a basis of the Krylov space $\mathcal{K}_m$. Since in any LTPM the product vectors located on the diagonal staircase of the w-table are always computed (see Figure 2.1), it is obvious to use these vectors to form a basis. Thus, if we let

$$\begin{aligned} W_{2n} &:= [w_0^0\ w_1^0\ w_1^1\ \cdots\ w_n^{n-1}] \\ W_{2n+1} &:= [w_0^0\ w_1^0\ w_1^1\ \cdots\ w_n^{n-1}\ w_n^n] \end{aligned} \tag{3.4}$$

it follows that W_m is a basis for $\mathcal{K}_m$.

From the Lanczos recurrence (2.12) and the recursion (2.13) for the product vectors we obtain immediately that

$$A\left(w_n^n - \beta_n \eta_{n-1} w_{n-1}^{n-1}\right) = \gamma_n w_{n+1}^n + \alpha_n w_n^n + \beta_n \xi_{n-1} w_{n-1}^{n-1} + \beta_n (1-\xi_{n-1}) w_{n-1}^{n-2}$$

$$A\left(w_{n+1}^n - \frac{(1-\xi_n)}{\eta_n \gamma_n} w_n^{n-1}\right) = \frac{1}{\eta_n} w_{n+1}^{n+1} - \frac{\xi_n}{\eta_n} w_{n+1}^n + \frac{\alpha_n (1-\xi_n)}{\eta_n \gamma_n} w_n^{n-1} + \frac{\beta_n (1-\xi_n)}{\eta_n \gamma_n} w_{n-1}^{n-1}. \tag{3.5}$$

Writing (3.5) in compact form becomes

$$AW_m G_m = W_{m+1} \widetilde{H}_{m=1,m} \iff AW_m = W_{m+1} H_{m+1,m}, \tag{3.6}$$

with an upper tridiagonal matrix G_m, and with Hessenberg matrices $\widetilde{H}_{m+1,m}$ and $H_{m+1,m} := \widetilde{H}_{m+1,m} G_m^{-1}$. By means of (3.6) we can construct QMR iterates in the following way: Starting with $w_0^0 := b - Ax_0$ for an arbitrary initial guess x_0, we define $x_m := x_0 + W_m s_m$ where $s_m \in \mathbb{C}^m$ is the solution of

$$\min_{s \in \mathbb{C}^m} \|e_1 - H_{m+1,m} s\|_2, \tag{3.7}$$

with $e_1 := [1, 0, \ldots, 0]_{m+1}^T$. Note that the iterates x_m can be computed by a short-term recurrence without the necessity of storing all old product vectors in W_m, since the Hessenberg matrix $H_{m+1,m}$ is not full.

4. Acquiring Spectral Information. Although the left and the right Lanczos vectors $\widetilde{z}_n$ and y_n are not explicitly computed in an LTPM, we can still use the Lanczos coefficients α_n, β_n, and γ_n to form the tridiagonal Hessenberg matrix $T_{n,n} :=$ tridiag$(\gamma_k, \alpha_k, \beta_k)$ for which

$$AY_n = Y_n T_{n,n} + \gamma_n y_{n+1} \tag{4.1}$$

holds. The eigenvalues of $T_{n,n}$ can then be used as an approximation for the spectrum of the system matrix.

Another, more advanced way is to use, as proposed in [2], the QMR approach also to compute the roots of the B-orthogonal and the residual polynomials. They can then be employed to obtain an approximation of the spectrum that is a subset of $\mathcal{F}_B(A) \cap \mathcal{F}_B^{-1}(A^{-1})$, where $\mathcal{F}_B(A)$ denotes the B-filed of values of A [15].

A further possibility is the idea in [16], where it is exploited that $\mathcal{F}_B(A)$ and $\mathcal{F}_B^{-1}(A^{-1})$ are subsets of the intersection of strips in the complex plane.

5. Obtaining the Optimal Iteration Polynomial. To obtain the optimal iteration polynomial we consider the methods proposed in [13], [20], [23], [16] and [24].

6. Numerical Examples and Conclusions. In many numerical examples we compare different hybrid LTPMs with standard LTPMs. Hybrid LTPMs that are based on the coupled two-term recurrence version of the Lanczos process, which is more stable, are considered in a forthcoming paper.

7. Acknowledgments. The author would like to thank Teri Barth, Thomas A. Manteuffel, and Gerhard Starke for illuminating discussions.

REFERENCES

[1] , S.F. ASHBY, T.A. MANTEUFFEL AND P.E. SAYLOR, *A taxonomy for conjugate gradient methods*, SIAM J. Num. Anal. 27, (1990), pp. 1542-1568.

[2] T. BARTH AND T.A. MANTEUFFEL, *Estimating the spectrum of A using the roots of the polynomials associated with the QMR iterations*, SIAM J. Matrix Anal., (to appear).

[3] T. BARTH AND T.A. MANTEUFFEL, *Variable metric conjugate gradient methods*, in : N. Natori and T. Nodera, eds., Matrix Analysis and Parallel Computing, PCG'94, Advances in Numerical Methods for large sparse sets of linear equations 10, (Keio University, Yokohama, 1994), pp. 165-188.

[4] H.C. ELMAN, Y. SAAD AND P.E. SAYLOR, *A hybrid Chebyshev Krylov subspace algorithm for solving nonsymmetric systems of linear equations*, SIAM J. Sci. Stat. Comput. 7, (1986), pp. 840-855.

[5] V. FABER AND T. MANTEUFFEL, *Necessary and sufficient conditions for the existence of a conjugate gradient method*, SIAM J. Numer. Anal. 21, (1984), pp. 352-362.

[6] R. W. FREUND, M. H. GUTKNECHT AND N. M. NACHTIGAL, *An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices*, SIAM J. Sci. Comput. 14, (1993), pp. 137-158.

[7] R. W. FREUND AND N. M. NACHTIGAL, *QMR: a quasi-minimal residual method for non-Hermitian linear systems*, Numer. Math. 60, (1991), pp. 315-339.

[8] M. H. GUTKNECHT, *A completed theory of the unsymmetric Lanczos process and related algorithms, Part I*, SIAM J. Matrix Anal. Appl. 13, (1992), pp. 594-639.

[9] M. H. GUTKNECHT, *Variants of BiCGStab for matrices with complex spectrum*, SIAM J. Sci. Comput. 14, (1993), pp. 1020-1033.

[10] M.H. GUTKNECHT AND K.J. RESSEL *Look-Ahead Procedures for Lanczos-type Product Methods based on three-term recurrences*, in preparation.

[11] M.R. HESTENES AND E. STIEFEL, *Methods of conjugate gradients for solving linear systems*, J. Res. Nat. Bureau Standards 49, (1952), pp. 409-435.

[12] C. LANCZOS, *Solution of systems of linear equations by minimized iterations*, J. Res. Nat. Bureau Standards 49, (1952), pp. 33-53.

[13] T.A. MANTEUFFEL, *The Tchebychev iteration for nonsymmetric linear systems*, Numer. Math. 28, (1977), pp. 307-327.

[14] T.A. MANTEUFFEL, *Adaptive procedure for estimating parameters for the nonsymmetric Tchebychev iteration*, Numer. Math. 31, (1978), pp. 183-208.

[15] T.A. MANTEUFFEL AND J.S. OTTO, *On the roots of the orthogonal polynomials and residual polynomials associated with a conjugate gradient method*, Numerical Linear Algebra with Applications 1(5), (1994), pp. 449-475.

[16] T.A. MANTEUFFEL AND G. STARKE, *On Hybrid iterative methods for nonsymmetric systems of linear equations*, Technical Report, University of Colorado at Denver, 1993.

[17] N.M. NACHTIGAL, L. REICHEL AND L.N. TREFETHEN, *A hybrid GMRES algorithm for nonsymmetric matrix iterations*, SIAM J. Matrix Anal. Appl. 13, (1992), pp. 796-825.

[18] Y. SAAD, *Least-Squares polynomials in the complex plane and their use for solving nonsymmetric linear systems*, SIAM J. Numer. Anal. 24, (1987), pp. 155-169.

[19] Y. SAAD AND M. H. SCHULTZ, *GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput. 7, (1986), pp. 856-869.

[20] P.E. SAYLOR AND D.C. SMOLARSKI, *Implementation of an adaptive algorithm for Richardson's method*, Linear Algebra Appl. 154-156, (1991), pp. 615-646.

[21] G.L.G. SLEIJPEN AND D.R. FOKKEMA, *BICGSTAB(ℓ) for linear equations involving unsymmetric matrices with complex spectrum*, Electronic Transaction on Numerical Analysis 1, (1993), pp. 11-32.

[22] P. SONNEVELD, *CGS, a fast Lanczos-type solver for nonsymmetric linear systems*, SIAM J. Sci. Comput. 10, (1989), pp. 36-52.

[23] G. STARKE AND R.S. VARGA, *A hybrid Arnoldi-Faber method for nonsymmetric systems of linear equations*, Numer. Math. 64, (1993), pp. 213-240.

[24] P.T.P. TANG, *A fast Algorithm for linear complex Chebyshev approximation*, Math. Comp. 51, (1988), pp. 721-739.

[25] H. A. VAN DER VORST, *Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Comput. 13, (1992), pp. 631-644.

Topic: *Eigenvalues*	*Session Chair:* *Homer Walker*	*Room C*
8:00 - 8:30	K. Wu	Preconditioned Krylov Subspace Methods for Eigenvalue Problems
8:30 - 9:00	J. Baglama	A Numerical Method for Eigenvalue Problems in Modeling Liquid Crystals
9:00 - 9:30	A. Knyazev	A Subspace Preconditioning Algorithm for Eigenvector/Eigenvalue Computation
9:30 - 10:00	Z. Drmac	Stable Computation of Generalized Singular Values

Preconditioned Krylov Subspace Methods for Eigenvalue Problems

Kesheng Wu, Yousef Saad, and Andreas Stathopoulos
Computer Science Department, University of Minnesota

March, 1996

1 Introduction

Lanczos algorithm is a commonly used method for finding a few extreme eigenvalues of symmetric matrices [7, 17]. It is effective if the wanted eigenvalues have large relative separations. If separations are small, several alternatives are often used, including the shift-invert Lanczos method [11], the preconditioned Lanczos method [14], and Davidson method [6, 8, 9, 13]. The shift-invert Lanczos method requires direct factorization of the matrix, which is often impractical if the matrix is large. In these cases preconditioned schemes are preferred.

Many applications require solution of hundreds or thousands of eigenvalues of large sparse matrices, which pose serious challenges for both iterative eigenvalue solver and preconditioner [2, 3, 4, 5]. In this paper we will explore several preconditioned eigenvalue solvers and identify the ones suited for finding large number of eigenvalues. Methods discussed in this paper make up the core of a preconditioned eigenvalue toolkit under construction.

2 Preconditioned Algorithms

2.1 Generating the basis vectors

Conceptually, we separate an eigenvalue solver into two parts; one constructs an orthonormal basis set, the other uses the basis set in a Rayleigh-Ritz procedure to find approximate eigen-pairs. For symmetric matrices the Rayleigh-Ritz procedure is optimal in many aspects [15] and it is easy to use. There are numerous ways of building orthonormal bases, including Davidson method and Arnoldi method. Many of them can be formulated as one algorithm with different preconditioning strategies.

ALGORITHM Generating an orthonormal basis $V_m = [v_1, v_2, \ldots, v_m]$

1. **Start.** *Choose a vector* v_1. *Let* $z = v_1$.
2. **Iterate.** *For* $j = 1, 2, \ldots, m$ *do:*
   (a) $z = z - V_{j-1} V_{j-1}^T z$, $v_j = z/\|z\|$,
   (b) $w_j = A v_j$,
   (c) $h_{i,j-1} = (v_i, w_j)$ *for* $i = 1, \ldots, j$,
   (d) *If* $j < m$, *generate next* z.

We explore four schemes of generating new z.

1. Preconditioned Arnoldi. $z = M_j^{-1} w_j$.
2. Modified Arnoldi. Let $h_j = [h_{1j}, \ldots, h_{jj}]^T$, $z = M_j^{-1}(w_j - V * h_j)$.
3. Davidson.
   (a) Compute smallest eigen-pair (λ, y) of $H_j = V_j^T A V_j$,

(b) $u = V_j y$,

(c) $r = W_j y - \lambda u$, where $W_j = [w_1, \ldots, w_j]$,

(d) $z = M_j^{-1} r$.

4. Harmonic Davidson. Let $H_j = V_j^T A V_j$, $G_j = W_j^T W_j$,

(a) Solve the generalized eigenvalue problem $H_j Y = \Lambda G_j Y$,

(b) $\lambda = \min_{i=1}^{j} y_i^T H_j y_i$. Assume $\lambda = y_1^T H_j y_1$.

(c) $u = V_j y_1$,

(d) $r = W_j y_1 - \lambda u$, where $W_j = [w_1, \ldots, w_j]$,

(e) $z = M_j^{-1} r$.

Among these four schemes, the preconditioned Arnoldi scheme is the most straightforward one. In the unpreconditioned case, i.e., $M = I$, we simply have $z = w_j$. The second scheme is a modification of the Arnoldi scheme. The input to the preconditioner is the last column of $AV_j - V_j H_j$, which could be viewed as the "residual of the basis", and it is orthogonal to the current basis V_j. This scheme can be viewed as an Arnoldi scheme where an extra orthogonalization is applied before preconditioning.

The Davidson's scheme applies the preconditioner on the residual of the current approximate solution. Let n denote the matrix size. At the jth Davidson step, computing one residual vector requires about $2jn$ floating-point multiplications, $2jn$ additions, and the solution of a $j \times j$ eigenvalue problem. This preparation for preconditioning is about twice as expensive as scheme 2.

The Harmonic Ritz value scheme is adopted from [12]. Two alternative ways of computing the Harmonic Ritz values require either V_j and W_j to be bi-orthogonal, or V_j to be A^2-orthonormal, and do not fit into the above framework.

2.2 Some characteristics of the implementations

In practice, the above algorithm is almost always restarted. Usually more than one eigenvalue is wanted, another level of loop is placed outside of the restarting loop. This yields two nested loops outside the above algorithm.

In our implementation of the eigenvalue solvers, we always work on a small number of eigenvalues at a time. This make it possible to keep the maximum basis size independent of the number of eigenvalues desired. Therefore, the workspace does not have to increase as the number of eigenvalues increases.

At restarting, we save Ritz vectors as the initial guesses for the next basis set. The number of vectors saved is independent of the number of eigenvalues wanted. This technique has been shown to enhance the performance of the eigenvalue solver in many cases [18]. When more than one starting vector is given to the Arnoldi process, we build a Krylov subspace from the first vector, v_1. The rest of the initial guesses are used to deflate the Krylov subspace generated. The theoretical aspects of this technique are studied in [16], and some experiences from using it on linear system solvers are reported in [1]. For our experiments here, we always keep the workspace to be about $50n$, i.e., the maximum basis size is 25. When restart, we always save 12 vectors.

Preconditioning is a crucial feature of the programs. In linear system solvers, either M approximates A, or M^{-1} approximates A^{-1}. In the eigenvalue case, if (λ, x) is the current approximation to eigen-pair (λ^*, x^*), the preconditioner is often taken to be

$$M \approx (A - \lambda^* I) \approx (A - \lambda I).$$

A commonly used preconditioner is $M = diag(A) - \lambda I$. Usually it is helpful to use a biased estimate of λ^* as the shift in the preconditioner [19]. For example, if the smallest eigenvalue is sought, we under-estimate it as $\lambda^* \approx \lambda - \|r\|$, where r is the residual vector of the current approximation. Among the four preconditioning strategies we have mentioned, the last two compute residual vectors. In these two cases, we also compute the residual norms and provide updated shift for the preconditioner at every step.

	Arnoldi		Modified		Davidson		Harmonic	
	matvec	time	matvec	time	matvec	time	matvec	time
662BUS	3410	66.8	5000	67.4	4969	136.1	4267	160.0
685BUS	1629	33.5	3814	54.0	1771	48.6	3747	145.3
BCSSTK01	236	0.9	954	6.7	1095	10.2	369	4.9
BCSSTK02	126	0.7	374	2.8	198	2.3	172	2.8
BCSSTK04	3550	19.3	5000	30.4	5005	56.5	3928	65.5
BCSSTK05	563	3.4	1274	11.5	757	9.6	665	11.9
BCSSTK09	745	25.3	1084	30.6	927	43.3	861	55.2
BCSSTK16	964	257.6			952	282.7	718	257.3
BCSSTK22	2118	11.1	1903	12.6	3175	36.7	3266	57.5
BCSSTM10	5006	174.5	444	13.6	263	12.3	4033	276.1
BCSSTM27	4032	190.6	5000	244.4	3331	207.3	5005	409.4
GR3030	199	4.8	323	8.5	147	5.8	628	34.7
LUNDA	1012	5.9	3774	25.6	1485	17.4	1316	22.5
LUNDB	2301	13.1	5000	50.5	3136	37.2	2967	52.0
NOS3	537	15.8	994	24.6	484	20.0	640	35.7
NOS4	3074	15.7	344	3.0	198	2.4	1732	29.6
NOS5	5005	70.8	2453	46.5	1381	30.6	1797	53.1
ZENIOS	5006	604.0	144	13.7	120	15.1	262	46.4

Table 1: Results of unpreconditioned case.

	Arnoldi		Modified		Davidson		Harmonic	
	matvec	time	matvec	time	matvec	time	matvec	time
662BUS	5005	93.7	5000	99.3	991	29.8	5003	185.8
685BUS	5005	101.4	5000	103.0	1004	30.8	5004	194.2
BCSSTK01	1781	6.4	1024	5.0	133	1.4	2694	38.7
BCSSTK02	2194	10.0	5000	27.7	211	2.5	2577	41.6
BCSSTK03	5009	24.2	5000	25.2	2031	23.7	5004	82.1
BCSSTK04	5012	28.7	4003	23.3	198	2.5	5004	86.9
BCSSTK05	5012	31.0	5000	32.1	445	6.2	5003	90.4
BCSSTK08	5005	174.1	5000	181.6	900	44.8	5004	323.2
BCSSTK09	4671	170.5	5000	184.9	900	45.9	4552	305.1
BCSSTK16	5003	1393.9			887	284.4	5005	1971.4
BCSSTK21	5002	820.3	5000	866.1	2227	422.6	5004	1177.2
BCSSTK22	5012	27.4	5000	31.7	1680	21.5	5004	87.7
BCSSTM07	5005	72.3	5000	54.3	367	8.0	5003	145.9
BCSSTM10	5005	179.4	3103	102.4	276	14.2	5004	338.6
BCSSTM13	5003	379.6			303	31.8	5004	588.7
BCSSTM27	5004	246.2	5000	254.0	3045	206.8	5004	428.5
GR3030	159	4.2	323	7.9	159	6.7	680	39.0
LUNDA	5013	32.0	4993	32.9	237	3.2	5005	89.4
LUNDB	5012	29.7	5000	32.0	367	4.7	5004	91.6
NOS3	2370	73.7	5000	130.0	666	29.9	2863	171.2
NOS4	3074	14.7	2144	8.4	237	2.9	3565	61.8
NOS5	4788	71.4	5000	77.0	861	20.3	4318	136.5
NOS6	5005	99.5	5000	106.5	4605	143.8	5004	189.6
NOS7	5005	105.0	5000	113.8	211	7.3	5004	195.5
ZENIOS	172	20.5	134	13.3	120	16.0	250	46.1

Table 2: Results of diagonal preconditioned case.

3 Experimental Comparisons

Our numerical experiments are conducted on a set of symmetric matrices from the Harwell-Boeing sparse matrix collection [10]. We used all the matrices of RSA type. In the following tables, we have dropped the results of diagonal matrices and those which can not be solved by any of the methods tested. The eigenvalues are considered converged if the residual norm of the approximation is less than $10^{-12}\|A\|_F$, where $\|A\|_F$ is the Frobenius norm of the matrix.

3.1 Without preconditioning

The first experiment is performed without any preconditioning (see table 1). The table shows the number of matrix-vector multiplications (matvec) and the CPU time used to compute 5 smallest eigenvalues. The unit of time is second. The limit on the number of matrix-vector multiplications is 5000. Entries in the table showing 5000 or more matrix-vector multiplications indicate that the eigenvalue solver did not compute all five wanted eigenvalues. The results from Arnoldi method, Modified Arnoldi method, Davidson method and Harmonic Davidson method are shown.

There are 50 test matrices. Without preconditioning, we solved 18 of them by one of the four eigenvalue solvers. Comparing the columns shown in table 1, we notice that Arnoldi method and Davidson method generally use less matrix-vector multiplications and less time than the other two. When solving the same eigenvalue problem, Arnoldi method often takes less time than Davidson method on the matrices tested.

3.2 Simple preconditioning

This experiment is performed with diagonal preconditioning, see table 2. Here again we look for 5 smallest eigenvalues. With diagonal preconditioning, we are able to solve half of the problem set. Our experiment indicates that Davidson method can take advantage the diagonal preconditioner better than others.

3.3 Finding more eigenvalues

Table 3 demonstrates the behavior of Davidson eigenvalue solver when more eigenvalues are wanted. The number of matrix-vector multiplications allowed is 20,000 for finding 20 eigenvalues, 40,000 for finding 40 eigenvalues. For all 25 matrices where Davidson method was successful in finding 5 eigenvalues, it is also able to find 20 eigenvalues with more matrix-vector multiplications. There are 3 case where Davidson method failed to find 40 eigenvalues. For matrix 685BUS, we are able to find 39. In the other two cases, less than 30 eigenvalues have converged.

The average ratio of the time spent on finding 20 eigenvalues versus time spent on finding 5 eigenvalues is about 3.8 (table 3). For the 23 cases where 40 eigenvalues are found by Davidson method, the average ratio time spent on finding 40 eigenvalues versus time spent on finding 5 eigenvalues is about 9.5. This indicates that if Davidson method do find the desired number of eigenvectors, it is fairly efficient in finding them. However Davidson method could fail to find more eigenvalues as in the case of 685BUS, BCSSTM27 and NOS6.

Acknowledgment. This research was supported by National Science Foundation Grant DMR 95-25885 and NSF/ASC 95-04038. The authors would also like to acknowledge the support of the Minnesota Supercomputer Institute which provided the computer facilities and an excellent research environment to conduct this research.

References

[1] A. Chapman and Y. Saad. Deflated and augmented Krylov subspace techniques. Technical Report UMSI 95/181, Minnesota Supercomputing Institute, Univ. of Minnesota, 1995.

[2] J. R. Chelikowsky, N. Troullier, and Y. Saad. Finite-difference-pseudopotential method: electronic structure calculations without a basis. *Phys. Rev. Lett.*, 72:1240–3, 1994.

	5		20 eigenvalues		40 eigenvalues	
	matvec	time	matvec	time	matvec	time
662BUS	991	29.8	3710	126.0	7773	316.7
685BUS	1004	30.8	3606	127.0	40012	2261.1
BCSSTK01	133	1.4	340	3.5	399	4.1
BCSSTK02	211	2.5	655	8.0	1247	15.5
BCSSTK03	2031	23.7	3294	40.0	4250	53.8
BCSSTK04	198	2.5	616	8.4	1559	22.9
BCSSTK05	445	6.2	1513	21.9	2859	43.4
BCSSTK08	900	44.8	4165	248.7	10880	828.9
BCSSTK09	900	45.9	2878	153.4	5342	332.8
BCSSTK16	887	284.4	3346	1037.0	5927	2071.1
BCSSTK21	2227	422.6	6050	1229.6	12271	2696.2
BCSSTK22	1680	21.5	3086	41.4	10763	164.0
BCSSTM07	367	8.0	1344	32.9	2729	77.3
BCSSTM10	276	14.2	1292	72.8	5108	362.4
BCSSTM13	303	31.8	915	108.1	1846	264.5
BCSSTM27	3045	206.8	15345	1129.3	40002	3579.1
GR3030	159	6.7	681	30.0	1663	86.4
LUNDA	237	3.2	720	9.8	1741	26.4
LUNDB	367	4.7	1227	16.8	2963	45.0
NOS3	666	29.9	2202	103.8	4302	237.2
NOS4	237	2.9	720	9.1	1416	18.5
NOS5	861	20.3	2436	59.3	3847	106.2
NOS6	4605	143.8	11471	386.4	40001	1721.7
NOS7	211	7.3	949	35.5	2105	91.4
ZENIOS	120	16.0	1006	152.1	2534	445.4

Table 3: Results of seeking different number of eigenvalues.

[3] J. R. Chelikowsky, N. Troullier, K. Wu, and Y. Saad. Higher order finite difference pseudopotential method: an application to diatomic molecules. *Phys. Rev. B*, 50:11355–11364, 1994.

[4] J. R. Chelikowsky, N. R. Troullier, X. Jing, D. Dean, N. Binggeli, K. Wu, and Y. Saad. Algorithms for the structural properties of clusters. *Computer Physics Communications*, 85:325–335, 1995.

[5] M. L. Cohen and J. R. Chelikowsky. *Electronic Structure and Optical Properties of Semiconditors.* Springer-Verlag, New York, Berlin, Heidelberg, 2nd edition, 1989.

[6] M. Crouzeix, B. Philippe, and M. Sadkane. The Davidson method. *SIAM J. Sci. Comput.*, 15:62–76, 1994.

[7] J. Cullum and R. A. Willoughby. *Lanczos Algorithms for Large Symmetric Eigenvalue Computations*, volume 1: Theory of *Progress in Scientific Computing; v. 3.* Birkhauser, Boston, 1985.

[8] Ernest R. Davidson. The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real-symmetric matrices. *J. Comput. Phys.*, 17:87, 1975.

[9] Ernest R. Davidson. super-matrix methods. *Computer Physics Communications*, 53:49–60, 1989.

[10] I. S. Duff, R. G. Grimes, and J. G. Lewis. Sparse matrix test problems. *ACM Trans. Math. Soft.*, pages 1–14, 1989.

[11] R. G. Grimes, J. G. Lewis, and H. D. Simon. A shifted block lanczos algorithm for solving sparse symmetric generalized eigenproblems. *SIAM J. Matrix Anal. Appl.*, 15(1):228–272, 1994.

[12] R. B. Morgan. Computing interior eigenvalues of large matrices. *Lin. Alg. Appl.*, pages 289–309, 1991.

[13] R. B. Morgan and D. S. Scott. Generalizations of davidson's method for computing eigenvalues of sparse symmetric matrices. *SIAM J. Sci. Statist. Comput.*, 7:817–825, 1986.

[14] R. B. Morgan and D. S. Scott. Preconditioning the Lanczos algorithm for sparse symmetric eigenvalue problems. *SIAM J. Sci. Comput.*, 14:585–593, 1993.

[15] Beresford N. Parlett. *The symmetric eigenvalue problem.* Prentice-Hall, Englewood Cliffs, NJ, 1980.

[16] Y. Saad. Analysis of augmented Krylov subspace techniques. Technical Report UMSI 95/175, Minnesota Supercomputing Institute, Univ. of Minnesota, 1995.

[17] Yousef Saad. *Numerical methods for large eigenvalue problems.* Manchester University Press, 1993.

[18] D. S. Sorensen. Implicit application of polynomial filters in a K-step Arnoldi method. *SIAM J. Matrix Anal. Appl.*, 13(1):357–385, 1992.

[19] A. Stathopoulos, Y. Saad, and C. F. Fischer. Robust preconditioning of large, sparse, symmetric eigenvalue problems. *Journal of Computational and Applied Mathematics*, 64:197–215, 1995.

A NUMERICAL METHOD FOR EIGENVALUE PROBLEMS IN MODELING LIQUID CRYSTALS *

J. BAGLAMA †, D. CALVETTI ‡, P. A. FARRELL†, L. REICHEL†, AND A. RUTTAN†

Abstract. Equilibrium configurations of liquid crystals in finite containments are minimizers of the thermodynamic free energy of the system. It is important to be able to track the equilibrium configurations as the temperature of the liquid crystals decreases. The path of the minimal energy configuration at bifurcation points can be computed from the null space of a large sparse symmetric matrix. We describe a new variant of the implicitly restarted Lanczos method that is well suited for the computation of extreme eigenvalues of a large sparse symmetric matrix, and we use this method to determine the desired null space. Our implicitly restarted Lanczos method determines adaptively a polynomial filter by using Leja shifts, and does not require factorization of the matrix. The storage requirement of the method is small, and this makes it attractive to use for the present application.

1. Introduction. Equilibrium configurations of liquid crystals in finite containments are minimizers of the thermodynamic free energy of the system. It is important to be able to track the equilibrium configurations as the temperature of the liquid crystals decreases. Standard methods for tracking these equilibrium configurations involve using Newton's to solve the associated stationary equations. At folds or bifurcation points the Jacobian, A, of the stationary equations is singular, and then the null space of A can be used to determine appropriate continuation vectors. In liquid crystal problems A is a large sparse matrix, resulting from a discretization on a 2 or 3 dimensional domain, so, in particular, direct methods for determining the null space of A are impractical, and iterative methods must be used.

We take a general approach to this problem: we consider methods for determining a few extreme eigenvalues and associated eigenvectors of a large sparse symmetric matrix $A \in \mathbf{R}^{n\times n}$ as this is an important computational problem that arises in many applications in addition to liquid crystal problems. A large number of algorithms for the solution of this problem are based on the Lanczos process. However, when applying the basic Lanczos method, one may encounter the following difficulties: large storage requirement for the Krylov subspace basis generated, and low accuracy of the computed approximate eigenvalues and eigenvectors due to loss of orthogonality of the computed Krylov subspace basis.

The difficulties associated with the basic Lanczos method have spurred considerable research aimed at improving the method or at developing alternative methods. Recently, Sorensen [?] proposed the Implicitly Restarted Lanczos (IRL) method for the computation of a few eigenvalues of a large sparse symmetric matrix, and the closely related Implicitly Restarted Arnoldi (IRA) method for the computation of a few eigenvalues of a large sparse nonsymmetric matrix. These methods can be regarded as curtailed QR algorithms for the symmetric and nonsymmetric eigenvalue problems, respectively. Similarly as in the QR algorithms, the choice of shifts is crucial for the performance of the IRL and IRA methods. However, the Rayleigh or Wilkinson shifts, popular choices of shifts for the QR algorithm, cannot be applied in the IRL and IRA methods, because the data required to compute these shifts is not available. Therefore other shift selection strategies have been studied by Calvetti et al. [?], Lehoucq [?] and Sorensen [?].

It is the purpose of the present paper to describe a new way to use the recursion formulas of the IRA method, and to apply the scheme obtained to the computation of a few of the smallest eigenvalues of a large sparse symmetric matrix that arises in the modeling of liquid crystals. The storage requirement of our scheme is smaller than for previously described algorithms based on the IRL recursions.

2. The implicitly restarted Lanczos method. The Lanczos process is a computational method for reducing an $n \times n$ symmetric matrix A to tridiagonal form, given an initial basis vector v_1, which we may assume to be of unit length. If we truncate the Lanczos process after $m < n$ steps, then we obtain the truncated reduction of A to tridiagonal form

$$AV_m = V_m T_m + f_m e_m^T, \tag{2.1}$$

where $V_m \in \mathbf{R}^{n\times m}$, $V_m e_1 = v_1$, $V_m^T V_m = I$, $T_m \in \mathbf{R}^{m\times m}$ is a symmetric tridiagonal matrix, and $f_m \in \mathbf{R}^n$ satisfies $V_m^T f_m = 0$. Throughout this paper e_j denotes the jth axis vector of appropriate dimension, and

* Research supported in part by NSF grants F377 DMR-8920147 ALCOM, DMS-9409422 and DMS-9404706.

† Department of Mathematics and Computer Science, Kent State University, Kent, OH 44242.

‡ Department of Mathematical Sciences, Stevens Institute of Technology, Hoboken, NJ 07030.

I denotes an identity matrix of suitable order. For future reference, we define

$$\beta_m = \|f_m\|, \tag{2.2}$$

where $\|\cdot\|$ denotes the Euclidean norm.

We describe the recursion formulas of the IRL method due to Sorensen [?]. Throughout this paper, k denotes the number of desired extreme eigenvalues. We assume that k is fixed and small. The number of steps of the Lanczos process taken between restarts is denoted by m, where we assume that $m > k$. After m steps of the Lanczos process with initial vector v_1, we have determined the quantities in formula (??). We now apply the following updating formulas, which are analogous to the explicitly shifted QR algorithm.

Let z be a chosen shift and determine the QR factorization $T_m - zI = QR$, where $Q, R \in \mathbf{R}^{m\times m}$, $Q^TQ = I$ and R is upper triangular. We put $V = V_m$ and $T = T_m$ and obtain

$$\begin{aligned}
(A - zI)V - V(T - zI) &= f_m e_m^T, \\
(A - zI)V - VQR &= f_m e_m^T, \\
(A - zI)(VQ) - (VQ)(RQ) &= f_m e_m^T Q, \\
A(VQ) - (VQ)(RQ + zI) &= f_m e_m^T Q.
\end{aligned}$$

After applying the $m-1$ shifts $z_1, z_2, \ldots, z_{m-1}$, we have

$$AV_m^+ = V_m^+ T_m^+ + f_m e_m^T Q^+, \tag{2.3}$$

where $V_m^+ = (v_1^+, v_2^+, \ldots, v_m^+) = V_m Q^+$, $T_m^+ = (Q^+)^T T_m Q^+$ and $Q^+ = Q_1 Q_2 \cdots Q_{m-1}$. Here Q_j denotes the orthogonal matrix associated with the shift z_j. Introduce the partitioning

$$T_m^+ = \begin{pmatrix} \alpha_1^+ & \beta_1^+ e_1^T \\ \beta_1^+ e_1 & T_{m-1}^+ \end{pmatrix},$$

and equate the first column on the right-hand side and left-hand side of (??). We then obtain

$$Av_1^+ = v_1^+ \alpha_1^+ + f_1^+ e_1^T,$$

where $f_1^+ = v_2^+ \beta_1^+ + f_m e_m^T Q^+ e_1$. It follows from $(v_1^+)^T f_1^+ = 0$ and $f_1^+ \in \mathrm{span}\{v_1^+, Av_1^+\}$ that f_1^+ can be determined by the Lanczos process applied to the matrix A with initial vector v_1^+. The mth shift z_m is applied according to

$$v_1^{++} = f_1^+ + (\alpha_1^+ - z_m)v_1^+.$$

By construction, $v_1^{++} = \psi_m(A)v_1$, where ψ_m is a polynomial of degree m with zeros $z_1, z_2, \ldots, z_m$. We refer to ψ_m as an accelerating polynomial. Note that v_1^{++} has been computed from v_1 without evaluation of matrix-vector products with the matrix A, except for the products required to determine the decomposition (??). Having computed v_1^{++} in the manner indicated, our scheme sets

$$v_1 = v_1^{++}/\|v_1^{++}\| \tag{2.4}$$

and restarts the Lanczos process with the vector (??) as initial vector. Our scheme proceeds in this manner to alternatively apply m steps of the Lanczos process and m shifts until an initial vector v_1 has been determined that lies in the invariant subspace associated with the k desired eigenvalues. Since we only keep at most $m+1$ orthogonal basis vectors of a Krylov subspace in memory during the computations, we can afford to secure their orthogonality by reorthogonalization whenever necessary.

As soon as an eigenvector has converged, it is stored, and subsequently generated Krylov subspace bases are orthogonalized against it. Assume that we have determined $j < k$ eigenvectors. Then the Lanczos process is only applied $m-j$ steps at a time, in order not to increase the memory requirement of the scheme. The orthogonalization of Krylov subspaces against converged eigenvectors makes it possible to determine multiple eigenvalues.

Our scheme differs from previous applications of the recursions of the IRL method in that we apply as many shifts as possible, i.e., until only the vector v_1^{++} remains. This makes deflation of converged eigenpairs trivial, and reduces the requirement of computer storage.

3. The iterative method. The rate of convergence of our iterative method is determined by the accelerating polynomial. We determine this polynomial by prescribing its zeros. Sometimes we refer to the zeros as shifts, because as shown in Section 2, they are shifts applied by a curtailed QR algorithm.

Let $\mathbf{K}$ be a compact interval on the real axis, and let $w(z)$ be a nonnegative continuous function on $\mathbf{K}$. We refer to $w(z)$ as a weight function. Define a sequence $\{z_j\}_{j=1}^{\infty}$ of points in $\mathbf{K}$ as follows. Let z_1 be a point such that

$$w(z_1)|z_1| = \max_{z\in\mathbf{K}} w(z)|z|, \qquad z_1 \in \mathbf{K}, \tag{3.1}$$

and let z_j satisfy

$$w(z_j)\prod_{l=1}^{j-1}|z_j - z_l| = \max_{z\in\mathbf{K}} w(z)\prod_{l=1}^{j-1}|z - z_l|, \qquad z_j \in \mathbf{K}, \qquad j = 2,3,\ldots . \tag{3.2}$$

We call any points $\{z_j\}_{j=1}^{\infty}$ that satisfy (??)-(??) Leja points or Leja shifts for $\mathbf{K}$. When $w(z) = 1$, the Leja points agree with the "classical" Leja points studied by Leja [?].

The sets $\mathbf{K}$ used in our scheme are chosen so that they contain none of the desired k extreme eigenvalues and all or most of the $n-k$ undesired ones. The motivation for choosing the shifts to be Leja points for such sets is that we want to dampen eigenvector components associated with undesired eigenvalues in the initial Lanczos vectors v_1 determined by (??). We now describe how the eigenvalues of the tridiagonal matrices T_m generated by the scheme help us determine such sets $\mathbf{K}$.

Assume that we want to compute the k smallest of the eigenvalues $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_k < \lambda_{k+1} \leq \ldots \leq \lambda_n$ of A, and let

$$\theta_1 < \theta_2 < \ldots < \theta_k < \theta_{k+1} < \ldots < \theta_m. \tag{3.3}$$

be eigenvalues of the matrix T_m defined by (??). We may assume that the subdiagonal elements of T_m are non-vanishing, because otherwise we have found an invariant subspace. By the Cauchy interlacing theorem $\lambda_k \leq \theta_k$ and $\theta_m \leq \lambda_n$, and therefore none of the k smallest eigenvalues of A lies in the interval $[\theta_{k+1}, \theta_m]$. This suggests the following choice of the set $\mathbf{K}$. The first time we determine a tridiagonal matrix T_m by the Lanczos process, we compute its spectrum (??) and define the endpoints of the interval $\mathbf{K} = [a, d]$ by

$$a = \theta_{k+1}, \qquad d = \theta_m.$$

We let the m shifts $\{z_j\}_{j=1}^{m}$ be Leja points for $\mathbf{K}$, i.e., we define the shifts by (??)-(??). Application of the shifts as described in Section 2 yields a new vector v_1 defined by (??). We now apply m steps of the Lanczos process to the matrix A with this vector v_1 as initial vector. This gives rise to a new tridiagonal matrix T_m, defined by (??). The eigenvalues (??) of this tridiagonal matrix are computed and the endpoints of $\mathbf{K} = [a, d]$ are updated according to

$$a = \theta_{k+1}, \qquad d = \max\{d, \theta_m\}.$$

We then select m shifts $z_{m+1}, z_{m+2}, \ldots, z_{2m}$ as Leja points for this new set $\mathbf{K} = [a, d]$ in the presence of the points $z_1, z_2, \ldots, z_m$. The weight function is chosen to be $w(z) = |z - \theta_{k+1}|$.

Let $\{\theta_j, y_j\}_{j=1}^{m}$ denote eigenvalue-eigenvector pairs of the symmetric tridiagonal matrix $T_m \in \mathbf{R}^{m\times m}$ defined by (??), and let the eigenvalues be ordered according to (??).

The vector $x_j = V_m y_j$ is an approximate eigenvector of the matrix A associated with the approximate eigenvalue θ_j, and

$$\|Ax_j - x_j\theta_j\| = |\beta_m e_m^T y_j|,$$

where β_m is defined by (??). In our scheme the computations are terminated as soon as

$$\max_{1\leq j\leq k} |\beta_m e_m^T y_j| \leq \epsilon, \tag{3.4}$$

where ϵ is a given positive constant. A formal algorithm based on the description in this and the previous sections can be found in [?]. There also modifications that allow the computation of k non-extreme eigenvalues are presented.

4. Liquid crystal modeling. We wish to determine the minimum energy equilibrium configuration of liquid crystals in a slab

$$\Omega = \{(x,y,z) : 0 \leq x \leq a, 0 \leq y \leq b, 0 \leq z \leq c\} \tag{4.1}$$

with surface $\partial\Omega$. Using the Landau-de Gennes formulation, the free energy can be expressed in terms of a tensor order parameter field Q; see Gartland [?] and Priestly et al. [?]. The free energy is given by

$$F(Q) = F_{\text{vol}}(Q) + F_{\text{surf}}(Q) = \int_{\Omega} f_{\text{vol}}(Q)dV + \int_{\partial\Omega} f_{\text{surf}}(Q)dS, \tag{4.2}$$

where $Q = Q(p)$, $p \in \Omega$, is a 3×3 symmetric traceless tensor, which is represented as a linear combination of 5 basis traceless 3×3 matrices Q_j,

$$Q(p) = \sum_{j=1}^{5} q_j(p) Q_j.$$

The q_j are real valued functions on Ω and are to be determined so that the free energy (??) is minimal. The representation

$$\begin{aligned} f_{\text{vol}}(Q) \;=\; & \tfrac{1}{2}L_1 Q_{\alpha\beta,\gamma} Q_{\alpha\beta,\gamma} + \tfrac{1}{2}L_2 Q_{\alpha\beta,\beta} Q_{\alpha\gamma,\gamma} + \tfrac{1}{2}L_3 Q_{\alpha\beta,\gamma} Q_{\alpha\gamma,\beta} \\ & + \tfrac{1}{2}A \text{ trace}(Q^2) - \tfrac{1}{3}B \text{ trace}(Q^3) + \tfrac{1}{4}C \text{ trace}(Q^2)^2 \\ & + \tfrac{1}{5}D \text{ trace}(Q^2)\text{trace}(Q^3) + \tfrac{1}{6}M \text{ trace}(Q^2)^3 + \tfrac{1}{6}M' \text{ trace}(Q^3)^2 \end{aligned}$$

uses the conventions that summation over repeated indices is implied and indices separated by commas represent partial derivatives. Here L_1, L_2 and L_3 are elastic constants; A, B, C, D, M and M' are bulk constants. Moreover,

$$f_{\text{surf}}(Q) = W \text{ trace}((Q - Q_0)^2),$$

where W is a constant and the tensor Q_0 is determined by the boundary conditions for the functions q_i. We impose the boundary condition

$$F_{\text{surf}}(Q) = \int_{\partial\Omega} f_{\text{surf}}(Q)dS = 0, \tag{4.3}$$

which implies that the values of the q_i are prescribed on $\partial\Omega$. This boundary condition models strong anchoring of the liquid crystals on the surface $\partial\Omega$. Details can be found in [?, ?, ?].

The minimum energy equilibrium configuration of the liquid crystals is determined by solving the Euler-Lagrange equations associated with (??) and (??). These equations yield a boundary value problem for a system of nonlinear partial differential equations for the q_i. Discretization by finite differences gives rise to a system of nonlinear equations of finite, but large, order. At nonsingular points, we solve this system by Newton's method. Each iteration of Newton's method requires the solution of a linear system of equations, the matrix of which is the Jacobian obtain from the discretized Euler-Lagrange equations. The purpose of the computations is to track the minimal energy equilibrium configuration as the temperature of the liquid crystals is varied. In order to recognize singular points and to determine continuation vectors at such points, it is essential to determine a few of the eigenvalues of the Jacobian closest to the origin. In order to reduce the storage space required, we do not store all the nonzero entries of the Jacobian matrix simultaneously, but rather we compute them as required. This approach is motivated by the fact that the order of the Jacobian matrices of interest is very large; see Example 2 of the next section.

5. Computed examples. This section describes some computed examples which illustrate the behavior of the scheme outlined in Sections 2-3. The computations were carried out on an HP 9000/770 computer using double precision arithmetic, i.e., with approximately 15 significant digits.

Our first example compares our scheme with the subroutine DNLASO of the FORTRAN package LASO2 by Scott [?] and with a subroutine in ARPACK by Lehoucq, Sorensen and Vu [?]. The subroutine DNLASO implements the Lanczos process with selective orthogonalization, see [?], and allows the user to specify the maximal amount of computer storage available for the code to use. Typically, the more storage available, the fewer restarts necessary and the faster convergence to the desired eigenvalues and eigenvectors. The subroutine allows the user to select block-size for the Lanczos process; if the block-size, denoted by NBLOCK, is larger than one, then DNLASO implements a block Lanczos algorithm. The parameter MAXJ of DNLASO specifies the order of the largest symmetric block-tridiagonal Lanczos matrix generated by the algorithm before restart. The largest order of this matrix is MAXJ*NBLOCK. The storage requirement for the (block) Lanczos vectors generated by DNLASO is n*MAXJ*NBLOCK storage locations. The columns labeled "# Lanczos vectors" in the tables display MAXJ*NBLOCK. The total storage requirement for DNLASO is larger than n*(MAXJ+2)*NBLOCK in addition to the storage needed to represent the matrix A.

The subroutine DNLASO is more advanced than the experimental code for our scheme and has multiple stopping criteria. The iterations may terminate before desired accuracy is achieved and this makes a comparison between the subroutine DNLASO and our scheme difficult. The performance of DNLASO is therefore displayed in tables for different choices of the maximal number of Lanczos vectors. The subroutine DNLASO allows the specification of a parameter NFIG, the number of desired correct decimal digits in the computed eigenvalue approximations. We show the performance of the DNLASO for NFIG=10. The columns "# of matrix-vector products" displays the number of matrix-vector products with the matrix A. The number of matrix-vector products shown in the tables is for vectors consisting of one column only, also when the block-size is larger than 1. The tables also display the magnitude of the errors in the computed eigenvalues. The stopping criterion is seen to be more reliable for block-size 3 than for block-size 1.

The iterations with the code for our scheme were terminated when condition (??) was satisfied. This stopping criterion gave in general at least $-2\log_{10}(\epsilon)$ correct decimal digits in the computed approximate eigenvalues. In all computed examples, the entries of the initial Lanczos vector v_1 were uniformly distributed random numbers in an interval $(0, \rho]$, where $\rho > 0$ is chosen so that $\|v_1\| = 1$. The initial vector is the same for all runs with all codes in each example, but it may differ between different examples. In experiments with DNLASO with block-size larger than 1, the first vector in the initial block is the initial vector used in experiments with block-size 1. The other vectors in the initial block have randomly generated uniformly distributed entries. The column "# Lanczos vectors" shows the parameter m; our scheme require storage of $m + 1$ vectors of length n.

From ARPACK we use a subroutine that implements the IRL method with "exact shifts" as described by Sorensen [?] and Calvetti et al. [?] and Lehoucq [?]. We refer to this method as ARPACK. Thus, when interested in computing the k smallest eigenvalues of A, we use the $m - k$ largest eigenvalues of the matrices T_m generated as shifts z_j. The application of exact shifts often requires m to be chosen substantially larger than k. This phenomenon is discussed in Example 1 as well as in [?]. More computed examples can be found in [?].

Example 1. We wish to compute the three smallest eigenvalues of the matrix

$$A = \mathrm{diag}(1, 2, 3, \ldots, n). \tag{5.1}$$

Tables 5.1-5.4 compare our scheme with DNLASO and ARPACK when the matrices (??) are of order $n = 2500$. Table 5.1 shows the performance of our scheme. Figure 5.1 displays the number of matrix-vector products required by our scheme for increasing values of m, and illustrates that the convergence of our scheme is fairly insensitive to the choice of m. The figure also displays the number of matrix-vector products required by ARPACK for different choices of m. We can see that ARPACK requires a larger value of m, i.e., storage of more n-vectors, in order to perform well. Analogous numerical experiments with ARPACK are reported in [?].

Table 5.2 shows the number of matrix-vector products required by ARPACK. The stopping criterion implemented in ARPACK is designed to terminate the computations when $|\lambda_e - \lambda_c| < \epsilon|\lambda_c|$, where λ_e and λ_c denotes an exact eigenvalue and a computed approximation, respectively. With the choice $\epsilon = 1 \cdot 10^{-4}$, ARPACK gives an approximation of the smallest eigenvalue λ_1 of about the same accuracy as our scheme. However, the computed approximations of λ_2 and λ_3 obtained by ARPACK were not as

TABLE 5.1
Example 1. Our scheme: $k = 3$, $\epsilon = 1 \cdot 10^{-4}$

# Lanczos vectors	# matrix-vector products	magnitude of error in computed approx. of		
		λ_1	λ_2	λ_3
5	525	$6.4 \cdot 10^{-11}$	$1.6 \cdot 10^{-10}$	$3.8 \cdot 10^{-10}$
10	583	$3.2 \cdot 10^{-10}$	$9.4 \cdot 10^{-11}$	$2.5 \cdot 10^{-11}$
15	497	$2.8 \cdot 10^{-11}$	$1.2 \cdot 10^{-11}$	$9.5 \cdot 10^{-11}$

accurate as those determined by our scheme. ARPACK required the evaluation of more matrix-vector products than our scheme. Decreasing ϵ, in order to obtain more accurate approximations of λ_2 and λ_3, increases the number of matrix-vector products required by ARPACK.

Table 5.3 displays the number of matrix-vector products required by the subroutine DNLASO when storage is limited to 10, 20 and 30 Lanczos vectors and NFIG=10. The block-size is 1. The accuracy is seen to increase with the storage size for the Krylov subspace basis. The storage requirement as well as the number of matrix-vector products are much larger than for our scheme. The accuracy achieved in the computed approximations of λ_1 and λ_2 when the iterations were terminated is lower than for our scheme.

The performance of the block Lanczos algorithm is illustrated by Table 5.4. The accuracy achieved is higher than in Table 5.3, but so is the number of matrix-vector products required. □

Example 2. Discretize Ω defined by (??) by a $40 \times 40 \times 40$ grid. This yields a Jacobian matrix of order 296,595. The smallest eigenvalues of this matrix are, roughly, $\lambda_1 = -1.2 \cdot 10^{-2}$, $\lambda_2 = 2.2 \cdot 10^{-1}$ and $\lambda_3 = 2.3 \cdot 10^{-1}$. We used our scheme with $\epsilon = 1 \cdot 10^{-5}$ and $m = 5$ to compute accurate approximations of these eigenvalues. This required the evaluation of 649 matrix-vector products with the Jacobian matrix. For comparison, we note that 664 matrix-vector products are required when $m = 10$. Thus, increasing the value of m does not decrease the computational work required. Due to the large size of the matrix, only a basis of a Krylov subspace of small dimension can be stored in fast computer memory. In particular, inverse iteration is infeasible. This examples illustrates that our scheme can determine eigenvalues of a very large matrix by using a sequence of Krylov subspaces of low dimension only. □

REFERENCES

[1] J. BAGLAMA, D. CALVETTI AND L. REICHEL, *Iterative methods for the computation of a few extreme eigenvalues of a large symmetric matrix*, BIT, to appear.

[2] D. CALVETTI, L. REICHEL AND D. C. SORENSEN, *An implicitly restarted Lanczos method for large symmetric eigenvalue problems*, Elec. Trans. Numer. Anal., 2 (1994), pp. 1–21.

[3] P. A. FARRELL, A. RUTTAN AND R. R. ZELLER, *Finite difference minimization of the Landau-de Gennes free energy for liquid crystals in rectangular regions*, Comp. Appl. Math., I, C. Brezinski and U. Kulish, eds., Elsevier, Amsterdam, 1992, pp. 137–146.

[4] E. C. GARTLAND, *On some mathematical and numerical aspects of the Landau-de Gennes minimization problem for liquid crystals*, Report, Institute for Computational Mathematics, Kent State University, Kent, 1993.

[5] R. LEHOUCQ, *Analysis and Implementation of an Implicitly Restarted Arnoldi Iteration*, Ph.D. Thesis, Rice University, Houston, 1995.

[6] R. LEHOUCQ, D. C. SORENSEN AND P.A. VU, ARPACK: An implementation of the implicitly restarted Arnoldi and the implicitly restarted Lanczos methods. Code available from Netlib, in directory scalapack.

[7] F. LEJA, *Sur certaines suits liées aux ensemble plan et leur application à la representation conforme*, Ann. Polon. Math., 4 (1957), pp. 8–13.

[8] B. N. PARLETT AND D. S. SCOTT, *The Lanczos algorithm with selective orthogonalization*, Math. Comp., 33 (1979), pp. 311–328.

[9] E. B. PRIESTLY, P. J. WOJYOWICZ, AND P. SHENG, eds., Introduction to Liquid Crystals, Plenum Press, New York, 1975.

[10] D. S. SCOTT, LASO2 - FORTRAN implementation of the Lanczos process with selective orthogonalization. Code and documentation available from Netlib.

[11] D. C. SORENSEN, *Implicit application of polynomial filters in a k-step Arnoldi method*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 357–385.

TABLE 5.2
Example 1. ARPACK: $k = 3$, $\epsilon = 1 \cdot 10^{-4}$

# Lanczos vectors	# matrix-vector products	magnitude of error in computed approx. of		
		λ_1	λ_2	λ_3
5	3724	$6.4 \cdot 10^{-10}$	$1.9 \cdot 10^{-4}$	$5.8 \cdot 10^{-5}$
10	904	$5.5 \cdot 10^{-10}$	$1.1 \cdot 10^{-4}$	$2.4 \cdot 10^{-5}$
15	614	$5.7 \cdot 10^{-10}$	$7.8 \cdot 10^{-5}$	$1.1 \cdot 10^{-4}$

TABLE 5.3
Example 1. DNLASO: NFIG=10, block-size = 1

# Lanczos vectors	# matrix-vector products	magnitude of error in computed approx. of		
		λ_1	λ_2	λ_3
10	1691	$5.7 \cdot 10^{-7}$	$5.6 \cdot 10^{-7}$	$1.7 \cdot 10^{-11}$
20	1922	$1.6 \cdot 10^{-8}$	$1.5 \cdot 10^{-8}$	$2.6 \cdot 10^{-11}$
30	1721	$1.3 \cdot 10^{-9}$	$1.2 \cdot 10^{-9}$	$6.1 \cdot 10^{-11}$

TABLE 5.4
Example 1. DNLASO: NFIG=10, block-size = 3

# Lanczos vectors	# matrix-vector products	magnitude of error in computed approx. of		
		λ_1	λ_2	λ_3
18	3813	$1.3 \cdot 10^{-11}$	$8.3 \cdot 10^{-12}$	$4.7 \cdot 10^{-11}$
30	2952	$5.6 \cdot 10^{-11}$	$1.2 \cdot 10^{-11}$	$2.4 \cdot 10^{-11}$
39	2274	$9.3 \cdot 10^{-12}$	$5.9 \cdot 10^{-12}$	$1.3 \cdot 10^{-11}$

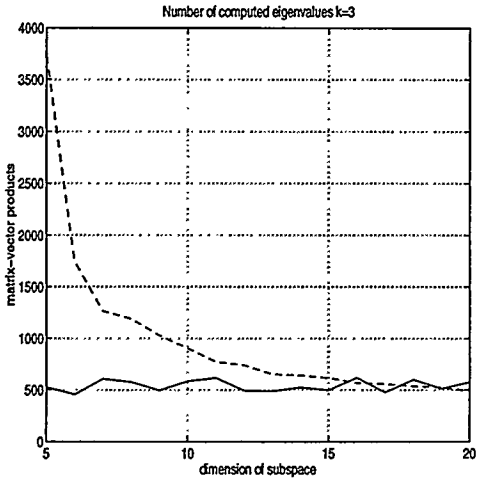


FIG. 5.1. *Number of matrix-vector products required by our scheme (—) and ARPACK (- -) for computing the* 3 *smallest eigenvalues for the matrix of Example 1 with* $n = 2500$ *as the dimension of the Krylov subspace is increased.*

A Subspace Preconditioning Algorithm for Eigenvector/Eigenvalue Computation

James H. Bramble,
Andrew V. Knyazev
Joseph E. Pasciak

We consider the problem of computing a modest number of the smallest eigenvalues along with orthogonal bases for the corresponding eigen-spaces of a symmetric positive definite matrix. In our applications, the dimension of a matrix is large and the cost of its inverting is prohibitive. In this paper, we shall develop an effective parallelizable technique for computing these eigenvalues and eigenvectors utilizing subspace iteration and preconditioning. Estimates will be provided which show that the preconditioned method converges linearly and uniformly in the matrix dimension when used with a uniform preconditioner under the assumption that the approximating subspace is close enough to the span of desired eigenvectors.

Stable Computation of Generalized Singular Values

Z. Drmač and E.R. Jessup
Department of Computer Science
University of Colorado
Boulder CO 80309–0430
zlatko@cs.colorado.edu
jessup@cs.colorado.edu

We study floating-point computation of the generalized singular value decomposition (GSVD) [8], [11] of a general matrix pair (A, B), where A and B are real matrices with the same numbers of columns. The GSVD is a powerful analytical and computational tool. For instance, the GSVD is an implicit way to solve the generalized symmetric eigenvalue problem $Kx = \lambda Mx$, where $K = A^T A$ and $M = B^T B$. Our goal is to develop stable numerical algorithms for the GSVD that are capable of computing the singular value approximations with the high relative accuracy that the perturbation theory says is possible [4], [6]. We assume that the singular values are *well-determined* by the data, i.e., that small *relative* perturbations δA and δB (pointwise rounding errors, for example) cause in *each* singular value σ of (A, B) only a small relative perturbation $|\delta\sigma|/\sigma$.

The first numerically attractive algorithms for the GSVD computation are based on reduction to a special case of the GSVD, namely the cosine–sine decomposition (CSD) [12]. These algorithms, devised by Stewart [12] and Van Loan [9], start with the QR factorization $\mathcal{G} \equiv [A^*, B^*]^* = QR$ and proceed with the CSD of the matrix Q. Another approach begins by applying Bai's and Zha's reduction algorithm [3] to extract a *regular pair* ((A, B) with A and B upper triangular and B nonsingular) from a general matrix pair. It then applies a Kogbetliantz–like algorithm [2], [10]. This latter combination is implemented as the LAPACK procedure `SGGSVD()` for GSVD computation [1]

All of the algorithms mentioned above are designed to use solely elementary orthogonal transformations. However, none of those algorithms is capable of computing the generalized singular values of (A, B) (eigenvalues of $K - \lambda M$) with high relative accuracy, even in cases where the singular values are well-determined by A and B. We discuss improvements to the LAPACK GSVD algorithm that guarantee high relative accuracy in the computed generalized singular values.

In future work, we intend to apply the developed technique to the product SVD (the SVD of $C = AB^T$). Since floating-point computation of the product AB^T can cause a loss of information in the singular values of C, a commonly used approach is to transform both A and B by a Jacobi (or Kogbetliantz) rotation scheme, see [7]. However, there is no theory that proves the relative accuracy properties of the algorithm [7]. Furthermore, the convergence of such a method may be slow, and the whole process is computationally expensive. We show

how to preprocess A and B to get an equivalent pair (A', B') for which the Jacobi SVD computation using the explicitly computed matrix $C' = A'(B')^T$ approximates the singular values with high relative accuracy, whenever possible. (Cf. [5].) Our algorithm has nice applications to the eigenvalue problem $KMx = \lambda x$ and to computation of system balancing transformations.

References

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. D. Croz, A. Greenbaum, S. Hammarling, A. McKenny, S. Ostrouchov, and D. Sorensen. *LAPACK users' guide.* SIAM, 1992.

[2] Z. Bai and J. Demmel. Computing the generalized singular value decomposition. LAPACK Working Note 46, University of Tennessee, Computer Science Department, May 1992.

[3] Z. Bai and H. Zha. A new preprocessing algorithm for the computation of the generalized singular value decomposition. *SIAM J. Sci. Stat. Comp.*, 14:1007–1012, 1993.

[4] J. Demmel and K. Veselić. Jacobi's method is more accurate than QR. *SIAM J. Matrix Anal. Appl.*, 13(4):1204–1245, 1992.

[5] Z. Drmač. Accurate computation of the product eigenvalue and singular value decompositions with application to system balancing transformations. Submitted to SIAM J. Numer. Anal., July 1995.

[6] Z. Drmač. The tangent against the cosine–sine algorithm for computing the generalized eigenvalue and singular value decompositions. Submitted to SIAM J. Numer. Anal., July 1995.

[7] M. T. Heath, A. J. Laub, C. C. Paige, and R. C. Ward. Computing the singular value decomposition of a product of two matrices. *SIAM J. Sci. Stat. Comp.*, 7:1147–1159, 1986.

[8] C. Van Loan. Generalizing the singular value decomposition. *SIAM J. Num. Anal.*, 13:76–83, 1976.

[9] C. Van Loan. Computing the CS and the generalized singular value decomposition. *Numer. Math.*, 46:479–491, 1985.

[10] C. C. Paige. Computing the generalized singular value decomposition. *SIAM J. Sci. Stat. Comp.*, 7:1126–1146, 1986.

[11] C. C. Paige and M. A. Saunders. Towards a generalized singular value decomposition. *SIAM J. Num. Anal.*, 18:398–405, 1981.

[12] G. W. Stewart. Computing the CS decomposition of a partitioned orthonormal matrix. *Numer. Math.*, 40:297–306, 1982.

Topic: *Navier-Stokes*	*Session Chair:* *Howard Elman*	*Room A*

10:30 - 11:00	I. Yavneh	Fast Multigrid Solution of the Advection Problem with Closed Characteristics
11:00 - 11:30	A.A. Lorber	Accelerated Solution of Non-Linear Flow Problems Using Chebyshev Iteration Polynomial Based Runge-Kutta Recursions
11:30 - 12:00	M. Murphy	Towards an Ideal Preconditioner for Linearized Navier-Stokes Problems

FAST MULTIGRID SOLUTION OF THE ADVECTION PROBLEM WITH CLOSED CHARACTERISTICS

Irad Yavneh, Technion---Israel Institute of Technology, Haifa, Israel.

Joint work with:
Cornelis H. Venner, University of Twente, Enschede, The Netherlands.
Achi Brandt, The Weizmann Institute of Science, Rehovot, Israel.

The numerical solution of the advection-diffusion problem in the inviscid limit with closed characteristics is studied as a prelude to an efficient high Reynolds-number flow solver. It is demonstrated by a heuristic analysis and numerical calculations that using upstream discretization with downstream relaxation-ordering and appropriate residual weighting in a simple multigrid V cycle produces an efficient solution process. We also derive upstream finite-difference approximations to the advection operator, whose truncation terms approximate "physical" (Laplacian) viscosity, thus avoiding spurious solutions to the homogeneous problem when the artificial diffusivity dominates the physical viscosity.

Accelerated Solution of Non-Linear Flow Problems using Chebyshev Iteration Polynomial Based RK Recursions

A. A. Lorber, G. F. Carey, S. W. Bova, C. H. Harlé

Computational Fluid Dynamics Laboratory
The University of Texas at Austin

1 Introduction

The connection between the solution of linear systems of equations by iterative methods and explicit time stepping techniques is used to accelerate to steady state the solution of ODE systems arising from discretized PDEs which may involve either physical or artificial transient terms. Specifically, a class of Runge-Kutta (RK) time integration schemes with extended stability domains has been used to develop recursion formulas which lead to accelerated iterative performance. The coefficients for the RK schemes are chosen based on the theory of Chebyshev iteration polynomials in conjunction with a local linear stability analysis.

We refer to these schemes as Chebyshev Parameterized Runge Kutta (CPRK) methods. CPRK methods of one to four stages are derived as functions of the parameters which describe an ellipse $\mathcal{E}$ which the stability domain of the methods is known to contain [5]. Of particular interest are two-stage, first-order CPRK and four-stage, first-order methods. It is found that the former method can be identified with any two-stage RK method through the correct choice of parameters. The latter method is found to have a wide range of stability domains, with a maximum extension of 32 along the real axis. Recursion performance results are presented below for a model linear convection-diffusion problem as well as non-linear fluid flow problems discretized by both finite-difference and finite-element methods.

The model problem is presented to demonstrate a novel "local" method that has been developed for applying the CPRK recursions. In the usual non-local or "global" method, the CPRK method chosen is the same for each point in the solution domain. For the local method, the CPRK recursion used at each point is allowed to vary, and is determined from a local von Neumann linear stability analysis applied to each point in the domain. This local method is conceptually similar to local point-relaxation techniques used for the iterative solution of stationary problems and local time-stepping techniques used in the integration of time-dependent PDEs.

2 CPRK Recursions

Here, Runge-Kutta recursions are used to solve to a steady-state the ODE system

$$\frac{d\boldsymbol{V}}{dt} = \boldsymbol{f}(t) - (\boldsymbol{A}(t)(\boldsymbol{V}(t)) \equiv \boldsymbol{R}(t, \boldsymbol{V}), \tag{1}$$

where $\boldsymbol{V}$ is the solution vector containing the discrete solution values and the matrix $\boldsymbol{A}$ and vector $\boldsymbol{f}$ result from the discretization of a governing PDE. We use the following "reduced

storage" form of the general explicit RK algorithm for the integration of (1) from time level n to $n+1$, with $\boldsymbol{V}(t^{(n)}) = \boldsymbol{V}^{(n)}$:

$$\begin{aligned} \boldsymbol{V}_0 &= \boldsymbol{V}^{(n)} \\ \boldsymbol{V}_m &= \boldsymbol{V}_0 + \varepsilon_m \Delta t^{(n)} \boldsymbol{R}\left(t^{(n)} + \varepsilon_{m-1}\Delta t^{(n)}, \boldsymbol{V}_{m-1}\right) \text{ for } m = 1 \text{ to } s \\ \boldsymbol{V}^{(n+1)} &= \boldsymbol{V}_s, \end{aligned} \tag{2}$$

where $\Delta t = t^{(n+1)} - t^{(n)}$ and s is the number of stages in the RK scheme [4].

The "classical" values of ε_m that give high orders of time accuracy are given by $\varepsilon_m = 1/(s-m+1)$. For steady-state solution, the choice of order of the integration method is not as significant as the maximum allowable value of Δt to maintain stability. Therefore, we choose the ε_m based on the equivalence between RK ODE recursions and Chebyshev polynomials applied to the iterative solution of linear systems. This increases the size of the associated stability domain at the expense of reduced time accuracy. Below, the notation RK(s)-p is used to refer to a s-stage RK scheme which, for (1) with for $\boldsymbol{A}$ and $\boldsymbol{f}$ constant, is of order accuracy p. For convenience, the values of ε_m used in the solutions below are presented in terms of values of γ_i, $i = 1, 2, \ldots, s$, where

$$\varepsilon_m = \frac{\gamma_{s-m+1}}{\gamma_{s-m}} \tag{3}$$

and $\gamma_0 \equiv 1$. The values γ_i are the coefficients of the stability polynomial $g_s(z) = (1 - \gamma_1 z + \gamma_2 z^2 - \cdots + \gamma_s z^s)$ where, for stability $|g_s(\Delta t \lambda_j)| < 1 \;\; \forall\, \lambda_j \in \sigma(\boldsymbol{A})$. For a RK($s$)-$p$ method, $\gamma_i = 1/(i!)$, $i = 1, 2, \ldots, p \leq s$.

3 Results

Linear Convection Diffusion

To illustrate the approach for a case with known analytic steady state solution, and test the local parameter solution, we first solve a model, 2D, linear convection diffusion equation of the form

$$-\frac{\partial \Phi}{\partial t} + \alpha(x,y)\left(\frac{\partial \Phi}{\partial x} + \frac{\partial \Phi}{\partial y}\right) = \frac{\partial^2 \Phi}{\partial x^2} + \frac{\partial^2 \Phi}{\partial y^2} + g(x,y) \tag{4}$$

where $\Phi = \Phi(t,x,y)$, $t \geq 0$.

In the discretization of (4), standard second-order central differencing is used for the spatial terms. The domain is discretized over the domain $\Omega = [0,1] \times [0,1]$ using a uniform 51×51 grid. We consider the steady-state solution of a modification of the model problem [6] given by

$$\begin{array}{ll} \alpha = -16Re\,(x-x^2)^2\,(y-y^2)\,(1-2y), & 0 \leq x,y \leq 1, \quad t \geq 0 \\ \phi(0,x,y) = 1, & 0 < x,y < 1 \\ \phi(t,x,0) = \phi(t,x,1) = 16(x-x^2)^2, & 0 \leq x \leq 1, \quad t \geq 0 \\ \phi(t,0,y) = \phi(t,1,y) = 0, & 0 \leq y \leq 1, \quad t \geq 0 \end{array} \tag{5}$$

where

$$g(x,y) = \alpha\left(\frac{\partial \Phi_{ex}}{\partial x} + \frac{\partial \Phi_{ex}}{\partial y}\right) - \left(\frac{\partial^2 \Phi_{ex}}{\partial x^2} + \frac{\partial^2 \Phi_{ex}}{\partial y^2}\right) \tag{6}$$

and the exact solution Φ_{ex} is

$$\Phi_{ex} = -8\left[4\left(1-2x\right)\left(x-x^2\right)\left(1-2y\right)\left(y-y^2\right) - 2\left(x-x^2\right)^2\left(1-6y+6y^2\right)\right], \qquad (7)$$

which is shown in Figure 1. The problem is of particular interest here because the convection

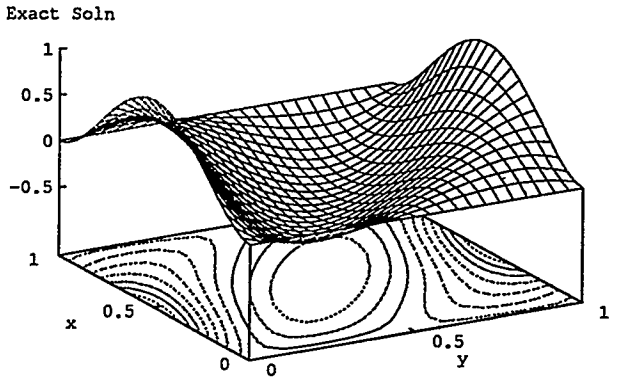


Figure 1: Surface and contour plots of the exact solution to the model problem.

coefficient α varies over the domain, producing a good test case for the local CPRK method discussed above. Convergence histories for the solution of the model problem using RK(4)-4, global CPRK(4)-1 and local CPRK(4)-1 recursions are shown in Figure 2. For the global

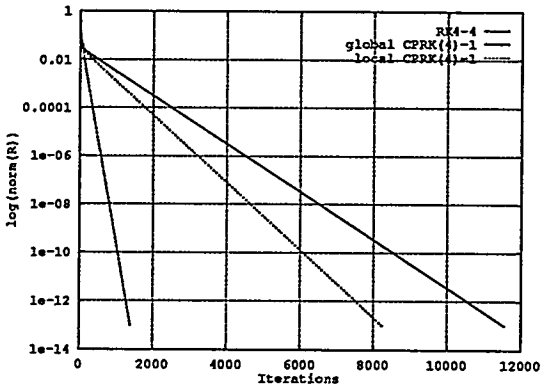


Figure 2: Convergence histories for $Re = 1200\sqrt{3}$ model problem using RK(4)-4 and both global and local CPRK(4)-1.

CPRK(4)-1 scheme, the number of iterations needed to reach a residual level of 10^{-13} was reduced by a factor of 1.40 over RK(4)-4. For local CPRK(4)-1, the reduction is 8.29 over RK(4)-4. For the global CPRK(4)-1 method the values of γ_i used were $\gamma_1 = 1$, $\gamma_2 = 0.46125$, $\gamma_3 = 1.0506\text{e-}2$ and $\gamma_4 = 1.3461\text{e-}2$.

Incompressible Viscous Flow in a Curved Channel

The second problem considered is $Re = 250$ incompressible viscous flow through a curved channel, for which the solution is shown in Figure 3. The flow is from left to right, with

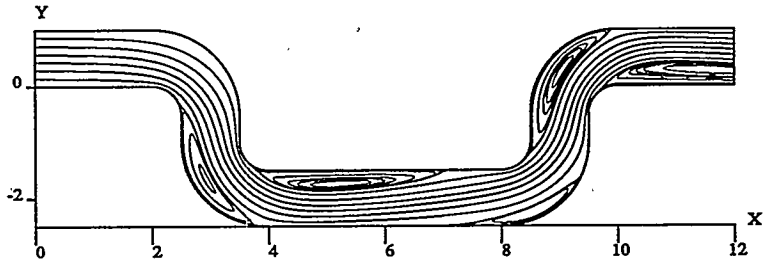


Figure 3: Stream function contours for curved-channel domain at $Re = 250$ over 512×64 grid.

uniform flow at the inlet. The flow is modeled using a non-linear, stream function vorticity formulation of the laminar, 2D, incompressible Navier-Stokes equations as described in [4]. The equations are discretized using standard second-order central differencing over a clustered 512×64 grid. An artificial transient formulation of the stream function Poisson and vorticity transport equations is used to allow solution via time integration. Convergence histories obtained using RK(4)-4 and CPRK(4)-1 recursions are shown in Figure 4. Separate

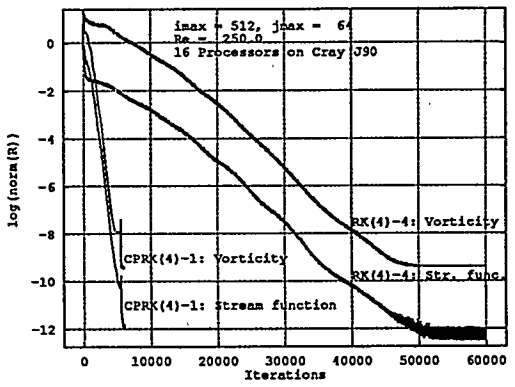


Figure 4: Convergence histories for 512×64 grid, $Re = 250$ computations using CPRK(4)-1 and RK(4)-4 recursions. In CPRK(4)-1 calculation, switch is made to RK(4)-1 at $n = 5500$.

convergence histories are shown for the decoupled stream function and vorticity equations. The use of the CPRK(4)-1 recursion reduces the number of iterations needed to reach a residual level of 10^{-7} by a factor of 8.83 over RK(4)-4. For the stream function equation, the values of γ_i used were $\gamma_1 = 1$, $\gamma_2 = 0.18080$, $\gamma_3 = 1.0707\text{e-}3$ and $\gamma_4 = 1.9972\text{e-}4$. For the vorticity equation, the values were $\gamma_1 = 1$, $\gamma_2 = 0.80158$, $\gamma_3 = 0.19815$ and $\gamma_4 = 6.4289\text{e-}2$.

Supercritical Flume Flow

Next, we consider a streamline-upwind Petrov-Galerkin (SUPG) finite-element discretization of the depth-integrated, 2D, shallow water equations [2, 1]. The flow problem solved is supercritical flow through a rectangular flume which transitions from a width of two feet to one foot. The length of the flume is 63.3 ft. The flume is smooth so that Manning's friction factor is 0.005, with a constant bed slope of 0.0125. The Froude number is 4.0, based on

the incoming depth of 0.1 ft. The mesh used has 4,585 nodes and 8,628 linear triangles. Computed depth contours are shown in Figure 5(a). The contour increment is 0.01868 ft. Figure 5(b) presents the depth along the centerline of the flume computed using RK(5)-5 and CPRK(5)-1 schemes. The two solutions are identical to the accuracy of the plot. Figure 6

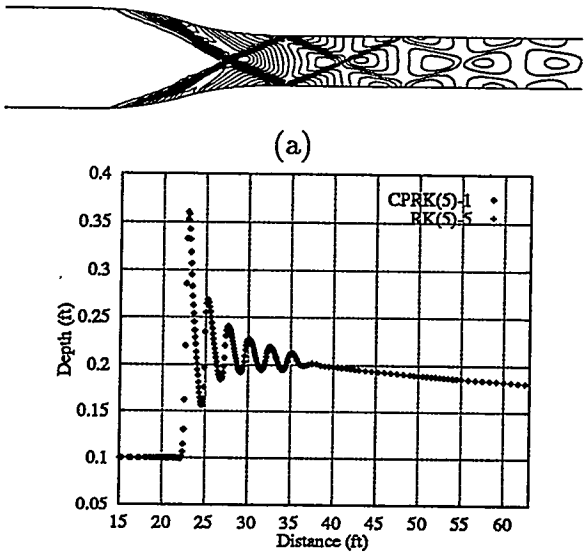


Figure 5: (a) Detail of depth contours for curved wall transition problem. (b) Depth profile along center of flume.

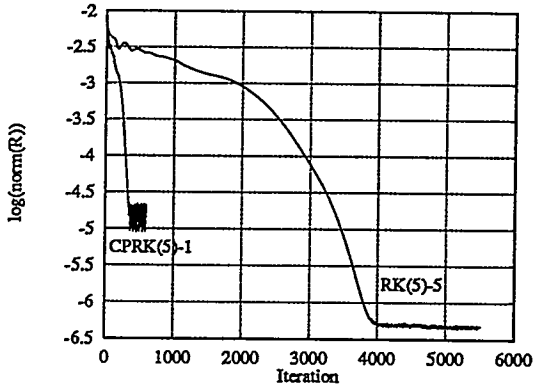


Figure 6: Convergence histories for flume solution.

contains the convergence histories for the problem using RK(5)-5 and CPRK(5)-1 recursions. For the CPRK(5)-1 scheme, the computation costs are reduced by a factor of approximately 8 over RK(5)-5. The values of γ_i used were $\gamma_1 = 1$, $\gamma_2 = 0.2262$, $\gamma_3 = 1.9121\text{e-}2$, $\gamma_4 = 6.7534\text{e-}4$ and $\gamma_5 = 8.4418\text{e-}6$.

Non-equilibrium Viscous Reacting Gas Flow

Finally, the CPRK(4)-1 scheme is tested in the computation of viscous reacting gas flow over a hemispherical blunt body [3]. The flow is modeled using a 2D, mixed finite-element/finite-volume discretizaton of the Navier-Stokes equations which are augmented to include chemical reactions. The incoming flow characteristics are: $M = 12.7$, $\rho = 1.6\times10^{-3}$ kg/m^3 and $T = 196$ K. The wall is non-catalytic, and the wall temperature is constant at $T_w = 300$ K. The mesh is structured with (60×60) nodes. Flow isotherms are shown in Figure 7, where the maximum temperature behind the shock reaches 7000 K. In Figure 8 the convergence

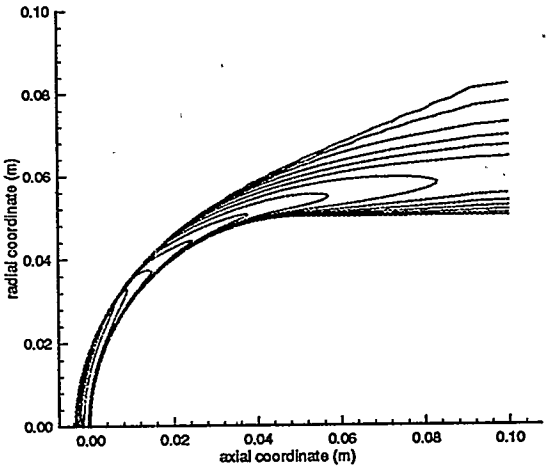


Figure 7: Translational temperature isolines for $M = 12.7$ blunt body solution.

histories are shown for solutions using CPRK(4)-1 and RK(4)-4 schemes. The mesh for this calculation is coarse and, in order to maintain convergence, solution averaging was done every 10 iterations, which induces oscillations in the residual. The number of iterations required to reach a residual level of 10^{-4} was reduced by approximately a factor of 2 through the use of the CPRK(4)-1 scheme. The values of γ_i used were $\gamma_1 = 1$, $\gamma_2 = 0.19738$, $\gamma_3 = 1.2976$e-2 and $\gamma_4 = 2.7033$e-4.

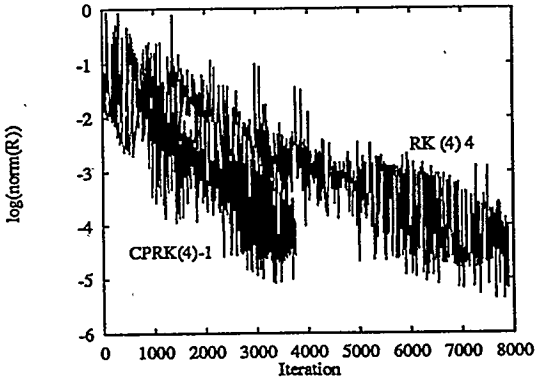


Figure 8: Residual history for blunt body solution.

4 Conclusions

Convergence results were presented for a model problem and three classes of flow problems representing a broad range of CFD applications: incompressible viscous flow, supercritical shallow water flow, and reacting hypersonic flow. In all cases the use of CPRK methods was seen to result in accelerated steady-state convergence. Finally, we note that in all the CFD applications, the programs were originally written to use standard RK integration schemes. Implementing the CPRK schemes simply involves changing the coefficients used in the integration algorithm. Hence, the new scheme can be easily retrofitted to existing software to yield dramatic improvements.

References

[1] BOVA, S., AND CAREY, G. F. An Entropy Variable Formulation and Applications for the Two-Dimensional Shallow Water Equations. In press: *International Journal for Numerical Methods in Fluids*, 1996.

[2] BOVA, S. W., LORBER., A. A., AND CAREY, G. F. An RK Iterative Recursion and SUPG Method for the Stationary Shallow Water Equations. In *Conference Proceedings, Coastal '95* (1995).

[3] HARLÉ, C., CAREY, G. F., VARGHESE, P. L., AND REINECKE, W. G. Analysis of High Speed Non-Equilibrium Chemically Reacting Gas Flows. Report IAT R-0083, Institute for Advanced Technology, June 1995.

[4] LORBER, A. A., AND CAREY, G. F. A Vector-Parallel Scheme for Navier Stokes Computations at Multi-Gigaflop Performance Rates. *International Journal for Numerical Methods in Fluids 21*, 6 (September 1995), 445–466.

[5] LORBER, A. A., CAREY, G. F., AND JOUBERT, W. D. ODE Recursions and Iterative Solvers for Linear Equations. *SIAM Journal on Scientific Computing 17*, 1 (January 1996).

[6] SPOTZ, W. F., AND CAREY, G. F. High-Order Compact Finite Difference Methods With Applications to Viscous Flows . Report 94-03, TICAM, 1994.

Towards an ideal preconditioner for linearized Navier-Stokes problems

Malcolm F. Murphy*

December 1995

Abstract

Discretizing certain linearizations of the steady-state Navier-Stokes equations gives rise to nonsymmetric linear systems with indefinite symmetric part. We show that for such systems there exists a block diagonal preconditioner which gives convergence in three GMRES steps, independent of the mesh size and viscosity parameter (Reynolds number). While this "ideal" preconditioner is too expensive to be used in practice, it provides a useful insight into the problem. We then consider various approximations to the ideal preconditioner, and describe the eigenvalues of the preconditioned systems. Finally, we compare these preconditioners numerically, and present our conclusions.

1 Introduction

One way of linearizing the steady state Navier-Stokes equations

$$\left.\begin{aligned} -\nu\nabla^2 \boldsymbol{u} + \boldsymbol{u}.\nabla\boldsymbol{u} + \nabla p &= \boldsymbol{f} \\ \nabla.\boldsymbol{u} &= 0 \end{aligned}\right\} \text{ in some domain } \Omega \qquad (1)$$

(together with suitable boundary conditions) is to replace the convective term $\boldsymbol{u}.\nabla\boldsymbol{u}$ with an advective term $\boldsymbol{w}.\nabla\boldsymbol{u}$, where the applied field $\boldsymbol{w}$ is known. This gives rise to the Oseen equations

$$\left.\begin{aligned} -\nu\nabla^2 \boldsymbol{u} + \boldsymbol{w}.\nabla\boldsymbol{u} + \nabla p &= \boldsymbol{f} \\ \nabla.\boldsymbol{u} &= 0 \end{aligned}\right\} \text{ in } \Omega \qquad (2)$$

*School of Mathematics, University of Bristol, University Walk, Bristol, BS8 1TW, England. Email Malcolm.Murphy@bristol.ac.uk

These equations also arise naturally when applying a fixed-point iteration to the Navier-Stokes equations (1), in which case we have further information about $\boldsymbol{w}$; it will satisfy the boundary conditions and the weak form of the zero divergence condition. However, we do not require that $\boldsymbol{w}$ has these properties in this paper.

Mixed finite element methods for solving (2) seek an approximate solution $(\boldsymbol{u}^h, p^h)$, from some finite dimensional space $V^h \times Q^h$, to a weak formulation of (2). If $\{\boldsymbol{v}_i\}_{i=1}^N$ and $\{q_i\}_{i=1}^M$ are bases for V^h and Q^h respectively, then we can write

$$\boldsymbol{u}^h = \sum_{i=1}^{N} u_i \boldsymbol{v}_i, \qquad p^h = \sum_{i=1}^{M} p_i q_i$$

so that the finite element approximation reduces to a system of algebraic equations:

$$\begin{bmatrix} \nu A + C & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{p} \end{bmatrix} = \begin{bmatrix} \boldsymbol{g} \\ \boldsymbol{0} \end{bmatrix} \tag{3}$$

where $A \in \mathbb{R}^{N \times N}$ represents diffusion $(-\nabla^2)$, and is therefore a symmetric and positive definite matrix; $B \in \mathbb{R}^{M \times N}$ is a discrete representation of the divergence operator $-\nabla .$; and $C \in \mathbb{R}^{M \times N}$ represents the advection term—by judiciously selecting the weak formulation of (2) [4, §4.1] we can ensure that C is skew-symmetric, then $\boldsymbol{z}^H C \boldsymbol{z}$ is strictly imaginary for all $\boldsymbol{z} \in \mathbb{C}$.

Typically, the boundary conditions will only specify values for the velocity $\boldsymbol{u}$: in these cases both equations (1) and and (2) only determine the pressure up to a constant, which means that B will have rank $M - 1$. To ensure that B is full rank, an extra condition is required, such as prescribing the value of p at some point in the domain, or requiring that the mean of the pressure over Ω be zero. We also have the following spectral information:

- $h^2 \sim \lambda_{min}(A),\ \lambda_{max}(A) \sim 1$, [7, p.637],
- $h \sim \sigma_{min}(B),\ \sigma_{max}(B) \sim 1$, [7, equations (1.9) and (1.10)],
- For a div-stable element [4, p.12], there exist γ, Γ such that

$$\gamma^2 \leq \frac{\boldsymbol{p}^T (B A^{-1} B^T) \boldsymbol{p}}{\boldsymbol{p}^T Q \boldsymbol{p}} \leq \Gamma^2 \quad \forall \boldsymbol{p} \in \mathbb{R}^M \backslash \{\boldsymbol{0}\} \tag{4}$$

  i.e. the matrix $BA^{-1}B^T$ is spectrally equivalent to the pressure mass matrix Q, whose eigenvalues are known to be $\mathcal{O}(h^2)$ for a 2-dimensional problem.

- The eigenvalues of $A^{-1}C$ are bounded in modulus by a constant δ independent of the mesh parameter h [1], i.e.

$$\rho(A^{-1}C) \leq \delta \tag{5}$$

For convenience, we write

$$\mathcal{A} = \begin{bmatrix} F & B^T \\ B & 0 \end{bmatrix}, \qquad F = \nu A + C.$$

2 Krylov subspace methods and preconditioning

Krylov subspace methods are attractive candidates for soving these linear systems, especially when M and N are large. Typically, a preconditioner is required to transform the original system into one for which the convergence of the method is faster. The convergence of Krylov subspace methods is governed by the eigenvalues and eigenvectors of the (preconditioned) system—one convergence estimate for GMRES applied to a matrix $G = V\Lambda V^{-1}$ is given [5] by

$$\frac{\|r_n\|}{\|r_0\|} \leq \kappa(V) \min_{p_n \in \mathcal{P}_n,\ p_n(0)=1)} \max_{\lambda \in \Lambda} |p_n(\lambda)| \tag{6}$$

where r_n is the nth residual, $\mathcal{P}_n$ is the set of polynomials of degree n, and $\kappa(V)$ is the condition number of the eigenvector matrix V.

Unfortunately, estimates such as (6) are rarely descriptive in practice, making it difficult to identify exactly what one is trying to achieve when preconditioning nonsymmetric systems.

In this paper, we consider a particular class of preconditioners for the matrix $\mathcal{A}$, namely block diagonal preconditioners of the form

$$\mathcal{M} = \begin{bmatrix} F & 0 \\ 0 & M_L \end{bmatrix}$$

We make this choice as it follows naturally from earlier work on Stokes equations [6, 7], which can be thought of as a special case of equations (2) with $\boldsymbol{w} \equiv \mathbf{0}$ and $\nu = 1$. However, we note that it may not be a trivial matter to invert F; in the case of the Stokes equations the matrix F reduces to a discrete Laplacian, for which there are various "fast" solution techniques available, but for the Oseen equations the matrix F represents an advection-diffusion operator, which is more difficult to deal with. We recognize that this is an issue, but we assume here that it is possible to form the action of F^{-1}, via some sort of advection-diffusion solver.

Having decided to use F as the upper block of the preconditioner, we are concerned with what to use for the lower block, i.e. what is a "good" choice for M_L?

2.1 The "ideal" block diagonal preconditioner

Choosing $M_L = BF^{-1}B^T$ yields the following preconditioner:

$$\mathcal{M}_S = \begin{bmatrix} F & 0 \\ 0 & BF^{-1}B^T \end{bmatrix}$$

Suppose that λ is an eigenvalue of $\mathcal{M}_S^{-1}\mathcal{A}$, with corresponding eigenvector $[\boldsymbol{u}^T, \boldsymbol{p}^T]^T$. Then

$$\begin{bmatrix} I & F^{-1}B^T \\ (BF^{-1}B^T)^{-1}B & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{p} \end{bmatrix} = \lambda \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{p} \end{bmatrix}$$

$$\left.\begin{aligned} \boldsymbol{u} + F^{-1}B^T\boldsymbol{p} &= \lambda\boldsymbol{u} \\ (BF^{-1}B^T)^{-1}B\boldsymbol{u} &= \lambda\boldsymbol{p} \end{aligned}\right\} \tag{7}$$

We can eliminate $\boldsymbol{u}$ from these equations, to obtain

$$(\lambda^2 - \lambda - 1)BF^{-1}B^T\boldsymbol{p} = \boldsymbol{0} \tag{8}$$

so that either $\lambda^2 - \lambda - 1 = 0$ or $BF^{-1}B^T\boldsymbol{p} = \boldsymbol{0}$. In the former case, we have that $\lambda = \frac{1}{2}(1 \pm \sqrt{5})$, and the corrensponding eigenvectors must satisfy

$$BF^{-1}B^T\boldsymbol{p} = (\lambda - 1)B\boldsymbol{u} \tag{9}$$

which has M independent solutions for each value of λ. In the latter case, $BF^{-1}B^T\boldsymbol{p} = \boldsymbol{0}$ implies that $\boldsymbol{p} = \boldsymbol{0}$, then equations (7) imply that $\lambda = 1$ and $B\boldsymbol{u} = \boldsymbol{0}$. This second equation has $N - M$ independent solutions, so that we have $N - M$ eigenvectors of the form $\begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{0} \end{bmatrix}$ associated with the eigenvalue 1.

Thus, we see that the preconditioned system $\mathcal{M}_S^{-1}\mathcal{A}$ has three distinct eigenvalues, and a complete set $(N - M + M + M)$ of independent eigenvectors. We can construct the characteristic polynomial:

$$P(x) = (x - 1)(x^2 - x - 1)$$

and observe that $P(0) = 1$.

Recall that, when applying GMRES to a matrix G, the nth residual r_n satisfies

$$r_n = p_n(G)r_0$$

where $p_n \in \mathcal{P}_n,\ p_n(0) = 1$ and

$$\|r_n\| \leq \|q_n(G)r_0\| \quad \forall q_n \in \mathcal{P}_n,\ q_n(0) = 1$$

By the Cayley-Hamilton theorem, we have that $P(\mathcal{M}_S^{-1}\mathcal{A}) = 0$, implying that GMRES applied to the matrix $\mathcal{A}$ with preconditioner $\mathcal{M}_S$, terminates after three steps. Of course, $\mathcal{M}_S$ is too expensive to be used in practice, but it is "ideal" in the sense of minimizing the number of GMRES iterations required.

Remark 1 *We could also take $M_L = \theta BF^{-1}B^T$, changing the values of the eigenvalues in equation (9). This makes no difference, except in the case $\theta = -\frac{1}{4}$, when the preconditioned system has a repeated eigenvalue $\lambda = \frac{1}{2}$. In this case, the system has only two distinct eigenvalues, but does not have a complete set of eigenvectors, so the Cayley-Hamilton theorem cannot be applied as above, and we cannot conclude that GMRES would converge in two iterations. In fact, three iterations are still required.*

2.2 Lower blocks of the form $BX^{-1}B^T$

The Schur complement $BF^{-1}B^T$ is too expensive to be used in practice, even under the assumption that we have some method of solving systems based on the advection-diffusion matrix F. However, since we know what the best choice for M_L would be, we can consider approximating it by some (hopefully cheaper) alternative. In particular, we consider a lower block of the form $M_L = BX^{-1}B^T$, giving a preconditioner of the form

$$\mathcal{M}_X = \begin{bmatrix} F & 0 \\ 0 & BX^{-1}B^T \end{bmatrix}$$

where the matrix X is to be determined. The idea is to choose X so that we can either solve linear systems with coefficient matrix $BX^{-1}B^T$ cheaply, or construct a good approximation to $BX^{-1}B^T$ which we can solve cheaply.

Analysis similar to that of section 2.1 shows that if λ is an eigenvalue of $\mathcal{M}_X^{-1}\mathcal{A}$, then either $\lambda = 1$, or

$$\lambda(\lambda - 1) = \frac{x^H x}{x^H X^{-1} F x} \quad \text{for some } x \in \mathbb{C}^N \tag{10}$$

Equivalently,

$$\eta = \frac{1}{\lambda(\lambda-1)} = \frac{\boldsymbol{x}^H X^{-1} F \boldsymbol{x}}{\boldsymbol{x}^H \boldsymbol{x}} \quad \text{for some } \boldsymbol{x} \in \mathbb{C}^N$$

Now, we examine some particular choices for X

1. $X = A$. In this case

$$\eta = \frac{\boldsymbol{x}^H(\nu I + A^{-1}C)\boldsymbol{x}}{\boldsymbol{x}^H \boldsymbol{x}} = \nu + \frac{\boldsymbol{x}^H A^{-1} C \boldsymbol{x}}{\boldsymbol{x}^H \boldsymbol{x}} \tag{11}$$

   The right hand side of (11) describes a line segment in the complex plane, with real part ν, and imaginary part bounded in modulus by a mesh-independent constant, so the eigenvalues of the preconditioned system are contained in a region that is independent of the mesh parameter h, but that does depend on ν.

   Here, we have a cheap and easily invertible approximation available, since equation (4) tells us that the pressure mass matrix Q is spectrally equivalent to $BA^{-1}B^T$.

2. $X = \nu A$. In this case

$$\eta = \frac{\boldsymbol{x}^H(I + \frac{1}{\nu}A^{-1}C)\boldsymbol{x}}{\boldsymbol{x}^H \boldsymbol{x}} = 1 + \frac{1}{\nu}\frac{\boldsymbol{x}^H A^{-1} C \boldsymbol{x}}{\boldsymbol{x}^H \boldsymbol{x}} \tag{12}$$

   Again, the right hand side of (12) describes a line segment, in this case with real part 1, and imaginary part proportional to ν^{-1}. This estimate does not depend on h, but the ν-dependence differs from that in the case $X = A$. Here, $\frac{1}{\nu}Q$ can be used as an approximation to $B(\nu A)^{-1}B^T$. The use of this scaled mass matrix has been previously discussed in [2], where mesh independent estimates for the eigenvalues were also given.

3. $X = C$. In this case there is a complication: the matrix C will typically be singular, since the boundary conditions for the original problem are not appropriate for an advection problem. One way of avoiding this is to apply a different set of boundary conditions to the advection matrix C to ensure that C^{-1} exists. Assuming that we can invert C, then we have

$$\eta = \frac{\boldsymbol{x}^H(I + \nu C^{-1}A)\boldsymbol{x}}{\boldsymbol{x}^H \boldsymbol{x}} = 1 + \nu\frac{\boldsymbol{x}^H C^{-1} A \boldsymbol{x}}{\boldsymbol{x}^H \boldsymbol{x}} \tag{13}$$

   The right hand side of (13) again describes a line segment, this time with real part 1, and imaginary part bounded (in modulus) below

by ν/δ, and above by ν/h. It is conceivable that we can construct a matrix with a spectrum similar to that of $BC^{-1}B^T$, which will be reasonably cheap to invert, although we have not done so here.

4. $X = \tau I + C$. For this choice of X, we choose an equivalent (but more convenient) expression for $\lambda(\lambda - 1)$, viz.

$$\begin{aligned} \lambda(\lambda-1) &= \frac{y^H X y}{y^T F y} \text{ for some } y \in \mathbb{C}^N \\ &= \frac{\tau y^H y + y^H C y}{\nu y^H A y + y^H C y} \end{aligned}$$

Without loss of generality, we assume that $y^H y = 1$, and let $i\zeta = y^H C y$, $\alpha = y^H A y$. Then

$$\lambda(\lambda-1) = \frac{\alpha\tau\nu + \zeta^2}{\nu^2\alpha^2 + \zeta^2} + i\frac{\zeta(\alpha\nu - \tau)}{\nu^2\alpha^2 + \zeta^2} \tag{14}$$

The question now is what to choose for τ. If we take $\tau > \rho(C)$, e.g. by taking $\tau = \|C\|_1$, then we can write

$$(\tau I + C)^{-1} = \frac{1}{\tau}\left\{I - \frac{1}{\tau}C + \frac{1}{\tau}C^2 - \frac{1}{\tau}C^3 + \frac{1}{\tau}C^4 - + \ldots\right\}$$

A partial sum of this series could then be used as an approximation to $(\tau I + C)^{-1}$. However, a more natural choice would seem to be $\tau = \nu$, in which case, examining the right hand side of (14), we can see that

- The real part of $\lambda(\lambda - 1)$ is bounded away from 0 by a constant independent of h and ν.
- The real part of $\lambda(\lambda-1)$ is bounded above by a quantity proportional to $(\nu/h)^2$.
- The imaginary part is proportional to ν for small ν.

Here, we don't have a cheap approximation to $B(\tau I+C)^{-1}B^T$, but we note that its spectrum is contained in a line segment in the complex plane. For such matrices, the solution of the the polynomial minimax problem contained in the convergence estimate (6) is explicitly known [3], and we can conclude that the work required in solving systems with coefficient matrix $B(\tau I+C)^{-1}B^T$ grows linearly as $\tau \to 0$. Thus, taking $\tau = \|C\|_1$ implies a constant (as ν varies) amount of work to do the preconditioning, but taking $\tau = \nu$ implies that the work required to precondition grows as ν gets small. If, however, this leads to fewer (hopefully constant) outer GMRES iterations, then this might be an acceptable cost.

3 Numerical Experiments

We consider a lid driven cavity problem on a square domain $\Omega = [-1,1] \times [-1,1]$. We use a regular grid of square $Q2-Q1$ (Taylor-Hood) elements [4, p.34], and choose a hydrostatic pressure level by eliminating one of the pressure variables from the system—effectively setting it to zero. The applied field w is a unidirectional flow in the x-direction. While this choice for w may not be consistent with the full Navier-Stokes equations, it is a valid choice for the Oseen equations, and it makes it easier to specify boundary conditions for the advection problem that guarantee that C is invertible—in particular, we enforce Dirichlet conditions on the top boundary when we wish to form C^{-1}. All computations were performed using Matlab 4.2a on a Silicon Graphics Indigo 2 workstation.

Although no results are presented here for $\mathcal{M}_S$, we have observed numerically that using this preconditioner, GMRES does indeed converge in three iterations, and the norm of the residual is of the order of the unit roundoff.

Table 1 shows the performance of using $\mathcal{M}_X$, for $X = \nu A$, $X = C$, $X = \nu I + C$, and $X = \|C\|_1 I + C$ on an 8×8 grid of elements for various values of the ν, and Table 2 shows the same information for a 12×12 grid. The number in the table is the number of orthogonal directions GMRES computes before the norm of the residual was less than 10^{-7}. In all cases, the initial guess was the zero vector. Although no figures are given here, choosing $X = A$ was consistently worse than choosing $X = \nu A$. Table 3 compares, on a 16×16 grid, the effect of choosing different values for τ.

We see that $X = \nu A$ is a better choice than $X = C$ when ν is large, but as ν gets smaller, $X = C$ is to be preferred. This is perhaps as one might expect, since F is dominated by νA when ν is large, and by C when ν is small. We see that, for $X = \tau I + C$, the choice $\tau = \nu$ is to be preferred to $\tau = \|C\|_1$, in terms of the number of GMRES iterations required. Indeed, the choice $\tau = \|C\|_1$ gets progressively worse as both ν and h get smaller, whereas choosing $\tau = \nu$ appears to give a mesh independent preconditioner. Table 3 also suggests that the number of iterations required could possibly be bounded independent of ν, although further investigation is needed.

4 Conclusions

We have identified the block diagonal preconditioner, $\mathcal{M}_S$, which gives the best performance in terms of the number of GMRES iterations required.

ν	$\frac{1}{\nu}BA^{-1}B^T$	$BC^{-1}B^T$	$B(\nu I+C)^{-1}B^T$	$B(\|C\|I+C)^{-1}B^T$
0.5	19	161	49	50
0.1	51	159	43	39
0.05	85	145	33	33
0.01	155	81	43	75
0.005	159	63	51	123
0.001	161	57	69	161

Table 1: 8×8 grid of $Q_2 - Q_1$ elements; 578+81=659 unknowns.

ν	$\frac{1}{\nu}BA^{-1}B^T$	$BC^{-1}B^T$	$B(\nu I+C)^{-1}B^T$	$B(\|C\|I+C)^{-1}B^T$
0.5	19	329	63	67
0.1	51	329	57	55
0.05	87	309	45	47
0.01	245	137	39	53
0.005	275	101	49	117
0.001	319	69	73	295

Table 2: 12×12 grid of $Q_2 - Q_1$ elements; 1250+169=1419 unknowns.

ν	$B(\nu I+C)^{-1}B^T$	$B(\|C\|I+C)^{-1}B^T$
0.5	77	81
0.1	71	69
0.05	57	59
0.01	39	57
0.005	45	89
0.001	71	347
0.0005	91	455
0.0001	91	457

Table 3: 16×16 grid of $Q_2 - Q_1$ elements

Although too costly to use in practice, this preconditioner gives us a reference point from which to assess other, more practical, preconditioners, and allows us to identify what features we want in a preconditioner for the Oseen equations. We believe that identifying $\mathcal{M}_S$ is an important step in understanding the problem of preconditioning these equations.

Acknowledgments

This research has been carried out under the funding of an EPSRC studentship. The computing facilities used were provided under an EPSRC distributed computing grant. I am grateful to my Ph.D. advisor, Andy Wathen, for his guidance while this research has been carried out.

References

[1] H. C. Elman and M. H. Schultz, *Preconditioning by fast direct methods for nonself-adjoint noseparable elliptic equations*, SIAM Journal of Numerical Analysis, 23 (1986), pp. 44–57.

[2] H. C. Elman and D. J. Silvester, *Fast nonsymmetric iteration and preconditioning for Navier-Stokes equations*, report 263, Manchester Centre for Computational Mathematics, 1995.

[3] R. Freund and S. Ruscheweyh, *On a class of Chebyshev approximation problems which arise in connection with a conjugate gradient type method*, Numerische Mathematik, 48 (1986), pp. 525–542.

[4] M. D. Gunzburger, *Finite element methods for viscous incompressible flows*, Academic Press, inc., 1989.

[5] N. M. Nachtigal, S. C. Reddy, and L. N. Trefethen, *How fast are nonsymmetric matrix iterations?*, SIAM Journal of Matrix Analysis and Applications, 13 (1992), pp. 778–795.

[6] D. Silvester and A. Wathen, *Fast iterative solution of stabilised Stokes systems Part II: Using general block preconditioners*, SIAM Journal of Numerical Analysis, 31 (1994), pp. 1352–1367.

[7] A. Wathen and D. Silvester, *Fast iterative solution of stabilised Stokes systems Part I: Using simple diagonal preconditioners*, SIAM Journal of Numerical Analysis, 30 (1993), pp. 630–649.

Topic:	*Session Chair:*	*Room B*
Krylov Methods	*M. Gutknecht*	

10:30 - 11:00	M. Gutknecht	Look-Ahead Procedures for Lanczos-Type Product Methods Based on Three-Term Recurrences
11:00 - 11:30	E. Gallopoulos	Solving Modified Systems with Multiple Right-Hand Sides
11:30 - 12:00		

LOOK-AHEAD PROCEDURES FOR LANCZOS-TYPE PRODUCT METHODS BASED ON THREE-TERM RECURRENCES

MARTIN H. GUTKNECHT* AND KLAUS J. RESSEL†

Abstract. Lanczos-type product methods for the solution of large sparse non-Hermitian linear systems either square the Lanczos process or combine it with a local minimization of the residual. They inherit from the underlying Lanczos process the danger of breakdown. For various Lanczos-type product methods that are based on the Lanczos three-term recurrence, look-ahead versions are presented, which avoid such breakdowns or near breakdowns with a small computational overhead. Different look-ahead strategies are discussed and their efficiency is demonstrated in several numerical examples.

Key words. Lanczos-type product methods, look-ahead steps, iterative methods, non-Hermitian matrices, sparse linear systems

AMS subject classifications. 65F15, 65F10

1. Introduction. Lanczos-type product methods (LTPMs) like (Bi)CGS [9], BiCGStab [10], BiCGStab2 [5], and BiCGStab(ℓ) [8] are among the most efficient methods for solving large systems of equations $Ax = b$ with nonsymmetric sparse system matrix $A \in \mathbb{C}^{N\times N}$. These methods either square the Lanczos process or combine it with a minimization of the residual. Compared to the plain Lanczos process, LTPMs have the advantage of converging roughly twice as fast and of not requiring a routine for applying the adjoint system matrix A^H to a vector. Nonetheless, they inherit from the underlying Lanczos process the danger of breakdown. Although exact breakdowns are very rare in practice, it has been observed that near-breakdowns can slow down or even prevent convergence [3]. Look-ahead techniques for the plain Lanczos process [3], [4], [7] allow to avoid this problem with very small overhead. However, the look-ahead procedures that have so far been proposed for CGS or other LTPMs are either limited to exact breakdowns or are extremely complicated and have considerable overhead [1], [2]. In the following work look-ahead procedures are derived for a general class of LTPMs that make use of the Lanczos three-term recurrences and the Lanczos biorthogonalization (BiO) algorithm. Different look-ahead criteria are discussed and a complete analysis of the computational overhead in terms of matrix vector multiplications (MVs), and inner products (IPs) is presented. Various numerical examples demonstrate that the proposed look-ahead procedures work very well.

2. The Look-Ahead Lanczos Process. We recall that the BiO algorithm generates a pair of finite sequences, $\{\widetilde{y}_n\}_{n=0}^{\nu}$, $\{y_n\}_{n=0}^{\nu}$, of *left* and *right Lanczos vectors*, such that

$$\begin{aligned} y_n \in \mathcal{K}_{n+1} &:= \text{span}\,\{y_0, Ay_0, \ldots, A^n y_0\}, \\ \widetilde{y}_n \in \mathcal{L}_{n+1} &:= \text{span}\,\{\widetilde{y}_0, A^H\widetilde{y}_0, \ldots, (A^H)^n\widetilde{y}_0\}, \end{aligned} \tag{2.1}$$

and

$$\widetilde{y}_n \perp \mathcal{K}_n, \quad y_n \perp \mathcal{L}_n. \tag{2.2}$$

* **Speaker:** Swiss Center for Scientific Computing (SCSC), ETH Zürich, ETH-Zentrum, CH-8092 Zürich, Switzerland (mhg@scsc.ethz.ch).

† Swiss Center for Scientific Computing (SCSC), ETH Zürich, ETH-Zentrum, CH-8092 Zürich, Switzerland (kjr@scsc.ethz.ch).

This sequence of pairs of Lanczos vectors can be constructed by the following three-term recursion

$$
\begin{aligned}
y_{n+1} &= \left(Ay_n - y_n\alpha_n - y_{n-1}\beta_n\right)/\gamma_n \\
\widetilde{y}_{n+1} &= \left(A^H\widetilde{y}_n - \widetilde{y}_n\overline{\alpha}_n - \widetilde{y}_{n-1}\overline{\beta}_n\right)/\overline{\gamma}_n,
\end{aligned}
\tag{2.3}
$$

with coefficients α_n and β_n that are determined from the orthogonality condition (2.2) and freely choosable nonvanishing scale factors γ_n. The Lanczos vectors can then be written in the form

$$
y_n = \rho_n(A)y_0, \qquad \widetilde{y}_n = \overline{\rho}_n(A^H)\widetilde{y}_0,
\tag{2.4}
$$

where ρ_n denotes the n-th Lanczos polynomial. Since we aim here at LTPMs, we consider for the Krylov spaces $\mathcal{L}_n$ more general basis vectors of the form

$$
\widetilde{z}_n = \overline{\tau}_n(A^H)\widetilde{z}_0,
\tag{2.5}
$$

with arbitrarily chosen polynomials τ_n of exact degree n and $\widetilde{z}_0 := \widetilde{y}_0$. In general, $\widetilde{z}_n \perp \mathcal{K}_n$ will no longer hold, but $y_n \perp \mathcal{L}_n$ can be still obtained by choosing the coefficients α_n and β_n in (2.3) in the following way:

$$
\beta_n = \frac{\langle \widetilde{z}_{n-1}, Ay_n\rangle}{\langle \widetilde{z}_{n-1}, y_{n-1}\rangle}, \qquad \alpha_n = \frac{\langle \widetilde{z}_n, Ay_n\rangle - \langle \widetilde{z}_n, y_{n-1}\rangle\,\beta_n}{\langle \widetilde{z}_n, y_n\rangle}.
\tag{2.6}
$$

Clearly, the recursive process terminates with $y_\nu = 0$ or $\widetilde{z}_\nu = 0$, or it breaks down with $\delta_\nu^\tau := \langle \widetilde{z}_\nu, y_\nu\rangle = 0$ and $y_\nu \neq 0$, $\widetilde{z}_\nu \neq 0$. The look-ahead Lanczos process [6, §9], [3] overcomes a breakdown if curable, i.e., if for some k, $\langle \widetilde{z}_\nu, A^k y_\nu\rangle \neq 0$, by enforcing the orthogonality condition (2.2) only partially for a couple of steps. To be more precise, in the look-ahead Lanczos process the the condition $y_n \perp \mathcal{L}_n$ is replaced by

$$
y_n \perp \mathcal{L}_{n_j}, \quad \text{if } n \geq n_j \ (j = 0, \ldots, J),
\tag{2.7}
$$

where $0 = n_0 < n_1 < \cdots < n_J = \nu \leq N$ is the subsequence of indices of *regular* Lanczos vectors y_{n_j}, while for the other indices the vectors are *inner* vectors. Thus, the Gramian matrix $D := (\langle \widetilde{z}_m, y_n\rangle)$ becomes block-lower triangular instead of triangular. Forming the blocks

$$
Y_j := \left[y_{n_j}, y_{n_j+1}, \ldots, y_{n_{j+1}-1}\right], \quad \widetilde{Z}_j := \left[\widetilde{z}_{n_j}, \widetilde{z}_{n_j+1}, \ldots, \widetilde{z}_{n_{j+1}-1}\right],
\tag{2.8}
$$

and denoting their length in the following by $h_j := n_{j+1} - n_j$, we can express relation (2.7) by

$$
\widetilde{Z}_l^H Y_j = \begin{cases} D_j & \text{if } l = j, \\ 0 & \text{if } l < j. \end{cases}
\tag{2.9}
$$

In the look-ahead case the Lanczos vectors $\{y_n\}$ can be generated by the following recursions [6, §9], [3]. If y_{n+1} is an inner vector we have

$$
y_{n+1} = \left(Ay_n - \widehat{Y}_j\alpha_n - Y_{j-1}\beta_n\right)/\gamma_n \qquad (n_j < n+1 < n_{j+1}),
\tag{2.10}
$$

where $\widehat{Y}_j := \left[y_{n_j}, \ldots, y_n\right]$ denotes the not yet fully completed j-block. The coefficient vector α_n can be chosen arbitrarily, whereas for the coefficient vector β_n the orthogonality condition (2.9) leads to

$$
\beta_n = D_{j-1}^{-1}\widetilde{Z}_{j-1}^H Ay_n.
\tag{2.11}
$$

On the other hand, if y_{n+1} is a regular vector, we have

$$y_{n+1} = \left(Ay_n - Y_j\alpha_n - Y_{j-1}\beta_n\right)/\gamma_n \qquad (n+1 = n_{j+1}), \tag{2.12}$$

where α_n is now also determined by the orthogonality condition (2.9), thus

$$\alpha_n = D_j^{-1}\left(\widetilde{Z}_j^H A y_n - \widetilde{Z}_j^H Y_{j-1}\beta_n\right). \tag{2.13}$$

3. Lanczos-Type Product Methods (LTPMs). By introducing the *product vectors* w_n^l and *product iterates* x_n^l:

$$w_n^l := \tau_l(A)y_n := \tau_l(A)\rho_n(A)y_0, \qquad b\dot{\rho}_n^l - Ax_n^l := w_n^l, \tag{3.1}$$

with $\dot{\rho}_n^l := \tau_l(0)\rho_n(0)$, and by rewriting the inner products in (2.6) in terms of product vectors, i.e.,

$$\beta_n = \frac{\langle \widetilde{z}_0, Aw_n^{n-1}\rangle}{\langle \widetilde{z}_0, w_{n-1}^{n-1}\rangle}, \qquad \alpha_n = \frac{\langle \widetilde{z}_0, Aw_n^n\rangle - \langle \widetilde{z}_0, w_{n-1}^n\rangle\,\beta_n}{\langle \widetilde{z}_0, w_n^n\rangle}, \tag{3.2}$$

LTPMs are derived. Different choices of the polynomials $\tau_n(\zeta)$ lead to different LTPMs. In BiOStab, the three-term version of Van der Vorst's BiCGStab [10], the polynomials $\tau_n(\zeta)$ are chosen as a product of polynomials of degree 1:

$$\tau_0(\zeta) \equiv 1, \qquad \tau_{n+1}(\zeta) = (1 - \chi_n\zeta)\tau_n(\zeta) \quad \text{for } n \geq 0, \tag{3.3}$$

where χ_n is defined by minimizing the norm of $w_{n+1}^{n+1} = w_{n+1}^n - \chi_n A w_{n+1}^n$. The choice

$$\begin{array}{lcll} \tau_0(\zeta) & \equiv & 1, & \\ \tau_{n+1}(\zeta) & = & (1-\chi_n\zeta)\,\tau_n(\zeta) & \text{if } n \text{ is even,} \\ \tau_{n+1}(\zeta) & = & (\xi_n + \eta_n\zeta)\,\tau_n(\zeta) + (1-\xi_n)\,\tau_{n-1}(\zeta) & \text{if } n \text{ is odd,} \end{array} \tag{3.4}$$

where χ_n is defined as above, whereas ξ_n and η_n are obtained by minimizing the norm of $w_{n+1}^{n-1} + \left(w_{n+1}^n - w_{n+1}^{n-1}\right)\xi_n + Aw_{n+1}^n\eta_n$ leads to BiOStab2, the three-term version of Gutknecht's BiCGStab2 [5]. Generating the polynomials $\tau_n(\zeta)$ also for n even by the three-term recurrence

$$\tau_{n+1}(\zeta) = (\xi_n + \eta_n\zeta)\,\tau_n(\zeta) + (1-\xi_n)\,\tau_{n-1}(\zeta), \tag{3.5}$$

with the above definition of ξ_n and η_n, results in BiOxMR2, where the Lanczos process is combined with a local two dimensional minimization in every step. In BiOS, the three-term version of Sonneveld's (Bi)CGS, the Lanczos polynomial is squared by the choice $\tau_n(\zeta) = \rho_n(\zeta)$.

The aim of all introduced algorithms in every iteration step is to obtain an improved new approximation x_{n+1}^{n+1} by computing a new product vector w_{n+1}^{n+1} from previously computed product vectors in a stable way. To visualize the different strategies used by the algorithms, we arrange the product vectors w_n^l in a *w-table*, where the n-axis of the table points downwards and the l-axis to the right. We describe then how every algorithm moves in this table from the upper left corner downwards right. By multiplying the three-term recursion (2.3) for the right Lanczos vectors y_{n+1} with $\tau_l(A)$ we obtain for the product vectors the recursion

$$w_{n+1}^l = \left(Aw_n^l - w_n^l\alpha_n - w_{n-1}^l\beta_n\right)/\gamma_n, \tag{3.6}$$

which can be applied to move forward in vertical direction in the w-table. To obtain a recursion formula for moving forward in horizontal direction, the recursion formula for the chosen polynomials $\tau_n(\zeta)$ is capitalized upon. For example, by multiplying y_n with $\tau_{l+1}(A)$ and applying then (3.5) leads to the recurrence

$$w_n^{l+1} = \eta_l A w_n^l + \xi_l w_n^l + (1 - \xi_l) w_n^{l-1}. \tag{3.7}$$

Form the recurrence formulas for the product vectors w_n^l and from relation (3.1) easily recurrence formulas for the product iterates x_n^l and for ρ_n^l are obtained.

4. Look-Ahead Procedures for LTPMs. To derive a look-ahead version of an LTPM, we need to express the recurrence formulas (2.10) and (2.12) for the inner and regular Lanczos vectors in terms of the product vectors w_n^l.

LOOP 4.1. *(Look-Ahead for Three-Term Recurrence for $h_{j-1} \geq h_j$) Let* $\nu := \min\{n-1, n_j\}$.

Regular Loop: $(n+1 = n_{j+1})$

1. Use (4.1) to compute $w_{n+1}^{\nu}, \ldots, w_{n+1}^{n}$.
2. Compute Aw_{n+1}^{ν}.
3. Use (4.6) to compute indirectly $Aw_{n+1}^{\nu+1}, \ldots, Aw_{n+1}^{n-1}$.
4. Compute Aw_{n+1}^{n} and ξ_n, η_n.
5. Use (4.6) to compute $w_{n_j}^{n+1}, \ldots, w_{n+1}^{n+1}$.
6. Compute $Aw_{n_j}^{n+1}, \ldots, Aw_{n+1}^{n+1}$.

Inner Loop: $(n_{j+1} < n+1 < n_{j+2})$

1. Use (4.5) to compute $w_{n+1}^{\nu}, \ldots, w_{n+1}^{n}$.
2. Compute Aw_{n+1}^{ν}.
3. Use (4.6) to compute indirectly $Aw_{n+1}^{\nu+1}, \ldots, Aw_{n+1}^{n-1}$.
4. Compute Aw_{n+1}^{n} and ξ_n, η_n.
5. Use (4.6) to compute $w_{n_j}^{n+1}, \ldots, w_{n+1}^{n+1}$.
6. Compute $Aw_{n_j}^{n+1}, \ldots, Aw_{n_{j+1}-1}^{n+1}$.
7. Use (4.5) to compute indirectly $Aw_{n_{j+1}}^{n+1}, \ldots, Aw_{n}^{n+1}$.
8. Compute Aw_{n+1}^{n+1}.

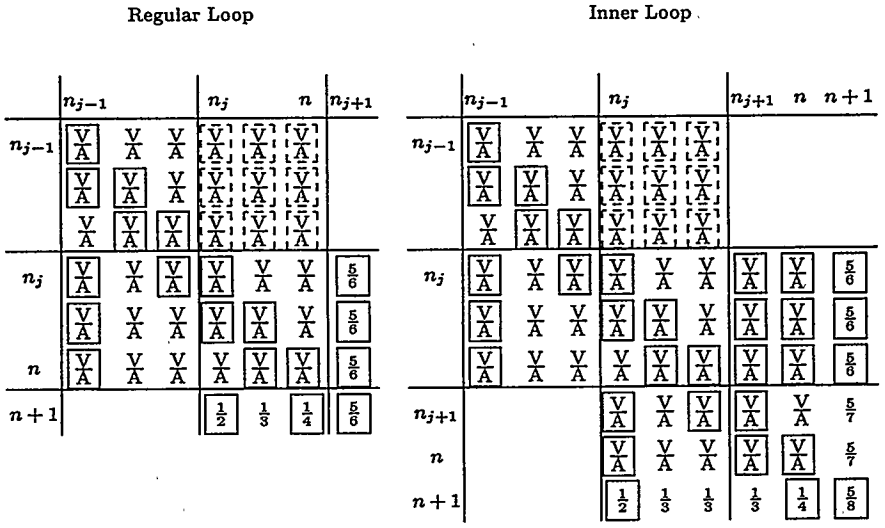


FIG. 4.1. *Action of Loop 4.1 (Look-Ahead Three-Term Recurrence) in the w-table for $h_{j-1} \geq h_j$.*

LOOP 4.2. *(Alternative Look-Ahead for Three-Term Recurrence for $h_{j-1} < h_j$)*

Regular Loop: $(n+1 = n_{j+1})$

0. Perform steps 1. up to 5. of the regular Loop 4.1.
6. Use (4.6) to compute $w_{n_{j-1}}^{n+1}, \ldots, w_{n_j-1}^{n+1}, \ldots,$.
7. Compute $Aw_{n_{j-1}}^{n+1}, \ldots, Aw_{n_j-1}^{n+1}$.
8. Use (4.5) to compute indirectly $Aw_{n_j}^{n+1}, \ldots, Aw_{n-1}^{n+1}$.
9. Use (4.1) to compute indirectly Aw_n^{n+1}.
10. Compute Aw_{n+1}^{n+1}.

Inner Loop: $(n_{j+1} < n+1 < n_{j+2})$

0. Perform steps 1. up to 5. of the inner Loop 4.1.
6. Use (4.6) to compute $w_{n_{j-1}}^{n+1}, \ldots, w_{n_j-1}^{n+1}$.
7. Compute $Aw_{n_{j-1}}^{n+1}, \ldots, Aw_{n_j-1}^{n+1}$.
8. Use (4.5) to compute indirectly $Aw_{n_j}^{n+1}, \ldots, Aw_{n_{j+1}-2}^{n+1}$.
9. Use (4.1) to compute indirectly $Aw_{n_{j+1}-1}^{n+1}$.
10. Use (4.5) to compute indirectly $Aw_{n_{j+1}}^{n+1}, \ldots, Aw_n^{n+1}$.
11. Compute Aw_{n+1}^{n+1}.

Regular Loop

	n_{j-1}		n_j		n	n_{j+1}
n_{j-1}	V/A	V/A	V/A	V/A	V/A	6/7
	V/A	V/A	V/A	V/A	V/A	6/7
n_j	V/A	V/A	V/A	V/A	V/A	5/8
	V/A	V/A	V/A	V/A	V/A	5/8
n	V/A	V/A	V/A	V/A	V/A	5/9
$n+1$			1/2	1/3	1/4	5/10

Inner Loop

	n_{j-1}		n_j			n_{j+1}	n	$n+1$
n_{j-1}	V/A	V/A	V/A	V/A	V/A	V/A	V/A	6/7
	V/A	V/A	V/A	V/A	V/A	V/A	V/A	6/7
n_j	V/A	V/A	V/A	V/A	V/A	V/A	V/A	5/8
	V/A	V/A	V/A	V/A	V/A	V/A	V/A	5/8
	V/A	V/A	V/A	V/A	V/A	V/A	V/A	5/9
n_{j+1}			V/A	V/A	V/A	V/A	V/A	5/10
n			V/A	V/A	V/A	V/A	V/A	5/10
$n+1$			1/3	1/2	1/2	1/2	1/4	5/11

FIG. 4.2. *Action of Loop 4.2 (Alternative Look-Ahead for Three-Term Recurrence) in the w-table for $h_{j-1} < h_j$.*

By multiplying (2.12) by $\tau_l(A)$ we obtain:

$$w_{n+1}^l = \left(Aw_n^l - W_j^l \alpha_n - W_{j-1}^l \beta_n\right)/\gamma_n \qquad (n+1 = n_{j+1}), \tag{4.1}$$

where we use the notation

$$W_j^l := \left[w_{n_j}^l, \ldots, w_{n_{j+1}-1}^l\right], \quad W_{j-1}^l := \left[w_{n_{j-1}}^l, \ldots, w_{n_j-1}^l\right].$$

The coefficient vectors α_n in (2.13) and β_n in (2.11) are easily expressed in terms of the product vectors w_n^l by rewriting all inner products in such a way that the part $\overline{\tau}_l(A^H)$ of $\widetilde{z}_l$ on the left side is transfered to the right side of the inner product. For the diagonal blocks D_j of the Gramian this results in

$$D_j = \left[\langle \widetilde{z}_{n_j+i}, y_{n_j+k} \rangle\right]_{i,k=0}^{h_j} = \left[\left\langle \widetilde{z}_0, w_{n_j+k}^{n_j+i} \right\rangle\right]_{i,k=0}^{h_j}, \tag{4.2}$$

and (2.11) becomes

$$\beta_n = D_{j-1}^{-1} \begin{bmatrix} \langle \widetilde{z}_0, A w_n^{n_j-1} \rangle \\ \vdots \\ \langle \widetilde{z}_0, A w_n^{n_j-1} \rangle \end{bmatrix}. \tag{4.3}$$

Likewise, (2.13) changes to

$$\alpha_n = D_j^{-1} \left(\begin{bmatrix} \langle \widetilde{z}_0, A w_n^{n_j} \rangle \\ \vdots \\ \langle \widetilde{z}_0, A w_n^{n_{j+1}-1} \rangle \end{bmatrix} - \begin{bmatrix} \langle \widetilde{z}_0, w_{n_{j-1}}^{n_j} \rangle & \cdots & \langle \widetilde{z}_0, w_{n_j-1}^{n_j} \rangle \\ \vdots & & \vdots \\ \langle \widetilde{z}_0, w_{n_{j-1}}^{n_{j+1}-1} \rangle & \cdots & \langle \widetilde{z}_0, w_{n_j-1}^{n_{j+1}-1} \rangle \end{bmatrix} \beta_n \right). \tag{4.4}$$

In the case where $n+1$ is an inner index, we multiply (2.10) by $\tau_l(A)$ to get

$$w_{n+1}^l = \left(A w_n^l - \widehat{W}_j^l \alpha_n - W_{j-1}^l \beta_n \right) / \gamma_n \qquad (n_j < n+1 < n_{j+1}), \tag{4.5}$$

where $\widehat{W}_j^l := \left[w_{n_j}^l, \ldots, w_n^l \right]$ denotes the not yet fully completed block W_j^l. The coefficient vector α_n can now be chosen arbitrarily, in particular, equal to the zero vector, whereas the coefficient vector β_n is again given by (4.3). Combining now the recurrence formulas (4.1) and (4.5) with those for the chosen polynomials $\tau_n(\zeta)$ leads to the look-ahead version of a LTPM. For example, let us consider here the case, where the polynomials $\tau_l(\zeta)$ are generated by a general three-term recurrence (3.5). Then obtain the following recursion to move horizontally in the w-table:

$$w_n^{l+1} = \xi_l w_n^l + \eta_l A w_n^l + (1 - \xi_l) w_n^{l-1}. \tag{4.6}$$

We describe now how to use (4.1), (4.5), and (4.6) to compute a new regular or inner product vector w_{n+1}^{n+1} from previously computed vectors w_k^l and $A w_k^l$. Let us assume that we have a starting situation as shown in Figure 4.1. Here the symbol 'V' indicates that the corresponding product vector is already known, whereas the symbol 'A' indicates that the product of this entry with the system matrix A is known. Surrounding an entry by a box is used to point out that a direct multiplication of this entry with the system matrix A was carried out, otherwise this product was obtained indirectly by applying a recurrence formula. Surrounding an entry by a dashed box indicates that whether this multiplication was done directly or indirectly depends on the way, the following block was computed. A number as entry elucidates in which step of the loop this entry is calculated. To compute a new product vector w_{n+1}^{n+1} Loop 4.1 is performed.

If $h_{j-1} < h_j$, so if the $(j-1)$-block is smaller than the j-block, the following modification, which is given by Loop 4.2 and is illustrated in Figure 4.2, can save computational work in terms of MVs.

Because of the many additional vectors w_n^l and $A w_n^l$, involved in the recurrence formulas (4.1) and (4.5) the look-ahead version of an LTPM needs more computational

work in terms of MVs and IPs. However, in practice the number of look-ahead steps as well as the size of their blocks is small if a good look-ahead criterion is used, so that the additional cost is rather moderate.

5. Numerical Example. In Figure 5.1 the convergence history of different look-ahead versions of LTPMS is shown for a problem with a 4-cyclic system matrix of order 400. Look-ahead is for this problem essential, since exact breakdowns appear at least in steps 2 and 3.

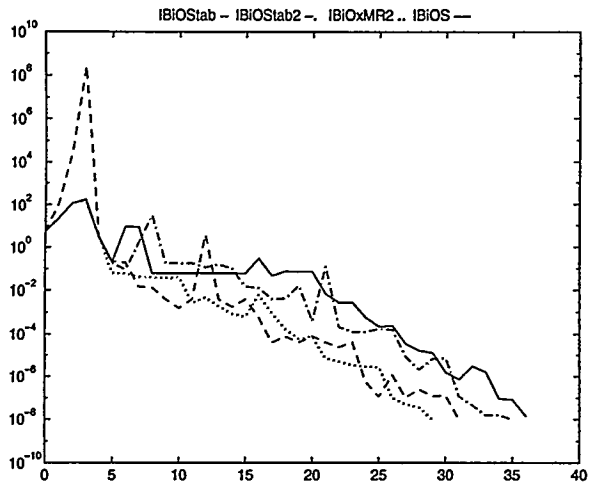


FIG. 5.1. *The residual norm history (i.e. log($\|r_n\|$) vs. iteration number n) for a linear system with the 4-cyclic system matrix solved by different LTPMs with look-ahead.*

REFERENCES

[1] C. BREZINSKI AND M. REDIVO-ZAGLIA, *Treatment of near-breakdown in the CGS algorithms*, Numerical Algorithms 7, (1994), pp. 33-73.

[2] C. BREZINSKI AND M. REDIVO-ZAGLIA, *Look-ahead in Bi-CGSTAB and other product-type methods for linear systems*, BIT 35, No. 2, (1995), pp. 169-201.

[3] R. W. FREUND, M. H. GUTKNECHT AND N. M. NACHTIGAL, *An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices*, SIAM J. Sci. Comput. 14, (1993), pp. 137-158.

[4] M. H. GUTKNECHT, *A completed theory of the unsymmetric Lanczos process and related algorithms, Part I*, SIAM J. Matrix Anal. Appl. 13, (1992), pp. 594-639.

[5] M. H. GUTKNECHT, *Variants of BiCGStab for matrices with complex spectrum*, SIAM J. Sci. Comput. 14, (1993), pp. 1020-1033.

[6] M.H. GUTKNECHT, *A completed theory of the unsymmetric Lanczos process and related algorithms, Part II*, SIAM J. Matrix Anal. Appl. 15, 15–58, 1994.

[7] B. N. PARLETT, D. R. TAYLOR AND Z. A. LIU, *A look-ahead Lanczos algorithm for unsymmetric matrices*, Math. Comp. 44, (1985), pp. 105-124.

[8] G.L.G. SLEIJPEN AND D.R. FOKKEMA, *BICGSTAB(ℓ) for linear equations involving unsymmetric matrices with complex spectrum*, Electronic Transaction on Numerical Analysis 1, (1993), pp. 11-32.

[9] P. SONNEVELD, *CGS, a fast Lanczos-type solver for nonsymmetric linear systems*, SIAM J. Sci. Comput. 10, (1989), pp. 36-52.

[10] H. A. VAN DER VORST, *Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Comput. 13, (1992), pp. 631-644.

Solving modified systems with multiple right-hand sides

V. Simoncini*and E. Gallopoulos†

January 1995

Abstract

In this talk we discuss the iterative solution of large linear systems of the form

$$(A + USV^H)X = B,$$

where A is an $n \times n$ non-Hermitian matrix, USV^H is a rank-r modification of A and B is of rank s with $s, r \ll n$. We analyze several approaches that exploit the structure of the coefficient matrix so as to solve the systems more efficiently than if one were to apply a non-hermitian solver to the original systems. In the development of procedures, we take into account the presence of both the low-rank modification and the several right-hand sides. Interesting issues connected to this problem originate from the quest for techniques that accelerate the underlying iterative solvers: preconditioning (e.g. inner-outer iteration strategies), domain decomposition, and continuation methods. Experiments are provided to analyze the behavior of the methods depending on the structure of the rectangular matrices. Preconditioning strategies are explored for an efficient implementation on the transformed systems.

*IMGA-CNR, Modena, Italy. E-mail: simoncin@deis.unibo.it.

†Department of Computer Engineering and Informatics, University of Patras, 26500 Patras, Greece. E-mail: stratis@daidalos.ceid.upatras.gr.

Topic: *Eigenvalues*	*Session Chair:* *Homer Walker*	*Room C*
10:30 - 11:00	A. Stathopoulos	Thick Restarting of the Davidson Method: an Extension to Implicit Restarting
11:00 - 11:30	X. Zou	Splitting the Determinants of Upper Hessenberg Matrices & the Hyman Method
11:30 - 12:00	M. Fernandes	A Combined Modification of the Newton's Method for Systems of Nonlinear Equations

Thick Restarting of the Davidson Method: an Extension to Implicit Restarting

Andreas Stathopoulos, Yousef Saad, Kesheng Wu
Computer Science Department, University of Minnesota

March 15, 1996

1 Introduction

The solution of the large, sparse, eigenvalue problem $Ax = \lambda x$, for a few eigenpairs is central to many scientific applications [15]. The Arnoldi method, and its equivalent in the symmetric case the Lanczos method, have been the traditional approach to solving these problems. Preconditioning, through some shift-and-invert technique [18], is frequently employed, because of the difficulty of these problems. A different approach is followed by the Generalized Davidson (GD) method [6, 12, 4] which is a popular preconditioned variant of the Lanczos iteration. Instead of using a three-term recurrence to build an orthonormal basis for the Krylov subspace, the GD algorithm obtains the next basis vector by explicitly orthogonalizing the preconditioned residual $(M - \lambda I)^{-1}(A - \lambda I)x$ against the existing basis. A straightforward extension to the non-symmetric case has also been studied in [17]. The GD method can be regarded as a way of improving convergence and robustness at the expense of a more complicated step.

Often, a large number of steps is required for convergence, necessitating restarting GD every m iterations (GD(m)). If the l lowest eigenvalues are needed, the l lowest Ritz values are computed at the end of the m_{th} step, and their corresponding Ritz vectors are used as initial guesses for the restarted GD iteration. Traditionally, the Lanczos and Arnoldi processes had been considered without restarting, but because of orthogonality problems and spurious solutions many restarting variants are used in practice [5, 14, 20, 2].

Truncating the Krylov sequence is expected to impair the convergence rate of the method. There are two main reasons: the new vectors entering the basis repeat some of the information that was discarded when restarting, and the Rayleigh-Ritz procedure does not minimize over the whole Krylov subspace. There has been much discussion about the problems caused by restarted methods for both linear systems and eigenvalue problems [7, 16, 20]. Some methods tend to save additional information at each restarting [11, 3]. For the Davidson method, Murray et al. [13], and Van Lenthe et al. [23], have proposed restarting with two vectors per required Ritz vector with some success. In an effort to minimize execution time, Crouzeix et al. [4], have proposed a dynamically chosen size m.

Recently, 'implicit restarting' has gained popularity as a means of improving convergence of the Arnoldi procedure [20]. By using $p = m - k$ steps of the implicit QR algorithm on the Hessenberg matrix, the basis is truncated down to k vectors. It turns out that the k new basis vectors can be considered to be the Arnoldi vectors obtained from a polynomially transformed starting vector $v' = \pi(A)v$. This is the basis of the popular ARPACK eigenvalue package [10]. Preconditioners for eigenvalue problems usually vary between steps, in which case the Implicitly Restarted Arnoldi (IRA(k,m)) is not straightforward to apply. Further, in case of the GD(m) where the residual is preconditioned, the Ritz vectors can not be described with a polynomial of A. Clearly, a new restarting scheme is needed.

In this paper, we propose an extension to the IRA(k,m) technique for the GD(m), called 'thick restarting' and noted GD(k,m), which depends on an integer parameter k. GD(k,m) restarts with k Ritz vectors instead of the l wanted ones, where $l \leq k < m$. A similar idea is mentioned by Sleijpen et al. in [19]. After briefly presenting the Arnoldi, IRA(k,m) and GD(k,m) algorithms in section 2, in section 3 we prove that in the absence of preconditioning, the IRA(k,m) using the Ritz-values as shifts, and GD(k,m) are

equivalent, in the sense that their basis vectors span exactly the same space. In section 4 a theorem is proved that relates the p steps of IRA(k,m), and thus GD(k,m), with an Arnoldi process applied on an approximately deflated initial vector. This extends the ideas that appeared recently in [8]. Usually, the benefits of IRA(k,m) are studied in relation to the polynomially transformed initial vector. The deflation relation justifies the use of a thicker restarting. In section 5, a dynamic choice of k is derived for the symmetric case, where the rate of convergence is described by well-known bounds. Although the above results are proved for the non-preconditioned case, the idea of thick restarting is readily applicable to the preconditioned GD(k,m) and similar behavior is expected. Compared with IRA(k,m), GD(k,m) can also assume any number of initial guesses, and/or enhancements of the basis through arbitrary vectors, during the procedure. In section 6, numerical experiments on the symmetric matrices of the Harwell Boeing collection demonstrate the effectiveness of GD(k,m).

The proofs for the theoretical results in this extended abstract can be found in the unabridged version of this paper.

2 The restarted Arnoldi and Davidson methods

Throughout this paper we assume that the matrix A is diagonalizable, of order N, with eigenpairs (λ_i, x_i). We look for l eigenpairs and the Arnoldi and Davidson methods use a basis size of $m > l$. The following descriptions of the algorithms establish the notation, and do not reflect implementation details.

Restarted Arnoldi's method in its simplest form can be expressed as follows:

ALGORITHM **2.1 Restarted Arnoldi**

0. Start: Choose initial unit vector $v^{(0)}$
1. For $s = 0, 1, \ldots$ *Do*
2. $v_1 = v^{(s)}$, $V^{(s)} = \{v_1\}$
3. *For* $j = 1, \ldots, m$ *Do*
4. $h_{ij} = (Av_i, v_j)$, $i = 1, \ldots, j$,
5. $w_j = Av_j - \sum_{i=1}^{j} h_{ij} v_i$
6. $h_{j+1,j} = \|w_j\|_2$, *if* $h_{j+1,j} = 0$ *stop.*
7. $v_{j+1} = w_j / h_{j+1,j}$
8. *Enddo*
9. *Compute the wanted eigenpairs* $(\mu_i^{(s)}, y_i^{(s)})$ *of* $H_m^{(s)}$
and the Ritz vectors $x_i^{(s)} = V_m^{(s)} y_i^{(s)}$
10. $v^{(s+1)} = \sum c_i x_i^{(s)}$, *for some* c_i
11. Enddo

The algorithm builds a Hessenberg matrix, from which the approximate eigenpairs are extracted through the Rayleigh-Ritz procedure. For the symmetric case, $H_m^{(s)}$ is a tridiagonal matrix, and a three-term recurrence replaces the above orthogonalization step. A linear combination of the wanted Ritz vectors is used to restart the algorithm. However, this discards a lot of information from the previous step and results in degradation of the convergence rate.

Implicitly restarted Arnoldi (IRA), applies the implicit QR algorithm with the $m-l$ unwanted eigenvalues as shifts to the Hessenberg matrix, and uses the generated orthogonal transformations to truncate the basis down to l vectors. These vectors can be considered to be the Arnoldi vectors of another Arnoldi process which started with a polynomially transformed initial vector. Thus, IRA avoids the need to explicitly restart the Arnoldi algorithm. The number of vectors in the new basis after restart, may also be larger than l, say k. For the rest of the paper we assume that $l \leq k < m$, and IRA(k,m) denotes the associated method.

The Davidson method first appeared as a diagonally preconditioned version of the Lanczos method for symmetric eigenproblems. Extensions, to both general preconditioners and to the non-symmetric case have been given since. We describe the algorithm for the symmetric case. For the non symmetric, line 5 should also include the computation of the last row of $D_j^{(s)}$.

ALGORITHM **2.2 Generalized Davidson**

0. Choose initial unit vectors $U_l^{(0)} = \{u_1^{(0)}, \ldots, u_l^{(0)}\}$
1. For $s = 0, 1, \ldots$ *Do*
2. $w_i^{(s)} = Au_i^{(s)}$, $i = 1, \ldots, l-1$
3. For $j = l, \ldots, m$ *Do*
4. $w_j^{(s)} = Au_j^{(s)}$.
5. $d_{i,j} = (w_j^{(s)}, u_i^{(s)})$, $i = 1, \ldots, j$, *the last column of* $D_j^{(s)}$
6. Compute some wanted eigenpair, say (μ_1, z_1) *of* $D_j^{(s)}$.
7. $x_1 = U_j^{(s)} z_1$ *and* $r = Ax_1 - \mu_1 x_1$, *the Ritz vector and its residual*
8. Test $\|r\|$ *for convergence. If satisfied target a new vector.*
9. $t = M_{(s,j)}^{-1} r$.
10. $b_{j+1}^{(s)} = MGS(U_j^{(s)}, t)$
11. Enddo
12. Set $U_k^{(s+1)} = \{x_1, \ldots, x_k\}$, $k < m$, *and restart*
13. Enddo

The Davidson algorithm has a more expensive step than the Lanczos and Arnoldi algorithms, to allow for the benefits of preconditioning. In addition, the Davidson algorithm can start with any number of initial vectors, and include in the basis any extra information that can be available during the execution. Finally, it can restart with the approximate eigenvectors, so it does not share the problems of the original Restarted Arnoldi. As in IRA(k,m), the Davidson method can also restart with more Ritz vectors than needed. This version is called 'thick restarting' and denoted GD(k,m) where l, k, and m are defined as in IRA(k,m). In the following sections, we show that IRA(k,m) and GD(k,m) are equivalent in the non-preconditioned case, but GD(k,m) offers all the aforementioned advantages and extensions. We also provide an explanation for the improvements associated with GD(k,m) and give a choice for the parameter k.

3 Thick and implicit restarting

It is known that the Lanczos and the Davidson methods are equivalent when no preconditioning is used. However, this has been pointed out only for the non-restarted case, where one eigenvalue is sought [12]. In this section we prove that in the non-preconditioned case, IRA(k,m) is equivalent to GD(k,m), if GD(k,m) starts with one initial vector. The proofs are omitted, and can be found in the full version of this paper.

The first Lemma is an extension of Lemma 3.10 in [20], and it is the basis for the equivalence proof. The implicit QR algorithm is applied on any diagonalizable matrix H.

Lemma 3.1 *Let* $\lambda(H) = \{\lambda_1, \ldots, \lambda_k\} \cup \{\mu_1, \ldots, \mu_p\}$ *be a disjoint partition of the eigenvalue set of a diagonalizable matrix* H. *Let*

$$H_+ = Q^T H Q,$$

where $Q = Q_1 Q_2 \cdots Q_p$, *and* Q_i *is the orthogonal matrix implicitly defined by the shift* μ_i *in the implicit QR algorithm. Then, the first* k *columns of* Q *span the same space as the* k *eigenvectors* y_i *of* H *associated with the eigenvalues* λ_i.

An immediate consequence is the following:

Lemma 3.2 *If at step* s *the basis vectors* $U_m^{(s)}$ *and* $V_m^{(s)}$ *of GD(k,m) and IRA(k,m) respectively, span the same space, then, after restarting both methods,*

$$\text{span}(V_k^{(s+1)}) = \text{span}(U_k^{(s+1)}).$$

Theorem 3.1 *If GD(k,m) without preconditioning and IRA(k,m) are applied to the same initial vector* $v^{(0)}$, *and at each restarting the* p *shifts used in IRA(k,m) are the Ritz values of the Ritz vectors discarded by GD(k,m), then the basis vectors produced by the two methods span the same space, and thus the methods are equivalent.*

4 The deflation connection

Krylov methods for linear systems, such as conjugate gradient (CG) and frequently GMRES, demonstrate a superlinear convergence at later iterations. One explanation for the phenomenon is the convergence of the outer eigenpairs of the matrix, so that each method behaves as if deflation has occurred. The result is a better condition number for CG, or a smaller ellipse containing the eigenvalues for GMRES. Such observations have appeared as early as in [1], but actual quantification of the behavior appears in [16] and [7, 8]. In the latter papers, the optimality of the CG and GMRES polynomials is employed to relate each method after some iterations with a similar process of the same method on a deflated residual.

Results similar to [8] can not be applied directly to the residual and eigenvalues in the non-symmetric Arnoldi, since there is no optimality principle. In the following, we extend the results found in [8] to the Arnoldi method, by considering the distance of some eigenvector from the Arnoldi Krylov subspace. Again preconditioning is not considered since the space that it creates is not a Krylov subspace.

Assume A diagonalizable, $X^{-1}AX = \Lambda = \text{diag}(\lambda_i)$, of order N, and let $v = X\gamma$ be the expansion of the starting Arnoldi vector to the eigenvector basis. Define three numbers satisfying $l < k' \leq k$, where $k-1$ is the number of steps that an Arnoldi method takes starting from v. We can assume an eigenvalue ordering so that the first l ones are wanted, and the $l+1, \ldots, k'$ eigenvalues are well approximated by the $k-1$ steps of Arnoldi. Let μ_i be the k Ritz values from this $\mathcal{K}_k(v)$ space. At this point we let the Arnoldi process take p more steps and build the space $\mathcal{K}_{k+p}(v)$. Define $\mathcal{D}^{(k)}$ a diagonal matrix with elements:

$$\mathcal{D}^{(k)}_{jj} = \begin{cases} 0, & \text{for } j = l+1, \ldots, k' \\ \prod_{i=l+1}^{k'} \frac{\lambda_j - \lambda_i}{\lambda_j - \mu_i}, & \text{for } j \leq l \text{ or } j = k'+1, \ldots, N \end{cases} . \tag{1}$$

The following theorem relates the distance of an eigenvector x_j from the Krylov subspace $\mathcal{K}_{k+p}(v)$, to the distance from the lower degree Krylov subspace $\mathcal{K}_p(\tilde{x}_j)$. $\tilde{x}_j$ is the corresponding Ritz vector from $\mathcal{K}_k(v)$, whose components of $x_{l+1}, \ldots, x_{k'}$ have been extracted. Assuming these definitions and $\mathcal{D}^{(k)}$ as defined in (1) we have:

Theorem 4.1 *If the following Krylov subspaces can be built, then for any $j = 1, \ldots, l$ it holds:*

$$\text{dist}(x_j, \mathcal{K}_{k+p}(v)) \leq |1 - \mathcal{D}^{(k)}_{jj}| + \|X\mathcal{D}^{(k)}X^{-1}\| \text{ dist}(x_j, \mathcal{K}_p(\tilde{x}_j)).$$

Let $\mathcal{P}_{k+p}$ be the orthogonal projector onto $\mathcal{K}_{k+p}(v)$, and $\mathcal{P}_p$ the orthogonal projector onto $\mathcal{K}_p(\tilde{x}_j)$. We define $A_{k+p} = \mathcal{P}_{k+p}A\mathcal{P}_{k+p}$, and $A_p = \mathcal{P}_pA\mathcal{P}_p$. A similar result holds for the residuals of the exact eigenpair from the projected matrices.

Corollary 4.1 *Let $\rho = \|\mathcal{P}_{k+p}A(I - \mathcal{P}_{k+p})\|$. Then, for the exact eigenpair (λ_j, x_j) we have:*

$$\|(A_{k+p} - \lambda_j I)x_j\| \leq \sqrt{\lambda_j^2 + \rho^2} \left(|1 - \mathcal{D}^{(k)}_{jj}| + \|X\mathcal{D}^{(k)}X^{-1}\| \|(A_p - \lambda_j I)x_j\| \right).$$

The term $\|X\mathcal{D}^{(k)}X^{-1}\|$ is bounded by the following [8]:

$$\|X\mathcal{D}^{(k)}X^{-1}\| < k_2(X) \max_{j \neq l+1, \ldots, k'} \prod_{i=l+1}^{k'} \frac{\lambda_j - \lambda_i}{\lambda_j - \mu_i} = k_2(X)F_k,$$

where $k_2(X) = \|X\|\|X^{-1}\|$ is the condition number of the eigenvectors. If k is large enough, then the approximations μ_i converge to λ_i for $i = 1, \ldots, k'$. Thus, $F_k \to 1$, and $|1 - \mathcal{D}^{(k)}_{jj}| \to 0$. Even when these are not accurately converged, provided that $\mathcal{O}(\text{dist}) < \mathcal{O}(|1 - \mathcal{D}^{(k)}_{jj}|)$, the distance behaves similarly to the distance from a deflated Krylov subspace.

In the case of the IRA(k,m) and GD(k,m) methods, the above results suggest there are advantages in keeping more vectors at each restarting, i.e., a thicker restart. As previously, l eigenpairs are needed, k pairs are retained after each restart, and both methods build $p = m - k$ additional vectors. We also assume

that the retained eigenpairs converge. This has been proved for the symmetric case, and under certain assumptions for the non-symmetric case [20]. It is known that at the s_{th} restarted step, the k retained vectors can be considered the Arnoldi vectors of a process starting with a normalized initial vector $v^{(s)} = \prod_{i=k+1}^{m}(A-\mu_i^{(s-1)})v^{(s-1)}$, where $\mu_i^{(s-1)}$ are the dumped Ritz values before restarting. Thus, Theorem 4.1 applies with the same l, k, p and $k' = k$. Moreover, the Krylov space $\mathcal{K}_{k+p}(v^{(s)})$ is built implicitly by only p steps, after retaining k vectors. Therefore, Theorem 4.1 relates the p steps of the deflated method, to p, rather than $k+p$ steps of the original method. Because of the assumption that the retained eigenpairs converge, $F_k^{(s)} \to 1$ and $|1-\mathcal{D}_{jj}^{(k)}| \to 0$ as $s \to \infty$. Therefore, after several restarts, the IRA(k,m) method builds a space close to the one built by an IRA(k,m) applied on a system deflated from the eigen-components $l+1, \ldots, k$. Because of Theorem 3.1, the GD(k,m) performs in a similar way.

If only the wanted eigenpairs $(1, \ldots, l)$ are retained at restart, the method does not demonstrate the deflation behavior for any other eigenpairs. At every restarting the current approximations of eigenpairs $(l+1, \ldots, k+p)$ are annihilated, and thus they do not converge. Frequently, some eigenvalues close to the wanted ones or close to the other end of the spectrum are relatively well approximated before restarting, and if retained, they would have soon converged. Even more undesired is the fact that these approximations will slowly reappear in the Krylov subspace, since their approximations are not accurate enough to completely annihilate the corresponding eigenvectors. Therefore, thick restarting should almost always be beneficial.

5 Dynamic thick restarting in symmetric cases

In this section we restrict the discussion to the symmetric case where explicit bounds for convergence rates are known. Two difficulties are associated with thick restarting: the choice of which eigenpairs to retain, and how many of them. Sleijpen et al. in [19] argue that the restarted Arnoldi method repeats the information close to the required eigenpairs, and they propose keeping $l+1, \ldots, k$ eigenvalues closest to the wanted ones. A similar strategy is followed in the implicit restarting of the ARPACK code. We denote this special case of GD(k,m) as TR(k), implying the basis size m.

The preceding discussion suggests that thick restarting should aim at improving the convergence of the method through deflation. TR(k) attempts to increase the gap of the wanted eigenvalues from the rest of spectrum by keeping close eigenpairs. However, convergence depends on the gap ratios of the eigenvalues and therefore the other end of the spectrum is also of importance. A more general form of thick restarting would be TR(L,R), where L lowest (leftmost) and R highest (rightmost) eigenvectors are kept. We need to address the issue of choosing optimal restarting parameters. In ARPACK, k is chosen dynamically, starting from a relatively small number and increasing it every time an eigenvalue converges. This is slightly different from the strategy reported in [20], where values of k close to $m/2$ usually gave the best results.

Because of the deflation relation, the thicker the restarting, the larger the part of the spectrum that is deflated. However, the basis size m is limited, and if too many vectors are retained when restarting, the Lanczos process can not effectively build additional basis vectors. A dynamic choice of the parameters L and R should be able to capture this trade-off. For the Lanczos procedure, convergence is governed by a term involving a Chebyshev polynomial. If p Lanczos steps are taken, and the gap ratio of the i_{th} eigenvalue, $g_i = (\lambda_i - \lambda_{i+1})/(\lambda_{i+1} - \lambda_N)$ is small, the corresponding eigenvalue error is approximated by:

$$\frac{1}{T_p^2(1+2g_i)} \approx 2e^{-2p\sqrt{g_i}}. \tag{2}$$

The L and R thick restarting parameters should maximize the deflated gap ratio $g_i = (\lambda_i - \lambda_{L+1})/(\lambda_{i+1} - \lambda_{N-R})$ and also maximize the number of new Lanczos steps $p = m - L - R$. The trade-off is captured by minimizing the error approximation equation (2). Since the actual eigenvalues are not known, the m approximate Ritz values (μ_i) before restarting should give an estimate of the spectrum. Thus, assuming the l lowest eigenpairs are sought, L and R are obtained dynamically by maximizing the following expression:

$$\max_{L=l,\ldots,m,\ R=0,\ldots,m-l,\ L+R<m} (m-L-R)\sqrt{\frac{\lambda_i - \lambda_{L+1}}{\lambda_{i+1} - \lambda_{m-R}}}.$$

It has been observed that if some unwanted eigenvector has converged it is usually beneficial to include it in restarting. We do not consider this option and let the dynamic choice of L and R take care of such cases. For the non-symmetric GD(k,m) a similar expression may be maximized, but the choice is more ad-hoc because of lack of general expressions for convergence rates. Finally, this dynamic strategy can be used in case of preconditioning. Its effects are expected to be less pronounced, since the spectrum of the varying operator is transformed by the preconditioners. Often, however, this transformation may not affect the eigenvalue order significantly, and thick restarting should perform as well in this case.

6 Numerical experiments

In the first part of this section we give an artificial example that demonstrates the increasing effect of deflation in thick restart TR(k). In the second part, we present results from a large number of tests on the symmetric matrices of the Harwell-Boeing collection [9]. The GD(k,m) code is based on a program published in [21] and the extensions proposed in [22]. It implements a block generalized Davidson method, using the reverse communication protocol for matrix-vector multiplication and preconditioning operations.

6.1 Deflation works

The GD(k,m) is applied on an artificially generated diagonal matrix of order 100, and elements:

$$A_{jj} = \begin{cases} j/55, & \text{for } j = 1,\ldots,8 \\ 19/55 + j/55, & \text{for } j = 9,\ldots,16 \\ j - 16, & \text{for } j = 17,\ldots,100 \end{cases} . \tag{3}$$

The lowest eigenvalues of this matrix are grouped in two clusters of 8 equidistant eigenvalues each. The separation between the two groups is equal to the separation of the second group from eigenvalue 17. Figure 1 depicts the lowest part of this spectrum. We look for the lowest eigenvalue and allow for 20 basis vectors in all versions of GD(k,m). The history of the logarithm of the eigenvalue error is plotted in Figure 2 for various restarting thicknesses of TR(k).

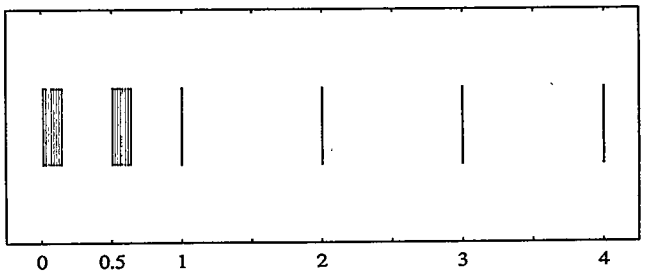


Figure 1: The lowest 20 eigenvalues of the 100×100 matrix. The first two clusters contain 8 equidistant eigenvalues each. The rest 80 eigenvalues are the integers from 5 up to 84.

As expected, the poor separation of the lowest eigenvalue results in a very slow original GD(20) (or TR(1)) method. A very good approximation of the second eigenvalue is available quite early, and thus when retained (TR(2)), the convergence rate improves by 30%. Similarly with TR(4) and TR(8). The superlinear convergence is more evident in TR(8). In early iterations, higher eigenvalues are not well approximated and TR(8) behaves similarly to TR(1) and TR(2). Later, as better approximations for eigenvalues 2–4 appear, TR(8) is similar to TR(4), and as higher eigenvalues settle down, TR(8) exhibits a concave convergence curve.

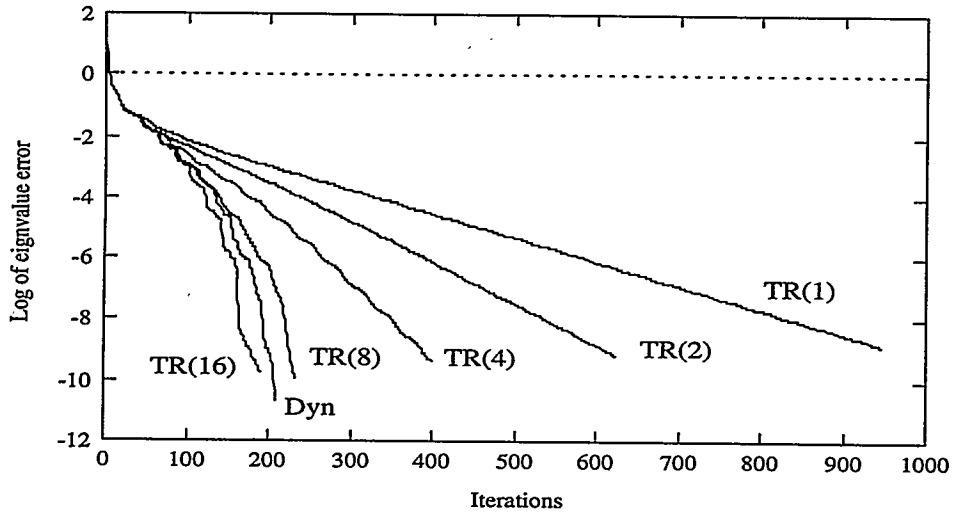


Figure 2: Effects of thick restarting to the convergence of the Generalized Davidson. No preconditioning is used, and the lowest eigenvalue is sought. TR(k) denotes GD(k,20).

Methods TR(k), with $8 < k < 16$ are similar to TR(8) since there is no significant improvement to the deflated gap ratio. In theory, TR(16) should be different, because of the large separation between eigenvalues 16 and 17. In practice, TR(16) does not perform significantly better than TR(8), because of the small size of the Krylov subspace. The dynamic thick restarting, shown as Dyn in the figure, takes advantage of both ends of the spectrum and performs better than TR(8) and close to TR(16), requiring no prior knowledge about the spectrum.

6.2 Harwell-Boeing Tests

To confirm the theoretical benefits of thick and dynamic thick restaring, a wide variety of tests has been performed on the symmetric matrices of the Harwell-Boeing collection. For almost all of these matrices, the lowest end of spectrum is very poorly conditioned. The higher end of the spectrum usually consists of well separated, very large eigenvalues providing a good test for easy or intermediate problems.

We compare three different versions of GD(k,m) and the ARPACK code for both the lower and the higher part of the spectrum. Five eigenvalues are sought and the basis size $m = 20$ for all GD methods. ARPACK is assigned a basis size of 25. For the highest eigenvalues only the non-preconditioned versions of GD(k,m) are considered, while for the lowest ones, we also consider diagonal preconditioning. The GD(k,m) methods stop when the norm of the residual is less than $10^{-12}\|A\|_F$, where $\|A\|_F$ is the Frobenious norm of the matrix. The ARPACK tolerance is set to 10^{-12}.

In Table 6.2, the results from the lower part of the spectrum are reported, for all the matrices in which at least one method has converged. A maximum number of 5000 matrix vector multiplications is allowed. The table does not include any of the diagonal matrices. As it is easily seen, TR(11) outperforms the original Davidson method (TR(5)). It is usually several times faster, and offers better robustness, converging for 6 additional matrices. Because no preconditioning is used, ARPACK behaves similarly. Further, dynamic thick restarting, improves both the robustness, and the speed in almost all cases. With diagonal preconditioning TR(11) still outperforms TR(5) in both convergence and robustness. Dynamic thick restarting improves convergence further, although the improvements are not as significant as in the non-preconditioned case.

Similar behavior of the methods is shown in Table 6.2. Dynamic thick restarting improves on the performance of TR(10) which in turn improves on the performance of TR(5). ARPACK performs between the range of TR(10) and Dyn. However, the few steps required for the problems in this table do not yield

Table 1: Comparison of thick (TR(11)) and dynamic thick restarting (Dyn) with original Davidson (TR(5)) and ARPACK on Harwell-Boeing matrices. The number of matrix vector multiplications is reported, with a maximum of 5000. Five smallest eigenvalues are sought. The GD codes use basis size of 20, and ARPACK uses basis size of 25.

	No preconditioning				Diagonal preconditioning		
Matrix	TR(5)	TR(11)	Arpack	Dyn	TR(5)	TR(10)	Dyn
BCSSTK01	-	1675	2929	396	288	132	127
BCSSTK02	-	209	233	205	-	194	191
NOS4	321	178	233	174	405	261	241
BCSSTK03	-	-	-	-	-	3697	1911
BCSSTK04	-	-	-	1913	-	189	190
BCSSTK22	4054	-	-	3299	-	931	710
LUND A	-	2017	4285	888	858	271	261
LUND B	-	-	-	1424	774	396	390
BCSSTK05	1174	975	801	629	1322	465	442
BCSSTK06	-	-	-	-	-	-	2324
BCSSTK07	-	-	-	-	-	-	2324
BCSSTM07	-	-	-	3323	1018	406	371
NOS5	-	2016	2789	960	2659	1401	969
662 BUS	-	-	3729	-	3220	1482	1286
NOS6	-	-	-	-	-	-	1522
685 BUS	-	-	2327	1950	2473	987	807
NOS7	-	-	-	-	200	216	191
GR 30 30	259	228	274	215	248	224	216
NOS3	2179	620	735	463	2096	878	727
BCSSTK08	-	-	-	-	1177	1012	1200
BCSSTK09	-	1206	1028	799	2283+	1508	1136
BCSSTK10	-	-	-	-	-	-	4970
BCSSTM10	498	226	329	206	448	258	253
BCSSTK27	-	-	-	-	-	-	3334
BCSSTM27	-	4455	3665	2529	-	4304	1771
BCSSTK14	-	-	-	-	-	-	2438
BCSSTM13	-	-	-	-	381	285	284
ZENIOS	87	88	82	87	88	89	90
BCSSTK21	-	-	-	-	-	2568+	1349
BCSSTK16	3962	1333	1600	718	2410	905	842
BCSSTK18	-	-	-	-	-	-	3686
BCSSTM25	-	-	-	-	62	64	56

+ denotes that one eigenpair has been skipped

the same impressive improvements as in Table 6.2. In fact, a few cases are noted to perform worse with dynamic than with simple thick restarting.

We should point out that the above results compare the number of matrix vector multiplications of the methods. This is an acceptable performance metric if the matrix-vector operation is expensive. Since on average, thick restarting uses more vectors in the basis than the original Davidson, its Davidson step is also more expensive. Although improvements like the ones in Table 6.2 justify any increase in the expense of the Davidson step, for easier cases a less aggressive choice of restarting might be more effective.

Acknowledgements

This research has been supported by NSF grant No. NSF/ ASC 95-04038 and NSF/DMR-95 25885, and the Minnesota Supercomputer Institute.

References

[1] J. R. Bunch and D. J. Rose, editors. *A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations.* Academic Press, New York, NY, 1976.

Table 2: Comparison of thick (TR(10)) and dynamic thick restarting (Dyn) with original Davidson (TR(5)) and ARPACK on Harwell-Boeing matrices. The number of matrix vector multiplications is reported. Five largest eigenvalues are sought. The GD codes use basis size of 20, and ARPACK uses basis size of 25.

	No preconditioning			
Matrix	TR(5)	TR(10)	Arpack	Dyn
BCSSTK01	57	45	43	41
BCSSTK02	71	58	60	53
NOS4	245	151	101	119
BCSSTK03	44	43	41	43
BCSSTK04	163	105	98	92
BCSSTK22	134	76	62	70
LUND A	215	149	118	138
LUND B	121	84	63	84
BCSSTK05	78	67	64	59
NOS1	286	157	221	126
PLAT362	166	134	130	122
BCSSTK06	437	144	200	132
BCSSTK07	437	144	200	132
BCSSTM07	365	231	200	219
NOS5	197	113	114	112
662 BUS	72	59	43	57
NOS6	133	112	102	101
685 BUS	35	34	41	34
NOS7	120	74	60	75
BCSSTK19	114	94	96	92
GR 30 30	660	519	463	489

	No preconditioning			
Matrix	TR(5)	TR(10)	Arpack	Dyn
NOS2	2306	959	1054	544
NOS3	329	256	158	214
BCSSTK08	33	31	25	32
BCSSTK09	383	274	260	276
BCSSTK10	152	96	121	90
BCSSTM10	569	219	178	172
1138 BUS	100	88	62	86
BCSSTK27	144	111	81	104
BCSSTM27	144	100	82	101
BCSSTK11	355	231	387	208
BCSSTM12	204	175	175	148
BCSSTK14	244	99	99	95
PLAT1919	127	111	113	106
ZENIOS	45	42	44	42
BCSSTK24	44+	103	114	97
BCSSTK21	1603	661	467	583
BCSSTK15	5006	1374	642	4067
BCSSTK16	94	77	97	77
BCSSTK17	68	55	62	52
BCSSTK18	100	66	79	63
BCSSTK25	15	15	43	57

+ denotes that one eigenpair has been skipped

[2] D. Calvetti, L. Reichel, and D. Sorensen. An implicitly restarted lanczos method for large symmetric eigenvalue problems. *Electronic Trans. Numer. Anal.*, 2:1–21, 1994.

[3] A. Chapman and Y. Saad. Deflated and augmented krylov subspace techniques. Technical Report 95-181, University of Minnesota, Supercomputing Institute, 1995.

[4] M. Crouzeix, B. Philippe, and M. Sadkane. The Davidson method. *SIAM J. Sci. Comput.*, 15:62–76, 1994.

[5] J. Cullum and R. A. Willoughby. *Lanczos algorithms for large symmetric eigenvalue computations*, volume 2: Programs of *Progress in Scientific Computing; v. 4*. Birkhauser, Boston, 1985.

[6] Ernest R. Davidson. The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real-symmetric matrices. *J. Comput. Phys.*, 17:87, 1975.

[7] A. Van der Sluis and H. A. Van der Vorst. The rate of convergence of conjugate gradients. *Numer. Math.*, 48:543–560, 1986.

[8] H. A. Van der Vorst and C. Vuik. The superlinear convergence behavior of GMRES. *Journal of computational and applied mathematics*, 48:327–341, 1993.

[9] I. S. Duff, R. G. Grimes, and J. G. Lewis. Sparse matrix test problems. *ACM Trans. Math. Soft.*, pages 1–14, 1989.

[10] R. B. Lehoucq, D. C. Sorensen, and P. Vu. An implementation of the implicitly restarted arnoldi iteration that computes some of the eigenvalues and eigenvectors of a large sparse matrix. Technical report, University of Tennessee, Netlib, 1995.

[11] R. B. Morgan. A restarted gmres method augmented with eigenvectors. *SIAM J. Matrix Anal. Appl.*, page 16, 1995.

[12] R. B. Morgan and D. S. Scott. Generalizations of davidson's method for computing eigenvalues of sparse symmetric matrices. *SIAM J. Sci. Comput.*, 7:817–825, 1986.

[13] C. W. Murray, S. C. Racine, and E. R. Davidson. Improved algorithms for the lowest eigenvalues and associated eigenvectors of large matrices. *J. Comput. Phys.*, 103(2):382–389, 1992.

[14] Yousef Saad. Chebyshev acceleration techniques for solving nonsymmetric eigenvalue problems. *Math. Comp.*, 42:567–588, 1984.

[15] Yousef Saad. *Numerical methods for Large eigenvalue problems.* Manchester University Press, 1993.

[16] Yousef Saad. Analysis of augmented krylov subspace methods. Technical Report 95-176, University of Minnesota, Supercomputing Institute, Minneapolis, MN, 1995.

[17] Miloud Sadkane. Block-arnoldi and davidson methods for unsymmetric large eigenvalue problems. *Numer. Math.*, 64(2):195–211, 1993.

[18] D. S. Scott. The advantages of inverted operators in rayleigh-ritz approximations. *SIAM J. Sci. Comput.*, 3:102, 1982.

[19] G. L. G. Sleijpen and H. A. Van der Vorst. A jacobi-davidson iteration method for linear eigenvalue problems. *SIAM J. Matrix Anal. Appl.*, 17(2), 1996.

[20] D. S. Sorensen. Implicit application of polynomial filters in a K-step Arnoldi method. *SIAM J. Matrix Anal. Appl.*, 13(1):357–385, 1992.

[21] Andreas Stathopoulos and Charlotte Fischer. A davidson program for finding a few selected extreme eigenpairs of a large, sparse, real, symmetric matrix. *Computer Physics Communications*, 79(2):268–290, 1994.

[22] Andreas Stathopoulos, Yousef Saad, and Charlotte Fischer. Robust preconditioning of large, sparse, symmetric eigenvalue problems. *Journal of Computational and Applied Mathematics*, 64:197–215, 1995.

[23] J.H. van Lenthe and Peter Pulay. A space-saving modification of davidson's eigenvector algorithm. *J. Comput. Chem.*, 11(10):1164, 1990.

Student paper competition

Splitting the determinants of upper Hessenberg matrices and the Hyman method

Xiulin Zou
Department of Mathematics
Michigan State University
East Lansing, MI. 48824
xzou@math.msu.edu

December 15, 1995

Abstract

In this article, an iterative algorithm is established that splits the evaluation of determinant of an upper Hessenberg matrix into two independent parts so that the evaluation can be done in parallel. This algorithm has application in parallel non-symmetric eigenvalue problems.

1 Introduction

A matrix whose elements below the lower sub-diagonal are all zero is called an upper Hessenberg matrix. Any non-symmetric matrix can be reduced to an upper Hessenberg matrix using a series of orthogonal Householder transformation [2]. That is, any matrix is similar to an upper Hessenberg matrix. So the eigenvalue problem of a general non-symmetric matrix can be reduced to the eigenvalue problem of the corresponding upper Hessenberg matrix. A Homotopy-Determinant algorithm, which is parallelable in nature, has been proposed by Li and Zeng [1] for solving non-symmetric eigenvalue problem. The algorithm heavily depends on the evaluation of the determinant of the upper Hessenberg matrix using Hyman's method. Here we introduce an algorithm to split the determinant evaluation so that computation can be done in parallel. In this paper, we only report the fundamental part: how to split the determinant. Application of this method to the parallel non-symmetric eigenvalue problem will be reported in future paper.

Our method relies on the Hyman's method [2]. The whole idea of the Hyman's method to evaluate the determinant of an upper Hessenberg matrix is to reduce, by row transformation (or column transformation, resp.), the first row (or the last column, resp.) of

the upper Hessenberg matrix into a vector whose elements are all zero except the last one (or the first one, resp.), then the determinant of this upper Hessenberg matrix equals the product of this last (or first, resp.) element of the vector and the lower sub-diagonal elements of the matrix, multiplied by -1 if the order of the matrix is odd.

2 The formula

Let A be an upper Hessenberg matrix of order $k+m$, with the form

$$A = \begin{pmatrix} A_1 & B_1 \\ B_2 & A_2 \end{pmatrix}, \tag{1}$$

where

$$A_1 = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdot & \cdot & \cdot & a_{1k} \\ b_1 & a_{22} & a_{23} & \cdot & \cdot & \cdot & a_{2k} \\ 0 & b_2 & a_{33} & \cdot & \cdot & \cdot & a_{3k} \\ & & \ddots & \ddots & & \vdots & \vdots \\ & 0 & & & b_{k-2} & a_{(k-1)(k-1)} & a_{(k-1)k} \\ & & & & & b_{k-1} & a_{kk} \end{pmatrix}, \tag{2}$$

$$A_2 = \begin{pmatrix} c_{11} & c_{12} & c_{13} & \cdot & \cdot & \cdot & c_{1m} \\ d_1 & c_{22} & c_{23} & \cdot & \cdot & \cdot & c_{2m} \\ 0 & d_2 & c_{33} & \cdot & \cdot & \cdot & c_{3m} \\ & & \ddots & \ddots & & \vdots & \vdots \\ & 0 & & & d_{m-2} & c_{(m-1)(m-1)} & c_{(m-1)m} \\ & & & & & d_{m-1} & c_{mm} \end{pmatrix}, \tag{3}$$

$B_1 = (b_{ij})_{k\times m}$, and

$$B_2 = \begin{pmatrix} 0 & 0 & \cdots & 0 & b \\ 0 & 0 & \cdots & 0 & 0 \\ & & \cdots & & \\ 0 & 0 & \cdots & 0 & 0 \end{pmatrix}_{m\times k}. \tag{4}$$

If one the lower sub-diagonal elements is zero, then the determinant splits naturally. So we assume A is unreduced, that is, none of b_i $(for\ i = 1, \cdots, k)$, b, and d_j $(for\ j = 1, \cdots, m)$ is zero.

First of all, we consider the case where A_1 is non-singular. We will remove this requirement later. Applying the following row transformation to matrix A, we have

$$\begin{pmatrix} I & 0 \\ -B_2A_1^{-1} & I \end{pmatrix}\begin{pmatrix} A_1 & B_1 \\ B_2 & A_2 \end{pmatrix} = \begin{pmatrix} A_1 & B_1 \\ 0 & A_2 - B_2A_1^{-1}B_1 \end{pmatrix} \tag{5}$$

So we have the following lemma.

Lemma 2.1 *If A_1 is nonsingular, then*

$$det(A) = det(A_1)det(A_2 - B_2A_1^{-1}B_1). \tag{6}$$

Denote A_1^{-1} by $(a_{ij}^*)_{k\times k}$. Then the following lemma is easy to verify due to the special structure of B_2.

Lemma 2.2

$$B_2A_1^{-1} = b\begin{pmatrix} a_{k1}^* & a_{k2}^* & \cdots & a_{kk}^* \\ 0 & 0 & \cdots & 0 \\ & \cdot & \cdot & \cdot \\ 0 & 0 & \cdots & 0 \end{pmatrix}. \tag{7}$$

Since $a_{ki}^* = cof(a_{ik})/det(A_1)$, where $cof(a_{ik})$ stands for the cofactor of a_{ik} in matrix A_1, we have.

Lemma 2.3 *If A_1 is non-singular and is of the following form*

$$A_1 = \begin{pmatrix} 0 & 0 & 0 & \cdot & \cdot & 0 & a_{1k} \\ b_1 & a_{22} & a_{23} & \cdot & \cdot & a_{2(k-1)} & a_{2k} \\ 0 & b_2 & a_{33} & \cdot & \cdot & a_{3(k-1)} & a_{3k} \\ & & \ddots & \ddots & & \vdots & \vdots \\ & 0 & & & b_{k-2} & a_{(k-1)(k-1)} & a_{(k-1)k} \\ & & & & & b_{k-1} & a_{kk} \end{pmatrix}, \tag{8}$$

then

$$B_2A_1^{-1}B_1 = \frac{b}{a_{1k}}\begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1m} \\ 0 & 0 & \cdots & 0 \\ \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \cdots & 0 \end{pmatrix}. \tag{9}$$

Proof. If A_1 is of the special form (8), then $det(A_1) = (-1)^{k+1}a_{1k}\prod_{i=1}^{k-1} b_i$, $cof(a_{1k}) = (-1)^{k+1}\prod_{i=1}^{k-1} b_i$ and $cof(a_{ik}) = 0$. So $a_{k1}^* = ((-1)^{k+1}\prod_{i=1}^{k-1} b_i)/((-1)^{k+1}a_{1k}\prod_{i=1}^{k-1} b_i) = \frac{1}{a_{1k}}$, and $a_{ki}^* = 0$ for $i = 2, \cdots, k$. Hence it follows from Lemma 2.2 that

$$B_2A_1^{-1} = \frac{b}{a_{1k}}\begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \cdots & 0 \end{pmatrix},$$

and

$$B_2A_1^{-1}B_1 = \frac{b}{a_{1k}}\begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1m} \\ 0 & 0 & \cdots & 0 \\ & \cdot & \cdot & \cdot \\ 0 & 0 & \cdots & 0 \end{pmatrix}.$$

□

Define

$$A_2' = \begin{pmatrix} b_{11} & b_{12} & b_{13} & \cdot & \cdot & \cdot & & b_{1m} \\ d_1 & c_{22} & c_{23} & \cdot & \cdot & \cdot & & c_{2m} \\ 0 & d_2 & c_{33} & \cdot & \cdot & \cdot & & c_{3m} \\ & & \ddots & \ddots & & \vdots & & \vdots \\ & 0 & & & d_{m-2} & c_{(m-1)(m-1)} & & c_{(m-1)m} \\ & & & & & d_{m-1} & & c_{mm} \end{pmatrix}. \tag{10}$$

That is, A_2' is A_2 with the first row of A_2 replaced by the first row of B_1. We prove the following lemma.

Lemma 2.4 *If A_1 is non-singular and is of the special form (8), then*

$$det(A_2 - B_2 A_1^{-1} B_1) = det(A_2) - \frac{b}{a_{1k}} det(A_2') \tag{11}$$

Proof. If A_1 is of the special form (8), then Lemma 2.3 shows that the first row of the matrix $A_2 - B_2 A_1^{-1} B_1$ is the component-wise difference of the first row of A_2 and the first row of B_1 multiplied by $\frac{b}{a_{1k}}$, while all the other rows of $A_2 - B_2 A_1^{-1} B_1$ are the same as A_2. So the determinant splits into two determinants as stated in the lemma. □

Now we handle the case where A_1 is not necessarily of the special form (8). We still assume A_1 is nonsingular at this stage. Denote the rows of A_1 by $\alpha_1, \alpha_2, \cdots, \alpha_k$. Let e_k be the unit vector of dimension k with the k^{th} element being 1, all others being 0.

Lemma 2.5 *There exist $x_1, x_2, \cdots, x_{k-1}$ such that*

$$\alpha_1 + x_1\alpha_2 + \cdots + x_{k-1}\alpha_k = \alpha e_k, \tag{12}$$

where

$$\alpha = a_{1k} + x_1 a_{2k} + x_2 a_{3k} + \cdots + x_{k-1} a_{kk}. \tag{13}$$

Proof. Let $x_1, x_2, \cdots, x_{k-1}$ be the solution to the following equation system.

$$\begin{aligned} b_1 x_1 + a_{11} &= 0 \\ b_2 x_2 + a_{22} x_1 + a_{12} &= 0 \\ &\cdots \\ b_{k-2} x_{k-2} + \cdots + a_{2(k-2)} x_1 + a_{1(k-2)} &= 0 \\ b_{k-1} x_{k-1} + a_{(k-1)(k-1)} x_{k-2} + \cdots + a_{2(k-1)} x_1 + a_{1(k-1)} &= 0 \end{aligned} \tag{14}$$

Because we have assumed the matrix is unreduced, solution to the above equation system exists and can be found by forward substitution.

Let T_1 be the row transformation that multiplies the $(i+1)^{st}$ row of the unit matrix $I_{k\times k}$ by x_i for each $i = 1, 2, \cdots, k-1$, and add the result to the first row of $I_{k\times k}$. Then T_1 transforms matrix A_1 into the following form, where α is given by (13),

$$T_1 A_1 = \begin{pmatrix} 0 & 0 & \cdots & 0 & \alpha \\ b_1 & a_{22} & \cdots & a_{2(k-1)} & a_{2k} \\ 0 & b_2 & \cdots & a_{3(k-1)} & a_{2k} \\ 0 & 0 & \ddots & & \\ 0 & 0 & \cdots & b_{k-1} & a_{kk} \end{pmatrix}. \tag{15}$$

□

Denote matrix (15) by A_1', i.e. $T_1 A_1 = A_1'$. A_1' is in the special form (8). The transformation T_1 does not change the determinant of A_1. So the following lemma is obvious.

Lemma 2.6

$$det(A_1) = (-1)^{k+1}\alpha \prod_{i=1}^{k-1} b_i, \tag{16}$$

where α is given by formula (13). Furthermore, A_1 is singular if and only if $\alpha = 0$.

Apply the same row transformation T_1 to B_1, and denote the new matrix by B_1', i.e., $T_1 B_1 = B_1'$. Let the first row of matrix B_1' be $(b_{11}', b_{12}', \cdots, b_{1m}')$. Then

$$b_{1i}' = b_{1i} + \sum_{j=2}^{k} x_{j-1} b_{ji}, \ i = 1, 2, \cdots, m. \tag{17}$$

In the following theorem, A_1 may be singular, that is, the theorem is true no matter whether A_1 is singular or not.

Theorem 2.7 *For an upper Hessenberg matrix A of the form (1),*

$$det(A) = (-1)^{k+1}\left(\prod_{i=1}^{k-1} b_i\right)\left(\alpha \cdot det(A_2) - b \cdot det(A_2'')\right), \tag{18}$$

where A_2'' is A_2 with the first row replaced by the first row of B_1' (see (17)), and α is given by (13).

Proof. First of all, we assume A_1 is nonsingular. Since the row transformation T_1 does not change the determinant of matrix A, we have

$$det(A) = det\left(\begin{pmatrix} T_1 & 0 \\ 0 & I \end{pmatrix}\begin{pmatrix} A_1 & B_1 \\ B_2 & A_2 \end{pmatrix}\right)$$

$$= det\begin{pmatrix} A_1' & B_1' \\ B_2 & A_2 \end{pmatrix}.$$

Since A_1' is of the special form (8), we apply the previous lemmas to obtain the following

$$det\begin{pmatrix} A_1' & B_1' \\ B_2 & A_2 \end{pmatrix} = det(A_1')det(A_2 - B_2 A_1'^{-1} B_1')$$

$$= (-1)^{k+1}\alpha \prod_{i=1}^{k-1} b_i \left(det(A_2) - \frac{b}{\alpha}\, det(A_2'') \right).$$

$$= (-1)^{k+1}(\prod_{i=1}^{k-1} b_i)\left(\alpha \cdot det(A_2) - b \cdot det(A_2'')\right).$$

Now we prove the theorem for A_1 (or A_1') singular. If A_1 is singular, then the above $\alpha = 0$ and we can not directly apply the previous lemmas which require A_1' to be nonsingular. However, we can perturb matrix A_1 to make it nonsingular. Consider matrix $A_{1\varepsilon}$, which is matrix A_1 with the element a_{1k}(on the upper-right corner of A_1) perturbed by $\varepsilon \neq 0$. That is

$$A_{1\varepsilon} = A_1 + \varepsilon e_1 e_k^T, \tag{19}$$

$A_{1\varepsilon}$ is non-singular, and

$$T_1 A_{1\varepsilon} = \begin{pmatrix} 0 & 0 & \cdots & 0 & \alpha_\varepsilon \\ b_1 & a_{22} & \cdots & a_{2(k-1)} & a_{2k} \\ 0 & b_2 & \cdots & a_{3(k-1)} & a_{2k} \\ 0 & 0 & \ddots & & \\ 0 & 0 & \cdots & b_{k-1} & a_{kk} \end{pmatrix},$$

where $\alpha_\varepsilon = \alpha + \varepsilon (= \varepsilon$ actually, since $\alpha = 0$ if A_1 is singular). Let A_ε be matrix A with A_1 replaced by $A_{1\varepsilon}$. Now we can apply the previous lemmas and methods to matrix A_ε to obtain

$$det(A_\varepsilon) = (-1)^{k+1} \prod_{i=1}^{k-1} b_i \left(\alpha_\varepsilon \cdot det(A_2) - b \cdot det(A_2'')\right). \tag{20}$$

Let $\varepsilon \to 0$, we obtain (18) □

We proceed to evaluate $det(A_2)$ and $det(A_2'')$ using the Hyman's method as what we did for $det(A_1)$ in Lemmas 2.5 and 2.6. However, column transformation is used this time. Since A_2 and A_2'' only differs by the first row, we can find a common column transformation to reduce the elements of the last column of A_2 and A_2'' to zeros, except the first element. Denote the columns of matrix A_2 by $\beta_1, \beta_2, \cdots, \beta_m$, and the columns of matrix A_2'' by $\gamma_1, \gamma_2, \cdots, \gamma_m$.

Lemma 2.8 *There exist* $y_1, y_2, \cdots, y_{m-1}$ *such that*

$$\beta_m + y_1\beta_1 + \cdots + y_{m-1}\beta_{m-1} = \beta e_1, \tag{21}$$

$$\gamma_m + y_1\gamma_1 + \cdots + y_{m-1}\gamma_{m-1} = \gamma e_1, \tag{22}$$

where e_m *is the unit vector of dimension* m *with the first element being* 1, *all other elements being* 0 *and*

$$\beta = c_{11}y_1 + c_{12}y_2 + \cdots + c_{1(m-1)}y_{m-1} + c_{1m} \tag{23}$$

$$\gamma = b'_{11}y_1 + b'_{12}y_2 + \cdots + b'_{1(m-1)}y_{m-1} + b'_{1m} \tag{24}$$

Proof. Let $y_1, y_2, \cdots, y_{m-1}$ be the solution to the following equation system.

$$\begin{aligned} d_1y_1 + c_{22}y_2 + \cdots + c_{2(m-1)}y_{m-1} + c_{2m} &= 0 \\ d_1y_2 + \cdots + c_{3(m-1)}y_{m-1} + c_{3m} &= 0 \\ \vdots & \\ d_{m-1}y_{m-1} + c_{mm} &= 0 \end{aligned} \tag{25}$$

Since $\prod_{i=1}^{m-1} d_i \neq 0$, there is a unique solution to the above system and the solution can be found by backward substitution. □

Let T_2 be the column transformation that multiplies the i^{th} column of the unit matrix $I_{m\times m}$ by y_i, for each $i = 1, 2, \cdots, m-1$, and add the result to the last column of $I_{m\times m}$. Then it follows from Lemma 2.8 that

$$T_2A_2 = \begin{pmatrix} c_{11} & c_{12} & c_{13} & \cdot & \cdot & \cdot & \beta \\ d_1 & c_{22} & c_{23} & \cdot & \cdot & \cdot & 0 \\ 0 & d_2 & c_{33} & \cdot & \cdot & \cdot & 0 \\ & & \ddots & \ddots & & \vdots & \vdots \\ & \mathbf{0} & & & d_{m-2} & c_{(m-1)(m-1)} & 0 \\ & & & & & d_{m-1} & 0 \end{pmatrix}, \tag{26}$$

and

$$T_2A''_2 = \begin{pmatrix} b'_{11} & b'_{12} & b'_{13} & \cdot & \cdot & \cdot & \gamma \\ d_1 & c_{22} & c_{23} & \cdot & \cdot & \cdot & 0 \\ 0 & d_2 & c_{33} & \cdot & \cdot & \cdot & 0 \\ & & \ddots & \ddots & & \vdots & \vdots \\ & \mathbf{0} & & & d_{m-2} & c_{(m-1)(m-1)} & 0 \\ & & & & & d_{m-1} & 0 \end{pmatrix}. \tag{27}$$

So the following lemma is obvious.

Lemma 2.9

$$det(A_2) = (-1)^{m+1}\beta \prod_{i=1}^{m-1} d_i, \tag{28}$$

$$det(A''_2) = (-1)^{m+1}\gamma \prod_{i=1}^{m-1} d_i, \tag{29}$$

So Theorem 2.7 can be stated as follows

Theorem 2.10

$$det(A) = (-1)^{k+m}(\prod_{i=1}^{k-1} b_i)(\prod_{j=1}^{m-1} d_j)(\alpha \cdot \beta - b \cdot \gamma), \tag{30}$$

where α, β and γ are computed according to (13), (23) and (24) respectively.

3 The algorithm

Let A be an upper Hessenberg matrix of form (1). We describe the algorithm for computing the determinant of A in the following.

1 Solve equation system (15) for $(x_1, x_2, \cdots, x_{k-1})$. and compute $(b'_{11}, b'_{12}, \cdots, b'_{1m})$ according to (17).

2 Solve equation system (25) for $(y_1, y_2, \cdot, y_{m-1})$.

3 Compute α, β and γ according to (13), (23) and (24) respectively.

4 Compute $det(A)$ according to (30).

Note that step 1 and step 2 are independent of each other, hence can be done in parallel. The number of multiplications and additions in step 1 is $\frac{1}{2}(k-1)(k-2)+m(k-1)$, the number of divisions in step 1 is $k-2$. The number of multiplications and additions in step 2 is $\frac{1}{2}(m-1)(m-2)$, the number of divisions in step 2 is $m-2$. The number of multiplications and additions in step 3 is $(k-1)+2(m-1)$ The number of multiplications in step 4 is $k+m+1$.

4 Load Balance

If we want to compute the determinant of an upper Hessenberg matrix by the above algorithm concurrently on two processors, we would like to split it in such a way that the computation load is approximately the same for each processor. Assume the order of the upper Hessenberg matrix A is n. We choose a sub-diagonal element $a_{(k+1)k}$ as the anchoring element to split the determinant. Then step 1 of the algorithm involves $\frac{1}{2}(k-1)(k-2)+(n-k)(k-1)$ multiplications and additions, and $k-2$ divisions. Step 2 of the algorithm, which is carried out at the same time as step 1 on a different processor, involves $\frac{1}{2}(n-k-1)(n-k-2)$ multiplications and additions, and $n-k-2$ divisions. We want to choose k such that

$$\frac{1}{2}(k-1)(k-2)+(n-k)(k-1)+k-2 \approx \frac{1}{2}(n-k-1)(n-k-2)+n-k-2.$$

Solve the above equation for approximate integer solution k. The solution tells us where to split the determinant.

References

[1] T. Y. Li and Z. Zeng, *Homotopy-determinant algorithm for solving non-symmetric eigenvalue problem* , Mathematics of Computation, Vol. 59, No. 200 (1992), pp. 483-502.

[2] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Oxford University Press, Oxford, 1965.

A COMBINED MODIFICATION OF NEWTON'S METHOD FOR SYSTEMS OF NONLINEAR EQUATIONS

M.T. Monteiro and Edite M.G.P. Fernandes
Universidade do Minho, Braga, Portugal

Abstract: To improve the performance of Newton's method for the solution of systems of nonlinear equations a modification to the Newton iteration is implemented. The modified step is taken as a linear combination of Newton step and steepest descent directions. In the paper we describe how the coefficients of the combination can be generated to make effective use of the two component steps. Numerical results that show the usefulness of the combined modification are presented.

1. Introduction

For relatively small problems Newton's method seems to be a good choice for solving systems of nonlinear equation,

$$F(x) = 0 \tag{1}$$

where $x \in R^n$, $F : \Gamma \to R^n$, $\Gamma \subset R^n$ and $F_i, i = 1, \ldots, n$ are nonlinear continuous functions of x. Let $\|.\|$ denote a norm on R^n.

If the functions F_i are differentiable at x, we define the Jacobian matrix $\nabla F(x) : \Gamma \to R^{n \times n}$ by

$$\nabla F(x)_{ij} = \frac{\partial F_i(x)}{\partial x_j} \quad i, j = 1, \ldots, n.$$

We shall always assume that the following two assumptions on F hold:

Assumption A1 : Equation (1) has a solution x^*.

Assumption A2: The Jacobian at the solution is nonsingular.

The affine approximation to F at $x^{(k)} + d$ is $l(x^{(k)} + d) = F(x^{(k)}) + \nabla F(x^{(k)})d$. The step d that makes $l(x^{(k)} + d) = 0$ defines the Newton iteration for solving (1)

$$d_N = -(\nabla F(x^{(k)}))^{-1} F(x^{(k)}), \quad k = 0, 1, \ldots \tag{2}$$

where $x^{(k+1)} = x^{(k)} + d_N$. It is known that (2) is locally quadratically convergent to x^* (see proof in Dennis and Schnabel (1983)).

It is reasonable to expect that $\|F(x^{(k+1)})\|$ should be less that $\|F(x^{(k)})\|$, for some norm. If the 2-norm is used then a decrease of $\|F(x)\|_2$ is required at each iteration and our attention could be focussed on the related problem

$$min_{x \in R^n} f(x) = \frac{1}{2} F(x)^T F(x). \tag{3}$$

This function f takes on its least value at a solution point of (1). Therefore, in finding the minimum for f, we could find a solution of $F = 0$. The Newton step (2) also minimizes the quadratic function

$$m(x^{(k)} + d) = \frac{1}{2} F(x^{(k)})^T F(x^{(k)}) + [\nabla F(x^{(k)})^T F(x^{(k)})]^T d + \frac{1}{2} d^T \nabla F(x^{(k)})^T \nabla F(x^{(k)}) d \tag{4}$$

which is not the quadratic model approximation to $f(x)$. When the Jacobian ∇F is singular or badly conditioned, the Newton iteration does not give a reliable step and need not converge. In this paper we are concerned with an algorithm which improves the Jacobian conditioning, retains as much as possible the local convergence property of Newton's method and introduces a global convergence characteristic. It is based on the Newton equation and combines Newton and steepest descent directions for f for solving (1).

The paper is organized as follows. In the next Section the modified Newton equation is defined and some properties are established. Results of some comparative experiments are discussed in Section 3 and Section 4 is devoted to concluding remarks.

2. The combined modification of the Newton equation

In this section we describe the modified Newton step direction and state the main result of the algorithm.

Definition 1 : The vector $d \in R^n$ is a descent direction for $f(x)$, in (3), at a point x if it satisfies

$$F(x)^T \nabla F(x)\, d < 0$$

where $\nabla F(x)^T F(x)$ is the gradient of f computed at x.

Definition 2 : The direction d_{sd} defined by

$$d_{sd} = -\nabla F(x)^T F(x) \tag{5}$$

is called the steepest descent direction for f to solve $F = 0$.

Definition 3 : The direction d_{GN} is said to be the Gauss-Newton direction for f if

$$d_{GN} = -(\nabla F(x)^T \nabla F(x))^{-1} \nabla F(x)^T F(x). \tag{6}$$

Since the Gauss-Newton direction is the step which goes to the minimum of the model (4), from now on it would be referred to as Newton step. Since forming the matrix $\nabla F^T \nabla F$ can square the conditioning of the problem in comparison to a method that uses ∇F directy, the structure of equation (2) for d_N will be used instead.

Definition 4 : Let ∇F satisfy Assumptions A1 and A2. A function defined by

$$w(x) = w_0(x) - \gamma w_0(x) \tag{7}$$

where

$$w_0(x) = \frac{F(x)^T (\nabla F(x) \nabla F(x)^T) F(x)}{F(x)^T (\nabla F(x) \nabla F(x)^T - I) F(x)} \tag{8}$$

is said to be a weight function, for some positive constant $0 < \gamma < 1$.

Definition 5 : Let $\alpha_1(x)$ and $\alpha_2(x)$ be continuous coefficient functions on R^n. A direction d_{MN} is said to be a modified Newton step direction if

$$d_{MN} = \alpha_1(x) d_N + \alpha_2(x) d_{sd} \tag{9}$$

where $\alpha_1(x) = w(x), \alpha_2(x) = 1 - w(x)$ and $w(x)$ is the weight function of (7).

A similar linear combination has been implemented to optimization problems with constraints (Fernandes (1995)).

Proposition : The direction d_{MN} is a descent direction for f at x to solve (1), provided that Assumption 2 is verified.

Proof: From (9) with d_N of (2) and d_{sd} of (5), we have

$$(\nabla F(x)^T F(x))^T d_{MN} = -[w(x)F(x)^T F(x) + (1-w(x))F(x)^T \nabla F(x) \nabla F(x)^T F(x)]$$

which using (7) and (8) yields

$$= -[\gamma \frac{(F(x)^T \nabla F(x) \nabla F(x)^T F(x))(F(x)^T (\nabla F(x) \nabla F(x)^T - I) F(x))}{F(x)^T (\nabla F(x) \nabla F(x)^T - I) F(x)}]$$

or

$$= -\gamma F(x)^T \nabla F(x) \nabla F(x)^T F(x)$$

which is negative as long as ∇F is nonsingular.

At this point two step directions could be used, the Newton or the Modified Newton directions. Progress towards the solution is measured in terms of the values of the non-linear functions. To decide which step direction should be used at each iteration we compare $\|F(x^{(k)} + d_N)\|_2$ with $\|F(x^{(k)} + d_{MN})\|_2$. The step direction which gives the least value of F is accepted.

When $\nabla F(x)$ is nearly singular, the Newton direction should not be computed, $w(x) \approx 0$ from (8) and $d_{MN} = -\nabla F(x)^T F(x)$ is the steepest descent direction. A step in this direction minimizes the quadratic function

$$\hat{m}(x^{(k)} + d) = \frac{1}{2} F(x^{(k)})^T F(x^{(k)}) + [\nabla F(x^{(k)})^T F(x^{(k)})]^T d + \frac{1}{2} d^T H(x^{(k)}) d$$

where $H(x^{(k)})$ is the identity matrix. This model proved to be unsatisfactory and a perturbation model is used instead. The $H(x)$ is now $\nabla F(x)^T (\frac{\nabla F(x)^T}{\|\nabla F(x)\|_\infty} + I)^{-1}$ and this corresponds to the following step

$$d = -(\frac{\nabla F(x)^T}{\|\nabla F(x)\|_\infty} + I) F(x). \tag{10}$$

3. Comparative experiments

In order to evaluate the efficiency of combined modification Newton's algorithm some numerical experiments were performed based on twenty seven test problems. For some of them more than one starting approximation were used. These are shown in the Appendix and the results are summarized in the Table. The column identified with $nit(MN)$ contains the number of iterations taken by the combined modification Newton's algorithm. The first term corresponds to accepted Newton directions and the second to Modified Newton directions. For comparative purposes, the table contains columns with results of a basic Newton implementation, $nit(N)$, and results taken from a report by Machado and Pereira (1995) where a Broyden secant method was implemented, $nit(B)$.

In the table, *nprob* corresponds to the problem number as in the Appendix, n is the number of equations, a) means singularity of the Jacobian and FC means failure to converge within 50 iterations. nfe gives the number of function evaluations and $\|F\|_\infty$ is the least value attained by F when the algorithm terminated due to the following stopping criteria (Dennis and Schnabel (1983)):

$$max_{1\leq i\leq n}|F_i(x^{(k+1)})| \leq tol \quad \text{and} \quad max_{1\leq i\leq n}\frac{|x_i^{(k+1)} - x_i^{(k)}|}{max\{|x_i^{(k+1)}|, typx_i\}} \leq tol$$

where $tol = 10^{-9}$. $typx_i$ is the user estimate of a typical magnitude of x_i.

4. Concluding remarks

The following points are worthy of mention.

(i) There are problems (5, 10, 15, 17 and 18) in which combined modification Newton's algorithm required less iterations than Newton's algorithm.

(ii) Jacobian singularity has occurred on problems 8, 16 and 19. The step defined in (10) solved them successfully.

(iii) On the remaining problems the overall number of iterations is the same for the two Newton based algorithms.

(iv) There were three cases where the combined modification Newton's algorithm failed to converge.

One run of problem 8 required a backtracking scheme to prevent large steps d to be taken. The nfe increased to 104. We conclude this paper with the following comment about the γ constant in the definition of the weight function. We have tested three values $0.1, 0.001$ and 10^{-9}. For some problems the value provided could be critical for convergence purposes.

References

1. J.E. Dennis and R.B. Schnabel, *Numerical methods for unconstrained optimization and nonlinear equations*, Prentice-Hall Inc., 1983.
2. L. Eldén and L. Wittmeyer-Koch, *Numerical analysis. An introduction*, Academic Press Inc., 1990.
3. E.M.G.P. Fernandes, *Newton based exact penalty techniques for nonlinear optimization with constraints*, in U. Derigs, A. Bachem and A. Drexl (Eds.), Operations Research Proceedings 1994 (39-44), Springer-Verlag, 1995.
4. D. Kahaner, C. Moler and S. Nash, *Numerical methods and software*, Prentice-Hall Inc., 1989.
5. G.J. Lastman and N.K. Sinha, *Microcomputers-based numerical methods for science and engineering*, Holt, Rinehart and Winston, Inc., 1989.
6. G.J.B.Q.A Machado and R.M.S. Pereira, *Método da secante - Equação de Broyden*, Internal Report, Mestrado em Matemática Computacional, Braga, 1995.
7. J.J. Moré, B.S. Garbow and K.E. Hillstrom, *Testing unconstrained optimization software*, ACM Transactions on Mathematical Software, Vol 7, No. 1, 1981.

TABLE - Numerical results

nprob	n	$nit(N)$	$nit(B)$	$nit(MN)$	γ	nfe	$\|F\|_\infty$
1	2	4	5	3+1	0.001	8	.25D-17
2	2	5	9	5+0	0.1	10	.33D-15
		7	11	5+2	0.1	14	
3	2	6	7	6+0	0.1	12	.00D00
		7		7+0	10^{-9}	14	
4	2	6	8	6+0	0.1	12	.00D00
5	2	7	12	7+0	0.1	14	.00D00
		9		9+0	0.1	18	
		8	18	7+1	0.001	16	
		overfl.		11+8	0.001	38	.16D-15
6	2	6	11	6+0	0.1	12	.00D00
7	2	13	15	FC			.21D-16
		6		6+2	10^{-9}	16	.58D-16
8	2	*a*)	6	3+3	10^{-9}	12	.51D-16
		a)		9+27	10^{-9}	104	
9	2	13		13+0	0.001	26	.26D-16
		14		13+1	10^{-9}	28	
10	2	6	10	6+0	0.1	12	.39D-15
		15	15	13+1	0.1	28	.10D-14
11	4	7	17	7+0	0.1	14	.10D-14
		14	FC	13+1	0.001	28	.89D-14
12	2	5	FC	5+0	0.1	10	.42D-14
13	3	6	12	6+0	0.1	12	.54D-19
		15	FC	11+4	10^{-9}	30	.00D00
14	3	8	8	8+0	10^{-9}	16	.15D-15
15	2	7		6+1	0.1	14	.12D-15
		25		6+2	0.1	16	
		5		5+0	0.1	10	
16	2	*a*)		8+4	0.001	24	.00D00
		8		8+0	0.1	16	
17	3	20		9+1	0.001	20	.60D-15/.15D-15
		5		5+0	0.1	10	.00D00
		30		12+3	0.001	30	.24D-16/.12D-15
18	3	8		7+1	0.001	16	.00D00
		60		34+4	10^{-9}	76	.60D-18
		FC		15+12	10^{-9}	54	.49D-33
		26		20+1	10^{-9}	42	.00D00
19	2	*a*)		10+1	10^{-9}	22	.11D-15
20	2	8		8+0	0.1	16	.86D-16
21	2	FC		FC			
22	4	32		32+0	0.1	64	.69D-18
23	3	7		6+1	0.1	14	.17D-17
24	3	4		4+0	0.1	8	.22D-16
25	3	4		4+0	0.1	8	.15D-16
26	2	6		FC			.15D-16
27	3	5		5+0	0.1	10	.23D-15

Appendix - Test functions

1. $F_1(x) = 0.2sin(x_1 + x_2) - x_1$
   $F_2(x) = 0.2cos(x_1 - x_2) - x_2$
   $x_0 = (0.1, 0.2)^T$
2. $F_1(x) = -5x_1 + 2sin(x_1) + cos(x_2)$
   $F_2(x) = 4cos(x_1) + 2sin(x_2) - 5x_2$
   $x_0 = (1, 1)^T; (10, 10)^T$
3. $F_1(x) = x_1 + x_2 - 3$
   $F_2(x) = x_1^2 + x_2^2 - 9$
   $x_0 = (1, 5)^T; (2, 7)^T$
4. $F_1(x) = e^{x_1} - 1$
   $F_2(x) = e^{x_2} - 1$
   $x_0 = (1, 1)^T$
5. $F_1(x) = x_1^2 + x_2^2 - 2$
   $F_2(x) = e^{x_1 - 1} + x_2^3 - 2$
   $x_0 = (1.5, 2)^T; (0.5, 0.5)^T; (2, 3)^T; (2, 0.5)^T$
6. $F_1(x) = 2(x_1 + x_2)^2 + (x_1 - x_2)^2 - 8$
   $F_2(x) = 5x_1^2 + (x_2 - 3)^2 - 9$
   $x_0 = (2, 0)^T$
7. $F_1(x) = 0.1x_1x_2 - 1$
   $F_2(x) = e^{-10^{-5}x_1} + e^{-x_2} - 1.0001$
   $x_0 = (0, 1)^T; (0, 10)^T$
8. $F_1(x) = 10^{-5}x_1x_2 - 1$
   $F_2(x) = e^{-10^{-10}x_1} + e^{-10x_2} - 1.0001$
   $x_0 = (0, 1)^T; (0, 0.1)^T$
9. Powell badly scaled function
   $F_1(x) = 10^4x_1x_2 - 1$
   $F_2(x) = e^{-x_1} + e^{-x_2} - 1.0001$
   $x_0 = (0, 1)^T$
10. $F_1(x) = x_1^2 + x_2^2 - 5$
    $F_2(x) = e^{x_1} + x_2 - 3$
    $x_0 = (1, 1)^T; (10, 10)^T$
11. $F_1(x) = x_1^2 - x_2^2 + x_3^2 - x_4^2 - 5$
    $F_2(x) = 2x_1x_2 + 2x_3x - 4$
    $F_3(x) = e^{x_1}cos(x_2) + x_3 - 3$
    $F_4(x) = e^{x_1}sin(x_2) + x_4$
    $x_0 = (1, 1, 1, 1)^T; (10, 10, 10, 10)^T$
12. $F_1(x) = 10x_1^{1.5} + 20(x_1 - x_2) - 75$
    $F_2(x) = 30x_2 - 20x_1 + 20e^{0.1x_2} - 25$
    $x_0 = (5, 10)^T$
13. $F_1(x) = x_1$
    $F_2(x) = x_2^2 + x_2$
    $F_3(x) = e^{x_3} - 1$
    $x_0 = (1, 1, 1)^T; (10, 10, 10)^T$
14. $F_1(x) = 3.1x_2 - cos(x_1x_3) - 0.6$
    $F_2(x) = 1.03e^{-x_1x_2} + 19.5x_3 + 11$
    $F_3(x) = x_2^2 - 83(x_1 + 0.11)^2 + sin(x_3) + 0.97$
    $x_0 = (1, 1, 1)^T$
15. $F_1(x) = x_1^2 - x_2 - 1$
    $F_2(x) = (x_1 - 2)^2 + (x_2 - 0.5)^2 - 1$
    $x_0 = (0, 0)^T; (0.1, 2)^T; (1, 0)^T$
16. $F_1(x) = x_1x_2 - x_2^3 - 1$
    $F_2(x) = x_1^2x_2 + x_2 - 5$
    $x_0 = (0, 0)^T; (2, 3)^T$
17. $F_1(x) = x_1cos(x_2) - x_3$
    $F_2(x) = x_1^2 + x_3$
    $F_3(x) = e^{x_1 + x_3}sin(\frac{x_2}{2}) + x_3$
    $x_0 = (1, 2, 3)^T; (0.1, 0.1, 0.1)^T; (0.5, 2, -0.5)^T$
18. $F_1(x) = x_1^5 + x_2^3x_3^4 + 1$
    $F_2(x) = x_1^2x_2x_3$
    $F_3(x) = x_3^4 - 1$
    $x_0 = (1, 1, 1)^T; (0.1, 0.1, 0.1)^T;$
    $(-0.01, -0.01, -0.01)^T; (-100, 0, 100)^T$
19. $F_1(x) = x_1^4 + x_2^4 - 6$
    $F_2(x) = e^{-x_1} + sin(x_2) - 2$
    $x_0 = (0, 0)^T$
20. $F_1(x) = x_1 - 1 - 0.005(e^{x_2\sqrt{x_1}} + 5x_1^3x_2^2)$
    $F_2(x) = x_2 + 2 - 0.005tan(e^{x_2} + x_1^2)$
    $x_0 = (1, -2)^T$
21. $F_1(x) = 10(x_2 - x_1^2)$
    $F_2(x) = 1 - x_1$
    $x_0 = (-1.2, 1)^T$
22. Powell singular function
    $F_1(x) = x_1 + 10x_2$
    $F_2(x) = \sqrt{5}(x_3 - x_4)$
    $F_3(x) = (x_2 - 2x_3)^2$
    $F_4(x) = \sqrt{10}(x_1 - x_4)^2$
    $x_0 = (3, -1, 0, 1)^T$

23. Brown almost linear function

$F_1(x) = 2x_1 + x_2 + x_3 - 4$

$F_2(x) = x_1 + 2x_2 + x_3 - 4$

$F_3(x) = x_1x_2x_3 - 1$

$x_0 = (0.5, 0.5, 0.5)^T$

24. Discrete boundary value function

$F_1(x) = 2x_1 - x_2 + 0.03125(x_1 + 1.25)^3$

$F_2(x) = -x_1 + 2x_2 - x_3 + 0.03125(x_2 + 1.5)^3$

$F_3(x) = -x_2 + 2x_3 + 0.03125(x_3 + 1.75)^3$

$x_0 = (-0.1875, -0.25, -0.1875)^T$

25. Discrete integral equation function

$F_1(x) = x_1 + 0.0234375(x_1 + 1.25)^3 + 0.015625(x_2 + 1.5)^3 + 0.0078125(x_3 + 1.75)^3$

$F_2(x) = x_2 + 0.015625(x_1 + 1.25)^3 + 0.03125(x_2 + 1.5)^3 + 0.015625(x_3 + 1.75)^3$

$F_3(x) = x_3 + 0.0078125(x_1 + 1.25)^3 + 0.015625(x_2 + 1.5)^3 + 0.0234375(x_3 + 1.75)^3$

$x_0 = (-0.1875, -0.25, -0.1875)^T$

26. Trigonometric function

$F_1(x) = 3 - 2cos(x_1) - cos(x_2) - sin(x_1)$

$F_2(x) = 4 - 3cos(x_2) - cos(x_1) - sin(x_2)$

$x_0 = (0.5, 0.5)^T$

27. Broyden tridiagonal function

$F_1(x) = (3 - 2x_1)x_1 - 2x_2 + 1$

$F_2(x) = -x_1 + (3 - 2x_2)x_2 - 2x_3 + 1$

$F_3(x) = -x_2 + (3 - 2x_3)x_3 + 1$

$x_0 = (-1, -1, -1)^T$

Topic: *Student Papers Winners*	*Session Chair:* *T. Manteuffel & S. McCormick*	*Room A*
4:45 - 5:15	S. Knapek (3rd)	Matrix-Dependent Multigrid-Homogenization for Diffusion Problems
5:15 - 5:45	M. Horn (2nd)	A Superlinear Convergence Estimate for an Iterative Method for the Biharmonic Equation
5:45 - 6:15	A. Klawonn (1st)	Triangular Preconditioners for Saddle Point Problems with a Penalty Term

MATRIX-DEPENDENT MULTIGRID-HOMOGENIZATION FOR DIFFUSION PROBLEMS

S. KNAPEK
ADVISER: M. GRIEBEL
INSTITUT FÜR INFORMATIK
TU MÜNCHEN
D-80290 MÜNCHEN

Abstract. We present a method to approximately determine the effective diffusion coefficient on the coarse scale level of problems with strongly varying or discontinuous diffusion coefficients. It is based on techniques used also in multigrid, like Dendy's matrix-dependent prolongations and the construction of coarse grid operators by means of the Galerkin approximation. In numerical experiments, we compare our multigrid-homogenization method with homogenization, renormalization and averaging approaches.

1. Introduction. Exact solutions for problems, which model locally strong varying phenomena on a micro-scale level, require that all length scales appearing in the problem are completely resolved. In numerical simulation however, due to reasons of storage requirements and numerical complexity, this requirement can often not be met, i.e. the grid for numerical simulation can not be chosen sufficiently fine enough. A typical example for such kind of problem is the diffusion equation with strongly varying diffusion coefficient, as it arises from Darcy's law in reservoir simulation and related problems for flow in porous media. There, the diffusion coefficient models the local permeability of the medium.

However, in many practical applications, the fine-scale details of the solution are not of interest, but merely a coarse-scale solution is sought. Therefore, it is necessary, to work with averaged equations, which describe directly the large-scale behavior of the problem under consideration. Thus, the first step in the direction of a numerical treatment of the coarse scale problem is the determination of equations which model the influence of the unresolved fine scales sufficiently.

In the earlier times of numerical simulation of reservoir performance this upscaling process was done by simply averaging the (measured or randomly generated) fine-scale diffusion coefficients using the arithmetic, geometric or harmonic mean. However, this approach turned out to be invalid for systems with strong permeability variations. Therefore, more advanced techniques like renormalization or homogenization are used nowadays.

The discretization of an elliptic differential equation can also be interpreted as a certain kind of averaging or filtering, where smaller scales than the respective grid size are eliminated. In multigrid methods, discretizations on successive coarser grids are needed. To obtain them, beside direct discretization on these grids, the Galerkin coarse grid discretization method is used frequently. It involves the discretization on the finest grid only and produces the operators and equations on the coarser levels relatively to it by only using an interpolation operator between two successive grids. This corresponds

basically to just an averaging of the fine grid equations. Now, in the multigrid field, there exist methods to determine the interpolation involved in the Galerkin coarsening in a *matrix-dependent* way. This results in stable and "physically meaningful" coarse grid operators which are necessary for good convergence rates of the corresponding multigrid method also in the case of interface problems and singular perturbed operators.

Thus, on the one hand, methods known from modelling (homogenization, renormalization) could be used for the determination of coarse grid operators in multigrid methods, by directly discretizing the homogenized or renormalized continuous equations, and using them as coarse grid operators. On the other hand, matrix-dependent Galerkin approximations used in some multigrid methods to obtain robust solvers, lead to coarse grid operators, that directly describe the behavior of the coarse scale solution. These discrete coarse grid operators then lead to approximations of the effective diffusion equation, by interpreting them as discrete coarse scale operators. This is illustrated in Figure 1.

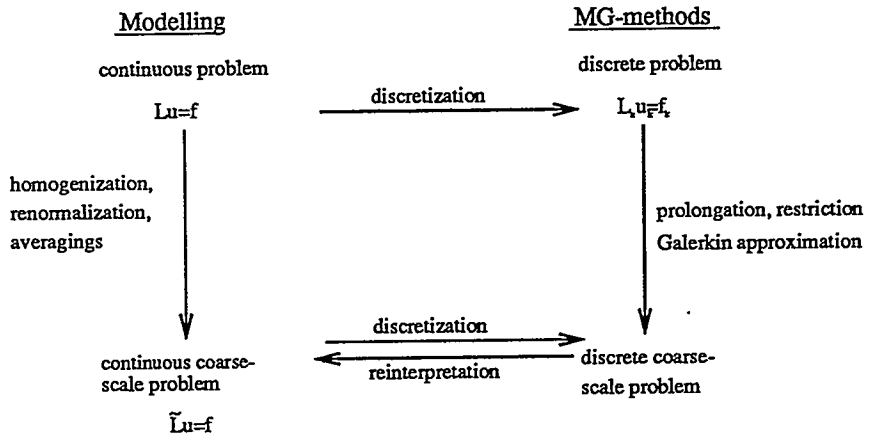


FIG. 1. *Relation between mathematical modelling methods and multigrid coarsening*

In this paper, we present the idea to use such multigrid Galerkin coarsening methods to derive a technique for modelling the effective, i.e. coarse-scale diffusion coefficient. It gives results that are better than the ones obtained with renormalization and nearly as accurate as the ones obtained with homogenization. However, our method can also be applied in the case of non-periodic structures where homogenization is not more directly applicable.

First, we consider a simple diffusion problem and describe the approaches and problems related to the mathematical modelling of coarse scale equations. Then, we consider the multigrid method and describe how matrix-dependent prolongations and Schur complement approximations lead to energy dependent averaging procedures and to averaged equations. Finally, we give some results from numerical experiments for the diffusion equation with different types of diffusion fields and compare our new method with averaging, renormalization and homogenization.

2. The problem of effective diffusion on large scales. We consider the model problem

$$(1) \qquad \begin{array}{cc} -\nabla(D\nabla u) = f & \text{in } \Omega = [0,1]^2 \\ u = 0 & \text{on } \delta\Omega \end{array} \quad \text{where } D = \begin{pmatrix} d_{11}(x,y) & d_{12}(x,y) \\ d_{21}(x,y) & d_{22}(x,y) \end{pmatrix}.$$

In porous media flow, under the assumption $\vec{g} = 0$ (zero gravity), the conservation of mass, the slight incompressibility assumption $u = u_0 e^{cp}$ and the restriction to the stationary case, this equation results results from Darcy's law $\vec{q} = -\frac{D}{\mu}(\nabla p - u\vec{g})$.[1]

There, the diffusion coefficient matrix D describes the permeability of the medium. In general, D is not continuous, highly varying and anisotropic. For isotropic media, D is a diagonal matirix with $\text{diag}(D) =: d \cdot I$, where d is a scalar function[2].

For locally strong varying D, a problem arises when it comes to numerical simulation: Due to storage requirements and numerical complexity, the grid Ω_k used in the discretization process can in general not be made fine enough to fully resolve the diffusion coefficient D. However, since fine scale details of the solution are usually not of interest, only a coarse scale solution is sought anyway. Therefore, the task of mathematical modelling is now to find a coarse scale problem describing directly the coarse scale solution of the problem. Thus, we are left with the problem to determine a so-called *effective* diffusion coefficient $\tilde{D}$, where

- $\tilde{D}$ is slowly varying or even constant,
- $\nabla\tilde{D}\nabla$ is a direct description of the problem on the coarse scale,
- $\tilde{D}$ takes the influence of the unresolved fine scales as good as possible into account.

Replacing D by $\tilde{D}$ in (1) then gives a continuous coarse scale problem, whose solution directly describes u on a coarse scale.

One way to gain an approximation of the effective diffusion coefficient is by simple averaging. There, for example the arithmetic mean

$$(2) \qquad d_{i,j}^{arith} = \frac{1}{|U|}\int_U d_{i,j}(x,y)d\Omega, \quad U \subset \Omega,$$

or the geometric or harmonic mean can be applied (component-wise). However, these simple methods are not very accurate – especially in the case of strongly varying diffusion coefficients – and therefore are of little or even no use. This can easily be seen from the example of a diagonal matrix D for which any averaging procedure must result in again an effective diagonal matrix $\tilde{D}$. However, for a diffusion coefficient involving layers in diagonal direction (which still can be described on the fine scale by a diagonal D) this is not more adequate, as $\tilde{D}$ must contain non-diagonal entries which describe cross-diffusion.

---

[1] Here, p denotes the pressure, u the density, μ the dynamic viscosity of the fluid, $\vec{g}$ describes the gravity and $\vec{q}$ the volume of the fluid transported per time unit and area unit.

[2] If D is a bounded, symmetric and elliptic function, that is $\exists\alpha \in \mathrm{R}^+ : d_{ij}(x)\xi_i\xi_j \geq \alpha\xi_i\xi_j \forall x \in \Omega, \xi = (\xi_i) \in \mathrm{R}^n$ and $d_{ij} = d_{ji}, (1 \leq i,j \leq n)$ and if f in $L^2(\Omega)$, then (1) is an elliptic boundary value problem.

Another, more sophisticated approach, is the so-called renormalization, described in [Kin89]. The idea is, to locally compute an approximation of the effective diffusion coefficient by treating it as a resistance network (with the inverse of the diffusion as resistance, that is $R \equiv D^{-1}$). However, a severe drawback of this approach is, that only diagonal matrices for D can be used. Again the resulting $\tilde{D}$ is diagonal and, therefore, this method also fails for layered structures in the diagonal directions.

Finally, there is the method of homogenization. For a general description of this approach[3], see [Bab76], [San80] or [BLP87]. There, one assumes a periodic structure of the diffusion coefficient (with the parameter ϵ as a characteristic length). Is u_ϵ the solution of

$$-\nabla(D^\epsilon(x)\nabla u_\epsilon) = f(x) \text{ in } \Omega, \tag{3}$$

$$u_\epsilon = 0 \text{ on } \partial\Omega \tag{4}$$

with D^ϵ periodic with characteristic length ϵ, then, the homogenization approach is to find the equations describing u_ϵ as $\epsilon \searrow 0$, and to find u_0, the limit of u_ϵ. The problem with homogenization is, that it is only useable in a periodic setting and that one must solve a local problem to obtain the homogenized coefficient. But, in contrast to simple averaging and to renormalization, the homogenized diffusion coefficient is in general no longer a diagonal matrix.

3. Matrix-dependent multigrid coarsening as an approximate homogenization method. Now, we turn to our approach for the computation of an effective diffusion coefficient. It is based on methods used already for some time in robust multigrid algorithms which use matrix-dependent prolongations. The main point is, that the coarse grid operators built with matrix-dependent prolongations by means of the Galerkin approximation can be viewed as some sort of discrete homogenized operators, see also Figure 1. From the coarse grid limit operator, approximations of the effective diffusion coefficient can be read of, which are (in the periodic case) in 1D the exact homogenized diffusion coefficients, and in 2D quite good approximations of the homogenized coefficient (depending on the strategy how the matrix-dependent prolongation is chosen). Note that the coarse grid limit operator, where we assume a periodic extension of the domain Ω and its fine level diffusion values to the whole $\mathbb{R}^2$, differs from the computed operator of the coarsest level due to boundary effects. Additionally, however, the coarse grid diffusion value gives us an effective diffusion coefficient for the case of a bounded Ω with generally given values on the finest level. Thus, our approach favourably associates the advantages of renormalization and homogenization, but does not have the drawbacks of these approaches, i.e. the limitation to periodic settings as homogenization and the problem of diagonal matrices $\tilde{D}$ like renormalization.

In the following we shortly consider multigrid and then discuss matrix-dependent prolongations. We give a general principle on which all schemes we know of can be

[3] One main field of its application is in the modelling of 'fissured media', see for example [Arb89], [ADH90], [Hor90], [Pes92], [SW91].

based. This shows their relationship to incomplete factorization and approximate block Gaussian elimination.

3.1. Multigrid discretization. Multigrid is the most efficient method for solving partial differential equations. There, a sequence of grids

$$\Omega_0 \subset \Omega_1 \subset \Omega_2 \subset \ldots \subset \Omega_{k-1} \subset \Omega_k$$

is set up. Furthermore, mappings between the associated grid function spaces are needed. Here, R_l^{l-1} denotes the restriction from level l to $l-1$ and P_{l-1}^l denotes the prolongation from level $l-1$ to l which is a (possibly weighted) interpolation. Then, given a discretization L_k on the finest level k, coarse grid operators $L_l, l = k-1, ..0$, must be determined. Beside direct discretization on level l, often the Galerkin approximation is used, i.e. L_{l-1} is computed from L_l by the application of restriction and prolongation:

$$L_{l-1} = R_l^{l-1} L_l P_{l-1}^l, \text{ where mostly } R_l^{l-1} := (P_{l-1}^l)^T. \tag{5}$$

Thus, the coarse grid operators are (recursively) determined by the discretization on the finest grid L_k and the prolongations P_{l-1}^l.

It is well known that, for singular perturbed problems and operator with locally highly varying or discontinuous coefficient functions, the performance and robustness of a multigrid solver depends on the right choice of the coarse grid operators, and thus, in the context of (5), on the fine grid discretization and the choice of the prolongations only. For example, for jumps in the coefficient matrix D, standard prolongation and restriction lead to a deterioration of the convergence rate. A remedy to this problem is the choice of matrix-dependent prolongations, which will be described later. They lead via (5) to coarse grid operators, which describe the large scale behavior of the problem more properly. Note, that in this way the Galerkin approximation can then be interpreted as a discrete method for calculating averaged equations, which also model the influence of the scales smaller than the grid size of the actual level.

3.2. Matrix-dependent prolongations. The often used standard prolongations

$${P_{l-1}^l}^* = \begin{bmatrix} 1/2 & 1 & 1/2 \end{bmatrix}, \text{ and } {P_{k-1}^k}^* = \begin{bmatrix} 1/4 & 1/2 & 1/4 \\ 1/2 & 1 & 1/2 \\ 1/4 & 1/2 & 1/4 \end{bmatrix}$$

(here in stencil notation for the 1D- and 2D-case) involving (bi-)linear interpolation correspond to standard nodal basis functions on every level. Now, a short calculation shows that with this choice in the Galerkin approach (5) just an *arithmetic averaging* of D is performed. As mentioned already, arithmetic averaging is however not suited for the computation of the effective diffusion coefficient in the case of strongly varying or discontinuous D. Therefore, we will not use standard prolongation in the multigrid coarsening process. Instead, we rely on the information stored in the matrix L_l to construct a *locally weighted* prolongation P_{l-1}^l that resembles an operator-induced interplation which leads via (5) to an implicitly weighted averaging of D.

Matrix-dependent prolongations P_{l-1}^l can be derived using a general principle: They can be interpreted as transformations to perform (approximately) a block Gaussian elimination of the operator L_l, which will be explained in the following. Let there be given a discrete problem $L_l u_l = f_l$ on a grid Ω_l. Partitioning of L_l in coarse grid and fine-without-coarse grid parts gives

$$\begin{pmatrix} L_{11} & L_{12} \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix} \tag{6}$$

where u_1 denotes the unknowns in $\Omega_l \setminus \Omega_{l-1}$ and u_2 the unknowns in the next coarser grid Ω_{l-1}. Now, the optimal matrix-dependent prolongation arises from the exact block Gaussian elimination of the outer diagonal blocks L_{12} and L_{21}, i.e. from transforming L_l to

$$\begin{pmatrix} I & -L_{11}^{-1}L_{12} \\ 0 & I \end{pmatrix}^T \begin{pmatrix} L_{11} & L_{12} \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} I & -L_{11}^{-1}L_{12} \\ 0 & I \end{pmatrix} = \begin{pmatrix} L_{11} & 0 \\ 0 & L_{22} - L_{21}L_{11}^{-1}L_{12} \end{pmatrix}. \tag{7}$$

The associated matrix-dependent prolongation operator between grid $l-1$ and l is then defined by

$$P_{l-1}^l = \begin{pmatrix} -L_{11}^{-1}L_{12} \\ I \end{pmatrix} \tag{8}$$

and the coarse grid operator obtained via the Galerkin approach (5) is then

$$L_{l-1} = (P_{l-1}^l)^T L_l P_{l-1}^l := L_{22} - L_{21}L_{11}^{-1}L_{12} \tag{9}$$

which is just the well known Schur complement of L_l with respect to Ω_{l-1}.

Of course, this approach can usually not be used for the construction of a coarse grid operator , since, in general, L_{11} is not a diagonal matrix, and therefore, L_{l-1} would be no local operator any more. The result would be an inefficient multigrid algorithm. However, in the 1D case, this is different: Let L_l be given by the 3-point stencil

$$L_k^*(x_i) = [\ L_{-1}(x_i) \quad L_0(x_i) \quad L_1(x_i)\ ]. \tag{10}$$

The partitioning of the associated matrix according to (6) now results in just a diagonal matrix L_{11}. Then L_{11}^{-1} is diagonal and, consequently, the prolongation P_{l-1}^l and the coarse grid operator L_{l-1} are local operators. In stencil notation, we obtain from (8)

$${P_{l-1}^l}^*(x_{i+1}) = \left[\ -\tfrac{L_1(x_i)}{L_0(x_i)} \quad 1 \quad -\tfrac{L_{-1}(x_{i+2})}{L_0(x_{i+2})}\ \right] \tag{11}$$

and by the Galerkin approach (9)

$${L_{l-1}}^*(x_{i+1}) = \left[\ -\tfrac{L_{-1}(x_i)L_{-1}(x_{i+1})}{L_0(x_i)} \quad \tfrac{L_{-1}(x_i)L_{-1}(x_{i+1})}{L_0(x_i)} + \tfrac{L_1(x_{i+1})L_1(x_{i+2})}{L_0(x_{i+2})} \quad -\tfrac{L_1(x_{i+1})L_1(x_{i+2})}{L_0(x_{i+2})}\ \right]$$

for $x_{i+1} \in \Omega_{l-1}$.

For the special case of the diffusion equation $(d(x)u(x)_x)_x = f(x)$, where $d(x)$ is assumed to be piecewise constant on Ω_k having jumps only at the grid points, we obtain, by discretizing with finite differences or finite elements on Ω_k, the stencil

$$L_k{}^*(x_{IF+1}) = \frac{1}{h_k}\left[\begin{array}{ccc} -d^k_{IF+1} & d^k_{IF+1}+d^k_{IF+2} & -d^k_{IF+2}\end{array}\right] \tag{12}$$

with $d^k_{IF+1} = d(x_{IF+1} - h_k/2)$, compare also Figure 2.

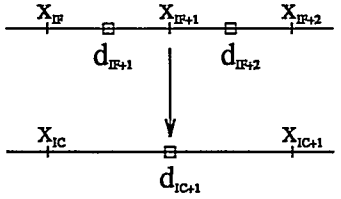


Fig. 2: 1D case: fine and coarse grid

Now, using the corresponding prolongation (11)

$$P^k_{k-1}{}^*(x_{IC}) = \left[\begin{array}{ccc} \frac{d^k_{IF}}{d^k_{IF-1}+d^k_{IF}} & 1 & \frac{d^k_{IF+1}}{d^k_{IF+1}+d^k_{IF+2}}\end{array}\right] \text{ for } x_{IC} \in \Omega_{k-1}, \tag{13}$$

we obtain

$$L_{k-1}{}^*(x_{IC}) = \frac{2}{h_{k-1}}\left[\begin{array}{cccc} -\frac{d^k_{IF-1}d^k_{IF}}{d^k_{IF-1}+d^k_{IF}} & \frac{d^k_{IF-1}d^k_{IF}}{d^k_{IF-1}+d^k_{IF}}+\frac{d^k_{IF+1}d^k_{IF+2}}{d^k_{IF+1}+d^k_{IF+2}} & -\frac{d^k_{IF+1}d^k_{IF+2}}{d^k_{IF+1}+d^k_{IF+2}}\end{array}\right] \tag{14}$$

for $x_{IC} \in \Omega_{k-1}$. Then, comparing (14) with (12) gives the coarse grid diffusion coefficient on the interval $[x_{IC}, x_{IC+1}]$:

$$d^{k-1}_{IC+1} = 2\frac{d^k_{IF+1}d^k_{IF+2}}{d^k_{IF+1}+d^k_{IF+2}} \tag{15}$$

which is just the harmonic mean of the two corresponding fine grid values. In other words: The direct discretization on the coarse grid Ω_{k-1} using the *local harmonic average* of the fine grid diffusion coefficient would give the same coarse grid operator. Now, if we apply this approach recursively for the sequence of successively coarser grids, we obtain in the end on Ω_0 the global harmonic mean as effective diffusion coefficient. Here, we can use periodic conditions on the boundary as well. Then, we obtain the same result as with homogenization.

However, in the two-dimensional case, the exact matrix-dependent prolongation cannot be used any more, since now, the unknowns belonging to the points from $\Omega_k \setminus \Omega_{k-1}$ are coupled to the unknowns belonging to coarse grid points and neighbored fine grid points by the entries in $\left(\begin{array}{cc} L_{11} & L_{12}\end{array}\right)$. Then, L_{11} is no longer a diagonal matrix as it was in the one-dimensional case. Consequently, the matrix-dependent prolongation is no local operator any more and would need the computation of L_{11}^{-1}.

Therefore, instead of the optimal matrix-dependent prolongation, approximations to it are used, which result in local operators. They can be gained using the following general approach: Substitute the set of equations associated to the grid points $\Omega_k \setminus \Omega_{k-1}$ by an altered set of equations, i.e. substitute $\left(\begin{array}{cc} L_{11} & L_{12}\end{array}\right)$ by some $\left(\begin{array}{cc} \tilde{L}_{11} & \tilde{L}_{12}\end{array}\right)$, which meets the following demands:

- $\tilde{L}_{11}$ is 'easy' to invert,
- $\tilde{L}_{11}^{-1}\tilde{L}_{12}$ is a 'good' approximation of $L_{11}^{-1}L_{12}$, and
- $\begin{pmatrix} -\tilde{L}_{11}^{-1}\tilde{L}_{12} \\ I \end{pmatrix}$ is a local operator.

Then, the associated prolongation operator $\tilde{P}_{k-1}^{k}$ is defined by

$$\tilde{P}_{k-1}^{k} = \begin{pmatrix} -\tilde{L}_{11}^{-1}\tilde{L}_{12} \\ I \end{pmatrix}. \tag{16}$$

Thus, the approximation of the optimal matrix-dependent prolongation is the exact optimal matrix-dependent prolongation of an altered discretization, namely

$$\tilde{L}_k = \begin{pmatrix} \tilde{L}_{11} & \tilde{L}_{12} \\ L_{21} & L_{22} \end{pmatrix}. \tag{17}$$

Now, we consider two simple examples how to construct an approximate matrix-dependent prolongation. To this end, we proceed as follows: If $l > 0$, then every grid Ω_l can be divided in four disjunct grids:

$$\begin{aligned} \Omega_l^{00} &:= \{(x,y) \in \Omega_l : x = ih_l, y = jh_l, i,j \in \mathrm{Z}, i \text{ even}, j \text{ even}\} = \Omega_{l-1} \\ \Omega_l^{10} &:= \{(x,y) \in \Omega_l : x = ih_l, y = jh_l, i,j \in \mathrm{Z}, i \text{ odd}, j \text{ even}\} \\ \Omega_l^{01} &:= \{(x,y) \in \Omega_l : x = ih_l, y = jh_l, i,j \in \mathrm{Z}, i \text{ even}, j \text{ odd}\} \\ \Omega_l^{11} &:= \{(x,y) \in \Omega_l : x = ih_l, y = jh_l, i,j \in \mathrm{Z}, i \text{ odd}, j \text{ odd}\}. \end{aligned} \tag{18}$$

Then we can partition the matrix L_l as

$$L_l = \begin{pmatrix} L_{11} & L_{12} \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} l_{11} & l_{12} & l_{13} \\ l_{21} & l_{22} & l_{23} \\ l_{31} & l_{32} & l_{33} \end{pmatrix} = \begin{pmatrix} \begin{pmatrix} l_{11} & l_{12} \\ l_{21} & l_{22} \end{pmatrix} & \begin{pmatrix} l_{13} \\ l_{23} \end{pmatrix} \\ \begin{pmatrix} l_{31} & l_{32} \end{pmatrix} & l_{33} \end{pmatrix}. \tag{19}$$

Here, $\begin{pmatrix} l_{11} & l_{12} & l_{13} \end{pmatrix}$ corresponds to the equations belonging to the unknowns associated to the points in Ω^{11}, $\begin{pmatrix} l_{21} & l_{22} & l_{23} \end{pmatrix}$ corresponds to the equations belonging to the unknowns associated to the points in $\Omega^{10} \cup \Omega^{01}$ and $\begin{pmatrix} l_{31} & l_{32} & l_{33} \end{pmatrix}$ corresponds to the equations belonging to the unknowns associated to the points in $\Omega^{22} = \Omega_{l-1}$. Now, if L_l can be described by a 9-point stencil then l_{11} is a diagonal and l_{33} is a diagonal matrix.

Surely, the easiest way for a substitution is to replace L_{11} by its diagonal $diag\{L_{11}\}$. Another possibility, which is obviously more accurate, is to choose

$$\tilde{L}_{11} = \begin{pmatrix} l_{11} & l_{12} \\ 0 & \tilde{l}_{22} \end{pmatrix} \tag{20}$$

(remember that l_{11} is diagonal!). Then, the inverse is given by

$$\tilde{L}_{11}^{-1} = \begin{pmatrix} l_{11}^{-1} & -l_{11}^{-1}l_{12}\tilde{l}_{22}^{-1} \\ 0 & \tilde{l}_{22}^{-1} \end{pmatrix} \tag{21}$$

and the prolongation can be defined by

$$\tilde{P}_{l-1}^{l} = \begin{pmatrix} -\tilde{L}_{11}^{-1} \begin{pmatrix} l_{13} \\ \tilde{l}_{23} \end{pmatrix} \\ I \end{pmatrix} = \begin{pmatrix} -l_{11}^{-1}(l_{13} - l_{12}\tilde{l}_{22}^{-1}\tilde{l}_{23}) \\ -\tilde{l}_{22}^{-1}\tilde{l}_{23} \\ I \end{pmatrix}$$

Note that this is the optimal prolongation of the altered discretization

$$\tilde{L}_l = \begin{pmatrix} l_{11} & l_{12} & l_{13} \\ 0 & \tilde{l}_{22} & \tilde{l}_{23} \\ l_{31} & l_{32} & l_{33} \end{pmatrix}. \tag{22}$$

There is additionally the possibility to perform the following substitutions, which involve 'directional' lumping:

$$\begin{bmatrix} L_{-1,1} & L_{0,1} & L_{1,1} \\ L_{-1,0} & L_{0,0} & L_{1,0} \\ L_{-1,-1} & L_{0,-1} & L_{1,-1} \end{bmatrix} \longrightarrow \left[\begin{array}{c|c|c} 0 & 0 & 0 \\ \hline L_{-1,0} & L_{0,0} & L_{1,0} \\ +L_{-1,-1} & +L_{0,-1} & +L_{1,-1} \\ +L_{-1,1} & +L_{0,1} & +L_{1,1} \\ \hline 0 & 0 & 0 \end{array}\right] \text{ in } \Omega^{10},$$

$$\begin{bmatrix} L_{-1,1} & L_{0,1} & L_{1,1} \\ L_{-1,0} & L_{0,0} & L_{1,0} \\ L_{-1,-1} & L_{0,-1} & L_{1,-1} \end{bmatrix} \longrightarrow \begin{bmatrix} 0 & L_{0,1} + L_{-1,1} + L_{1,1} & 0 \\ 0 & L_{0,0} + L_{-1,0} + L_{1,0} & 0 \\ 0 & L_{0,-1} + L_{-1,-1} + L_{1,-1} & 0 \end{bmatrix} \text{ in } \Omega^{01}.$$

Then $\tilde{l}_{22}$ becomes a diagonal matrix, too. These substitutions lead to the prolongation described in [Den82].

Further prolongations can easily be constructed in a similar way. Such matrix-dependent prolongations are described, for example, in [ABDP81], [Den83], [Den82], [FG93], [Fuh94], [Re93a], [Re93b], and [AV89], [AV90], [AV91], [Zee90]. We state that all the matrix-dependent schemes we know of can be based on the principle mentioned above and can be denoted also in form of a substitution (17). This shows their relationship to incomplete factorization and approximate block Gaussian elimination. Since, for our simple model problem (1), most of these methods coincide with the scheme due to [Den82], we will restrict ourselves in the numerical experiments of the following section to it.

Then, the resulting coarse grid operator L_{l-1} built from a 9-point stencil matrix L_l and the associated matrix-dependent prolongation P_{l-1}^{l} by means of the Galerkin approximation (9) are again by 9-point stencils and our matrix-dependent coarsening procedure can be repeated recursively on coarser grids. The resulting operator on the coarse grid describes the large scale problem quite good.

In the two-dimensional case, for periodic boundary conditions, we thus have not more the exact equivalence of homogenization and recursive matrix-dependent Galerkin coarsening as we had in the one-dimensional case. Now we introduced a slight error compared with homogenization due to the substitution (17) which is necessary to keep the coarse grid operators in the recursive process local.

4. Numerical experiments. In the following, we compare the different methods to obtain an effective diffusion coefficient for our model problem (1). Here, if not indicated otherwise, we consider the periodic setting for the computation of the approximation D^{MG-hom} of the effective diffusion coefficient by repeated Galerkin approximation using matrix-dependent prolongations. To this end, the domain can be imagined to be implicitly repeated and to tile the whole $\mathbb{R}^2$ to get rid of the influences from boundary conditions. This is achieved by using periodic conditions on the boundary of the domain. In this way, we obtain 9-point stencils in *every* point of the coarse grid Ω_{k-1} (including boundary) and, after applying the respective Galerkin coarsening recursively[4] we obtain in the limit a stencil which is in every point the same[5]. From this star we directly read off the so-called MG-homogenized diffusion coefficient D^{MG-hom}. The same is done in the computations of the approximations D^{ren} using renormalization according to [Kin89] and D^{arith}, D^{geo} and D^{harm} using arithmetic, geometric and harmonic averaging, respectively. This makes it possible to compare our results with that of homogenization since there, a periodic structure is involved inherently.

4.1. Example 1: Domain with inclusion. Now, we define the diffusion coefficient of our first model problem by

$$D(x,y) = \begin{cases} \alpha \cdot I & \text{in subdomain } Y_I \\ I & \text{in subdomain } Y_{II}. \end{cases}$$

This is illustrated in Figure 3 (left). By mean of periodicity, structures like in Figure 3 (right) are associated.

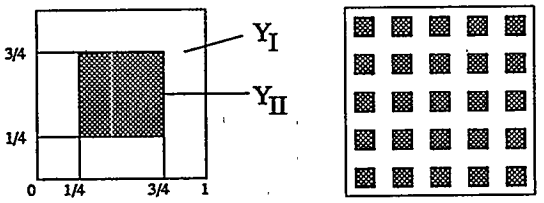


FIG. 3. *Diffusion coefficient of example* 1

Because of the symmetry of the problem and the diagonal structure of D, we obtain from all methods an approximation of the effective diffusion operator $\nabla \tilde{D} \nabla$ where

$$\tilde{D} = d \cdot I, \quad d \in \mathbb{R}.$$

The value for d depends on the value α and the respective method under consideration. For two different α the results are given in Table 1. We clearly see that the approach based on recursive matrix-dependent Galerkin coarsening gives values which are most close to the exact values obtainable for this periodic problem by homogenization. All other approaches give worse values.

[4] Here, we copy the stencils of Ω_{k-1} four times properly to the four quadrants of the fine grid Ω_k to obtain periodicity and repeat this process.

[5] Note however, that the method of successive Galerkin coarsening can be applied without this periodicity approach also directly on the sequence of grids $\Omega_k, \Omega_{k-1}, \ldots, \Omega_0$.

TABLE 1
Problem 1: Comparison of the approximations of the effective diffusion coefficient

α	d^{hom}	d^{MG-hom}	d^{ren}	d^{arith}	d^{geo}	d^{harm}
10	6.52	6.70088	6.16712	7.75	5.6234	3.0769
100	59.2	61.73646	54.24763	75.25	31.6228	3.8835

4.2. Example 2: Cross-wind diffusion. In our second considered example, we now set the diffusion coefficient to

$$D(x,y) = \begin{cases} 1 \cdot I & \text{if } (x,y) \in Y_I \\ 8 \cdot I & \text{if } (x,y) \in Y_{II} \end{cases}$$

where the subdomains Y_I and Y_{II} are indicated in Figure 4 (left). Then, by means of periodicity, layered structures like the one given in 4 (right) are built. Because of the

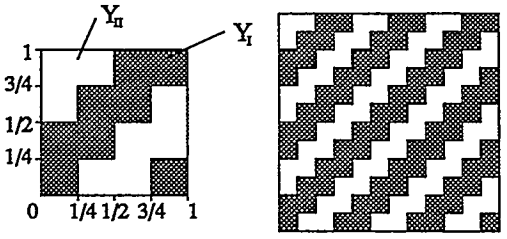


FIG. 4. *Subdomains for diffusion coefficient of problem 2*

layering of the diffusion coefficient in diagonal direction, the effective diffusion coefficient must be a full matrix, i.e. it contains diffusion in the off-diagonal entries of $\tilde{D}$ as well.

The approximations that we obtained with our different methods were

$$D^{hom} = \begin{pmatrix} 3.09 & 0.93 \\ 0.93 & 3.09 \end{pmatrix}, \quad D^{MG-hom} = \begin{pmatrix} 2.99 & 1.13 \\ 1.13 & 3.28 \end{pmatrix}$$

$$D^{ren} = \begin{pmatrix} 2.44 & 0 \\ 0 & 2.44 \end{pmatrix}, \quad D^{arith} = \begin{pmatrix} 4.5 & 0 \\ 0 & 4.5 \end{pmatrix}.$$

The approaches based on renormalization and the arithmetic mean (as well as the other simple averaging schemes) are not able to produce diffusion entries in the off-diagonal and, thus, give a completely wrong picture of the coarse scale equation. However, MG-homogenization as well as homogenization give cross-wind diffusion terms. (The inequality of the diagonal entries of D^{MG-hom} comes from the slight imbalance in the diffusion coefficient D induced by the different sizes of the subdomains Y_I and Y_{II}, see also Figure 4 (left).)

4.3. Example 3: Randomly distributed diffusion. The last example involves an isotropic diffusion coefficient $D(x,y) = d(x,y) \cdot I$ where the values of $d(x,y)$ are randomly chosen over Ω. Here, in contrast to the previous two examples, we apply no

periodicity trick in the coarsening process (since it would influence the results and spoil the effective diffusion coefficient) but merely use the sequence of grids $\Omega_k, \Omega_{k-1}, \ldots \Omega_0$ and determine the corresponding approximation of the effective diffusion coefficient on the second coarsest level. We took here a 257×256 grid on the finest level. Note that such types of problems are frequently used in oil industry to model porous media statistically. There, a log-distribution is often assumed. Homogenization can not be directly applied any more[6] due to its non-periodic nature. The results obtained with our different coarsening methods are given in Table 2. Here, the diffusion values were chosen equally distributed from the interval $[a, b]$. Therefore, the theoretically optimal values must be $(a+b)/2$. We see that MG-homogenization again gives the best approximation to the effective diffusion (beside arithmetic averaging of course which is exact for this specific example). It is clearly superior to the other methods. Note that, up to now, we did not obtain real life data from oil fields yet to test our method also in this respect. This will be done in the near future.

TABLE 2

Problem 3: Results for random diffusion coefficient which are equally distributed in $[a, b]$.

$[a, b]$	$[1, 9]$	$[1, 99]$
arith. mean	5.0	50.0
geom. mean	4.357	38.168
harm. mean	3.641	21.327
renormalization	4.28	37.9
MG- homogenization	4.61	44.09

5. Concluding remarks. We have introduced a method to compute an approximation to the effective diffusion coefficient. This type of problem arises frequently in the area of porous media flow. Our technique is based on the Galerkin coarsening approach involving matrix-dependent prolongations. The method can be applied both, in a periodic setting and in the non-periodic case. For the presented problems and also in many further experiments not discussed here [Kna95], our method produced quite good results. In the periodic case, it gave results close to the optimal values obtained from homogenization which had been better than that obtained from renormalization or simple averaging. In applications where no periodicity of the diffusion structure was present and homogenization is not applicable any more, it also gave results better than that obtained from renormalization or simple averaging.

To us it was interesting, that techniques which are applied successfully in multigrid algorithms to produce a 'good' coarse grid operator are also applicable in another area where multiscale questions arise. We conjecture that there exist other techniques used to maintain a fast and robust convergence rate in multigrid methods which can be exploited also in many areas of mathematical modelling where multiscale questions are important.

[6] Note that there exist recent attempts to generalize the theory of homogenization to stochastic diffusion fields.

REFERENCES

[ABDP81] Alcouffe R.E., Brandt A., Dendy J.E., Painter J.W.: *The multi-grid method for the diffusion equation with strongly discontinuous coefficients*, SIAM J. Sci. Stat. Comput. Vol. 2, 1981, 430-454.

[ABK91] Amaziane B., Bourgeat A., Koebbe A.J.: *Numerical simulation and homogenization of two-phase flow in heterogeneous porous media*, Transport in porous media 6, 519-547, 1991.

[Arb89] Arbogast T.: *Analysis of the simulation of single phase flow through a naturally fractured reservoir*, SIAM J. Numer. Anal., Vol. 26, No. 1, 12-29, 1989.

[ADH90] Arbogast T., Douglas J., Hornung U.: *Derivation of the double porosity model of single phase flow via homogenization theory*, SIAM J. Math Anal., Vol. 21, No. 4, 823-836, 1990.

[AV89] Axelsson O., Vassilevski P.S.: *Algebraic multilevel preconditioning methods. I*, Numer. Math. 56, 157-177, 1989.

[AV90] Axelsson O., Vassilevski P.S.: *Algebraic Multilevel Preconditioning Methods, II*, Siam J. Numer. Anal., Vol. 27, No. 6, 1569-1590, Dec. 1990.

[AV91] Axelsson O., Vassilevski P.S.: *Construction of variable preconditioners for inner-outer iteration methods*, Report 9107, Department of Math., Katholische Universität Nijmegen, Niederlande, 1991.

[Bab76] Babuska I.: *Homogenization and its Application. Mathematical and Computational Problems*, Num. Sol. of Part. Diff. Eqn.-III, B. Hubbard, ed., Academic Press, N.Y., 89-116, 1976.

[BLP87] Bensoussan A., Lions J.L., Papanicolaou G.: *Asymptotic Analysis for Periodic Structure*, North-Holland, Amsterdam, 1987.

[Den83] Dendy J.E.: *Black box multigrid for nonsymmetric problems*, Appl. Math. Comput., Vol. 13, 261-283, 1983.

[Den82] Dendy J.E.: *Black box multigrid*, J. Comp. Phys., Vol. 48, 366-386, 1982.

[FG93] Fuhrmann J., Gärtner K.: *On matrix data structures and the stability of multigrid algorithms*, Multigrid Methods, Vol. IV, Springer Lectures in Math., Springer-Verlag, 1993.

[Fuh94] Fuhrmann, J.: *Zur Verwendung von Mehrgitterverfahren bei der numerischen Behandlung elliptischer Differentialgleichungen mit variablen Koeffizienten*, Aachen: Shaker, 1995 (Berichte aus der Mathematik), Chemnitz-Zwickau, Techn. Univ., Diss., 1994.

[Hor90] Hornung U.: *Diffusion Models for fractured media*, J. of math. Anal. and Appl. 147, 69-80, 1990.

[Kes79] Kesavan S.: *Homogenization of Elliptic Eigenvalue Problems: Part 2*, Appl. Math. Optim. 5, 153-167, 1979.

[Kin89] King P.R.: *The use of renormalization for calculating effective permeability*, Transport in porous media 4, 37-58, 1989.

[Kir73] Kirkpatrick S.: Rev. Mod. Phys. 45, 574.

[Kna95] Knapek S.: *Multiskalenverfahren bei der Modellierung, Diskretisierung und Lösung von Diffusionsproblemen*, Diplomarbeit, Institut für Informatik, TU München, 1995.

[San80] Sanchez-Palencia E.: *Non-homogenious media and vibration theory*, Lecture Notes in Physics 127, Springer-Verlag, 1980.

[Pes92] Peszynska M.: *Flow through fissured media: Mathematical analysis and numerical approach*, Dissertation, Universität Augsburg, 1992.

[Re93a] Reusken A.: *Multigrid with matrix-dependent transfer operators for convection-diffusion problems*, Multigrid Methods, Vol. IV, Springer Lectures in Math., Springer-Verlag, 1993.

[Re93b] Reusken A.: *Multigrid with matrix-dependent transfer operators for a singular perturbation problem*, Computing 50, 199-211, 1993.

[SW91] Showalter R.E., Walkington N.J.: *Micro-structure models of diffusion in fissured media*, J. of Math. Anal. and Appl. 155, 1-20, 1991.

[Zee90] De Zeeuw P.M.: *Matrix-dependent prolongations and restrictions in a blackbox multigrid solver*, J. of Comp. and Appl. Math. 33, 1-27, 1990.

A Superlinear Convergence Estimate for an Iterative Method for the Biharmonic Equation

Mark A. Horn†‡,
Department of Mathematics and Statistics, Wichita State University,
Wichita, KS 67260-0033

1. Introduction. In [CDH] a method for the solution of boundary value problems for the biharmonic equation using conformal mapping was investigated. The method is an implementation of the classical method of Muskhelishvili [Musk]. In [CDH] it was shown, using the Hankel structure, that the linear system in [Musk] is the discretization of the identity plus a compact operator, and therefore the conjugate gradient method will converge superlinearly. The purpose of this paper is to give an estimate of the superlinear convergence in the case when the boundary curve is in a Hölder class.

The paper is organized as follows. Section 2 describes the original method for simply connected regions. The representation of the biharmonic function and the boundary conditions in terms of the analytic Goursat functions is given. Transplanting the boundary conditions to the unit disk with a conformal map then leads to a linear system for the Taylor coefficients of the Goursat functions on the disk. In Section 3, some results from conformal mapping are used to show that the linear system can be formulated in terms of a compact operator with a Hankel structure. In Section 4, the superlinear convergence rate of the conjugate gradient method applied to the linear system is established. In section 5, numerical results are given to illustrate the theorems.

2. The biharmonic equation. As in [CDH], we will follow the presentation in [KK] and [Musk]. We wish to find a function $u = u(\eta, \varsigma)$ which satisfies the biharmonic equation,

$$\Delta^2 u = 0,$$

for $\varsigma = \eta + i\mu \in \Omega$ where Ω is a region with a smooth boundary Γ and u satisfies the boundary conditions

$$u_\eta = G_1 \quad \text{and} \quad u_\mu = G_2$$

on Γ. This boundary value problem arises, for instance, in plane stress problems where u is the so-called Airy stress function. u can be represented as

$$u(\varsigma) = Re(\bar{\varsigma}\phi(\varsigma) + \chi(\varsigma)),$$

where $\phi(\varsigma)$ and $\chi(\varsigma)$ are analytic functions in Ω known as the *Goursat functions*. Letting $G = G_1 + iG_2$, the boundary conditions become

$$\phi(\varsigma) + \varsigma\overline{\phi'(\varsigma)} + \overline{\psi(\varsigma)} = G(\varsigma), \quad \varsigma \in \Gamma \tag{1}$$

† The author's research was partially supported by U. S. Department of Energy grant DE-FG02-92ER25124 and National Science Foundation EPSCoR grant OSR-9255223.

‡ Email address: mhorn@twsuvm.uc.twsu.edu

where $\psi(\zeta) = \chi'(\zeta)$. The problem is to find ϕ and ψ satisfying (1).

Let $\zeta = f(z)$ be the conformal map from the unit disk to Ω, fixing $f(0) = 0 \in \Omega$. Then with $d(z) := f(z)/\overline{f'(z)}$, $\phi(z) := \phi(f(z)), \psi(z) := \psi(f(z))$, and $G(z) := G(f(z))$, equation (1) transplants to the unit disk as

$$\phi(z) + d(z)\overline{\phi'(z)} + \overline{\psi(z)} = G(z), \quad |z| = 1. \tag{2}$$

Let

$$\phi(z) = \sum_{k=1}^{\infty} a_k z^k \quad \text{and} \quad \psi(z) = \sum_{k=0}^{\infty} b_k z^k.$$

The problem is now to find the a_k's and the b_k's.

For $|z| = 1$, define the Fourier series

$$d(z) := f(z)/\overline{f'(z)} = \sum_{k=-\infty}^{\infty} h_k z^k, \quad G(z) = \sum_{k=-\infty}^{\infty} A_k z^k.$$

Substituting into (2) gives a linear system of equations for the a_k's and b_k's,

$$a_j + \sum_{k=1}^{\infty} k\bar{a}_k h_{k+j-1} = A_j, \quad j = 1, 2, 3, \ldots \tag{3}$$

$$\bar{b}_j + \sum_{k=1}^{\infty} k\bar{a}_k h_{k-j-1} = A_{-j}, \quad j = 0, 1, 2, \ldots \tag{4}$$

If (3) is solved for the a_k's, then the b_k's can be easily computed from (4). These systems are derived in [Musk] and [KK] and solved for some simple examples. In [CDH], (3) was truncated after n terms and solved efficiently using the conjugate gradient method. The matrix-vector multiplications can be done in $O(N \log N)$ using fast Fourier transforms (FFTs).

There is a moment condition to be satisfied by the data. After transplantation to the disk, this condition can be stated as $Re[\int_{|z|=1} G(z)\overline{f'(z)}d\bar{z}] = 0$. Furthermore, ϕ and ψ are not unique. In short, one needs to specify $a_0 = \phi(0) = 0$ and $Im(a_1/f'(0)) = 0$ which will be incorporated into the derivations.

It should be noted that if our boundary data corresponds to $G = 0$ then the only possible (nonzero) choice for ϕ is $\phi(z) = Cif(z)$, for some nonzero $C \in \mathbf{R}$. This implies that the null space corresponding to the infinite system in (3) is one dimensional and the eigenvector spanning this space is given by $a_k = ic_k, k = 1, 2, 3, \cdots$ where $f(z) = \sum_{k=1}^{\infty} c_k z^k$.

3. Compact Operators. As in [CDH], we take real and imaginary parts of equation (3) to get

$$\alpha_j + \sum_{k=1}^{\infty} k(\eta_{k+j-1}\alpha_k + \gamma_{k+j-1}\beta_k) = B_j, \quad j = 1, 2, 3, \ldots \tag{5}$$

$$\beta_j + \sum_{k=1}^{\infty} k(\gamma_{k+j-1}\alpha_k - \eta_{k+j-1}\beta_k) = C_j, \quad j = 1,2,3,\ldots \tag{6}$$

where we have used the notation $a_k = \alpha_k + i\beta_k$, $h_k = \eta_k + i\gamma_k$, and $A_k = B_k + iC_k$. For visualization purposes, we combine equations (5) and (6) into a doubly infinite matrix equation in which the two sums are combined into a block Hankel matrix composed with a diagonal matrix. (A *Hankel matrix* is a matrix which is constant on the antidiagonals.) In fact, (5) and (6) can be written as

$$(I_\infty + H_{r,\infty}D_\infty)\underline{\alpha} + H_{i,\infty}D_\infty\underline{\beta} = \underline{B}$$

$$(I_\infty - H_{r,\infty}D_\infty)\underline{\beta} + H_{i,\infty}D_\infty\underline{\alpha} = \underline{C}$$

so that

$$\left(\begin{pmatrix} I_\infty & 0 \\ 0 & I_\infty \end{pmatrix} + \begin{pmatrix} H_{r,\infty} & H_{i,\infty} \\ H_{i,\infty} & -H_{r,\infty} \end{pmatrix}\begin{pmatrix} D_\infty & 0 \\ 0 & D_\infty \end{pmatrix}\right)\begin{pmatrix} \underline{\alpha} \\ \underline{\beta} \end{pmatrix} = \begin{pmatrix} \underline{B} \\ \underline{C} \end{pmatrix} \tag{7}$$

where $\underline{\alpha} = (\alpha_1, \alpha_2, \ldots)^T$, $\underline{\beta} = (\beta_1, \beta_2, \ldots)^T$, $\underline{B} = (B_1, B_2, \ldots)^T$, $\underline{C} = (C_1, C_2, \ldots)^T$, I_∞ is the infinite identity matrix, $D_\infty = diag(1,2,\ldots)$, $H_{r,\infty}$ is an infinite Hankel matrix generated by the η_k, and $H_{i,\infty}$ is an infinite Hankel matrix generated by the γ_k.

Now suppose $(\underline{\alpha}, \underline{\beta})$ represents a solution to (5),(6). Define

$$\underline{x} = \begin{pmatrix} D_\infty^{1/2}\underline{\alpha} \\ D_\infty^{1/2}\underline{\beta} \end{pmatrix}, \quad \underline{r} = \begin{pmatrix} D_\infty^{1/2}\underline{B} \\ D_\infty^{1/2}\underline{C} \end{pmatrix}.$$

Then (7) can be written as

$$(I_\infty + M_\infty)\,\underline{x} = \underline{r} \tag{8}$$

where M_∞ is given by

$$M_\infty = \begin{pmatrix} M_{r,\infty} & M_{i,\infty} \\ M_{i,\infty} & -M_{r,\infty} \end{pmatrix} = \begin{pmatrix} D_\infty^{1/2}H_{r,\infty}D_\infty^{1/2} & D_\infty^{1/2}H_{i,\infty}D_\infty^{1/2} \\ D_\infty^{1/2}H_{i,\infty}D_\infty^{1/2} & -D_\infty^{1/2}H_{i,\infty}D_\infty^{1/2} \end{pmatrix}.$$

Note that M_∞ is symmetric. We would now like to justify the formal manipulations above and show that M_∞ is a compact operator. This will require the following definitions and lemmas; see, e.g., [Po].

Definition 1. *$\gamma \in C^{n,\alpha}[0,2\pi]$ if γ is n-times differentiable for $0 < s < 2\pi$ and*

$$|\gamma^{(n)}(s_1) - \gamma^{(n)}(s_2)| \le C|s_1 - s_2|^\alpha, \quad n \ge 1,\ 0 < \alpha \le 1.$$

Definition 2. *The Jordan curve Γ is of class $C^{n,\alpha}$ if it has a parameterization $\Gamma : \gamma(s)$, $0 \le s \le 2\pi$, such that $\gamma \in C^{n,\alpha}[0,2\pi]$ and $\gamma'(s) \neq 0$.*

Next, we state a few well known results about Fourier coefficients and conformal maps.

Theorem 1. *Let f be periodic with Fourier series $f(e^{i\theta}) = \sum_{n=1}^{\infty} c_n e^{in\theta}$. Then $f \in C^{l,\alpha}[0,2\pi]$ implies*

$$|c_n| = O(n^{-l-\alpha}) \quad l \geq 1, \quad 0 < \alpha \leq 1.$$

Warschawski's Theorem. *Let f map the disk D conformally onto the inner domain of the Jordan curve Γ of class $C^{n,\alpha}$ where $n = 1,2,\ldots$ and $0 < \alpha < 1$. Then $f^{(n)}$ has a continuous extension to $\overline{D}$ and*

$$|f^{(n)}(z_1) - f^{(n)}(z_2)| \leq C|z_1 - z_2|^{\alpha}, \quad z_1, z_2 \in \overline{D}.$$

Note that in general one cannot take $\alpha = 1$ in the above theorem.

Lemma 1. *Let f be a conformal map from the unit disk to the region Ω with boundary Γ. Assume*

$$f(e^{i\theta})/\overline{f'(e^{i\theta})} = \sum_{n=-\infty}^{\infty} h_n e^{in\theta}.$$

Then $\Gamma \in C^{l+1,\alpha}$ implies

$$|h_n| = O(n^{-l-\alpha}), \quad l \geq 2, \quad 0 < \alpha < 1.$$

Proof: By Warschawski's Theorem Γ of class $C^{l+1,\alpha}$ implies $f(e^{i\theta}) \in C^{l+1,\alpha}[0,2\pi]$ and $\overline{f'}(e^{i\theta} \in C^{l,\alpha}[0,2\pi]$. The proof then follows from Theorem 1 since $f/\overline{f'} \in C^{l,\alpha}[0,2\pi]$, for $l \geq 2$, $0 < \alpha < 1$.

Theorem 2. *If Γ is of class $C^{l+1,\alpha}$, $l \geq 2$, $0 < \alpha < 1$, then $M_{r,\infty} : l^1 \to l^1$ and $M_{i,\infty} : l^1 \to l^1$ are compact operators where for $\underline{y} \in l^1$,*

$$M_{r,\infty}\underline{y} = \sum_{k=1}^{\infty} \sqrt{kj}\eta_{k+j-1}y_k, \quad j = 1,2,\ldots$$

$$M_{i,\infty}\underline{y} = \sum_{k=1}^{\infty} \sqrt{kj}\gamma_{k+j-1}y_k, \quad j = 1,2,\ldots$$

Proof: We will prove the theorem for $M_{r,\infty}$. As above, $M_{i,\infty}$ follows similarly. Define the finite rank operators $\{M_{r,n}\} = \{D_n^{1/2} H_n D_n^{1/2}\}$ by

$$M_{r,n}\underline{y} = \sum_{k=1}^{n} \sqrt{kj}\eta_{k+j-1}y_k, \quad j = 1,2,\ldots,n$$

for all $\underline{y} = (y_1, y_2, \ldots) \in l^1$. The goal is to show that $M_{r,\infty}$ can be approximated uniformly by these finite rank operators. (Then, e.g., a version of Theorem 4.4c [Con, p. 41] for

Banach spaces shows that $M_{r,\infty}$ is itself compact.) If $A = (a_{kj})$ is an infinite matrix, then the induced l^1 operator norm is given by

$$||A||_{l^1} = \sup_j \sum_{k=1}^{\infty} |a_{kj}|;$$

see, e.g., [Con, p. 171, prob. 8].

Assume Γ is of class $C^{l+1,\alpha}$. Let $m_{k,j}$ denote the $(k,j)th$ component of $M_{r,n}$. Then clearly, $m_{k,j} = \sqrt{kj}Re(h_{k+j-1})$. From Lemma 1 we obtain the following estimate. For any $j \geq 1$,

$$\begin{aligned}
\sum_{k=1}^{\infty} |m_{k,j}| &\leq C \sum_{k=1}^{\infty} \frac{\sqrt{kj}}{(k+j-1)^{l+\alpha}} \\
&\leq C\sqrt{j} \sum_{k=0}^{\infty} \frac{1}{(k+j)^{l-1/2+\alpha}} \\
&\leq C\sqrt{j}\Big\{ \frac{1}{j^{l-1/2+\alpha}} + \int_0^{\infty} \frac{dx}{(x+j)^{l-1/2+\alpha}} \Big\} \\
&\leq C \frac{1}{j^{l-2+\alpha}},
\end{aligned}$$

where C is a constant that depends only on the conformal map. Consequently,

$$\begin{aligned}
||M_{r,\infty} - M_{r,n}||_{l^1} &= \sup \Big\{ \sum_{k=1}^{\infty} |m_{k,n+1}|, \sum_{k=1}^{\infty} |m_{k,n+2}|, \dots \Big\} \\
&\leq C \sup \Big\{ \frac{1}{(n+1)^{l-2+\alpha}}, \frac{1}{(n+2)^{l-2+\alpha}}, \dots \Big\} \\
&= C \frac{1}{(n+1)^{l-2+\alpha}}.
\end{aligned}$$

The result follows.

Corollary 1. *Under the notation and assumptions of Theorem 1 M_∞ is compact on $l^1 \times l^1$ where for $\underline{x} = (\underline{x}^1, \underline{x}^2) \in l^1 \times l^1$,*

$$M_\infty \begin{pmatrix} \underline{x}^1 \\ \underline{x}^2 \end{pmatrix} = \begin{pmatrix} \sum_{k=1}^{\infty} \sqrt{kj}\eta_{k+j-1}x^1{}_k + \sum_{k=1}^{\infty} \sqrt{kj}\gamma_{k+j-1}x^2{}_k \\ \sum_{k=1}^{\infty} \sqrt{kj}\eta_{k+j-1}x^1{}_k - \sum_{k=1}^{\infty} \sqrt{kj}\gamma_{k+j-1}x^2{}_k \end{pmatrix}.$$

The norm on $l^1 \times l^1$ is given by $||\underline{x}||_{l^1 \times l^1} = ||\underline{x}^1||_{l^1} + ||\underline{x}^2||_{l^1}$.

Proof: From the notation of the problem, it is easily verified that

$$||M_\infty - M_n||_{l^1 \times l^1} \leq 2(||M_{r,\infty} - M_{r,n}||_{l^1} + ||M_{i,\infty} - M_{i,n}||_{l^1})$$

The result follows as in Theorem 2.

Though Corollary 1 shows that M_∞ is compact, a precise estimate on the eigenvalues of M_∞ is needed to obtain superlinear convergence rates for error vectors of the conjugate gradient method. This leads us to the next section.

4. Superlinear Convergence Rate. First, we will discretize the infinite systems above. As in [CDH], we do this by truncating (3) after n terms, replacing the h_k for $k = 1, \ldots, n$ by the values computed with the discrete Fourier transform, and setting $h_k = 0$ for $k = n+1, \ldots, N-1$, where $N = 2n$. The decay of the h_k's given in Lemma 1 will guarantee that the resulting finite system is only a small perturbation of the infinite system. We denote by $\underline{x}^{(n)}$ and $\underline{r}^{(n)}$, the N-vectors formed by truncating $\underline{x}$ and $\underline{r}$ and by M_n the $N \times N$ matrix formed by truncating M_∞, etc. Then, for instance,

$$M_n = \begin{pmatrix} M_{r,n} & M_{i,n} \\ M_{i,n} & -M_{r,n} \end{pmatrix}.$$

where the (k,j)th entry of $M_{r,n}$ and $M_{i,n}$ are respectively $\sqrt{kj}Re(h_{k+j})$ and $\sqrt{kj}Im(h_{k+j})$.

Our problem is then to solve the truncated version of (8), the $N \times N$ system

$$(I_n + M_n)\underline{x}^{(n)} = \underline{r}^{(n)}.$$

Recall that $\underline{x}^{(n)}$ is subject to a uniqueness condition. Since $f'(0) > 0$, the condition $Im(a_1/f'(0)) = 0$ implies $x_{n+1}^{(n)} = 0$.

Put $A_\infty = I_\infty + M_\infty$, and $A_n = I_n + M_n$. Then A_∞ has a one dimensional null space with null vector

$$\underline{v} = (-Im\ c_1, -\sqrt{2}Im\ c_2, \ldots, Re\ c_1, \sqrt{2}Re\ c_2, \ldots)^T.$$

Since M_∞ is compact, A_n has numerical rank $N-1$ for large $N = 2n$. Moreover, the corresponding near-null vector is approximated by the truncated version of $\underline{v}$; see [An]. We will assume that A_∞ is positive semidefinite. This will imply that A_n is positive semidefinite for large n. (This is the case in our numerical examples.) But then this implies A_n is positive definite on $\underline{v}^{(n)\perp}$ and conjugate gradient will converge in this subspace if the initial guess $\underline{x}_0^{(n)} = \underline{0}$ is chosen.

Our solution can be written in the form

$$\underline{y}^{(n)} = \underline{x}^{(n)} + \delta \underline{v}^{(n)},$$

where δ is to be determined from the uniqueness condition. Thus, conjugate gradient will be applied to find the $\underline{x}^{(n)}$ in $\underline{v}^{(n)\perp}$. Our goal is to give a precise estimate for the superlinear convergence of the method. Define the norm $\|\underline{x}\|_A^2 = \underline{x}^T A \underline{x}$ and the error vector at the qth step $\underline{e}_q = \underline{x}^{(n)} - \underline{x}_q^{(n)}$. We are now ready for the main result.

Theorem 3. *Assume A_∞ is positive semidefinite with exactly one null vector $\underline{v}$. If Γ is of class $C^{l+1,\alpha}$, $l \geq 2$, $0 < \alpha < 1$, then, for large n, the error vector $\underline{e}_q$ at the qth step of the conjugate gradient method applied to $\underline{v}^{(n)\perp}$ satisfies the estimate*

$$\|\underline{e}_{4q}\|_{A_n} \leq \frac{C^q}{((q-1)!)^{2(l-2+\alpha)}} \|\underline{e}_0\|_{A_n}. \tag{9}$$

Here C is a constant that depends on the conformal map.

Proof: The proof follows closely the proof of Theorem 3 in [Chan]. From the standard error analysis of the conjugate gradient method, we have

$$\|\underline{e}_q\|_{A_n} \leq [\min_{P_q} \max_{\lambda \in \sigma(A_n)} |P_q(\lambda)|\,]\, \|\underline{e}_0\|_{A_n}, \tag{10}$$

where the minimum is taken over polynomials of degree q with constant term 1 and the maximum is taken over $\sigma(A_n)$, the spectrum of A_n; see for instance [GVL]. Actually, we restrict A_n to $\underline{v}^{(n)\perp}$ as discussed above so that by [An], for large n, $\sigma(A_n) \subset (\epsilon, \infty)$, for some $\epsilon > 0$ which is independent of n. In the following, we will try to estimate the minimum in (10).

First, we write

$$M_n = W_n^{(k)} + U_n^{(k)}, \quad \forall k \geq 1, \tag{11}$$

where $U_n^{(k)}$ is the matrix obtained from keeping the number $1, 2, \ldots, k$ and number $n+1, n+2, \ldots, n+k$ columns and rows of M_n while replacing all other entries of M_n by zeros. Clearly, rank $(U_n^{(k)}) \leq 4k$. By Lemma 1, for all $k \geq 1$, $W_n^{(k)}$ satisfies

$$\begin{aligned}
\|W_n^{(k)}\|_1 &\leq \sup_{k+1 \leq j \leq n} \sum_{i=k+1}^{n} \sqrt{ij}\,|h_{i+j-1}| \\
&\leq \sup_{k+1 \leq j \leq n} \sum_{i=k+1}^{n} \frac{c\sqrt{ij}}{(i+j-1)^{l+\alpha}} \\
&\leq \sup_{k+1 \leq j \leq n} (\sqrt{j}) \int_k^\infty \frac{c\,dx}{(x+j-1)^{l-1/2+\alpha}} \\
&\leq \frac{c}{k^{l-2+\alpha}}.
\end{aligned}$$

Note that $W_n^{(k)}$ is symmetric since M_n is symmetric. Moreover, for any symmetric matrix we have $\|A\|_2 \leq \|A\|_1$. It follows that

$$\|W_n^{(k)}\|_2 \leq \frac{c}{k^{l-2+\alpha}}, \quad \forall k \geq 1. \tag{12}$$

Let us order the eigenvalues of $M_n =: W_n^{(0)}$ as

$$\mu_0^- \leq \mu_{1/2}^- \leq \mu_1^- \leq \mu_{3/2}^- \leq \cdots \leq \mu_{3/2}^+ \leq \mu_1^+ \leq \mu_{1/2}^+ \leq \mu_0^+.$$

By applying Cauchy Interlace Theorem [GVL] to (11) and using the bound of $\|W_n^{(k)}\|_2$ in (12), we see that

$$|\mu_k^{\pm}| \leq \frac{c}{\lfloor k \rfloor^{l-2+\alpha}}, \quad \forall \lfloor k \rfloor \geq 1.$$

Thus if we order the eigenvalues of $A_n = I_n + W_n^{(0)}$ as

$$0 < \lambda_0^- \leq \lambda_{1/2}^- \leq \lambda_1^- \leq \lambda_{3/2}^- \leq \cdots \leq \lambda_{3/2}^+ \leq \lambda_1^+ \leq \lambda_{1/2}^+ \leq \lambda_0^+,$$

then $\lambda_k^{\pm} = 1 + \mu_k^{\pm}$ for all $k \geq 0$ with

$$1 - \frac{c}{\lfloor k \rfloor^{l-2+\alpha}} \leq \lambda_k^- \leq \lambda_k^+ \leq 1 + \frac{c}{\lfloor k \rfloor^{l-2+\alpha}}, \quad \forall k \geq 1. \tag{13}$$

Having obtained the bounds for $\lambda_k^{\pm}$, we can now construct the polynomial that will give us a bound for (10). Our idea is to choose P_{4q} that annihilates the $2q$ extreme pairs of eigenvalues. Thus consider

$$p_k(x) = (1 - \frac{x}{\lambda_k^+})(1 - \frac{x}{\lambda_{k+1/2}^-})(1 - \frac{x}{\lambda_{k+1/2}^+})(1 - \frac{x}{\lambda_k^-}), \quad \forall k \geq 0.$$

Between the roots $\lambda_k^{\pm}$, the maximum of $|(1 - \frac{x}{\lambda_k^-})(1 - \frac{x}{\lambda_k^+})|$ is attained at the average $x = \frac{1}{2}(\lambda_k^+ + \lambda_k^-)$. Consequently,

$$\begin{aligned}
\max_{x \in [\lambda_{k+1/2}^-, \lambda_{k+1/2}^+]} |p_k(x)| &\leq \frac{(\lambda_k^+ - \lambda_k^-)^2}{4\lambda_k^+ \lambda_k^-} \frac{(\lambda_{k+1/2}^+ - \lambda_{k+1/2}^-)^2}{4\lambda_{k+1/2}^+ \lambda_{k+1/2}^-} \\
&\leq \frac{(\lambda_k^+ - \lambda_k^-)}{4\lambda_k^-} \frac{(\lambda_{k+1/2}^+ - \lambda_{k+1/2}^-)}{4\lambda_{k+1/2}^-}, \\
&\leq \frac{(\lambda_k^+ - \lambda_k^-)^2}{16(\lambda_0^-)^2} \quad \forall k \geq 0.
\end{aligned}$$

But then by (13) we have

$$\max_{x \in [\lambda_{k+1/2}^-, \lambda_{k+1/2}^+]} |p_k(x)| \leq \frac{(\lambda_k^+ - \lambda_k^-)^2}{16(\lambda_0^-)^2} \leq \frac{C}{k^{2(l-2+\alpha)}} \quad \forall k \geq 1.$$

Hence the polynomial $P_{4q} = p_0 p_1 \cdots p_{q-1}$, which annihilates the $2q$ extreme pairs of eigenvalues, satisfies

$$|P_{4q}(x)| \leq \frac{C^q}{((q-1)!)^{2(l-2+\alpha)}} \tag{14}$$

for all $x = \lambda_k^{\pm}$ in the inner interval between $\lambda_{q-1/2}^-$ and $\lambda_{q-1/2}^+$, where the remaining eigenvalues are. Here C is constant that depend only on f. It should be noted that due to the one dimensional null space of compact M_∞, we must have λ_0^- uniformly bounded

away from zero for large N [An, Thm 4.8]. Thus, C is uniformly bounded for large N. (9) now follows directly from (10) and (14).

In the case that A_∞ is not semidefinite we can solve the normal equations by the conjugate gradient method. It is clear that $(I_n + M_n)^2$ will then be positive definite on $\underline{v}^{(n)\perp}$. Using the techniques in the proof of Theorem 3 one can establish a similar result for the normal equations. This will not be done here. Also, in a future paper, we will show that if Γ is analytic, then $\|\underline{e}_{4q}\|_{A_n} \leq C^q r^{q^2} \|\underline{e}_0\|_{A_n}$ for some $r < 1$.

5. Numerical example. In our example we choose $\phi(\varsigma) = \varsigma^3$, and $\chi(\varsigma) = 0$. Then $u(\eta, \mu) = \eta^4 - \mu^4$. Note that, for the conformal map $f(z)$ from the disk, $\phi(z) = (f(z))^3$ and the boundary values at the mesh points are given by $G(z) = 4(Ref(z))^3 - i4(Imf(z))^3$. The discretization error in Table 1 is given by the sup norm

$$\max_{0 \leq j \leq N-1} |\phi(e^{i2\pi j/N}) - \phi_n(e^{i2\pi j/N})|,$$

where ϕ_n is our nth degree approximation to ϕ. In the Table 1 below, *iter* is the number of iterations required by conjugate gradient for the residuals to be $\leq 10^{-14}$. The computations were performed on a SUN Sparc 5 workstation using MATLAB.

For our boundary $\Gamma : \gamma(s)$, we take the Hölder continuous curve given in [Weg, example c]: Let r be a positive real number, $r = l + 1 + \alpha$, then

$$\gamma(s) = A \cdot e^{is} + B|2 \sin \frac{s}{2}|^r exp(ir\frac{(s-\pi)}{2}) - C, \quad 0 \leq s \leq 2\pi, \tag{15}$$

is for $0 < \alpha < 1$ a function in Hölder class $C^{l+1,\alpha}$. If $C = 0.4A + 0.6^r B$, and $|A| > |B| r 2^{r-1}$, then the function

$$f(z) = A\Psi(z) + B(1 - \Psi(z))^r - C,$$

where $\Psi(z) = (5z + 2)/(5 + 2z)$, maps the unit disk conformally onto the region bounded by the curve parameterized by (15). (A numerical approximation to f can be used with similar results.) In this example we take $A = 1$, $B = 0.01$, $l = 2$, $\alpha = 0.5$. (In a later paper we will show that the discrete error behaves like $O(N^{-2.5})$ which can be seen in Table 1.) According to Theorem 3 the error vectors should approach 0 rapidly. We illustrate this in Table 2, where we have computed $\|\underline{x}_{q+1} - \underline{x}_q\|_{A_n}$ as an approximation of $\|\underline{e}_q\|_{A_n}$ for several q. In Figure 1, we plot the eigenvalues of A_n. Note that they are well grouped around 1, as shown in the proof of Theorem 3.

References

[An] P. M. Anselone, *Collectively Compact Operator Approximation Theory,* Prentice-Hall, Inc., 1971.

[Chan] R. H. Chan, *Circulant preconditioners for Hermitian Toeplitz systems,* SIAM J. Matrix Anal. Appl., 10 (1989) 542–550.

[CDH] R. H. Chan, T. K. DeLillo, and M. A. Horn, *The numerical solution of the biharmonic equation by conformal mapping,* submitted for publication.

[Con] J. B. Conway, *A Course in Functional Analysis, 2nd ed.,* Springer-Verlag, 1990.

[GVL] G. Golub, C. Van Loan, *Matrix Computations, 2nd ed.,* John Hopkins U. Press, 1989.

[KK] L. V. Kantorovich and V. I. Krylov, *Approximate Methods of Higher Analysis,* P. Noordhoff Ltd, Groningen, The Netherlands, 1958.

[Musk] N. I. Muskhelishvili, *Some Basic Problems of the Mathematical Theory of Elasticity,* P. Noordhoff Ltd, Groningen, Holland, 1953.

[Po] Ch. Pommerenke, *Boundary Behavior of Conformal Maps,* Springer-Verlag, Berlin, Heidelberg, New York, 1992.

[Weg] R. Wegmann, *Convergence proofs and error estimates for an iterative method for conformal mapping,* Numer. Math. 44 (1984) 435–461.

N	disc. error	iter
32	$1.1 \cdot 10^{-3}$	6
64	$1.4 \cdot 10^{-6}$	5
128	$3.2 \cdot 10^{-8}$	5
256	$5.5 \cdot 10^{-9}$	5
512	$9.5 \cdot 10^{-10}$	5
1024	$2.1 \cdot 10^{-10}$	5

Table 1. A=1, B=0.01, r=3.5.

q	$\|x_{q+1} - x_q\|_{A_n}$
0	$2.1 \cdot 10^{0}$
1	$4.1 \cdot 10^{-1}$
2	$2.6 \cdot 10^{-2}$
3	$1.1 \cdot 10^{-5}$
4	$9.8 \cdot 10^{-10}$
5	$1.9 \cdot 10^{-14}$

Table 2. N=256.

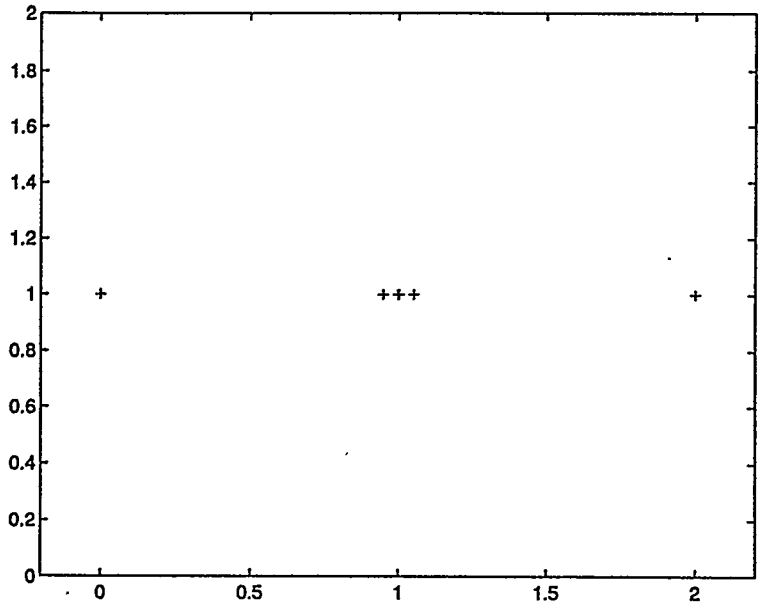


Figure 1. Eigenvalue distribution for example.

TRIANGULAR PRECONDITIONERS FOR SADDLE POINT PROBLEMS WITH A PENALTY TERM

AXEL KLAWONN *

December 1995

Abstract. Triangular preconditioners for a class of saddle point problems with a penalty term are considered. An important example is the mixed formulation of the pure displacement problem in linear elasticity. It is shown that the spectrum of the preconditioned system is contained in a real, positive interval, and that the interval bounds can be made independent of the discretization and penalty parameters. This fact is used to construct bounds of the convergence rate of the GMRES method used with an energy norm. Numerical results are given for GMRES and BI-CGSTAB.

Key words. saddle point problems, penalty term, triangular preconditioners

AMS(MOS) subject classifications.65F10,65N22,65N30

1. Introduction. In this article, we analyze a triangular preconditioner for saddle point problems with penalty terms. An important example of this type of problems is the linear system of equations resulting from a mixed formulation of the pure displacement method for the equations of linear elasticity; see e.g. Braess [1], pp. 252. The penalty parameter then depends on the Poisson ratio or, alternatively the Lamé parameter λ. We only consider symmetric saddle point problems. For related work on the non-symmetric case, see a paper by Elman and Silvester [6] that analyzes the Oseen operator which is obtained by applying a Picard iteration to the Navier-Stokes equations.

We introduce the notation

$$\mathcal{A} := \begin{pmatrix} A & B^t \\ B & -t^2 C \end{pmatrix}, \quad \mathcal{F} := \begin{pmatrix} f \\ g \end{pmatrix}, \quad t \in [0,1],$$

where A and C are positive definite and continuous operators on the Hilbert spaces V and M and the continuous operator B satisfies the inf-sup condition

$$\inf_{q \in M} \sup_{v \in V} \frac{v^t B^t q}{\|v\|_V \|q\|_M} \geq \beta_0^2.$$

We assume that $f \in V', g \in M'$, where V' and M' are the dual spaces of V and M. The product space $V \times M$ is denoted by X.

We consider the following problem

$$\mathcal{A}x = \mathcal{F}. \tag{1}$$

From the well-known Babuška-Brezzi theory, we know that (1) is well-posed.

A number of methods have been proposed to solve this type of problems. For a list of references, see e.g. the introduction in Klawonn [10].

The outline of the remainder of this article is as follows. In Section 2, we discuss the preconditioning strategy. In the case of exact solvers for A and C, we show that

* Westfälische Wilhelms-Universität, Institut für Numerische und instrumentelle Mathematik, Einsteinstraße 62, 48149 Münster, Germany. Electronic mail address: klawonn@math.uni-muenster.de. Dieser Artikel ist Teil einer geplanten Dissertation an der Mathematisch-Naturwissenschaftlichen Fakultät der Westfälischen Wilhelms-Universität Münster. This work was supported in part by the U.S. Department of Energy under contract DE-FG02-92ER25127.

the spectrum of the preconditioned system is bounded independently of the critical parameters by solving a generalized eigenvalue problem. This technique cannot be applied to inexact preconditioners. Neither can a perturbation argument be used as in Klawonn [10] since the preconditioner is indefinite. We find that the preconditioned system $A\hat{B}^{-1}$ is symmetric positive definite in a certain metric which is defined by a symmetric positive definite matrix $\mathcal{H}^{-1}$ and that the generalized eigenvalues of $\mathcal{H}^{-1}A\hat{B}^{-1} = \lambda\mathcal{H}^{-1}$ are bounded independently of the critical parameters. From this, we obtain the bounds of the eigenvalues of $A\hat{B}^{-1}$. We also use this fact to show that GMRES, in a certain metric defined by $\hat{\mathcal{H}}^{-1}$, converges independently of these parameters. Here $\hat{\mathcal{H}}$ is an arbitrary symmetric positive definite matrix with $\bar{C}_0^2\mathcal{H}^{-1} \leq \hat{\mathcal{H}}^{-1} \leq \bar{C}_1^2\mathcal{H}^{-1}$, where $\bar{C}_0, \bar{C}_1 > 0$ are constants independent of the discretization and the penalty parameters. To prove this convergence estimate, we introduce an apparently new technique. We exploit that both, GMRES and the method of conjugate residuals (CR), minimize the residual in the norm used and that $\mathcal{H}^{-1}$ and $\hat{\mathcal{H}}^{-1}$ define equivalent norms. Then, we use that the convergence rate of CR in the $\mathcal{H}^{-1}$–metric can be estimated by the generalized eigenvalues of $\mathcal{H}^{-1}A\hat{B}^{-1} = \lambda\mathcal{H}^{-1}$. Finally, we show that $A\hat{B}^{-1}$ is diagonalizable and use this property to give a convergence estimate for GMRES in the euclidean metric. We also discuss a connection between triangular preconditioners and an algorithm by Bramble and Pasciak [2]. In Section 3, we discuss numerical results for a problem of linear elasticity obtained by using GMRES and BI-CGSTAB.

2. Preconditioning Techniques. In this section, we consider a triangular preconditioner for problem (1). The preconditioned system is either of the form $A\hat{B}^{-1}$ or $\hat{B}^{-1}A$ where $\hat{B}$ is the triangular preconditioner.

We use the following notation

$$\hat{B}_U := \begin{pmatrix} \hat{A} & B^t \\ O & -\hat{C} \end{pmatrix}, \quad \hat{B}_L := \begin{pmatrix} \hat{A} & O \\ B & -\hat{C} \end{pmatrix},$$

Here $\hat{A}$ and $\hat{C}$ are positive definite. We make the following assumptions on A and $\hat{B}$: The matrix $\hat{A}$ is a good preconditioner for A, i.e.

$$\exists a_0, a_1 > 0 \quad a_0^2 u^t \hat{A} u \leq u^t A u \leq a_1^2 u^t \hat{A} u \quad \forall u \in V. \tag{2}$$

The constants a_0, a_1 should preferably be close to each other and be independent of the discretization parameters but there are also other interesting cases. Multigrid and domain decomposition methods are examples of preconditioners that meet these requirements.

We also require that $\hat{C}$ is a good preconditioner for the pressure mass matrix M_p, i.e.

$$\exists m_0, m_1 > 0 \quad m_0^2\, p^t \hat{C} p \leq p^t M_p p \leq m_1^2\, p^t \hat{C} p \quad \forall p \in M \tag{3}$$

and we finally assume that C is spectrally equivalent to $\hat{C}$, i.e.

$$\exists c_0, c_1 > 0 \quad c_0^2\, p^t \hat{C} p \leq p^t C p \leq c_1^2\, p^t \hat{C} p \quad \forall p \in M. \tag{4}$$

A good choice for $\hat{C}$ is a one-level overlapping Schwarz method; see Klawonn [10] and Section 3, Tables 3,6,7.

From the inf-sup condition for B and the continuity of B, we obtain the following inequality

$$\beta_0^2\, p^t M_p p \leq p^t B A^{-1} B^t p \leq \beta_1^2\, p^t M_p p \quad \forall p \in M, \tag{5}$$

with positive constants β_0, β_1; see Brezzi and Fortin [3]. These constants are independent of the discretization and the penalty parameters. We denote by μ an eigenvalue of the generalized eigenvalue problem

$$BA^{-1}B^t p = \mu\, Cp$$

and by μ_{min}, μ_{max} its extreme eigenvalues.

We first restrict our analysis to $\hat{\mathcal{B}}_U$ and drop the subscript U. In the case of $\hat{A} = A$ and $\hat{C} = C$, we use $\mathcal{B}$ rather than $\hat{\mathcal{B}}$.

LEMMA 1.

$$\sigma(\mathcal{A}\mathcal{B}^{-1}) \subset [\mu_{min}, \mu_{max} + 1] \cup \{1\}$$

Proof: The proof follows by considering the generalized eigenvalue problem

$$\mathcal{A}x = \lambda\, \mathcal{B}x.$$

□

From this lemma, we obtain

THEOREM 1.

$$\sigma(\mathcal{A}\mathcal{B}^{-1}) \subset [\beta_0^2 m_0^2, \beta_1^2 m_1^2 + 1] \cup \{1\}$$

Proof: Use the bounds for μ_{min} and μ_{max} obtained from (3) and (5).

□

To provide bounds for $\mathcal{A}\hat{\mathcal{B}}^{-1}$, we cannot solve the generalized eigenvalue problem easily anymore. Nor is it possible to use the techniques applied in Klawonn [10], since $\hat{\mathcal{B}}^{-1}$ is not positive definite.

By making the assumption that

$$1 < a_0 \leq a_1, \tag{6}$$

which can always be achieved by an appropriate scaling, we can show that the spectrum of $\mathcal{A}\hat{\mathcal{B}}^{-1}$ stays bounded independently of the discretization and the penalty parameters. Consider

$$\begin{aligned} \mathcal{A}\hat{\mathcal{B}}^{-1} &= \begin{pmatrix} A & B^t \\ B & -t^2 C \end{pmatrix} \begin{pmatrix} \hat{A}^{-1} & \hat{A}^{-1}B^t\hat{C}^{-1} \\ O & -\hat{C}^{-1} \end{pmatrix} \\ &= \begin{pmatrix} A\hat{A}^{-1} & (A-\hat{A})\hat{A}^{-1}B^t\hat{C}^{-1} \\ B\hat{A}^{-1} & (t^2 C + B\hat{A}^{-1}B^t)\hat{C}^{-1} \end{pmatrix}. \end{aligned}$$

Introduce the notation,

$$\mathcal{H} := \begin{pmatrix} A-\hat{A} & O \\ O & \hat{C} \end{pmatrix}.$$

Multiplying $\mathcal{A}\hat{\mathcal{B}}^{-1}$ by $\mathcal{H}$, we obtain

$$\mathcal{A}\hat{\mathcal{B}}^{-1}\mathcal{H} = \begin{pmatrix} A\hat{A}^{-1}(A-\hat{A}) & (A-\hat{A})\hat{A}^{-1}B^t \\ B\hat{A}^{-1}(A-\hat{A}) & t^2 C + B\hat{A}^{-1}B^t \end{pmatrix},$$

which is a symmetric positive definite matrix; cf. Lemma 2. Thus, the preconditioned system $\mathcal{A}\hat{\mathcal{B}}^{-1}$ is $\mathcal{H}$–normal(1). Introduce the notation

$$\mathcal{T} := \begin{pmatrix} I & 0 \\ BA^{-1} & I \end{pmatrix} \quad \text{and} \quad \tilde{\mathcal{H}} := \begin{pmatrix} A & 0 \\ 0 & S \end{pmatrix},$$

where $S := t^2C + BA^{-1}B^t$ denotes the Schur complement of $\mathcal{A}$ which is obtained by block Gaussian elimination.

LEMMA 2. *There exist positive constants $\tilde{C}_0, \tilde{C}_1$, independent of t, h, such that*

$$\tilde{C}_0\,\tilde{\mathcal{H}} \le \mathcal{A}\hat{\mathcal{B}}^{-1}\mathcal{H} \le \tilde{C}_1\,\tilde{\mathcal{H}},$$

with $\tilde{C}_0 = \min\{(a_0-1),1\}/3$ and $\tilde{C}_1 = 3\max\{(a_1-1),1\}$.[1]

Proof: The proof is based on a block Cholesky factorization of $\mathcal{A}\hat{\mathcal{B}}^{-1}\mathcal{H}$. A direct computation shows:

$$\mathcal{A}\hat{\mathcal{B}}^{-1}\mathcal{H} = \begin{pmatrix} I & 0 \\ BA^{-1} & I \end{pmatrix} \begin{pmatrix} A\hat{A}^{-1}A - A & 0 \\ 0 & S \end{pmatrix} \begin{pmatrix} I & A^{-1}B^t \\ 0 & I \end{pmatrix}. \tag{7}$$

From the assumptions (2), (6), we obtain

$$\hat{C}_0\tilde{\mathcal{H}} \le \begin{pmatrix} A\hat{A}^{-1}A - A & 0 \\ 0 & S \end{pmatrix} \le \hat{C}_1\tilde{\mathcal{H}}, \tag{8}$$

where $\hat{C}_0 := \min\{(a_0-1),1\}$ and $\hat{C}_1 := \max\{(a_1-1),1\}$ are positive constants.

We now show that the eigenvalues of $\mathcal{T}\tilde{\mathcal{H}}\mathcal{T}^t$ are bounded by the extreme eigenvalues of $\tilde{\mathcal{H}}$,

$$\frac{1}{3}\tilde{\mathcal{H}} \le \mathcal{T}\tilde{\mathcal{H}}\mathcal{T}^t \le 3\tilde{\mathcal{H}}. \tag{9}$$

We have

$$\begin{aligned} \mathcal{T}\tilde{\mathcal{H}}\mathcal{T}^t &= \begin{pmatrix} I & 0 \\ BA^{-1} & I \end{pmatrix} \begin{pmatrix} A & 0 \\ 0 & S \end{pmatrix} \begin{pmatrix} I & A^{-1}B^t \\ 0 & I \end{pmatrix} \\ &= \begin{pmatrix} A & B^t \\ B & S + BA^{-1}B^t \end{pmatrix}. \end{aligned}$$

From this, we get

$$\begin{aligned} x^t\mathcal{T}\tilde{\mathcal{H}}\mathcal{T}^t x &= \begin{pmatrix} u \\ p \end{pmatrix}^t \begin{pmatrix} A & B^t \\ B & S + BA^{-1}B^t \end{pmatrix} \begin{pmatrix} u \\ p \end{pmatrix} \\ &= u^tAu + 2p^tBu + p^tSp + p^tBA^{-1}B^tp \\ &\le u^tAu + 2|p^tBu| + 2p^tSp. \end{aligned} \tag{10}$$

Applying the Cauchy-Schwarz inequality and $ab \le a^2/2 + b^2/2$, we obtain

$$\begin{aligned} |p^tBu| &= |p^tBA^{-1/2}A^{1/2}u| \\ &\le (p^tBA^{-1}B^tp)^{1/2}(u^tAu)^{1/2} \\ &\le \frac{1}{2}p^tBA^{-1}B^tp + \frac{1}{2}u^tAu \\ &\le \frac{1}{2}(p^tSp + u^tAu). \end{aligned}$$

[1] As usual, $A \le B$ means $B - A$ is symmetric positive semi-definite.

Hence, we get from (10)

$$\begin{aligned} x^t\mathcal{T}\tilde{\mathcal{H}}\mathcal{T}^t x &\leq 2u^tAu + 3p^tSp \\ &\leq 3x^t\tilde{\mathcal{H}}x. \end{aligned}$$

To obtain a lower bound, consider the generalized Rayleigh-quotient

$$\frac{x^t\mathcal{T}\tilde{\mathcal{H}}\mathcal{T}^t x}{x^t\tilde{\mathcal{H}}x} = \frac{y^t\tilde{\mathcal{H}}y}{y^t\mathcal{T}^{-1}\tilde{\mathcal{H}}\mathcal{T}^{-t}y},$$

where we have used the substitution $y := \mathcal{T}^t x$. As in the case of the upper bound, we obtain

$$\begin{aligned} y^t\mathcal{T}^{-1}\tilde{\mathcal{H}}\mathcal{T}^{-t}y &= \begin{pmatrix} v \\ q \end{pmatrix}^t \begin{pmatrix} I & 0 \\ -BA^{-1} & I \end{pmatrix} \begin{pmatrix} A & 0 \\ 0 & S \end{pmatrix} \begin{pmatrix} I & -A^{-1}B^t \\ 0 & I \end{pmatrix} \begin{pmatrix} v \\ q \end{pmatrix} \\ &= \begin{pmatrix} v \\ q \end{pmatrix}^t \begin{pmatrix} A & -B^t \\ -B & S + BA^{-1}B^t \end{pmatrix} \begin{pmatrix} v \\ q \end{pmatrix} \\ &= v^tAv - 2q^tBv + q^tSq + q^tBA^{-1}B^tq \\ &\leq v^tAv + 2|q^tBv| + 2q^tSq \\ &\leq 3y^t\tilde{\mathcal{H}}y. \end{aligned}$$

From (7), (8) and (9), we obtain

$$\tilde{C}_0\tilde{\mathcal{H}} \leq \mathcal{A}\hat{\mathcal{B}}^{-1}\mathcal{H} \leq \tilde{C}_1\tilde{\mathcal{H}}.$$

□

From this lemma, we derive

LEMMA 3. *There exist positive constants C_0, C_1, independent of t, h, such that*

$$C_0\mathcal{H} \leq \mathcal{A}\hat{\mathcal{B}}^{-1}\mathcal{H} \leq C_1\mathcal{H}.$$

Proof: The lemma follows immediately from Lemma 2 since $\hat{C}$ is spectrally equivalent to $t^2C + BA^{-1}B^t$ and $A - \hat{A}$ to A.

□

REMARK 1. *This lemma could also be used to prove Theorem 1 of Bramble and Pasciak [2].*

Since we have made the assumption that $1 < a_0 \leq a_1$, $A - \hat{A}$ and $\hat{C}$ are positive definite, and $\mathcal{H}$ defines a new inner product on X. We are now able to give bounds for the spectrum of $\mathcal{A}\hat{\mathcal{B}}^{-1}$.

THEOREM 2. *There exist positive constants C_0, C_1, independent of t, h, such that*

$$\sigma(\mathcal{A}\hat{\mathcal{B}}^{-1}) \subset [C_0, C_1].$$

Proof: The constants C_0, C_1 in Lemma 3 provide lower and upper bounds for the eigenvalues of the generalized eigenvalue problem

$$\mathcal{A}\hat{\mathcal{B}}^{-1}\mathcal{H}x = \lambda\mathcal{H}x.$$

Since $\mathcal{H}$ is non-singular, this problem has the same eigenvalues as

$$\mathcal{A}\hat{\mathcal{B}}^{-1}y = \lambda y.$$

□

COROLLARY 1. *$\mathcal{H}^{-1}$ also defines an $X-$inner product and $A\hat{B}^{-1}$ is symmetric positive definite in this inner product, i.e. $\mathcal{H}^{-1}A\hat{B}^{-1}$ is symmetric and there exist positive constants C_0, C_1, such that $C_0\,\mathcal{H}^{-1} \leq \mathcal{H}^{-1}A\hat{B}^{-1} \leq C_1\,\mathcal{H}^{-1}$.*

Proof: The proof follows immediately from Lemma 3 by multiplying by $\mathcal{H}^{-1}$ from both sides.

□

REMARK 2. *Since it can be shown that*

$$A\hat{B}_U^{-1}\mathcal{H} = \mathcal{H}\hat{B}_L^{-1}A,$$

the previous results obtained for $A\hat{B}_U^{-1}$ also apply to the spectrum of $\hat{B}_L^{-1}A$.

We now use the bounds of Corollary 1 to provide bounds of the convergence rate of the GMRES method used with a norm equivalent to the $\mathcal{H}^{-1}-$norm. Here, we exploit the fact that both, CR and GMRES, minimize the residual in the norm used; see e.g. Bruaset [4], Freund, Golub, and Nachtigal [7] or Saad and Schultz [11].

THEOREM 3. *Let $\hat{\mathcal{H}}$ be a positive definite matrix, such that $\bar{C}_0^2\ \mathcal{H}^{-1} \leq \hat{\mathcal{H}}^{-1} \leq \bar{C}_1^2\ \mathcal{H}^{-1}$, where $\bar{C}_0, \bar{C}_1$ are positive constants independent of the discretization and penalty parameters. Then,*

$$\|r_n\|_{\hat{\mathcal{H}}^{-1}} \leq \frac{\bar{C}_1}{\bar{C}_0}\, 2\, \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^n\ \|r_0\|_{\hat{\mathcal{H}}^{-1}},$$

where r_n is the n-th residual, $r_0 = b - A\hat{B}^{-1}x_0$ and $\kappa := \kappa(A\hat{B}^{-1}) \leq \frac{C_1}{C_0}$ is the condition number of $A\hat{B}^{-1}$ in the $\mathcal{H}^{-1}-$inner product.

Proof:

$$\begin{aligned}
\|r_n\|_{\hat{\mathcal{H}}^{-1}} &= \min_{\Phi_n \in \mathcal{P}_n, \Phi_n(0)=1} \|\Phi_n(A\hat{B}^{-1})r_0\|_{\hat{\mathcal{H}}^{-1}} \\
&\leq \bar{C}_1 \min_{\Phi_n \in \mathcal{P}_n, \Phi_n(0)=1} \|\Phi_n(A\hat{B}^{-1})r_0\|_{\mathcal{H}^{-1}} \\
&\leq \bar{C}_1\, 2\, \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^n\ \|r_0\|_{\mathcal{H}^{-1}} \\
&\leq \frac{\bar{C}_1}{\bar{C}_0}\, 2\, \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^n\ \|r_0\|_{\hat{\mathcal{H}}^{-1}}.
\end{aligned}$$

□

REMARK 3. *Our convergence estimate only depends on the square root of the condition number of the preconditioned problem. Note that this estimate matches, except for a leading factor, the standard estimate for the conjugate gradient method applied to positive definite symmetric problems.*

Finally, we give a well-known convergence estimate for GMRES in the L_2-norm; see e.g. Saad and Schultz [11]. This estimate is based on the eigenvalues of the preconditioned system. We make use of the fact that $A\hat{B}^{-1}$ is diagonalizable. This can easily be seen from the following arguments:
$A\hat{B}^{-1}\mathcal{H}$ is symmetric positive definite; so is

$$\mathcal{H}^{-1/2}(A\hat{B}^{-1}\mathcal{H})\mathcal{H}^{-1/2} = \mathcal{H}^{-1/2}A\hat{B}^{-1}\mathcal{H}^{1/2}.$$

Thus, $\mathcal{H}^{-1/2}A\hat{B}^{-1}\mathcal{H}^{1/2}$ is diagonalizable, i.e. there exists an unitary matrix $\tilde{\mathcal{Q}}$, such that

$$\mathcal{H}^{-1/2}A\hat{B}^{-1}\mathcal{H}^{1/2} = \tilde{\mathcal{Q}}\mathcal{D}\tilde{\mathcal{Q}}^{-1},$$

where $\mathcal{D} := diag\{\lambda_i\}$ with λ_i the eigenvalues of $\mathcal{A}\hat{\mathcal{B}}^{-1}$. Note that $\mathcal{A}\hat{\mathcal{B}}^{-1}$ and $\mathcal{H}^{-1/2}\mathcal{A}\hat{\mathcal{B}}^{-1}\mathcal{H}^{1/2}$ have the same eigenvalues. We also obtain

$$\begin{aligned}\mathcal{A}\hat{\mathcal{B}}^{-1} &= \left(\mathcal{H}^{1/2}\tilde{\mathcal{Q}}\right)\mathcal{D}(\mathcal{H}^{1/2}\tilde{\mathcal{Q}})^{-1} \\ &=: \mathcal{Q}\mathcal{D}\mathcal{Q}^{-1},\end{aligned}$$

i.e. $\mathcal{A}\hat{\mathcal{B}}^{-1}$ is diagonalizable with the diagonalization matrix $\mathcal{Q} := \mathcal{H}^{1/2}\tilde{\mathcal{Q}}$. Unfortunately, our estimate depends on the condition number of the matrix $\mathcal{H}^{1/2}$.

THEOREM 4. *We have*

$$\|r_n\|_2 \leq \|\mathcal{H}^{1/2}\|_2 \ \|\mathcal{H}^{-1/2}\|_2 \ 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^n \ \|r_0\|_2,$$

where r_n is the n-th residual and $\kappa := \kappa(\mathcal{A}\hat{\mathcal{B}}^{-1}) := \lambda_{max}/\lambda_{min}$ is the spectral condition number of the preconditioned system.

Proof: Under the assumptions made, the following estimate is satisfied,

$$\|r_n\|_2 \leq \|\mathcal{Q}\|_2 \ \|\mathcal{Q}^{-1}\|_2 \min_{\Phi_n\in\mathcal{P}_n,\Phi_n(0)=1} \max_{\lambda\in\sigma(\mathcal{A}\hat{\mathcal{B}}^{-1})} |\Phi_n(\lambda)| \, \|r_0\|_2,$$

see Saad and Schultz [11], Proposition 4. Since the spectrum of $\mathcal{A}\hat{\mathcal{B}}^{-1}$ is real and positive, we can use the Chebyshev polynomials to construct an upper bound. The given estimate follows from the invariance of the euclidean norm under unitary transformations, i.e.

$$\begin{aligned}\|\mathcal{Q}\|_2 &= \|\mathcal{H}^{1/2}\tilde{\mathcal{Q}}\|_2 = \|\mathcal{H}^{1/2}\|_2, \\ \|\mathcal{Q}^{-1}\|_2 &= \|\tilde{\mathcal{Q}}^{-1}\mathcal{H}^{-1/2}\|_2 = \|\mathcal{H}^{-1/2}\|_2.\end{aligned}$$

□

Note that the convergence estimate in Theorem 4 is independent of the penalty parameter but unfortunately it normally still depends on h. To give an estimate of $\kappa(\mathcal{H}^{1/2})$ for a particular discretization, we now restrict ourselves to the mixed formulation of the equations of linear elasticity discretized by a $Q_1(h)$ iso $Q_1(2h)$ macro-element; see Section 3. In this case, we have $\kappa(\mathcal{H}^{1/2}) \in O(1/h)$. Since $\mathcal{H}^{1/2}$ is symmetric positive definite, this can be immediately seen from the extreme eigenvalues:

$$\begin{aligned}\lambda_{max}(\mathcal{H}^{1/2}) &\leq const\,(1/h), \\ \lambda_{min}(\mathcal{H}^{1/2}) &\geq const\,(1+h).\end{aligned}$$

A factor of $1/h$ appears in the bound given by Theorem 4. However, our numerical experiments do not reflect the presence of such a factor; see Section 3.

3. Numerical Examples. In this section, we apply the triangular preconditioner to the mixed formulation of planar, linear elasticity. For simplicity, we work with the following formulation

$$\begin{aligned}\mu\,(\nabla u, \nabla v)_0 + (\mathrm{div} v, p)_0 &= < f, v > \quad \forall v \in V := (H^1_\Gamma(\Omega))^2, \\ (\mathrm{div} u, q)_0 - \frac{1}{\lambda+\mu}\,(p,q)_0 &= 0 \qquad\quad \forall q \in M := L_2(\Omega),\end{aligned}$$

with $H^1_\Gamma(\Omega) := \{v \in H^1(\Omega) : v_{|\Gamma_0} = 0\}$. Γ_0 is the part of the boundary where Dirichlet conditions are imposed and λ, μ are the Lamé parameters. All results shown are for

TABLE 1
Iteration counts for exact solvers as preconditioners for A and C, $\nu = 0.3$.

Grid	20×10	40×20	60×30	80×40	100×50	120×60	140×70
GMRES	10	11	11	12	12	12	13
BI-CGSTAB	5	5	6	6	6	6	6

mixed boundary conditions with $\Gamma_0 := \{x = (x_1, x_2) \in \partial\Omega : x_1 < -0.8\}$ and the region $[-1,1] \times [-1,1]$. We note that our model is mathematically equivalent to the full elasticity problem only in the case of Dirichlet conditions on the whole boundary.

For growing λ, the considered material becomes more incompressible. Instead of using the Lamé constants λ and μ, we can also work with Young's elasticity modulus E and the Poisson ratio ν. These parameters are related to each other by the following equations

$$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}, \quad \mu = \frac{E}{2(1+\nu)}. \tag{11}$$

The relation between the penalty parameter t and the Poisson ratio ν is given by $t := (1+\nu)(1-2\nu)/(E\nu)$. Without loss of generality, we set $E = 1$. We discretize by a $Q_1(h)$ iso $Q_1(2h)$ macro-element, i.e. we use piecewise bilinear polynomials on quadrilaterals on a grid with mesh size h for the displacements u and piecewise bilinear polynomials on quadrilaterals with mesh size $2h$ for the Lagrange multiplier p. For a proof that the inf-sup condition of B^t holds for this element; see Girault and Raviart [8] or Brezzi and Fortin [3].

All computations were carried our on a SUN SPARC 10 workstation using the numerical software package PETSc developed by William Gropp and Barry Smith at the Argonne National Laboratory; see Gropp and Smith [9] or Smith [13]. The initial guess is 0, and the stopping criterion is $\|r_k\|_2/\|r_0\|_2 < 10^{-5}$, where r_k is the k-th residual.

We give numerical results for two Krylov space methods, GMRES and BI-CGSTAB; see e.g. [5, 11, 12, 14]. We use a version of GMRES without restarts but we also ran a version with a restart every 10 iterations. The number of iterations for this latter version was always just 1-2 iterations larger than for the GMRES method without a restart. We used right-oriented preconditioning with $\hat{\mathcal{B}}_U^{-1}$ for GMRES and left-oriented preconditioning with $\hat{\mathcal{B}}_L^{-1}$ for BI-CGSTAB and we only use the L_2- rather than the $\mathcal{H}^{-1}-$metric. The numerical results suggest that the number of iterations is bounded independently of the critical parameters h and t. We point out that BI-CGSTAB always converges smoothly in our experiments.

To see how the Krylov space methods behave under the best of circumstances, we first conducted some experiments using exact solvers, i.e. $\hat{A} = A$ and $\hat{C} = C$; see Tables 1 and 4.

In another series of experiments, we use different preconditioners for A and C. We present results with a two-level multigrid preconditioner with a V-cycle including one pre- and one post-smoothing symmetric Gauss-Seidel step defining $\hat{A}$, and a one-level symmetric multiplicative overlapping Schwarz method with the minimal overlap of one node as $\hat{C}$; see Tables 2, 3, 5, 6, 7.

Finally, we show that the assumption $a_0 > 1$ is not necessary for the convergence of the Krylov space methods. We conducted some experiments with $\hat{A} := \epsilon A$, where

TABLE 2

Iteration counts for a two-level multigrid preconditioner with a standard V-cycle defining $\hat{A}$, and $\hat{C} = C$, and $\nu = 0.3$.

Grid	20×10	40×20	60×30	80×40	100×50	120×60	140×70
GMRES	13	14	14	15	15	15	15
BI-CGSTAB	7	7	7	7	7	7	7

TABLE 3

Iteration counts for a two-level multigrid preconditioner with a standard V-cycle defining $\hat{A}$ and a one-level symmetric multiplicative overlapping Schwarz method with the minimal overlap of one node defining $\hat{C}$, and $\nu = 0.3$.

Grid	20×10	40×20	60×30	80×40	100×50	120×60	140×70
GMRES	13	14	14	15	15	15	15
BI-CGSTAB	7	7	7	7	7	7	7

TABLE 4

Iteration counts for exact solvers as preconditioners for A and C on a 80×40 grid.

ν	0.3	0.4	0.49	0.499	0.4999	0.49999	0.499999	0.5
GMRES	12	13	14	14	14	14	14	14
BI-CGSTAB	6	7	7	7	7	7	7	7

TABLE 5

Iteration counts for a two-level multigrid method with a standard V-cycle defining $\hat{A}$ and $\hat{C} = C$ on a 80×40 grid.

ν	0.3	0.4	0.49	0.499	0.4999	0.49999	0.499999	0.5
GMRES	15	16	17	17	17	17	17	17
BI-CGSTAB	7	7	7	7	7	7	7	7

TABLE 6

Iteration counts for an exact solver as $\hat{A}$ and a one-level symmetric multiplicative overlapping Schwarz method with the minimal overlap of one node as $\hat{C}$ on a 80×40 grid.

ν	0.3	0.4	0.49	0.499	0.4999	0.49999	0.499999	0.5
GMRES	12	13	14	14	14	14	14	14
BI-CGSTAB	6	7	7	7	7	7	7	7

TABLE 7

Iteration counts for a two-level multigrid method with a standard V-cycle as $\hat{A}$ and a one-level symmetric multiplicative overlapping Schwarz method with the minimal overlap of one node as $\hat{C}$ on a 80×40 grid.

ν	0.3	0.4	0.49	0.499	0.4999	0.49999	0.499999	0.5
GMRES	15	16	17	17	17	17	17	17
BI-CGSTAB	7	7	7	7	7	7	7	7

TABLE 8
Iteration counts for $\hat{A} = \epsilon A$ and $\hat{C} = C$ on a 80×40 Grid with $\nu = 0.3$.

ϵ	0.8	0.9	0.99	1.01	1.1	1.2
GMRES	13	13	13	13	14	14
BI-CGSTAB	6	6	6	6	6	6

ϵ is a parameter at our disposal. The numerical results show that the methods still converge, even when $\epsilon > 1$; see Table 8.

REFERENCES

[1] Dietrich Braess. *Finite Elemente.* Springer-Verlag, 1992.

[2] James H. Bramble and Joseph E. Pasciak. A preconditioning technique for indefinite systems resulting from mixed approximations of elliptic problems. *Math. Comp.*, 50:1–17, 1988.

[3] Franco Brezzi and Michel Fortin. *Mixed and Hybrid Finite Element Methods.* Springer-Verlag, 1991.

[4] Are Magnus Bruaset. *A survey of preconditioned iterative methods.* Pitman Research Notes in Mathematics Series. Longman Scientific and Technical, 1995.

[5] Stanley C. Eisenstat, Howard C. Elman, and Martin H. Schultz. Variational iterative methods for nonsymmetric systems of linear equations. *SIAM J. Numer. Anal.*, 20 (2):345–357, 1983.

[6] Howard Elman and David Silvester. Fast Nonsymmetric Iterations and Preconditioning for Navier-Sokes Equations. Technical Report UMIAC CS-TR-94-66, University of Maryland, June 1994.

[7] R. Freund, G. H. Golub, and N. Nachtigal. *Iterative Solution of Linear Systems*, pages 57–100. Acta Numerica 1992. Cambridge University Press, 1991.

[8] Vivette Girault and Pierre-Arnaud Raviart. *Finite Element Methods for Navier-Stokes Equations.* Springer-Verlag, 1986.

[9] William D. Gropp and Barry F. Smith. Scalable, extensible, and portable numerical libraries. In *Proceedings of Scalable Parallel Libraries Conference*, pages 87–93. IEEE, 1994.

[10] Axel Klawonn. An optimal preconditioner for a class of saddle point problems with a penalty term. Technical Report 676, Courant Institute of Mathematical Sciences, New York University, December 1994.

[11] Y. Saad and M. H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comp.*, 7:856–869, 1986.

[12] Gerard L.G. Sleijpen, Henk A. Van der Vorst, and Diederik R. Fokkema. BiCGstab(l) and other hybrid Bi-CG methods. Technical report, University of Utrecht, February 1994.

[13] Barry F. Smith. Extensible PDE solvers package users manual. Technical Report ANL 94/40, Argonne National Laboratory, September 1994.

[14] Henk A. van der Vorst. BI-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM J. Sci. Comput.*, 13(2):631–644, Mar. 1992.