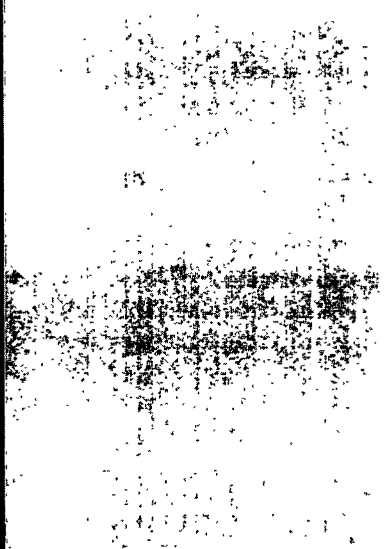
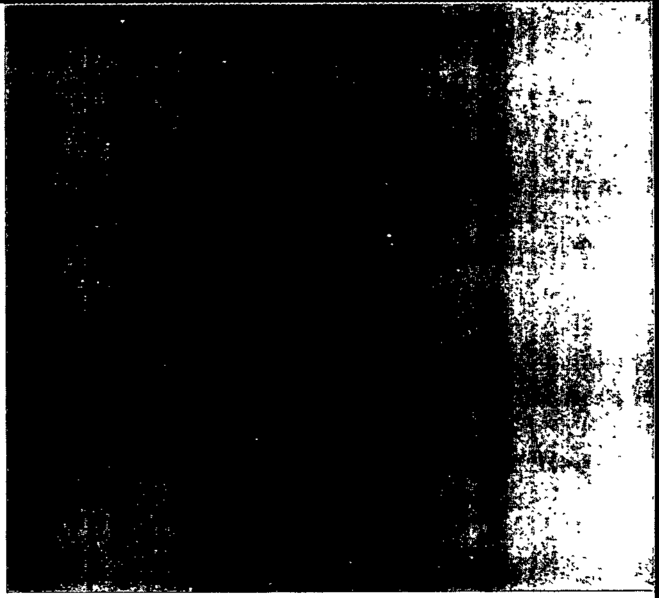
DE-FG03-94ER25201

Colorado Conference on Iterative Methods

Breckenridge, Colorado
April 5-9, 1994

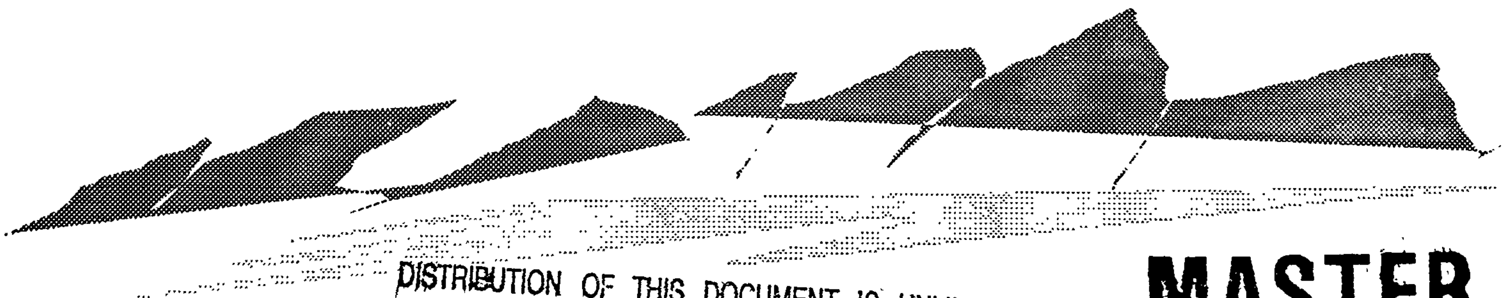
Volume 2





Sponsored by
The Department of Energy
and
The National Science Foundation

Organized By
The University of Colorado
Front Range Scientific Computations, Inc.



Colorado Conference on Iterative Methods

Conference Chairmen

Tom Manteuffel and
Steve McCormick
University of Colorado

Program Committee

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

Tuesday, April 5

Domain Decomposition Methods I — Chair: Seymour Parter — Room A

8:00 - 8:25	*Michael Pernice*	Domain Decomposed Preconditioners with Krylov Subspace Methods as Subdomain Solvers
8:25 - 8:50	*W.K. Tsui*	Domain Decomposition Methods for Solving an Image Problem
8:50 - 9:15	*Marc Garbey*	A Schwarz Alternating Procedure for Singular Perturbation Problems
9:15 - 9:40	*Christina Christara*	Schwarz and Multilevel Methods for Quadratic Spline Collocation

Nonlinear Problems I — Chair: Homer Walker — Room B

8:00 - 8:25	*Masao Igarasi*	On the Convergence Processes of Newton-Raphson Iteration Methods
8:25 - 8:50	*Homer F. Walker*	Choosing the Forcing Terms in an Inexact Newton Method

8:50 - 10:15 Coffee Break

Domain Decomposition Methods II — Chair: Seymour Parter — Room A

10:15 - 10:40	*Xiao-Chuan Cai*	Domain Decomposition Based Iterative Methods for Nonlinear Elliptic Finite Element Problems
10:40 - 11:05	*Seongjai Kim*	Parallel Iterative Procedures for Approximate Solutions of Wave Propagation by Finite Element and Finite Difference Methods
11:05 - 11:30	*Tony Chan*	Multigrid and Multilevel Domain Decomposition for Unstructured Grids
11:30 - 11:55	*Steven M. McKay*	The Use of the Spectral Method within the Fast Adaptive Composite Grid Method

Nonlinear Problems II — Chair: Homer Walker — Room B

10:15 - 10:40	*Scott Hutchinson*	A Two-Level Parallel Direct Search Implementation for Arbitrarily Sized Objective Functions
10:40 - 11:05	*Rosemary Renaut*	Parallel Algorithms for Unconstrained Optimization by Multisplitting with Inexact Subspace Search - The Abstract
11:05 - 11:30	*Randall Bramley*	Solving Linear Inequalities in a Least Squares Sense
11:30 - 11:55	*Matthias Heinkenschloss*	Numerical Solution of Control Problems Governed by Nonlinear Differential Equations

12:00 - 4:30 Informal Discussion

Integral Equations and Inverse Problems — Chair: Nick Trefethen — Room A

4:45 - 5:10	*J. White*	Comparing Precorrected-FFT and Fast Multipole Algorithms for Solving Three-Dimensional Potential Integral Equations
5:10 - 5:35	*Curt Vogel*	The Numerical Solution of Total Variation Minimization Problems in Image Processing
5:35 - 6:00	*C.T. Kelley*	GMRES and Integral Operators
6:00 - 6:25	*J.G. Wade*	Iterative Methods for Distributed Parameter Estimation in Parabolic PDE

Eigenvalue Problems — Chair: Roland Freund — Room B

4:45 - 5:10	*Clemens W. Brand*	Preconditioned Iterations to Calculate Extreme Eigenvalues
5:10 - 5:35	*Andreas Stathopoulos*	Overlapping Domain Decomposition Preconditioners for the Generalized Davidson Method for the Eigenvalue Problem
5:35 - 6:00	*Victor Pan*	New Algorithms for the Symmetric Tridiagonal Eigenvalue Computation

Workshop: Iterative Software Kernels Chair: Iain Duff Room A
(Evening: 8:00p - 10:00p)

Iain Duff	Current status of user level sparse BLAS
Michael A. Heroux	Current status of the Sparse BLAS Toolkit
Craig C. Douglas	Adding Matrix-Matrix and Matrix-Matrix-Matrix Multiply to the Sparse BLAS Toolkit

Wednesday, April 6

Nonsymmetric Solvers I Chair: Tom Manteuffel Room A

8:00 - 8:25	*Tobin Driscoll*	Conformal Mapping and Convergence of Krylov Iterations
8:25 - 8:50	*Kim-Chuan Toh*	Convergence Estimates for Iterative Methods Via the Kreiss Matrix Theorem on a General Complex Domain
8:50 - 9:15	*N. J. Meyers*	An Iterative Method for the Solution of Linear Systems Using the Faber Polynomials for Annular Sectors
9:15 - 9:40	*Gerhard Starke*	Subspace Orthogonalization for Substructuring Preconditioners for Nonsymmetric Systems of Linear Equations

Parallel Computation I Chair: Howard Elman Room B

8:00 - 8:25	*Wayne Joubert*	PCG: A Software Package for the Iterative Solution of Linear Systems on Scalar, Vector & Parallel Computers
8:25 - 8:50	*Claude Pommerell*	Migration of Vectorized Iterative Solvers to Distributed Memory Architectures
8:50 - 9:15	*Youcef Saad*	P_SPARSLIB: A Parallel Sparse Iterative Solution Package
9:15 - 9:40	*Barry Smith*	Portable, Parallel, Reusable Krylov Space Codes
9:40 - 10:15	Coffee Break	

Nonsymmetric Solvers II — Chair: Tom Manteuffel — Room A

10:15 - 10:40	*Emanuel Knill*	Minimal Residual Method Stronger than Polynomial Preconditioning
10:40 - 11:05	*Jane Cullum*	Peaks, Plateaus, Numerical Instabilities, and Achievable Accuracy in Galerkin and Norm Minimizing Procedures for Solving Ax=b
11:05 - 11:30	*Karl Gustafson*	Computational Trigonometry
11:30 - 11:55	*Olavi Nevanlinna*	Convergence of Arnoldi Method

Parallel Computation II — Chair: Howard Elman — Room B

10:15 - 10:40	*Anne E. Trefethen*	The Conjugate Gradient NAS Parallel Benchmark on the IBM SP1
10:40 - 11:05	*Gene Poole*	Advancements and Performance of Iterative Methods In Industrial Applications Codes on Cray Parallel/Vector Supercomputers
11:05 - 11:30	*Shu-Mei C. Richman*	A Component Analysis Based On Serial Results for Analyzing Performance of Parallel Iterative Programs
11:30 - 11:55	*Michael Heroux*	Performance Analysis of High Quality Parallel Preconditioners Applied to 3d Finite Element Structural Analysis

12:00 - 4:30 Informal Discussion

Iterative Methods: Theory — Chair: Roland Freund — Room A

4:45 - 5:10	*Eugene L. Wachspress*	Recent ADI Iteration Analysis and Results
5:10 - 5:35	*W.E. Boyse*	A Sparse Matrix Iterative Method for Efficiently Computing Multiple Simultaneous Solutions
5:35 - 6:00	*Tugral Dayar*	On the Effects of Using the GTH method in the Iterative Aggregation/Disaggregation Technique
6:00 - 6:25	*Eldar Giladi*	On the Interplay Between Inner and Outer Iterations for a Class of Iterative Methods

Software and Programming Environments — Chair: Steve Ashby — Room B

4:45 - 5:10	*Are Magnus Bruaset*	Object-Oriented Design of Preconditioned Iterative Methods
5:10 - 5:35	*Linda Hayes*	VOILA-A Visual Object-Oriented Iterative Linear Algebra Problem Solving Environment
5:35 - 6:00	*D. Kim*	Multilevel Adaptive Solution Procedure for Material Nonlinear Problems in Visual Programming Environment
6:00 - 6:25	*David R. Kincaid*	ITPACK Project: Past, Present, and Future

Workshop: Recent Progress and Advances in Iterative Software (Evening: 8:00p - 10:00p) — Chair: Graham Carey — Room A

David M. Young	Origins of the ITPACK Project
David Kincaid	Recent Developments on ITPACK
Graham Carey	Design Considerations for a Portable Parallel Package
Wayne Joubert	Adapting Iterative Software Libraries to Parallel Environments
Rossen Parashkevov	Operator-based Iterative Tools

Thursday, April 7

Nonsymmetric Solvers III — Chair: Anne Greenbaum — Room A

8:00 - 8:25	*Diederik Fokkema*	Generalized Conjugate Gradient Squared
8:25 - 8:50	*Roland Freund*	Block Quasi-Minimal Residual Iterations for Non-Hermitian Linear Systems
8:50 - 9:15	*Noel M. Nachtigal*	A Look-Ahead Variant of TFQMR
9:15 - 9:40	*Tedd Szeto*	Composite-Step Product Methods for Solving Nonsymmetric Linear Systems

Parallel Computation III — Chair: Dan Quinlan — Room B

8:00 - 8:25	*Lei Li*	A Divide-and-Inner Product Parallel Algorithm for Polynomial Evaluation
8:25 - 8:50	*Dan Hu*	Parallelizing Sylvester-Like Operations on a Distributed Memory Computer
8:50 - 9:15	*H.S. Kohli*	Maximizing Sparse Matrix Vector Product Performance in MIMD Computers
9:15 - 9:40	*John Shadid*	Parallel Performance of a Preconditioned CG Solver for Unstructured Finite Element Applications

9:40 - 10:15 Coffee Break

Nonsymmetric Solvers IV — Chair: Anne Greenbaum — Room A

10:15 - 10:40	*E. Gallopoulos*	Matrix-Valued Polynomials in Lanczos Type Methods
10:40 - 11:05	*Teri Barth*	Variable Metric Conjugate Gradient Methods
11:05 - 11:30	*Ron Morgan*	Some Comparison of Restarted GMRES and QMR for Linear and Nonlinear Problems
11:30 - 11:55	*David Young*	MGMRES: A Generalization of GMRES for Solving Large Sparse Nonsymmetric Linear Systems

Parallel Computation IV — Chair: Dan Quinlan — Room B

10:15 - 10:40	*Larry Reeves*	Adapting Implicit Methods to Parallel Processors
10:40 - 11:05	*A. Basermann*	Parallelizing Iterative Solvers for Sparse Systems of Equations and Eigenproblems on Distributed-Memory Machines
11:05 - 11:30	*R.P. Silva*	A Parallel Implementation of an EBE Solver for the Finite Element Method
11:30 - 11:55	*Martin Bucker*	An Implementation of the TFQMR-Algorithm on a Distributed Memory Machine

12:00 - 4:30 Informal Discussion

Student Paper Winners		Chair: Tom Manteuffel and Steve McCormick	Room A
4:45 - 5:10	*Qing He*	Parallel Algorithms for Unconstrained Optimizations by Multisplitting	
5:10 - 5:35	*Lina Hemmingsson*	Analysis of Semi-Toeplitz Preconditioners for First-Order PDEs	
5:35 - 6:00	*Johannes Tausch*	Equivariant Preconditioners for Boundary Element Methods	

ODE Solvers		Chair: Paul Saylor	Room B
4:45 - 5:10	*Vladimir Druskin*	Explicit and Implicit ODE Solvers Using Krylov Subspace Optimization: Application to the Diffusion Equation and Parabolic Maxwell's System	
5:10 - 5:35	*A. Lorber*	On the Relationship Between ODE Solvers and Iterative Solvers for Linear Equations	
5:35 - 6:00	*Andrew Lumsdaine*	Krylov-Subspace Acceleration of Time Periodic Waveform Relaxation	
6:00 - 6:25	*Yimin Kang*	Convergence Analysis of Combinations of Different Methods	
7:00 - 9:00	**Banquet**	Location to be announced	

Friday, April 8

Multigrid and Multilevel Methods I		Chair: Steve McCormick	Room A
8:00 - 8:25	*Jian Shen*	Implementations of the Optimal Multigrid Algorithm for the Cell-Centered Finite Difference on Equilateral Triangular Grids	
8:25 - 8:50	*Craig C. Douglas*	Constructive Interference II: Semi-Chaotic Multigrid Methods	
8:50 - 9:15	*Jan Janssen*	Multigrid Waveform Relaxation on Spatial Finite Element Meshes	
9:15 - 9:40	*Stefan Vandewalle*	Time-Parallel Iterative Methods for Parabolic PDEs: Multigrid Waveform Relaxation and Time-Parallel Multigrid	

Applications I

Chair: Jim Morel — Room B

8:00 - 8:25	*S.F. Ashby*	Modeling Groundwater Flow on Massively Parallel Computers
8:25 - 8:50	*M.J. Hagger*	Two Grid Iteration With a Conjugate Gradient Fine Grid Smoother Applied to a Groundwater Flow Model
8:50 - 9:15	*Jussi Rahola*	Solution of Dense Systems of Linear Equations in Electromagnetic Scattering Calculations
9:15 - 9:40	*Tom Cwik*	An Iterative Parallel Sparse Matrix Equation Solver with Application to Finite Element Modeling of Electromagnetic Scattering

9:40 - 10:15 Coffee Break

Multigrid and Multilevel Methods II

Chair: Steve McCormick — Room A

10:15 - 10:40	*Michael Griebel*	On the Relation Between Traditional Iterative Methods and Modern Multilevel/Domain Decomposition Methods
10:40 - 11:05	*Irad Yavneh*	Multigrid with Red Black SOR Revisited
11:05 - 11:30	*Michael Jung*	Implicit Extrapolation Methods for Multilevel Finite Element Computations
11:30 - 11:55	*Steve McCormick*	Multilevel First-Order System Least Squares for PDE'S

Applications II

Chair: Jim Morel — Room B

10:15 - 10:40	*Ray S. Tuminaro*	A Multigrid Preconditioner for the Semiconductor Equations
10:40 - 11:05	*R. Bauer*	Preconditioned CG-Solvers and Finite Element Grids
11:05 - 11:30	*Karen R. Baker*	Modeling the Diffusion of Phosphorus in Silicon in 3-D

12:00 - 4:30 Informal Discussion

Multigrid and Multilevel Methods III — Chair: Joel Dendy — Room A

4:45 - 5:10	*J.E. Dendy, Jr.*	Grandchild of the Frequency Decomposition Multigrid Methods
5:10 - 5:35	*Van Henson*	On Multigrid Methods for Image Reconstruction from Projections
5:35 - 6:00	*John Ruge*	A Nonlinear Multigrid Solver for a Semi-Lagrangian Potential Vorticity-Based Barotropic Model on the Sphere
6:00 - 6:25	*C Liu*	Implicit Multigrid Method for Numerical Simulation of the Whole Process of Flow Transition in 3-D Boundary Layers

Applications III — Chair: Howard Elman — Room B

4:45 - 5:10	*Thomas Hagstrom*	Experimental and Theoretical Studies of Iterative Methods for Nonlinear, Nonsymmetric Systems Arising in Combustion
5:10 - 5:35	*Colin Aro*	Preconditioned Time-Difference Methods for Advection-Diffusion-Reaction Equations
5:35 - 6:00	*D.Rh.Gwynllyw*	Preconditioned Iterative Methods for Unsteady Non- Newtonian Flow Between Eccentrically Rotating Cylinders
6:00 - 6:25	*David Silvester*	Fast Non-Symmetric Iterations and Efficient Preconditioning for Navier-Stokes Equations

Workshop: Robust Iterative Methods (Evening: 8:00p - 10:00p) — Chair: Youcef Saad — Room A

Youcef Saad	Iterative solvers in industrial applications: are we kidding ourselves?
Mike Heroux	Some current challenges for industrial CFD applications
Wei Pai Tang	Multi-stage ILU preconditioners for semiconductor device simulation
Larry Wigton	Experiences with Matrix-Iterative Solvers at Boeing
Alex Yeremin	Numerical experiences with advanced iterative solvers for industrial applications

Saturday, April 9

Preconditioners I — Chair: Steve Ashby — Room A

8:00 - 8:25	*Edmond Chow*	Approximate Inverse Preconditioners for General Sparse Matrices
8:25 - 8:50	*Xiaoge Wang*	CIMGS: An Incomplete Orthogonal Factorization Preconditioner
8:50 - 9:15	*L. Kolotilina*	Incomplete Block SSOR Preconditionings for High Order Discretizations
9:15 - 9:40	*Fernando Alvarado*	Block-Bordered Diagonalization and Parallel Iterative Solvers

Applications IV — Chair: — Room B

8:00 - 8:25	*S.W. Bova*	Iterative Methods for Stationary Convection-Dominated Transport Problems
8:25 - 8:50	*A.J. Meir*	Velocity-Vorticity Formulation of Three-Dimensional, Steady, Viscous, Incompressible Flows
8:50 - 9:15	*Louis Howell*	A Multilevel Approximate Projection for Incompressible Flow Calculations
9:15 - 9:40	*N.A. Hookey*	Simulation of Viscous Flows Using a Multigrid-Control Volume Finite Element Method

9:40 - 10:15 Coffee Break

Preconditioners II — Chair: Steve Ashby — Room A

10:15 - 10:40	*P. Amodio*	Parallel Preconditioning for the Solution of Nonsymmetric Banded Linear Systems
10:40 - 11:05	*J.E. Pasciak*	Preconditioning the Pressure Operator for the Time Dependent Stokes Problem
11:05 - 11:30	*S. Holmgren*	A Framework for the construction of preconditioners for systems of PDE

Applications V — Chair: — Room B

10:15 - 10:40	*Maryse Page*	Iterative Solvers for Navier-Stokes Equations - Experiments with Turbulence Model
10:40 - 11:05	*Eli Tziperman*	Multilevel Turbulence Simulations
11:05 - 11:30	*M. Kamon*	Preconditioning Techniques for Constrained Vector Potential Integral Equations, with Application to 3-D Magnetoquasistatic Analysis of Electron Packages

12:00 - 4:30 Informal Discussion

Toeplitz and Circulant Matrix Solvers — Chair: — Room A

4:45 - 5:10	*Thomas Huckle*	Iterative Methods for Toeplitz-Like Matrices
5:10 - 5:35	*Paul Saylor*	A Modified Direct Preconditioner for Indefinite Symmetric Toeplitz Systems
5:35 - 6:00	*Eugene E. Tyrtyshnikov*	Circulant Preconditioners with Unbounded Inverses: Why Non-Optimal Preconditioners may Possess a Better Quality than Optimal Ones
6:00 - 6:25	*Seymour Parter*	A Remark on Band-Toeplitz Preconditions for Hermitian Toeplitz Systems

Saddle Point Problems — Chair: — Room B

4:45 - 5:10	*Andy Wathen*	An Optimal Iterative Solver for the Stokes Problem
5:10 - 5:35	*Bruno Welfert*	On the Convergence of Inexact Uzawa Algorithms
5:35 - 6:00	*Xiezhang Li*	The Asymptotic Convergence Factor for a Polygon Under a Perturbation

6:25 Conference Adjourns

Evening Workshops

8:00pm–10:00pm

Tuesday:	Iterative Software Kernels Organizer: Iain Duff
Wednesday:	Recent Progress and Advances in Iterative Software (including Parallel Aspects) Organizer: Graham Carey
Friday:	Robust Iterative Solvers Organizer: Youcef Saad

Audience Participation is Encouraged!

Thursday, April 7

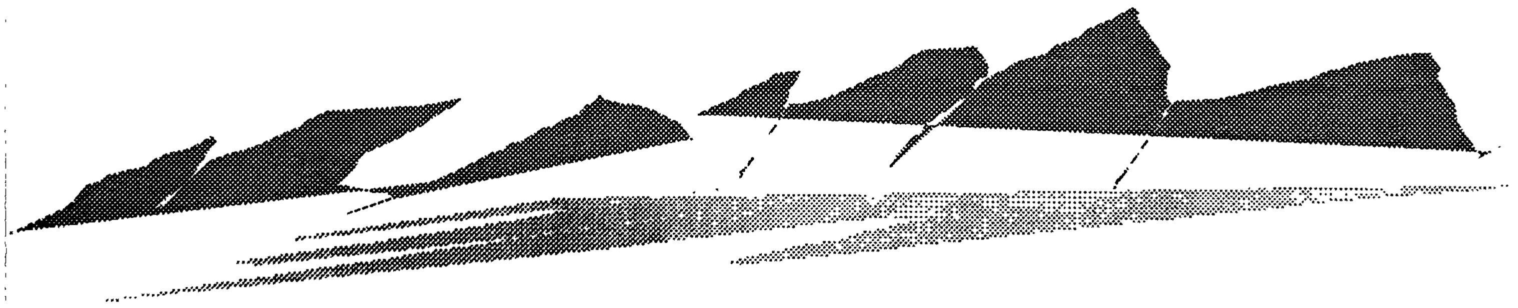
Nonsymmetric Solvers III
Chair: Anne Greenbaum
Room A

8:00 - 8:25 Diederik Fokkema
Generalized Conjugate Gradient Squared

8:25 - 8:50 Roland Freund
Block Quasi-Minimal Residual Iterations for Non-Hermitian Linear Systems

8:50 - 9:15 Noel M. Nachtigal
A Look-Ahead Variant of TFQMR

9:15 - 9:40 Tedd Szeto
Composite-Step Product Methods for Solving Nonsymmetric Linear Systems



Generalized Conjugate Gradient Squared

Diederik R. Fokkema* and Gerard L.G. Sleijpen

January 14, 1994

Abstract

In order to solve non-symmetric linear systems of equations, the Conjugate Gradient Squared (CGS) is a well-known and widely used iterative method. In practice the method converges fast, often twice as fast as the Bi-Conjugate Gradient method. This is what you may expect, since CGS uses the square of the BiCG polynomial. However, CGS may suffer from its erratic convergence behavior. The method may diverge or the approximate solution may be inaccurate. BiCGSTAB uses the BiCG polynomial and a product of linear factors in an attempt to smoothen the convergence. In many cases, this has proven to be very effective. Unfortunately, the convergence of BiCGSTAB may stall when a linear factor (nearly) degenerates. BiCGstab(ℓ) is designed to overcome this degeneration of linear factors. It generalizes BiCGSTAB and uses both the BiCG polynomial and a product of higher order factors. Still, CGS may converge faster than BiCGSTAB or BiCGstab(ℓ). So instead of using a product of linear or higher order factors, it may be worthwhile to look for other polynomials. Since the BiCG polynomial is based on a three term recursion, a natural choice would be a polynomial based on another three term recursion. Possibly, a suitable choice of recursion coefficients would result in method that converges faster or as fast as CGS, but less erratic. It turns out that an algorithm for such a method can easily be formulated. One particular choice for the recursion coefficients leads to CGS. Therefore one could call this algorithm generalized CGS. Another choice for the recursion coefficients leads to BiCGSTAB. It is therefore possible to mix linear factors and some polynomial based on a three term recursion. This way we may get the best of both worlds. We will report on our findings.

*Mathematical Institute, Utrecht University, P.O. Box 80.010, NL-3508 TA Utrecht, the Netherlands. E-mail: fokkema@math.ruu.nl

Block Quasi-Minimal Residual Iterations for Non-Hermitian Linear Systems

Roland W. Freund

AT&T Bell Laboratories
Room 2C–420
600 Mountain Avenue
Murray Hill, NJ 07974–0636

Abstract. Many applications require the solution of multiple linear systems that have the same coefficient matrix, but differ only in their right-hand sides. Instead of applying an iterative method to each of these systems individually, it is usually more efficient to employ a block version of the method that generates blocks of iterates for all the systems simultaneously. An example of such an iteration is the block conjugate gradient algorithm, which was first studied by Underwood and O'Leary. On parallel architectures, block versions of conjugate gradient-type methods are attractive even for the solution of single linear systems, since they have fewer synchronization points than the standard versions of these algorithms.

In this talk, we present a block version of Freund and Nachtigal's quasi-minimal residual (QMR) method for the iterative solution of non-Hermitian linear systems. We describe two different implementations of the block-QMR method, one based on a block version of the three-term Lanczos algorithm and one based on coupled two-term block recurrences. In both cases, the underlying block-Lanczos process still allows arbitrary normalizations of the vectors within each block, and we discuss different normalization strategies. To maintain linear independence within each block, it is usually necessary to reduce the block size in the course of the iteration, and we describe a deflation technique for performing this reduction. We also present some convergence results, and we report results of numerical experiments with the block-QMR method. Finally, we discuss possible block versions of transpose-free Lanczos-based iterations such as the TFQMR method.

A Look-Ahead Variant of TFQMR

by

Roland W. Freund (AT&T Bell Labs)

and

Noël M. Nachtigal (Oak Ridge Natl Lab)

Recently, Freund proposed a Krylov subspace iteration, the transpose-free quasi-minimal residual method (TFQMR), for solving general nonsingular non-Hermitian linear systems. The algorithm relies on a version of the squared Lanczos process to generate the basis vectors for the underlying Krylov subspace. It then constructs iterates defined by a quasi-minimization property, which leads to a smooth and nearly monotone convergence behavior. We investigate a variant of TFQMR that uses look-ahead to avoid some of the problems associated with breakdowns in the underlying squared Lanczos procedure. We also present some numerical examples that illustrate the properties of the new method, as compared to the original TFQMR algorithm.

COMPOSITE-STEP PRODUCT METHODS FOR SOLVING NONSYMMETRIC LINEAR SYSTEMS

TONY F. CHAN * AND TEDD SZETO †

Extended Abstract

The Biconjugate Gradient (BCG) algorithm [5] is the "natural" generalization of the classical Conjugate Gradient method to nonsymmetric linear systems. It is an attractive method because of its simplicity and its good convergence properties. Unfortunately, BCG suffers from two kinds of breakdowns (divisions by 0): one due to the non-existence of the residual polynomial, and the other due to a breakdown in the recurrence relationsip used. There are many look-ahead techniques in existence which are designed to handle these breakdowns. Although the step size needed to overcome an exact breakdown can be computed in principle, these methods can unfortunately be quite complicated for handling near breakdowns since the sizes of the look-ahead steps are variable (indeed, the breakdowns can be *incurable*).

Recently, Bank and Chan introduced the Composite Step Biconjugate Gradient (CSBCG) algorithm [1, 2], an alternative which cures only the first of the two breakdowns mentioned by skipping over steps for which the BCG iterate is not defined. This is done with a simple modification of BCG which needs only a maximum look-ahead step size of 2 to eliminate the (near) breakdown and to smooth the sometimes erratic convergence of BCG. Thus, instead of a more complicated (but less prone to breakdown) version, CSBCG cures only one kind of breakdown, but does so with a minimal modification to the usual implementation of BCG in the hope that its empirically observed stability will be inherited.

We note, then, that the Composite Step idea can be incorporated anywhere the BCG polynomial is used; in particular, in product methods such as CGS [6], Bi-CGSTAB [7], and TFQMR [4]. Doing this not only cures the breakdown mentioned above, but also takes on the advantages of these product methods, namely, no multiplications by the transpose matrix and a faster convergence rate than BCG. For example, if we take the resulting CSBCG polynomials and square them, we obtain the Composite Step CGS method [3]. Similarly, we can apply it to the Bi-CGSTAB algorithm by computing products of the CSBCG polynomial with a steepest descent polynomial to obtain a more stable basis for the Krylov subspace; we call the new method CS-CGSTAB.

In the implementation of these methods, we use a strategy for deciding whether to skip a step that does not involve any machine dependent parameters. Furthermore, this strategy is designed to skip near breakdowns as well as produce smoother iterates.

In [1], Bank and Chan also prove a "best approximation" result which establishes a bound on the error of CSBCG. Since the product methods involve the same BCG polynomial, we can extend this CSBCG result to prove convergence results for CSCGS and CS-CGSTAB.

* Dept. of Mathematics, Univ. of Calif. at Los Angeles, Los Angeles, CA 90024. E-mail: chan@math.ucla.edu.

† Dept. of Mathematics, Univ. of Calif. at Los Angeles, Los Angeles, CA 90024. E-mail: szeto@math.ucla.edu.

Numerical experiments compare the convergence behavior between these composite step methods and also show that these methods do produce improved performance over their non-look ahead couterparts on practical problems.

REFERENCES

[1] R. E. BANK AND T. F. CHAN, *An analysis of the composite step bi-conjugate gradient algorithm for solving nonsymmetric systems*, UCLA CAM Tech. Report 92-53(1992). Numer. Math., to appear.

[2] R. E. BANK AND T. F. CHAN, *A composite step bi-conjugate gradient algorithm for solving nonsymmetric systems*, UCLA CAM Tech. Report 93-21 (1993), Numerical Alg., to appear.

[3] T. F. CHAN AND T. SZETO, *A composite step conjugate gradients squared algorithm for solving nonsymmetric linear systems*, UCLA CAM Tech. Report 93-27 (1993). Numerical Alg., to appear.

[4] R. W. FREUND, *A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems*, SIAM J. Sci. Stat. Comput., 14(1993), pp. 470-482.

[5] C. LANCZOS, *Solution of linear equations by minimized iterations*, J. Res. Natl. Bur. Stand. 49 (1952), pp. 33-53.

[6] P. SONNEVELD, *CGS, a fast Lanczos-type solver for nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 10(Jan 1989), pp. 36-52.

[7] 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. Stat. Comput., 13(March 1992), pp. 631-644.

Thursday, April 7

Parallel Computation III
Chair: Dan Quinlan
Room B

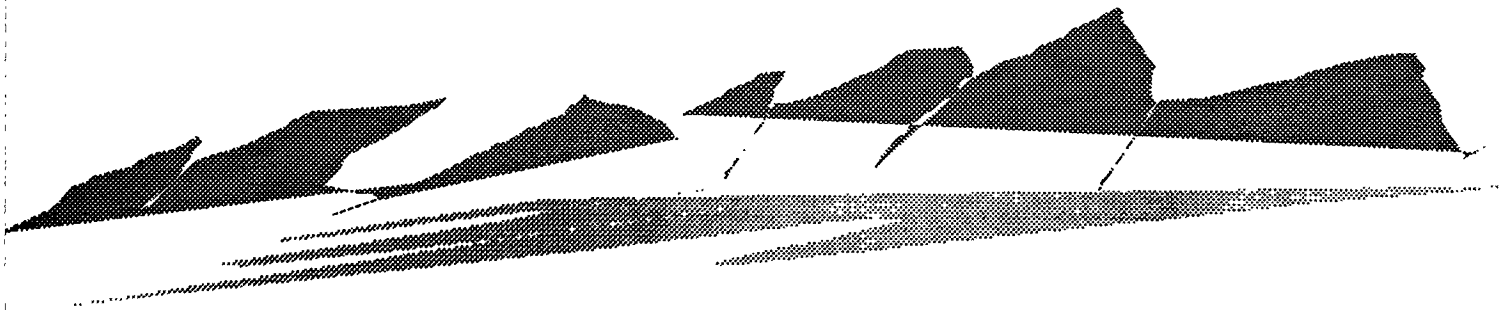
8:00 - 8:25 Lei Li
A Divide-and-Inner Product Parallel Algorithm for Polynomial Evaluation

8:25 - 8:50 Dan Hu
Parallelizing Sylvester-Like Operations on a Distributed Memory Computer

8:50 - 9:15 H.S. Kohli
Maximizing Sparse Matrix Vector Product Performance in MIMD Computers

9:15 - 9:40 John Shadid
Parallel Performance of a Preconditioned CG Solver for Unstructured Finite Element Applications

9:40 - 10:15 Coffee Break



A Divide-and-Inner Product Parallel Algorithm for Polynomial Evaluation

Jie HU* Lei LI* Tadao NAKAMURA**

* *Department of Information System Engineering, Aomori University, AOMORI, 030 JAPAN*

* * *Graduate School of Information Science, Tohoku University, Sendai, 980 JAPAN*

ABSTRACT --- In this paper, a divide-and-inner product parallel algorithm for evaluating a polynomial of degree N $(N+1=KL)$ on a MIMD computer is presented. It needs

$$2K + \log_2 L$$

steps to evaluate a polynomial of degree N in parallel on $L+1$ processors $(L \leqq 2K-2\log_2 K)$ which is a decrease of $\log_2 L$ steps as compared with the L-order Horner's method [1], and which is a decrease of $(2\log_2 L)^{1/2}$ steps as compared with the some MIMD algorithms [3,4]. The new algorithm is simple in structure and easy to be realized.

1. INSTRUCTION

There are many issues related to the parallel algorithm for polynomial evaluation of degree N. Estrin's algorithm [2] needs $T=2(\log_2 N+1)$ steps when using $p=[N/2]$ (here [x] is an integer satsfying $x \leqq [x] < x+1$) processors. L-order Horner's method needs $T= 2K+2\log_2 L$ steps when using $p=L$ processors [1]. For a MIMD processor, the number of steps can be further decreased but the algorithms become extremely complex to realize. For example, Munro and Paterson [3], Maruyama [4] proved that $p=N$, $T \leqq \log_2 N + (2\log_2 N)^{1/2} + O(1)$, and when $p=N^{1/2}$, $T \leqq 2N/p + \log_2 p + (2\log_2 p)^{1/2} + O(1)$. Muraoka [1] applied the method of folding and realized $T \leqq 1.44\log_2 N + O(1)$. Even when there are infinite number of processors, Kosaraju [5] derived a lower bound for $T(N)$ gives by $T(N) \geqq \log_2 N + (2\log 2N)^{1/2} - (\log_2 N)^{1/4} - C$, where constant $C>0$.

In this paper, we present a simple divide-and-inner product parallel algorithm for evaluating a polynomial of degree N $(N+1=KL)$ on a MIMD computer. This algorithm needs $T \leqq 2K+\log_2 L$ steps when using $p=L+1$ processors. This algorithm is simple, and yet improves [3] and [4] by $(2\log_2 L)^{1/2}$ steps.

2. ALGORITHM

Let f(x) be a polynomial of degree N

$$f(x) = \sum_{i=0}^{N} a_i x^i \qquad (1)$$

and $N+1=KL$. Suppose the number of processors $p=L+1$. First divide the $N+1$ items of $f(x)$ into L groups, i.e.,

$$f(x) = \sum_{i=0}^{L-1} b_i\, x^{iK},$$

$$b_i = a_{iK} + a_{iK+1}x + \dots + a_{iK+K-1}x^{K-1}, \qquad for \quad i = 0, 1, \dots, L-1. \qquad (2)$$

a) $b_0, b_1, \dots, b_{L-1}$ can be computed in $2(K-1)$ steps in parallel using L processors by vector Horner's method.

b) Serially compute $x^K, x^{2K}, \dots, x^{(L-1)K}$.

Let $2^r \leqq K < 2^{r+1}$ then K can be expressed in binary form:

$$K = d_0 + d_1 * 2 + d_2 * 2^2 + \dots + d_r * 2^r,$$

where $d_i \in (0,1)$, $i = 0, 1, \dots, r-1$, $d_r = 1$. Then

$$x^K = x^{d_0}(x^{d_1}(x^{d_2}\dots(x^{d_{r-1}}(x^{d_r})^2)^2\dots)^2)^2, \qquad (3)$$

thus, x^K can be computed serially in 2r ($\leqq 2[\log_2 K]$) steps using one processor. Again using L-2 multiplications, we can compute

$$x^{iK}, \quad for \quad i = 2, 3, \dots, L-1. \qquad (4)$$

c) Compute inner product

$$b_0 + b_1x^K + b_2x^{2K} + \dots + b_{L-1}x^{(L-1)K}. \qquad (5)$$

It can be evaluated in parallel in $1 + [\log_2 L]$ steps at most using L processors.

3. THE ANALYSIS OF TIME COMPLEXITY

Now let us analyze the number of steps of divide-and-inner product. Note that (2), (3) and (4) can be computed in parallel. Thus the whole process can be evaluated in parallel in

$$T \leqq \max(2K-2,\ 2\log_2 K + L - 2) + 2 + \log_2 L$$
$$= \begin{cases} 2K + \log_2 L, & (L \leqq 2K - 2\log_2 K) \\ L + \log2K + \log_2(N+1), & (L > 2K - 2\log_2 K) \end{cases}$$

steps. Let $Y_1 = f(L) = 2K + \log_2 L = 2(N+1)/L + \log2L$ $(L \leqq 2K - 2\log2K)$, then

$$Y_1' = -2(N+1)/L^2 + 1/\ln 2 * 1/L$$
$$= 1/L\,(\log_2 e - 2(N+1)/L) < 0.$$

Thus function Y_1 is monotone decreasing. Similarly, let

$$Y_2 = g(L) = L + \log_2 K + \log_2(N+1)$$
$$= L - \log_2 L + 2\log_2(N+1), \qquad (L > 2K - 2\log_2 K)$$
$$Y_2' = 1 - \log_2 e * 1/L > 0$$

and so Y_2 is monotone increasing. Therefore the number of steps takes the minimum $T = 2K + \log_2 L$ on $L = 2K - 2\log_2 K$.

THEOREM 1 A polynomial of degree N $(N+1 = KL)$ can be evaluated in parallel in $T = 2K + \log_2 L$ steps at most when using $L+1$ processors, where $L \leqq 2K - \log_2 K$. And when $L = 2K - 2\log_2 K$, T takes the minimum.

Let $L = 2K - 2\log_2 K$, then $2K^2 - 2K\log_2 K = N+1$. The relationship of the degree N of polynomial, the number $p = L+1$ of processors and the least steps T_{min} is given in Table 1. And using the interpolation method, we can get

$$K \doteqdot 1.218\, N^{0.43},$$

and thus the number of the processors is

$$L = (N+1)/K \doteqdot 0.821\, N^{0.57},$$

and the least number of steps is

$$T_{min} = 2K + \log_2 L \doteqdot 2.436N^{0.43} + 0.57\log_2 N,$$

and the speed up is

$$S_p = T_1/T_{min} = 2N/(2.436N^{0.43} + 0.57\log_2 N) \doteqdot 0.821N^{0.57} \qquad (N \rightarrow \infty),$$

and parallel efficiency

$$\varepsilon_p = S_p/P = 2.436N^{0.43}/(2.436N^{0.43} + 0.57\log_2 N) \doteqdot 1 \qquad (N \rightarrow \infty).$$

4. CONCLUSION

The divide-inner-product parallel algorithm for evaluating a polynomial of degree N $(N+1 = KL)$ is presented in this paper. This method needs $2K + \log_2 L$ parallel steps on $L+1$ $(L \leqq 2K - 2\log_2 K)$ processors and have decreased by $\log_2 L$ steps and $(2\log_2 L)^{1/2}$ steps as compared with the L-order Horner's method [1] and some MIMD methods [3,4] respectively. The new method is also simple on structure and easy to be realized.

Table 1

N	p	Tmin	N	p	Tmin
11	5	8	31	7	12
12	5	8	32	7	12
13	5	8	33	7	12
14	6	8	34	8	12
15	5	10	35	8	12
16	5	10	36	8	12
17	5	10	37	8	12
18	5	10	38	8	12
19	6	10	39	9	13
20	6	10	40	7	14
21	6	10	41	8	14
22	6	10	42	8	14
23	7	10	43	8	14
24	7	10	44	8	14
25	7	10	45	8	14
26	6	12	46	8	14
27	6	12	47	9	15
28	6	12	48	9	15
29	7	12	49	9	15
30	7	12	50	9	15

REFERENCE

[1] S.Lakshmivarahan, S.K.Dhall , Analysis and design of parallel Algorithms, McGraw-Hill Publishing Company, 1990, pp. 254-273.

[2] W.S.Dorn, Generalizations of Horner's Rule for Polynomial Evaluation, IBM J. Res. Develop., Vol. 6, 1962, pp.239-245.

[3] I. Munro, M.Paterson, Optimal Algorithms for Parallel Polynomial Evaluation, J. Comput. System Sci., 1973, pp. 189-198.

[4] K.Maruyama, On the Parallel Evaluation of Polynomials, IEEE Trans. Comp., Vol. C-22, 1973, pp. 2 - 5.

[5] Kosaraju, S. R., Parallel Evaluation of Division-free Arithmetic Expressions, Proceedings of the Symposium on Theory of Computing, 1986, pp. 231 - 239.

Parallelizing Sylvester-like Operations on a Distributed Memory Computer

Dan Y. Hu, Danny C. Sorensen
Department of Computational and Applied Mathematics
Rice University, P.O. Box 1892, Houston, TX 77251

Extended Abstract

1 Introduction

Discretization of linear operators arising in applied mathematics often leads to matrices with the following structure:

$$\hat{M}(x) = (D \otimes A + B \otimes I_n + \hat{V})x, \qquad (1)$$

where $x \in R^{mn}, B, D \in R^{n\times n}$, $A \in R^{m\times m}$ and $\hat{V} \in R^{mn\times mn}$; both D and $\hat{V}$ are diagonal. For the notational convenience, we assume that both A and B are symmetric. All the results through this paper can be easily extended to the cases with general A and B.

The linear operator on R^{mn} defined by (1) can be viewed as a generalization of the Sylvester operator: $\hat{S}(x) = (I_m \otimes A + B \otimes I_n)x$. We therefore refer it as a Sylvester-like operator. The schemes discussed in this paper therefore also apply to Sylvester operator.

Similar to the matrix form of the Sylvester operator $S(X) = AX + XB$, we have the matrix form equivalent to (1) as the follows:

$$M(X) = AXD + XB + V \odot X \qquad (2)$$

where $X \in R^{m\times n}$, $V \in R^{m\times n}$ and V is defined as: for $i = 1, \dots, n$, let V_{diag} be the mn vector consisting of the diagonal elements of $\hat{V}$, then $V(\ :\ ,\ i) = V_{diag}((i-1)m + 1 : nm, (i-1)m + 1 : nm)$. The operation denoted by '$\odot$' here is the element-by-element multiplication. More precisely, $V \odot X \in R^{m\times n}$, and $(V \odot X)(i,j) = V(i,j)X(i,j)$. The equivalence between the matrix form (2) and the 'tensor-product' form (1) is based on the following $1-1$ correspondence from $R^{m\times n}$ to R^{mn}:$x = [X(:,1)^t, X(:,2)^t, \dots, X(:,n)^t]^t$.

One example of the Sylvester-like operator is from the reactive scattering model in quantum chemistry (see [1], and [6]). In the study of the accurate state-to-state reaction cross sections for three-atom systems with modest number of energetically open quantum states, the following 2-D Schrödinger equation is arises.

$$H(\theta, \chi; \rho_\xi)\Phi_t(\theta, \chi; \rho_\xi) = \mathcal{E}_t(\rho_\xi)\Phi_t(\theta, \chi; \rho_\xi),$$

where ρ_ξ is a fixed center of a sphere in the space, and θ, χ are the polar and azimuthal angles, respectively. With the discrete variable representation, the Hamiltonian can be written in the form

$$\mathbf{H} = \mathbf{h}_\theta \otimes \mathbf{I}_\chi + \mathbf{f}_\theta \otimes \mathbf{h}_\chi + \hat{\mathbf{V}},$$

which has the same form as (1).

For many iterative algorithms of solving the linear system of equation $\hat{M}x = b$, or of solving the eigenvalue problem $\hat{M}x = \lambda x$, the linear operation $\hat{M}x$ are performed repeatedly. Therefore in the implementation of an iterative algorithm on a certain computer system, the efficiency of computing the operation $y = \hat{M}x$ will effect the efficiency of the whole algorithm.

In this paper, we present the SIMD scheme for parallelization of the Sylvester-like operator on a distributed memory computer. This scheme is designed to approach the best possible efficiency by avoiding unnecessary communication among processors. Throughout this paper we use form (2) and form (1) alternately in our presentation and analysis. Hereafter, the integers m and n are reserved for the sizes of the Sylvester-like operator in either (2) or (1).

2 The Standard Systolic Model on a Ring Structure

The matrix $\hat{M}$ in (1) has a very special structure. In general, we can write $\hat{M}$ as $\hat{M} = M_{diag} + M_{off}$, where M_{diag} consists of n nonzero diagonal blocks denoted by M_i's and M_{off} consists of nonzero off diagonal blocks written as $\alpha_{ij} I_m$.

For a distributed memory system with n processors connected in a ring network, the matrix-vector product $y := \hat{M}x$ can be naturally carried out by the standard Systolic Procedure (see [3]).

We denote the k-th processor in the ring by P_{k-1} For a certain processor, *left* and *right* refer to the indices of its left and right neighbors, respectively.

As usual, we distribute the vector x to the local array $x_{loc}(1 : m)$ of each processor. For P_k, $x_{loc} = x(km+1 : (k+1)m)$. Similarly, the result vector y will be distributed to the local array $y_{loc}(1 : m)$ with $y_{loc} = y(km+1 : (k+1)m)$. M_{k+1} and $\alpha_{k+1,i}(i = 1, \ldots, n; i \neq k+1)$ are also saved in $M_{loc}(1 : m, 1 : m)$ and $a(1 : n)$, respectively, such that $M_{loc} = M_{k+1}$, and $a(i) = \alpha_{k+1,i}(i = 1, \ldots, n; i \neq k+1)$.

The procedure is described as the follows:

Algorithm 1 **(for P_k)**

```
    $y_{loc} = M_{loc} x_{loc}$
    $ind = k$
    for $i = 1 : n-1$
        send $x_{loc}$ to $P_{left}$
        recv $x_{loc}$ from $P_{right}$
        $ind = ind + 1$
        if $ind = n, ind = 0$
        $y_{loc} = y_{loc} + a(ind+1) x_{loc}$
    end
```

To complete the procedure, each processor needs to send and receive $(n-1)m$ floating point numbers and perform $2m(m+n-1)$ floating point operations(flops). The total memory required is $m^2 + 3m + n$ including an m buffer for message passing.

Algorithm 1 is easy to implement and perfectly load balanced. However, it does have some disadvantages. First, the number of processors has to agree with n. It many be difficult or impossible to arrange this scheme on may applications. Secondly, the communication load on each processor is heavy. In fact, upon completion of the procedure, each processor has contacted information from x_{loc}'s of all other processors.

3 A New Systolic Model on a 2-D Mesh

Our new scheme is designed to reduce communication among processors by taking advantages of both special structure of the operator and the 2-D processor network structure of the system.

The cause of heavy communication involved in **Algorithm 1** can be traced to the fact that in operation $y = \hat{M}x$, all components of the vector x are needed to compute any component of y. In contrast with $y = \hat{M}x$, the equivalent matrix operation $Y = M(X)$ imposes a different dependency of Y on X. Only two 'strips' of X, namely, $X(i_1 : i_2, :)$ and $X(:, j_1 : j_2)$, are needed to obtain a submatrix $Y(i_1 : i_2, j_1 : j_2)$ of the result matrix Y. This special dependency relationship provides a motivation to distribute X and Y on a 2-D mesh processor network with a natural 2-D partition.

In brief, we partition the matrices X according to the 2-D mesh so that an equal-sized block of X can be distributed to every processor. The same rule applys to Y.

Before we discuss the scheme in more detail, we introduce the way of labeling the 2-D mesh of processors. We assume that in a mesh there are p_c columns and p_r rows of processors, and assume that $p = p_c p_r$ is the total number of processors in the mesh. Following the similar way used for the Intel Touchstone DELTA system, (see [4]) we identify each processor with a logical processor number i and the corresponding 2-D coordinates (i_x, i_y). The one in the upper left corner of the mesh has processor number 0 and denoted by P_0. The processor numbers then increase from left to right and top to bottom. The processor in the lower right corner has processor number $p-1$ and is denoted by P_{p-1}.

For a processor located at a certain position in the mesh, $left, right, up$ and $down$ refer to the indices of its left, right, upward and downward neighbors, respectively.

On processor P_k with coordinates (k_x, k_y), chunks of matrices A, B, D and V are distributed in the local arrays $A_{loc}(1 : m_l, 1 : n)$, $B_{loc}(1 : m, 1 : n_l)$, $D_{loc}(1 : n_l)$ and $V_{loc}(1 : m_l, 1 : n_l\ )$ so that $A_{loc} = A(k_y\ m_l + 1 : (k_y + 1)m_l,\ :\ )$, $V_{loc} = V(\ k_y\ m_l + 1 : (k_y + 1)m_l,\ k_x\ n_l + 1 : (k_x + 1)n_l)$, $B_{loc} = B(\ :\ ,\ k_x\ n_l + 1 : (k_x + 1)n_l)$, $D_{loc} = diag\{D\}(k_x\ n_l + 1 : (k_x + 1)n_l)$.

Where we assume that $m = m_l\ p_r$ and that $n = n_l\ p_c$. As described before, matrix X has its chunk distributed in the array $X_{loc}(m_l,\ n_l)$ with the relation $X_{loc} = X(k_y\ m_l+1 : (k_y+1)m_l,\ k_x n_l+1 : (k_x + 1)n_l)$. Similarly, the result matrix Y will have its chunk distributed in the array $Y_{loc}(m_l, n_l)$ with the relation $Y_{loc} = Y(k_y\ m_l + 1 : (k_y + 1)m_l, k_x n_l + 1 : (k_x + 1)n_l)$.

The follows is the description of the procedure.

Algorithm 2 (for P_k)

```
{Step 1. applying the operation $Y = V \odot X$}
    for $i = 1 : m_l$
        for $j = 1 : n_l$
            $Y_{loc}(i,j) = V_{loc}(i,j) X_{loc}(i,j)$
        end
    end
{Step 2. applying the operation $Y = Y + XB$}
    $ind = k_x$
    for $i = 1 : p_c - 1$
        send $X_{loc}$ to $P_{left}$
        recv $X_{loc}$ from $P_{right}$
        $ind = ind + 1$
        if $ind = p_c$, $ind = 0$
        $Y_{loc} = Y_{loc} + X_{loc} B_{loc}(\ ind\ n_l + 1 : (ind + 1)n_l, 1 : m_l)$
    end
{Step 3. recovering the original distribution of $X$}
    send $X_{loc}$ to $P_{left}$
    recv $X_{loc}$ from $P_{right}$
{Step 4. applying the operation $Y = Y + AXD$}
    for $i = 1 : n_l$
        $X_{loc}(:, i) = D_{loc}(i)\ X_{loc}(:, i)$
    end
    $Y_{loc} = Y_{loc} + A_{loc}(1 : n_l, k_y\ m_l + 1 : (k_y + 1)m_l\ ) X_{loc}$
    $ind = k_y$
    for $i = 1 : p_r - 1$
        send $X_{loc}$ to $P_{up}$
        recv $X_{loc}$ from $P_{down}$
        $ind = ind + 1$
        if $ind = p_r$, $ind = 0$
        $Y_{loc} = Y_{loc} + A_{loc}(1 : n_l, ind\ m_l + 1 : (ind + 1)m_l) X_{loc}$
```

	communication load	flops	memory
Algorithm 1	$n^2 - n$	$4n^2 - 4n$	$n^2 + 3n + n$
Algorithm 2	$2n^{3/2} - n$	$4n^2 + 2n^{3/2} + 2n$	$2n^{3/2} + 4n + \sqrt{n}$

Table 1: Comparison: both compute an $n \times n$ operator on n processors

end

Algorithm 2 is distinct from **Algorithm 1** with its 2-direction message passing pattern. In Step 2 and Step 3 each row of processors in the mesh form a horizontal ring. Message passing is restricted within processors in this row. Similarly, in Step 4 each column of processors in the mesh form a vertical ring. Message passing is restricted within processors in this column.

For each processor, in order to complete **Algorithm 2**, it takes p_c 'horizontal' message sending and receiving actions and $p_r - 1$ 'vertical' message sending and receiving actions. In each action the message load is $m_l\ n_l$. Therefore the total message sending and receiving load is $(p_c + p_r - 1)m_l\ n_l$. The total flops performed in one processor is $2m_l^2\ n_l\ p_r + 2m_l\ n_l^2\ p_c + (p_c + p_r + 2)m_l\ n_l$. The total memory required is $m_l n + m n_l + 4m_l n_l + n_l$ including a $m_l n_l$ buffer for message passing.

4 Comparisons

Like **Algorithm 1**, **Algorithm 2** is easy to implement and perfectly load balanced. Further more, it has several advantages over **Algorithm 1**.

First, **Algorithm 1** requires that p, the number of processors used, must agree with n. This would limit its applications on practical problems. For **Algorithm 2**, the requirement for the size of the processor mesh is much looser. In order keep the load balance, p_r and p_c, the number of rows and the number of columns in the mesh, have to be a factor of m and a factor of n, respectively. However, the algorithm can be arranged to deal with a general case where p_r and p_c are arbitrary with imbalanced communication and flops load on processors. Therefore, for many practical cases where **Algorithm 1** cannot be implemented, the comparison between the two algorithm is out of the question.

Secondly, even in a case where both algorithms can be used, they apply different message passing strategies which have different communication costs. In Step 2 and Step 3 of **Algorithm 2**, each row of processors carries out message passing actions independently. Also in Step 4 each column of processors carries out message passing actions independently. This 2-direction message passing pattern makes it more efficient on communications than **Algorithm 1**. In order to compare the flops and communication load of the two algorithms, we have to fix the sizes of the operators, as well as the numbers of processors being used in the ring network for **Algorithm 1** and the mesh network for **Algorithm 2**.

We set $m = n$, so that **Algorithm 1** would be well suited to a ring of n processors. Accordingly, we assume **Algorithm 2** is adapted on an $\sqrt{n}$ by $\sqrt{n}$ mesh of processors with total n processors. In Table 1, we compare the these two algorithms by communication load, flops, and memory required on each processor under the above assumptions.

It is clear from these numbers that while **Algorithm 2** keeps about the same flops load as **Algorithm 1** has (with respect to the leading terms of $flops_1$ and $flops_2$), it reduces both communication load and memory space dramatically.

5 Discussions on Speed-up

By definition, we say that a parallel algorithm for a particular problem achieves speed-up S if $S = T_s/T_p$ where T_p is the time required for execution of the parallel program on p processors and T_s is the time required by one processor when the best uniprocessor procedure is used(see [3]). We

number of processors	flops	communication time
$p_c \times p_c = {p_c}^2$	$4{n_l}^3 p_c + 2(p_c + 1){n_l}^2$	$(2p_c - 1)\alpha + (2p_c - 1){n_l}^2\beta$
$2p_c \times 2p_c = 4{p_c}^2$	${n_l}^3 p_c + (p_c + 1/2){n_l}^2$	$(4p_c - 1)\alpha + (p_c - 1/4){n_l}^2\beta$
$kp_c \times kp_c = k^2{p_c}^2$	$(4/k^2){n_l}^3 p_c + 2(p_c/k + 1/k^2){n_l}^2$	$(2kp_c - 1)\alpha + (2p_c/k - 1/k^2){n_l}^2\beta$

Table 2: Change in flops and communication time caused by changing of processor mesh sizes

will give the speed-up test data of **Algorithm 2** which is computed according to this concept in the next section. However, the statement 'the best uniprocessor procedure' is somewhat confusing in our case. Since our procedure is designed to distribute the original system onto a network of large number of processors, it is likely that a problem of certain sizes can be computed by p processors but cannot be computed by one processor due to insufficient memory on a single processor. In this case, the following ratio should be considered as the 'relative speed-up' $\hat{S} = T_{p_{min}}/T_p$ where p_{min} stands for the least number of processors on which the computation can be carried out. Hereafter, for an integer i, T_i stands for the time required for execution of a certain operation on i processors. More generally, we can define relative speed-up $\hat{S}$ as $\hat{S} = T_{p_0}/T_{\mu p_0}$ where p_0 is given and μ is an integer. In any case, it is important to know, for an operator with given sizes, how run time T_p changes when the processor number p changes. In the following analysis, we measure the change of flops and communication time to 'predict' the change of run time. In next section, we will compare our 'prediction's with the real run time data. An alternative way to measure speed-up would be to use Gustafson's [2] scaled speed-up idea. However, we think the data presented here are informative and they only involve measured quantities.

For simplicity, we assume that our operator has sizes n by n and that in our processor mesh $p_c = p_r$. We first double the sizes of the mesh in both directions, i.e. we enlarge the mesh into $2p_c$ by $2p_c$. Then we enlarge the mesh into kp_c by kp_c. Assuming that for each processor α is the time required to initiate a message and β is the rate that a message can be transferred, the time of sending l floating point numbers once between two processors is $\alpha + l\beta$. Table 2 shows the change in flops and communication time on one processor.

Thus, when the number of processors is raised by k^2 times, the flops load on each processor is reduced by about k^2 times. The reduction of the communication time is more complicated to calculate. In the situation when $\alpha \ll {n_l}^2\beta$ the reduction is about k times. However, we can set k^2 and k as the 'upper bound' and the 'lower bound' for the reductions in run time. In the next section, we will refer k and k^2 as 'ideal computation speed-up' and 'ideal communication speed-up' caused by raising the processor sizes in both directions k times, respectively.

6 Numerical Test Results

In this section, we provide some numerical results to demonstrate the behavior of **Algorithm 2** and to compare **Algorithm 2** with **Algorithm 1**. Both algorithms are implemented on the Intel Touchstone DELTA with number of processors up to 256. All the matrices involved in both matrix form (2) and (1) are formed by random numbers. All programs are run in double precision.

First we compare **Algorithm 1** and **Algorithm 2** by applying **Algorithm 1** and **Algorithm 2** to the equivalent operators (1) and (2) with the same operator sizes on the same processor mesh. We always set $n = m$ and the processor mesh sizes $\sqrt{n}$ by $\sqrt{n}$. Table 3 compares the performances of both algorithms in terms of average time per operation. In the last column of the table, the ratios t_1/t_2 show the relative speed-ups from Algorithm 1 to Algorithm 2 with respect to different operator sizes and processor mesh sizes. The results indicate that under the above assumptions on the operator sizes and the processor mesh sizes, Algorithm 2 reduced run time dramatically, and when the problem sizes and the processor mesh sizes increase, the amount of the reduction increases too.

We then test the speed-up behavior of **Algorithm 2** by fixing the operator sizes to 240×240,

problem sizes	processor number	t_1 (in sec.)	t_2(in sec.)	t_2/t_1
$256 \times 256 = 65536$	$16 \times 16 = 256$	0.3656	0.09688	3.774
$196 \times 196 = 38416$	$14 \times 14 = 196$	0.1879	0.05191	3.619
$144 \times 144 = 20736$	$12 \times 12 = 144$	0.1067	0.03152	3.385
$100 \times 100 = 10000$	$10 \times 10 = 100$	0.05785	0.01875	3.085
$64 \times 64 = 4096$	$8 \times 8 = 64$	0.03054	0.005352	2.313

Table 3: Comparison: run time t_1 of **Algorithm 1** vs run time t_2 of **Algorithm 2**

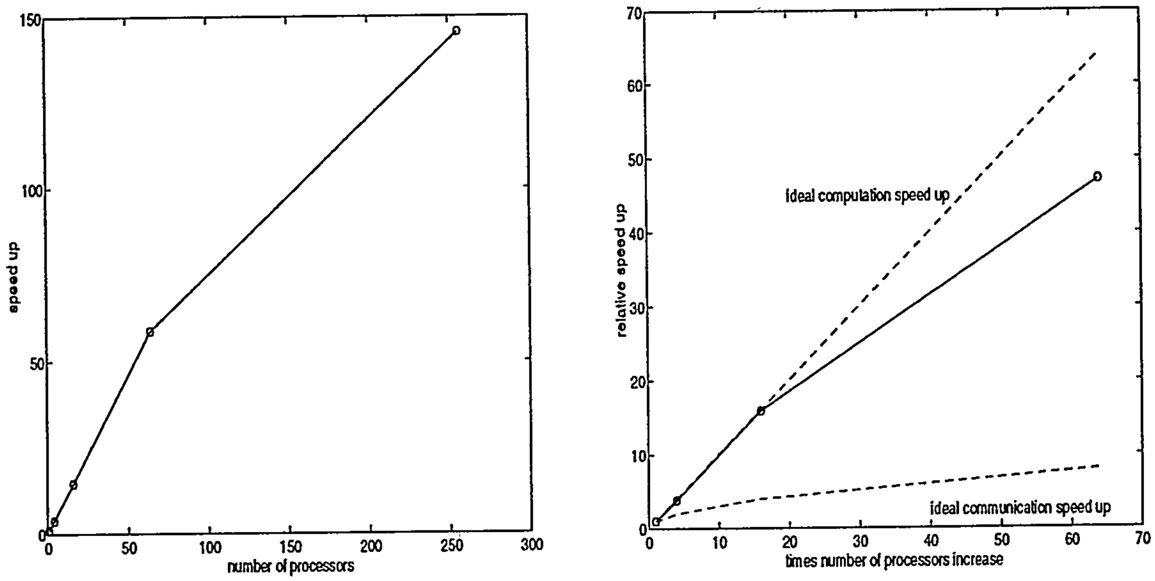


Figure 1: (a)Speed-up S: matrix sizes 240×240, left; (b)Relative speed-up $\hat{S} = T_4/T_{4k}$:matrix sizes 256×256

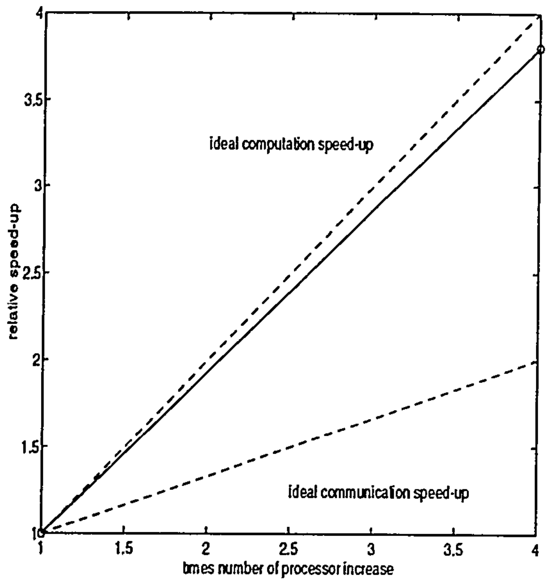
and applying **Algorithm 2** on processor meshes with different sizes. The single processor program is composed with *LAPACK* level 3 *BLAS* routines. Figure 1(a) shows the speed-up in this case.

Following the discussions in Section 4, we arrange tests to observe the 'relative speed-up'. This time we fix the operator sizes to 256×256. We start from using $2 \times 2 = 4$ processors and denote the observed run time by T_4. Then for $k = 2, 4, 8$ we use $2k \times 2k = 4k^2$ processors and denoted the run time by T_k. Figure 1(b) shows the 'relative speed-up' T_4/T_{4k}, and compares the curve with the 'ideal computation speed-up' k^2 and the 'ideal communication speed-up' k. However, the curves indicate that k^2 and k are good upper bound and lower bound for the real time speed-up.

In the following tests, we measure the relative speed-up caused by raising the processor mesh sizes once. In the first test, we take the operator with sizes 1024×1024 and we start from using $8 \times 8 = 64$ processors. Figure 2(a) shows the speed-up caused by raising the processor mesh sizes to 16×16. In the second test, we take a much smaller system of sizes 240×240 and start from $8 \times 8 = 64$ processors. Figure 2 (b)shows the speed-up caused by raising the processor mesh sizes to 16×16.

The different behaviors of **Algorithm 2** shown in Figure 2(a) and Figure 2(b) indicate that the change of system sizes will effect the speed-up behavior, the larger the system sizes are, the closer it looks to 'ideal computation speed-up'.

Algorithm 2 can be generalized to one that would deal with the 3-D array Sylvester-like operation $\hat{M}_{3D}(x) = (I \otimes I \otimes A + I \otimes B \otimes I + C \otimes I \otimes I + \hat{V})x$. With 3-direction message passing scheme, this generalized algorithm is well suited for distributed memory computers with 3-D torus topology such as Cray T3D. For detail, please see [5].



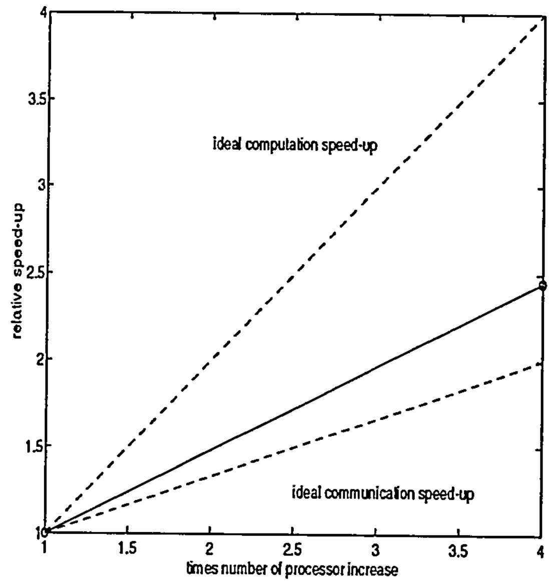


Figure 2: Relative speed-up $\hat{S} = T_{64}/T_{256}$:(a) matrix sizes 1024×1024, left; (b) matrix sizes 240×240

7 Acknowledgements

Financial support for this work was provided in part by the National Science Foundation cooperative agreement CCR-9120008 , by the Department of Energy contract DE-FG0f-91ER25103 and by ARPA through U.S. Army ORA4466.01.

References

[1] B. J. Archer, G. A. Parker and R. T Pack *Positron–Hydrogen Atom S–Wave Coupled–Channel Scattering at Low Energies Physical Review A* 41, no. 3 (1 Feb.1990): 1303–1310.

[2] J. Gustafson *Reevaluating Amdald's law* Comm. ACM, 31: 532-533, 1988.

[3] G.H. Golub and C.F. Van Loan *Matrix Computations*, 2nd ed., Johns Hopkins University Press, Baltimore, MD, 1989.

[4] Intel Co., *Touchstone DELTA System User's Guide*, Order No. 312125, Oct., 1991.

[5] D.Y. Hu and D. C. Sorensen *Parallelizing Sylvester-like Operations on a Distributed Memory Computer* Rice University Tech. Report, TR 94-10.

[6] P. Pendergast, Z. Darakjian, E. F. Hayes and D. C. Sorensen *Scalable Algorithms for Three-Dimensional Reactive Scattering: Evaluation of a New Algorithm for Obtaining Surface Functions*, preprint.

[illegible]

Maximizing Sparse Matrix Vector Product Performance in MIMD Computers

R. T. McLay, H. S. Kohli, S. L. Swift, G. F. Carey

A considerable component of the computational effort involved in conjugate gradient solution of structured sparse matrix systems is expended during the Matrix-Vector Product (MVP), and hence it is the focus of most efforts at improving performance. Such efforts are hindered on MIMD machines due to constraints on memory, cache and speed of memory-cpu data transfer.

This paper describes a strategy for maximizing the performance of the local computations associated with the MVP. The method focuses on single stride memory access, and the efficient use of cache by pre-loading it with data that is re-used while bypassing it for other data. The algorithm is designed to behave optimally for varying grid sizes and number of unknowns per gridpoint.

Results from an assembly language implementation of the strategy on the iPSC/860 show a significant improvement over the performance using FORTRAN.

Parallel Performance of a Preconditioned CG Solver for Unstructured Finite Element Applications[1]

John N. Shadid[2], Scott A. Hutchinson[2] and Harry K. Moffat[3]
Sandia National Laboratories
Albuquerque, New Mexico 87185

Abstract

A parallel unstructured finite element (FE) implementation designed for message passing MIMD machines is described. This implementation employs automated problem partitioning algorithms for load balancing unstructured grids, a distributed sparse matrix representation of the global finite element equations and a parallel conjugate gradient (CG) solver. In this paper a number of issues related to the efficient implementation of parallel unstructured mesh applications are presented. These include the differences between structured and unstructured mesh parallel applications, major communication kernels for unstructured CG solvers, automatic mesh partitioning algorithms, and the influence of mesh partitioning metrics on parallel performance. Initial results are presented for example finite element (FE) heat transfer analysis applications on a 1024 processor nCUBE 2 hypercube. Results indicate over 95% scaled efficiencies are obtained for some large problems despite the required unstructured data communication.

1. Introduction.

In recent years substantial progress has been made in the development of scientific applications on large-scale multiple instruction - multiple data (MIMD) machines. These machines utilize many independent processors, each with local memory, to provide a substantial performance increase over traditional vector supercomputers. Unfortunately, this performance increase is often obtained with considerable cost in programing complexity. In particular, users must now partition computational tasks into subtasks suitable for individual processors, map these tasks to the parallel architecture and then use message passing to achieve synchronization and communication. In general, algorithm development and performance analysis for the parallel solution of PDE systems has been focused on regular meshes [7,10,11]. For this case the static partitioning problem, of dividing the computational task into subtasks, is trivial. Due to the data locality inherent in these finite difference (FD) and finite element (FE) approximation procedures the task partitoning reduces to simple heuristics designed to minimize the perimeter-to-area (or surface -to-volume) ratio for subdomains of regular meshes. Once subdivided these tasks can be easily mapped onto hypercubes and mesh architectures so that neighboring subdomains in the computational domain are mapped to neighboring processors in the parallel machine. Thus, only structured - nearest neighbor communication is needed and contention for the communication channels is avoided.

In contrast, the partitoning and mapping problem for irregular and unstructured meshes can be much more difficult. Indeed, the determination of a partition that actually minimizes communication between balanced sets is know to be an NP-hard problem [3], so it is very unlikely that a general computationally efficient algorithm exists. In addition, the mapping problem for unstructured

1. This work was partially funded by the Applied Mathematical Sciences Program, US Department of Energy, Office of Energy Research, and was performed at Sandia National Laboratories operated for the US Department of Energy under contract no. DE-AC04-76DP00789.

2. Parallel Computational Sciences Department

3. Chemical Processing Sciences Department

meshes is also difficult since a general unstructured mesh cannot be mapped with only nearest neighbor communication on a hypercube or mesh architecture. As a result the required unstructured communication can produce contention for communication channels between processors. However, the central importance of the partitoning and mapping problem for unstructured mesh computations has motivated the use of a wide range of heuristic algorithms [4,6,8,12,13]. These algorithms produce a variety of partitions for which the quality of the partition is roughly correlated with computational cost. It is these essential differences between regular mesh and unstructured mesh parallel computations which make the efficient parallel implementation of unstructured FE codes challenging on distributed memory MIMD machines.

2. Parallel Implementation Overview

In this section the overall parallel implementation framework for a representative parallel unstructured mesh FE application and CG solver is briefly discussed. This FE application is capable of solving for steady and transient conduction with local volumetric heat generation in complex 2D/3D geometries. Considering a typical complex unstructured grid as in Figure 1 the following comments about the typical structure of a parallel FE application can be made. It is evident from Figure 1 that a general automated method for subdividing an unstructured computational mesh and mapping it to the parallel MIMD machine is necessary. An ad-hoc or by-hand method would prove to be unusable for a large number of meshes and the resulting parallel communication efficiency would be difficult to predict, assess and control. In our implementation we have used a general graph and mesh partitioning utility, CHACO[5], developed at Sandia. Using this utility the basic overall parallel MIMD implementation on P processors is as follows. Consistent with the choice of a nodal based FE scheme, it is necessary to load balance the work associated with the matrix setup and solution by partitioning the N nodes among the P processors[2]. The load balancing algorithm partitions the N nodes into P sets of $\lfloor N/P \rfloor$ or $(\lfloor N/P \rfloor + 1)$ FE nodes[3] and then maps these sets to the P processors of the parallel machine such that the overall interprocessor communication is minimized (see Section 4). After the required load balance and mapping the FE application can then set up the distributed FE coefficient matrix. On each processor this distributed sparse matrix corresponds to a rectangular submatrix, $\mathbf{A}_p$, of the global overall coefficient matrix, $\mathbf{A}$. Thus each processor, in parallel, performs the necessary element integrations and sets up a local set of equations for each of the FE nodes which it has been assigned. In this implementation, the equations are fully summed and actually correspond to a complete row entry from the corresponding global coefficient matrix. The union of these distributed rectangular matrices is therefore equivalent to a serial global coefficient matrix.

3. Parallel CG Solver

The parallel CG algorithm is essentially the same as the structured grid Krylov methods discussed in [10,11] with the exception of the unstructured matrix-vector product. The main kernels are the matrix-vector product, DAXPY type operations and vector inner products. As in the structured mesh case, the key to performance in these solvers is typically the efficiency of the matrix-vector multiply kernel and, within this, the communication it requires. For the parallel CG kernel to operate as efficiently as possible, the time for the interprocessor communication during the required matrix-vector multiplies must be minimized. In turn, key to this minimization is the data structures in which both the distributed sparse matrix and the distributed vectors are stored. Logically and as will be described in the next section, each processor is given a set of nodes for which it is responsible. Thus, in the formation of the sparse matrix and vectors, each processor will have a set of rows in both the sparse matrix and any associated vectors, each corresponding to

2. Other choices such as element based schemes are possible. The relative performance of such schemes should not vary greatly for reasonable implementations of either the nodal or elemental schemes.
3. $\lfloor R \rfloor$ is the floor function which returns the largest integer, m, such that $m < R$.

unknowns located at the nodes which have been assigned to this processor by the partitioning algorithm. Formally, let $v_l \in \mathbf{R}^{n_l}$ be the vector of unknowns which have been assigned to processor l where n_l is the number of these unknowns. The edges cut in the connectivity graph represent the data dependencies between subpartitions (processors). Thus, in order to complete the matrix-vector product, processor l will need additional values of **v** which reside on other processors. These values are required to complete the interactions between processor *l's* "border" unknowns and it's "external" unknowns. That is, if $\mathbf{v}_{lb} \subset \mathbf{v}_l$ are the border unknowns of processor l, these unknowns interact with border unknowns on neighboring processors via the connectivity of the FEM mesh. These border unknowns assigned to neighboring processors are referred to as processor *l*'s external unknowns. Figure 2 gives an illustration of this partitioning.

Given a specific partition and the assignment of the unknowns (internal, border and external) described above, a distributed sparse matrix storage scheme[10] is used. On each processor, the node numbers are reordered such that the first n_{in} nodes are the internal nodes of the processor, the next n_b are the border nodes and the last n_e are the external nodes. Therefore, n_l is defined as $n_l = n_{in} + n_b$. Thus, the last n_e members of the vector must be obtained by communicating with the processors which have been assigned these nodes. Once this portion of the vector is filled, the matrix-vector product may be computed. Note that the number of messages is equal to the number of neighboring processors and that the size of each message is directly proportional to the number of edges cut in the connectivity graph by interprocessor boundaries. In addition, if the processor is not a physical neighbor, the message may have to traverse several intermediate communication channels in order to arrive at its destination. Thus two processor based local metrics, the maximum number of messages and the maximum size of these messages along with two global network metrics, the total number of data items transmitted (termed cuts) and the total number of data items transmitted weighted by the distance they travel (termed hops) are useful to describe the effectiveness of partitioning algorithms. These factors are critical in determining the speed of the matrix-vector products. All of these factors are influenced to some degree by the partitioning of the problem. The next section briefly describes the partitioning methods used in the results presented in Section 5. For a more complete description of the partitioning methods available through the utility CHACO used in this study, see [5].

4. Partitioning Algorithms

The central importance of the partitoning problem to unstructured mesh computations has motivated the use of a wide range of heuristic algorithms [4,6,8,10,13]. These algorithms produce a wide spectrum of partitions for which the quality of the partition is roughly correlated with computational cost. The partitioning utility CHACO[5] implements a wide range of methods which can be used to produce load balanced partitions for distributed memory MIMD machines. These methods may be divided into two classifications geometric and graph based. Geometric methods use only information about the relative geometric distribution of vertices (or elements) of a mesh that is being partitioned. Graph based methods require only information on the inter-connectivity of the vertices (or elements) to produce a partition of the graph. Graph and geometric partitioners can further be subdivided into local and global methods to describe the scope of influence. Global methods use information from the entire graph (or geometric domain) to select the partition, whereas local methods consider only a small neighborhood of the vertex under consideration to produce the next candidate partition. In this study the following methods have been used. The *linear* and *scattered* partitioners use a simple assignment of the vertices (FE node in this study) to locally balance and map the vertices. The *linear* partitioner assigns groups of $\lfloor N/P \rfloor$ or $(\lfloor N/P \rfloor + 1)$ FE nodes to each processor in a sequential manner in accord with the numbering of the original graph. The *scattered* method assigns each vertex sequentially to a processor, after P vertices are assigned the method repeats cyclically until all vertices are assigned. The run time of the simple schemes is very small and the effectiveness of the partition depends on the original numbering of the graph. The *Inertial* method is a global geometric method which considers the vertices to be point masses with mass equal to a vertex weight. The principle axis of the distributed point masses is determined and a cut is made perpendicular to this direction to bisect the domain into two sets of equal mass. The *inertial* method is fast and can produce a very reasonable partition for a wide range of geometrically based problems. The global inertial method can be combined with a local graph based heuristic, Kernighan-Lin (KL), a greedy local optimization scheme, to produce a hybrid method. The KL local refinement scheme can significantly improve inertial method partitions, it does however increase the cost of this method substantially. The *multilevel spectral bisection* scheme (denoted by ML) is a global graph based heuristic. The ML method of Hendrickson and Leland[4,5] produces a coarse representation of the original graph, to which spectral partitioning is applied on the coarsest level, this partition is projected to the next finer level on which a refinement by KL is used to optimize the partition locally. This local refinement is done recursively up to the finest level. This algorithm is more expensive but usually produces the best partition for large problems. All of these methods are used as a preprocessing step in the parallel FEM implementation so that the individual cost of any partitioning method will be amortized over the number of times a specific parallel partition of a FE mesh is used.

5. Results

As described above there are a number of issues related to the performance of the parallel unstructured finite element application and in this section a number of these are briefly discussed. Figure 3 shows the fixed-size problem speedup for a small 3D unstructured FE problem with about 6000 FE nodes. This plot illustrates the transition from a computation to a communication bound application as the number of processors is increased. Clearly the matrix setup phase of the application scales well with increasing number of processors. The small deviation from linear speedup is due to a small amount of work that is redundantly done in parallel to obtain elemental Jacobians. The total time to setup and solve the problem is also shown in Figure 3. The point at which communication begins to overwhelm the computational cost is at about 256 processors. A transition point at 128 to 256 processors for a small problem indicates that the application scales reasonable well as the number of processors is increased despite the unstructured communication overhead. The effects of the various partitioning metrics for a small problem is shown in Table 1. Here it is evident that the local processor-based metrics of "maximum neighbors" and "maximum message size" are important in the overall performance of the unstructured solvers. The partitioner that produces the best run time is associated with a minimum in the "maximum neighbors" metric (Inertial). When two partitions tie with this metric (ML and Inertial + KL) the "maximum message size" metric becomes appropriate. Clearly the network based "cuts" and "hops" metrics have very little bearing due to the small size of the problem and the small number of processors.

However, distributed memory machines are most productive solving large problems. Tables 2 and 3 indicate the preliminary performance of the parallel application and solvers on two large test problems. In Table 3 the scaled parallel efficiency of the parallel unstructured solver is indicated. The scaled speedup (and thus efficiency) is estimated by using the single node maximum computational rate (Mflops) and the measured multiprocessor rate on two large FE problems. For these large problems it is evident that the network based metric of "hops" is an appropriate measure of expected performance. In terms of the time for the unstructured communication it is clear that having a good quality partition can reduce the required communication time substantially. Scaled efficiencies of over 95% indicate that the parallel solvers scale well for large problems on a large number of processors. Table 4 indicates performance for a 1,000,000 FE node problem run on 512 and 1024 processors. As indicated in the table the ML partitioner produces the highest performing partition. Again in this case the "hops" metric is a good predictor of the expected performance. The minimum time necessary to setup and solve the 1,000,000 node problem, 197 sec., is impressive. The overall performance can also be measure by the solver Mflop rate of 840 Mflops. This compares well to a 1,000,000 unknown finite difference calculation on the nCUBE2 that obtains about 1.4 Gflops. These preliminary results are encouraging.

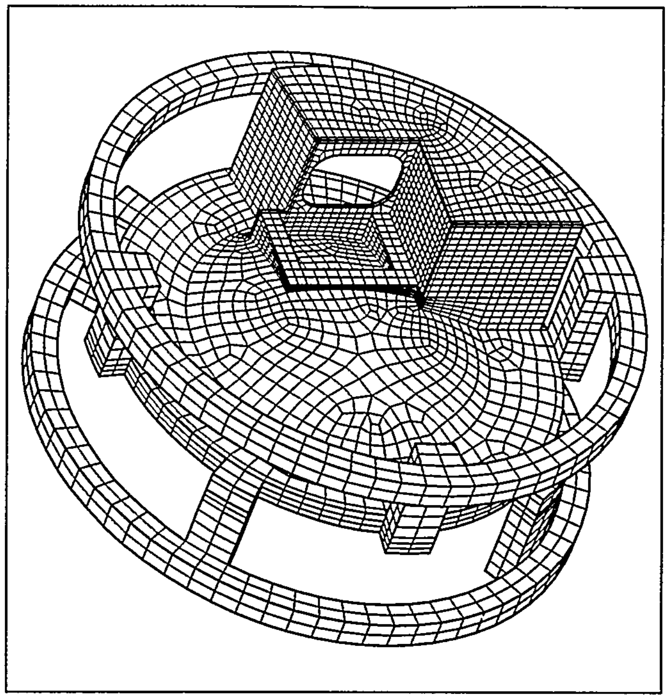


Figure 1: Coarse Investment Casting Mesh

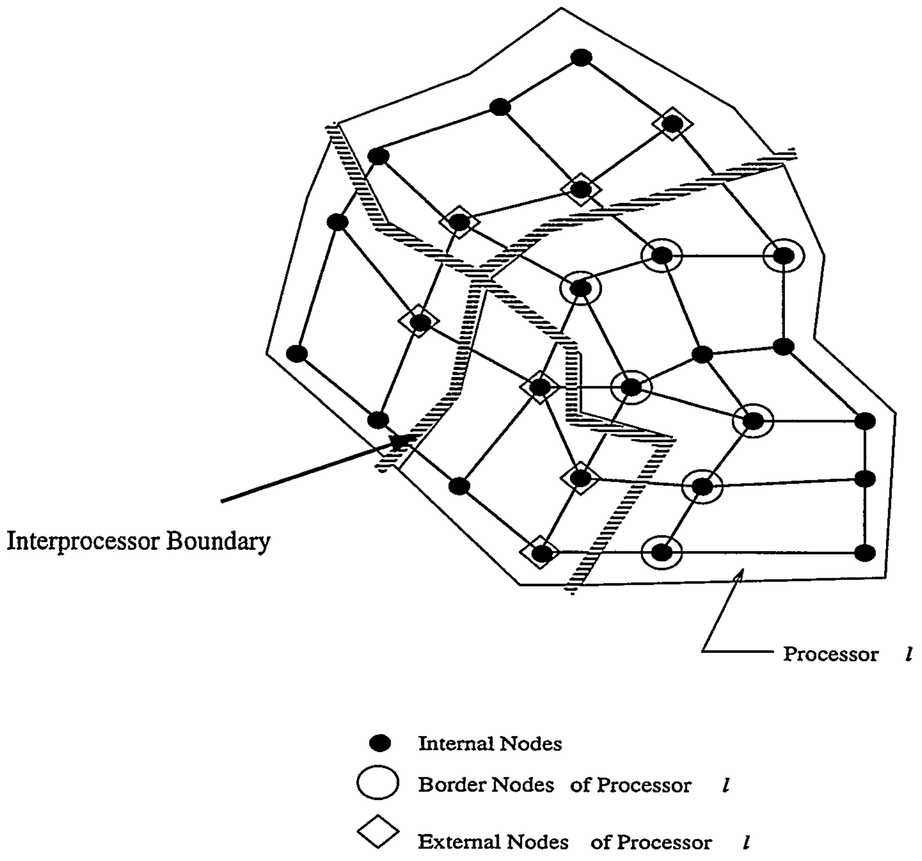

Interprocessor Boundary
Processor l
Internal Nodes
Border Nodes of Processor l
External Nodes of Processor l

Figure 2: Unstructured Mesh Partitioning Illustrating the Internal, Border and External Nodes.

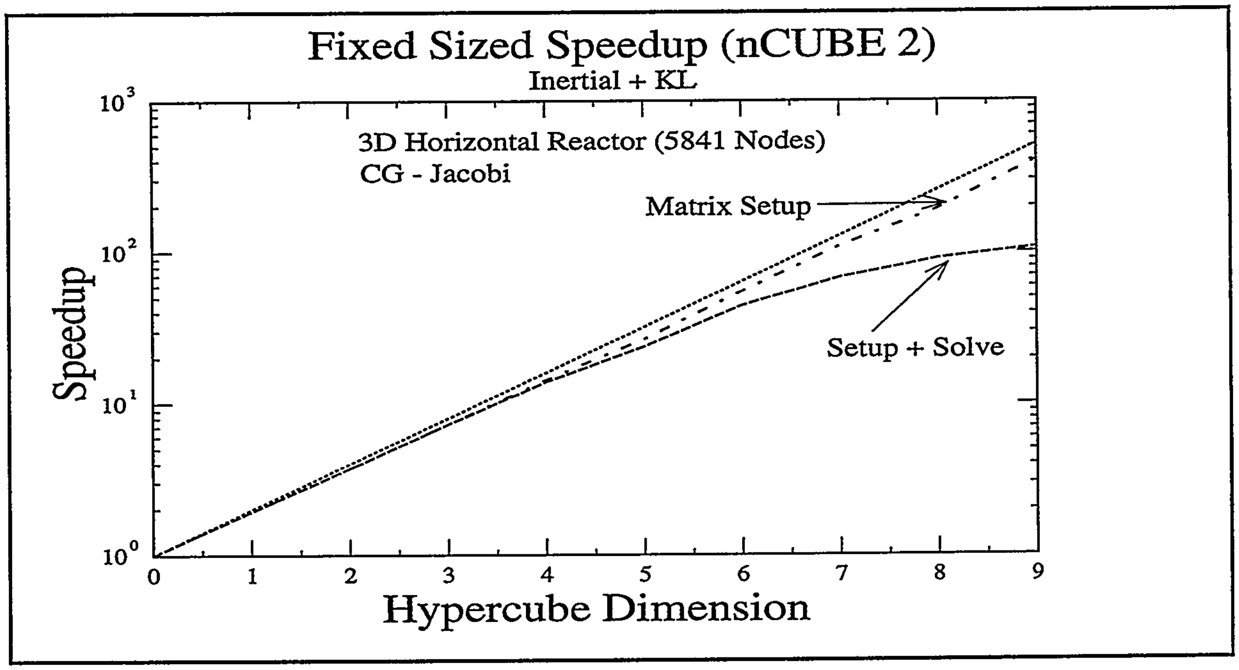


Figure 3: Fixed Sized Speedup for Coarse 3D Horizontal CVD Reactor Mesh

Algorithm	Cuts (10^3)	Hops (10^3)	Max. Neighbors	Max. Mesg. Length (Kbytes)	Partition Time (sec.)	Solve Time (sec.)
Scattered	51	88	7	6.6	0.0	26.9
Linear	7.3	13	7	2.3	0.0	21.2
Inertial	4.7	5.6	5	1.3	0.1	17.9
Inertial +KL	3.0	4.2	6	.46	2.1	18.1
ML	2.6	3.6	6	.70	3.0	19.7

Table 1: Comparison of Partition Metrics for Investment Casting Mesh -8 processors (6,673 FE Nodes)

	64 Processors (186,381 FE Nodes)[1]				256 Processors (1,088,019 FE Nodes)			
Algorithm	Hops (10^6)	t_P	t_{UC}	Solver Efficiency	Hops (10^6)	t_P	t_{UC}	Solver Efficiency
Linear	3.2	0.0	24.3	.69	--	--	---	--
Inertial	0.62	7.0	7.7	.87	7.6	58	101	.80
Inertial + KL	0.47	84.1	4.7	.90	4.7	688	54	.88
ML	0.27	373.1	2.2	.96	1.3	~3100	13	.97

Table 2: Scaled Parallel Efficiency for 3D Horizontal CVD Reactor Mesh

[1]This is the largest problem per processor (2912 FE Nodes/Proc) that can run with the Linear partitioner
-- Unable to run due to increased message buffer size
t_P - Partitioning Time (sec.) on a SGI ONYX with R4000 100/50 MHz MIPS cpu and 256 Mb memory.
t_{UC} - Unstructured communication time (sec.)

	512 Processors				1024 Processors			
Algorithm	Hops (10^6)	t_{UC}	t_S	Mflops	Hops (10^6)	t_{UC}	t_S	Mflops
Linear	-	-	--	--	41.1	145	397	371
Inertial	10.5	71	460	356	13.9	48	261	615
Inertial + KL	6.3	34.2	403	408	7.4	26	218	740
ML		12	365	451		11	197	840

Table 3: Parallel Performance for 3D Horizontal CVD Reactor Mesh (1,088,019 FE Nodes)

-- Unable to run due to increased message buffer size
t_{UC} - Unstructured communication time (sec.)
t_S - Total solution time (matrix setup + CG Jacobi solver) (sec.)

References:

1. S. F. Ashby, T. A. Manteuffel, and P. E. Saylor, "A taxonomy of Conjugate gradient methods", SIAM J. Numer. Anal, Vol. 27, No 6, 1542-1568, (1990)

2. R. W. Freund, G. H. Golub, N. M. Nachtuigal, "Iterative solution of linear systems", Acta Numerica, 57-100, (1991)

3. M. Garey, D. Johnson, and L. Stockmeyer, "Some simplified NP-complete graph problems", Theoretical Computer Science, 1, 619-633, (1976)

4. B. Hendrickson and R. Leland, "An improved Spectral Graph partitioning algorithm for mapping parallel computations", Technical Report SAND92-1460, Sandia National Laboratories, Albuquerque, NM, (1992)

5. B. Hendrickson and R. Leland, "A user's guide to the graph partitioning code Chaco", Technical Report SAND93-2339, Sandia National Laboratories, Albuquerque, NM, (1993)

6. S. Hammond, "Mapping unstructured grid computations to massively parallel computers", Ph. D. thesis, Rensselaer Polytechnic Institute, Dept. of Computer Science, Troy, NY, (1992)

7. M. T. Jones and P. E. Plassmann, "The efficient parallel iterative solution of large sparse linear systems", Technical Report, Argonne National Laboratory, June 1992

8. B. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs", Bell System Technical Journal, 29, 291-307, (1970)

9. Y. Saad, "Krylov subspace methods on supercomputers", SIAM J. Sci. Stat. Comput., Vol 10, No. 6, 1200-1232, (1989)

10. J. N. Shadid and R. S. Tuminaro, "Sparse iterative algorithm software for large-scale MIMD machines: an initial discussion and implementation", Concurrency: Practice and Experience, 4(6), 481-497, (1992)

11. J. N. Shadid and R. S. Tuminaro, "A comparison of preconditioned nonsymmetric Krylov methods on a large-scale MIMD machine", SIAM J. Sci. Comput., Vol. 15, No. 2, pp 440-459, March 1994

12. H. Simon, "Partitioning of unstructured problems for parallel processing", in Proc. Conference on Parallel Methods on Large Scale Structural Analysis and Physics Applications, Pergammon Press, (1991)

13. R. Williams, Performance of dynamic load balancing algorithms for unstructured mesh calculations, Concurrency, 3, 457-481, (1991)

Thursday, April 7

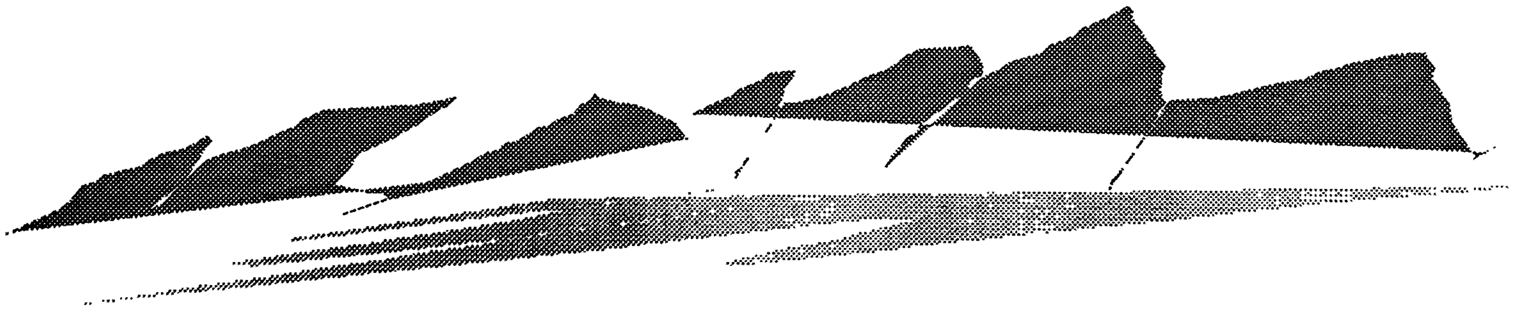
Nonsymmetric Solvers IV
Chair: Anne Greenbaum
Room A

10:15 - 10:40 E. Gallopoulos
Matrix-Valued Polynomials in Lanczos Type Methods

10:40 - 11:05 Teri Barth
Variable Metric Conjugate Gradient Methods

11:05 - 11:30 Ron Morgan
Some Comparison of Restarted GMRES and QMR for Linear and Nonlinear Problems

11:30 - 11:55 David Young
MGMRES: A Generalization of GMRES for Solving Large Sparse Nonsymmetric Linear Systems



Matrix-valued polynomials in Lanczos type methods *

V. Simoncini[†] E. Gallopoulos[‡]

January 3, 1994

Abstract

It is well known that convergence properties of iterative methods can be derived by studying the behavior of the residual polynomial over a suitable domain of the complex plane. Block Krylov subspace methods for the solution of linear systems

$$A[x_1, \ldots, x_s] = [b_1, \ldots, b_s] \quad (1) \qquad (1)$$

lead to the generation of residual polynomials $\phi_m \in \bar{P}_{m,s}$ where $\bar{P}_{m,s}$ is the subset of matrix-valued polynomials of maximum degree m and size s such that $\phi_m(0) = I_s$, $R_m := B - AX_m = \phi_m(A) \circ R_0$, where $\phi_m(A) \circ R_0 := R_0 - A\sum_{j=0}^{m-1} A^j R_0 \xi_j$, $\xi_j \in \mathrm{R}^{s \times s}$. An effective method has to balance adequate approximation with economical computation of iterates defined by the polynomial. Matrix valued polynomials can be used to improve the performance of block methods for solving (1). Another approach is to solve for a single right-hand side at a time and use the generated information in order to update the approximations of the remaining systems. In light of this, a more general scheme is as follows: A subset of residuals (seeds) is selected and a block short term recurrence method is used to compute approximate solutions for the corresponding systems. At the same time the generated matrix valued polynomial is implicitly applied to the remaining residuals. Subsequently a new set of seeds is selected and the process is continued as above, till convergence of all right-hand sides. The use of a quasi-minimization technique ensures a smooth convergence behavior for all systems. In this talk we discuss the implementation of this class of algorithms and formulate strategies for the selection of parameters involved in the computation. Experiments and comparisons with other methods will be presented.

*This research was supported by the National Science Foundation under grant NSF CCR-9120105 and by ARPA under grant COM-NIST-UNIVMINN.

[†]Dipartimento di Matematica Pura ed Applicata, Universitá di Padova, Italy. E-mail `simoncin@deis20.cineca.it`.

[‡]Center for Supercomputing Research and Development and Coordinate Sciences Laboratory, Univ. of Illinois at Urbana-Champaign, Urbana, Illinois 61801. E-mail `stratis@csrd.uiuc.edu`.

Variable Metric Conjugate Gradient Methods

Teri Barth

Abstract not available

Speaker / corresponding author: Ron Morgan
(817) 755-3561
morganr@baylor.edu

Some Comparison of restarted GMRES and QMR for Linear and Nonlinear Problems

Comparisons are made between the following methods: QMR including its transpose-free version, restarted GMRES, and a modified restarted GMRES that uses approximate eigenvectors to improve convergence. For some problems, the modified GMRES is competitive with or better than QMR in terms of the number of matrix-vector products. Also, the GMRES methods can be much better when several similar systems of linear equations must be solved, as in the case of nonlinear problems and ODE problems.

Ron Morgan
Department of Mathematics
Baylor University
Waco, TX 76798-7328

Wayne Joubert
Los Alamos National Laboratory
Group C-3, MS-B265
Los Alamos, NM 87545

Abstract

MGMRES: A Generalization of GMRES for Solving Large Sparse Nonsymmetric Linear Systems

by

David M. Young and Jen Yuan Chen
Center for Numerical Analysis
The University of Texas at Austin
Austin, Texas

We are concerned with the solution of the linear system (1): $Au = b$, where A is a real square nonsingular matrix which is large, sparse and nonsymmetric. We consider the use of Krylov subspace methods. We first choose an initial approximation $u^{(0)}$ to the solution $\bar{u} = A^{-1}b$ of (1). We also choose an auxiliary matrix Z which is nonsingular. For $n = 1, 2, ...$ we determine $u^{(n)}$ such that $u^{(n)} - u^{(0)} \varepsilon K_n(r^{(0)}, A)$ where $K_n(r^{(0)}, A)$ is the (Krylov) subspace spanned by the Krylov vectors $r^{(0)}, Ar^{(0)}, ..., A^{n-1}r^{(0)}$ and where $r^{(0)} = b - Au^{(0)}$. If ZA is SPD we also require that $(u^{(n)} - \bar{u}, ZA(u^{(n)} - \bar{u}))$ be minimized. If, on the other hand, ZA is not SPD, then we require that the Galerkin condition, $(Zr^{(n)}, v) = 0$, be satisfied for all $v \varepsilon K_n(r^{(0)}, A)$, where $r^{(n)} = b - Au^{(n)}$.

With the GMRES method, which was developed by Saad and Schultz [1986], and which has for many years been used extensively for solving large sparse nonsymmetric systems one lets $Z = A^T$. One generates a set of mutually orthogonal vectors $w^{(0)}, w^{(1)}, ... w^{(n)}$ such that $w^{(0)} = r^{(0)}$ and such that $Sp(w^{(0)}, w^{(1)}, ..., w^{(k-1)}) = K_k(r^{(0)}, A)$ for $k = 0, 1, 2, \ldots, n$. To do this, for each k we let $w^{(k)}$ be a linear combination of $Aw^{(k-1)}, w^{(k-1)}, ..., w^{(0)}$. Next, for each n we choose $c_o^{(n)}, c_1^{(n)}, \ldots, c_{n-1}^{(n)}$ so that $u^{(n)} = u^{(0)} + c_o^{(n)} w^{(0)} + \ldots + c_{n-1}^{(n)} w^{(n-1)}$ and so that $(r^{(n)}, r^{(n)})$ is minimized. The $c_i^{(n)}$ are determined by solving a related system of linear equations in the least squares sense. This is done in a stable manner using Givens rotations.

In this paper we consider a generalization of GMRES. This generalized method, which we refer to as "MGMRES", is very similar to GMRES except that we let $Z = A^T Y$ where Y is a nonsingular matrix which is symmetric but not necessarily SPD. We require that the $w^{(i)}$ be mutually orthogonal

with respect to Y. Of course if Y is not SPD it is possible that the process of generating the $w^{(i)}$ may break down. Also, unless Y is SPD we must replace the minimization condition on $(r^{(n)}, r^{(n)})$ by a Galerkin condition which requires that $(Zr^{(n)}, w^{(i)}) = 0$ for $i = 0, 1, \ldots, n-1$. The determination of the coefficients $c_i^{(n)}$ can be carried out using Givens rotations, as in the case of GMRES, though the overall procedure is somewhat more complicated. It can, however be shown that, for given n^* and $u^{(0)}$, one can uniquely determine $u^{(1)}, u^{(2)}, \ldots, u^{(n*)}$ provided that the process of computing $w^{(0)}, w^{(1)}, \ldots, w^{(n^*-1)}$ does not break down and provided that for $n = 1, 2, \ldots, n^*$ there actually exists a unique vector $u^{(n)}$ such that $u^{(n)} - u^{(0)} \varepsilon K_n(r^{(0)}, A)$ and such that the Galerkin condition is satisfied.

The MGMRES algorithm is considerably simplified if YA as well as Y is symmetric. Under this assumption one can determine $w^{(n)}$ in terms of $w^{(n-1)}$ and $w^{(n-2)}$ instead of in terms of $w^{(n-1)}, w^{(n-2)}, \ldots, w^{(0)}$ as would be required in the general case. The determination of the coefficients $c_i^{(n)}$ which are involved in the Galerkin condition is also considerably simplified. An example of a case where Y and YA are symmetric is the "double system" which corresponds to the Lanczos method for solving (1). Thus given the linear system (1) one can consider the double system $\{A\}\{u\} = \{b\}$ where for some $\tilde{b}$ and $\tilde{u}$ we have

$$\{A\} = \begin{pmatrix} A & 0 \\ 0 & A^T \end{pmatrix}, u = \begin{pmatrix} u \\ \tilde{u} \end{pmatrix}, b = \begin{pmatrix} b \\ \tilde{b} \end{pmatrix} \qquad (1)$$

We also choose

$$\{Y\} = \begin{pmatrix} 0 & I \\ I & 0 \end{pmatrix} \qquad (2)$$

Evidently $\{Y\}$ and $\{Y\}\{A\}$ are symmetric. The application of MGMRES with $Y = \{Y\}$ to the double system yields the "LANGMRES" method given by Young and Chen [1994].

An important feature of the GMRES algorithm is that one can determine $(r^{(n)}, r^{(n)})$ for a given n and test for convergence without actually carrying out the complete GMRES process to determine $u^{(n)}$. Thus, one can compute $(r^{(n)}, r^{(n)})$ for each iteration and only actually compute $u^{(n)}$ when $(r^{(n)}, r^{(n)})$ is smaller than a prescribed tolerance level. We describe a similar procedure for MGMRES. For each n we first compute the residual $\tilde{r}^{(n)}$ for ORTHORES

(Y) by determining the scaling factor c_n such that $\tilde{r}^{(n)} = c_n w^{(n)}$. The residual $r^{(n)}$ for MGMRES can be determined from $\tilde{r}^{(n)}$ by a short series of elementary vector operations. No matrix-vector operations or inner products are required to get $r^{(n)}$.

References

[1] Saad, Youcef and Schultz, Martin H. [1986], "GMRES, A Generalized Minimum Residual Algorithm for Solving Non Symmetric Linear Systems", *SIAM J. Sci. Stat. Comput.* 7, 856-869.

[2] Young, David M. and Chen, Jen Yuan [1994], "LANGMRES, An Alternative to the Biconjugate Gradient Algorithm for Solving Large Sparse Nonsymmetric Linear Systems", to appear in the proceedings of a conference held at North Carolina State University in December 1993 in honor of Cornelius Lanczos.

Thursday, April 7

Parallel Computation IV
Chair: Dan Quinlan
Room B

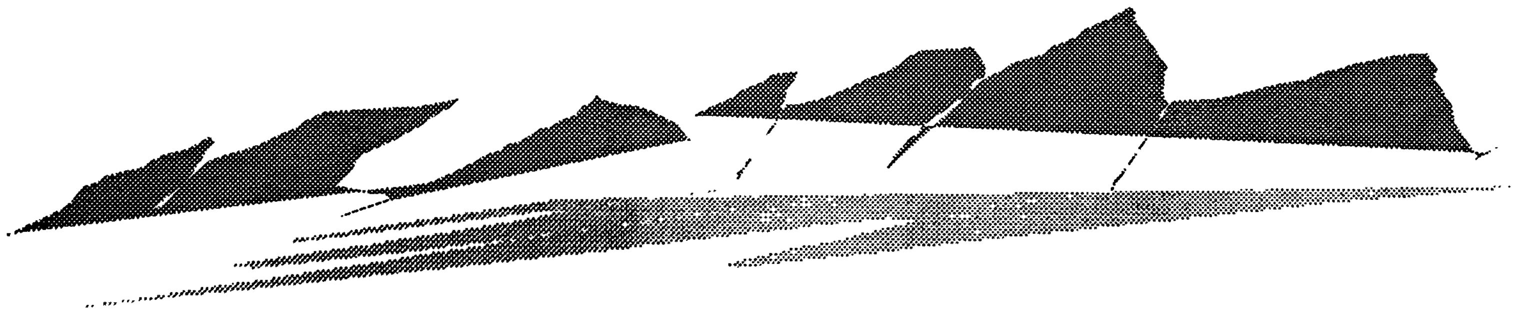
10:15 - 10:40 Larry Reeves
Adapting Implicit Methods to Parallel Processors

10:40 - 11:05 A. Basermann
Parallelizing Iterative Solvers for Sparse Systems of Equations and Eigenproblems on Distributed-Memory Machines

11:05 - 11:30 R.P. Silva
A Parallel Implementation of an EBE Solver for the Finite Element Method

11:30 - 11:55 Martin Bucker
An Implementation of the TFQMR-Algorithm on a Distributed Memory Machine

12:00 - 4:30 Informal Discussion



Adapting Implicit Methods to Parallel Processors

Larry Reeves (larryr@cs.umr.edu) Bruce McMillin (ff@cs.umr.edu) [*]
Daniel Okunbor (okunbor@cs.umr.edu)
Dept. of Computer Science
David Riggins (riggins@voyager.larc.nasa.gov) [†]
Dept. of Mechanical and Aerospace Eng.
University of Missouri-Rolla

February 10, 1994

When numerically solving many types of partial differential equations, it is advantageous to use implicit methods because of their better stability and more flexible parameter choice, (e.g. larger time steps). However, since implicit methods usually require simultaneous knowledge of the entire computational domain, these methods are difficult to implement directly on distributed memory parallel processors. This leads to infrequent use of implicit methods on parallel/distributed systems.

The usual implementation of implicit methods is inefficient due to the nature of parallel systems where it is common to take the computational domain and distribute the grid points over the processors so as to maintain a relatively even workload per processor. This creates a problem at the locations in the domain where adjacent points are not on the same processor.

[*]This work was supported in part by the National Science Foundation under Grant Numbers MSS-9216479 and CDA-9222827, and, in part, from the Air Force Office of Scientific Research under contract numbers F49620-92-J-0546 and F49620-93-1-0409.

[†]Supported by NASA grant NAG-1189

In order for the values at these points to be calculated, messages have to be exchanged between the corresponding processors. Without special adaptation, this will result in idle processors during part of the computation, and as the number of idle processors increases, the lower the effective speed improvement by using a parallel processor.

We can see this problem by examining the one-dimension diffusion equation, $\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$. Using a standard second-order implicit method, this results in the discrete equation:

$$\frac{w_i^{t+1} - w_i^t}{\Delta t} = \frac{w_{i+1}^{t+1} - 2w_i^{t+1} + w_{i-1}^{t+1}}{(\Delta x)^2}, \tag{1}$$

or, rearranged,

$$-\alpha\overline{w}_{i+1}^{t+1} + (1+2\alpha)\overline{w}_i^{t+1} - \alpha\overline{w}_{i-1}^{t+1} = \overline{w}_{i,t}, \tag{2}$$

where $\alpha = \frac{\Delta t}{(\Delta x)^2}$. Now, solving this equation on a sequential machine is quite easy using a standard tridiagonal sweep method, similar to the Thomas algorithm [1]. However, on a parallel machine with the domain decomposed across processors, only one processor at a time will be active if the sweep method is programmed to match the computations on the sequential machine. Using other matrix methods such as a straightforward implementation of LU decomposition has the same effect since they also need full domain information.

In our scheme, we consider each processor to temporarily be an independent sub-domain, and use the appropriate implicit operator for that domain, allowing all processors to operate simultaneously. Then, after each iteration, processors that have adjacent points exchange messages, updating the values for that iteration. To examine our method, we discretize the domain into $nk+2$ grid points and distribute them over n processors such that points $0, \ldots, k+1$ are on processor 1; points $k, \ldots, 2k+1$ are on processor 2; $\ldots$, and points $(n-1)k, \ldots, nk+1$ are on processor n (See Figure 1). Note that points 0 and $nk+1$ are the boundary points of the physical domain and the overlapping points are used to store values from the adjacent processors.

For simplicity, assume that there are only two processors ($n = 2$) and there are only three grid points per processor that are updated by that processor during each iteration ($k = 3$).

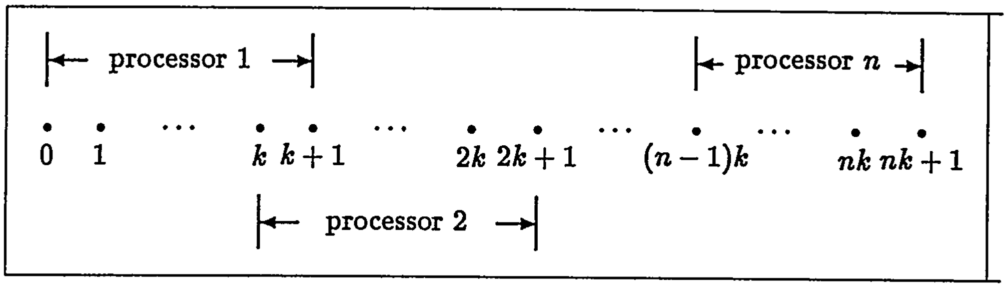


Figure 1: Domain Distribution

Examining the processor with points $0, 1$ and 2, we can write the matrix equation as:

$$\begin{bmatrix} (1+2\alpha) & -\alpha & 0 \\ -\alpha & (1+2\alpha) & -\alpha \\ 0 & -\alpha & (1+2\alpha) \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}_{t+1} = \begin{bmatrix} w_1 \\ w_2 \\ w_3 + \alpha w_4 \end{bmatrix}_t + \begin{bmatrix} \alpha w_0 \\ 0 \\ 0 \end{bmatrix} \tag{3}$$

where the subscripts on the w vectors represent the iteration level. The values for w_4 are at iteration t since they are passed by a message after each iteration. Similarly, looking at the processor with points $4, 5$ and 6, the resulting matrix equation is:

$$\begin{bmatrix} (1+2\alpha) & -\alpha & 0 \\ -\alpha & (1+2\alpha) & -\alpha \\ 0 & -\alpha & (1+2\alpha) \end{bmatrix} \begin{bmatrix} w_4 \\ w_5 \\ w_6 \end{bmatrix}_{t+1} = \begin{bmatrix} w_4 + \alpha w_3 \\ w_5 \\ w_6 \end{bmatrix}_t + \begin{bmatrix} 0 \\ 0 \\ \alpha w_7 \end{bmatrix} \tag{4}$$

Combining equations 3 and 4, we can get the global equation for this example:

$$\begin{bmatrix} (1+2\alpha) & -\alpha & 0 & & & \\ -\alpha & (1+2\alpha) & -\alpha & & & \\ 0 & -\alpha & (1+2\alpha) & & & \\ & & & (1+2\alpha) & -\alpha & 0 \\ & & & -\alpha & (1+2\alpha) & -\alpha \\ & & & 0 & -\alpha & (1+2\alpha) \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \end{bmatrix}_{t+1} = \begin{bmatrix} w_1 \\ w_2 \\ w_3+\alpha w_4 \\ w_4+\alpha w_3 \\ w_5 \\ w_6 \end{bmatrix}_t + \begin{bmatrix} \alpha w_0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \alpha w_7 \end{bmatrix} \tag{5}$$

where all the elements off of the main diagonals of the left-hand matrix are 0. Let's represent this equation in the form

$$P\overline{w}_{(t+1)} = B\overline{w}_{(t)} + \overline{b}, \tag{6}$$

where B is

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & \alpha & 0 & 0 \\ 0 & 0 & \alpha & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

Upon examination, it is easy to see that B can be written as $Q + I$, where Q is all 0's except for the two α's just off the main diagonal, and I is the 6x6 identity matrix. Thus, we can rearrange (6) as

$$(P - Q)\overline{w}_{(t+1)} = \overline{w}_{(t)} + \overline{b}. \tag{8}$$

But, this is exactly the form the implicit method would take if all of the domain was on a single processor. The main thing to notice is that the left-hand operator has been split in such a manner as to uncouple the two sections of the domain so that both processors can be calculating new values simultaneously. This improves the efficiency over common implicit implementations, for example [2], [3] and [4]. Although this example is small, it can be easily generalized to a larger problem on many processors.

There are 3 points to consider in analyzing this method:

- First, how does the splitting of the operator affect the convergence/convergence rate as compared to the original operator of Equation 2? We conjecture that the actual convergence should not be affected, but that the convergence rate will be less than that of the original operator. The critical part will be to discover if this method will give convergence rates better than those of the best of the explicit methods such as conjugate gradient or cyclic reduction.

- Second, if the convergence rate is better, then how much does the message passing affect the overall run-time of the method compared to explicit codes that can overlap communication with computation? If the message passing uses too much time, then there will be significant amounts of time when all the processors are compuatationally idle. In this case, the explicit methods may use less total run time then our method, since they can usually maintain almost continuous computation. Even if our method uses fewer iterations, the longer run time would make our code impractical for actual use.

- Third, how easily can this method be generalized to higher-order differencing methods and/or higher dimensional problems? A large number of the problems that could make use of this method are 2- and 3-dimensional problems that require greater accuracy then simple first-order differences. If our method does not generalize well to these more complicated problems, then there will not be any distinct advantage to use it over other methods that are more directly applicabile.

Hopefully, this method will be both rapidly convergent and very adaptable so that the parallel community will be able to use the 'better' algorithms from the large background of sequential computing while still allowing for the efficient use of parallel systems.

References

[1] William F. Ames. *Numerical Methods for Partial Differential Equations.* Computer Science and Applied Mathematics. Academic Press, Inc., second edition, 1977. 4th printing.

[2] Peter G. Eltgroth and Mark K. Seager. The sub-implicit method: New multiprocessor algorithms for old implicit codes. *Parallel Computing*, 8(1–3):155–163, October 1988.

[3] William D. Gropp and David E. Keyes. Domain decomposition on parallel computers. *Impact of Computing in Science and Engineering*, 1(4):421–439, December 1989.

[4] James S. Ryan and Sisira K. Weeratunga. Parallel computation of 3-D Navier-Stokes flowfields for supersonic vehicles, AIAA paper 93-0064. In *31st Aerospace Sciences Meeting & Exhibit*, Reno, NV, January 11-14, 1993. AIAA.

Parallelizing Iterative Solvers for Sparse Systems of Equations and Eigenproblems on Distributed-Memory Machines

A. Basermann [a]

[a] Central Institute for Applied Mathematics
Research Centre Jülich GmbH, 52425 Jülich, Germany
email: a.basermann@kfa-juelich.de

For the solution of discretized ordinary or partial differential equations it is necessary to solve systems of equations or eigenproblems with coefficient matrices of different sparsity pattern, depending on the discretization method; using the finite element method (FE) results in largely unstructured systems of equations.

Sparse eigenproblems play particularly important roles in the analysis of elastic solids and structures [7] [12] [17]. In the corresponding FE models, the natural frequencies and mode shapes of free vibration are determined as are buckling loads and modes. Another class of problems is related to stability analysis, e.g. of electrical networks. Moreover, approximations of extreme eigenvalues are useful for solving sets of linear equations, e.g. for determining condition numbers of symmetric positive definite matrices or for conjugate gradients methods with polynomial preconditioning [3].

Iterative methods for solving linear systems and eigenproblems mainly consist of matrix-vector products and vector-vector operations; the main work in each iteration is usually the computation of matrix-vector products. Therein, accessing the vector is determined by the sparsity pattern and the storage scheme of the matrix.

For parallelizing iterative solvers on a multiprocessor system with distributed memory, in particular the data distribution and the communication scheme depending on the used data structure for sparse matrices are of greatest importance for the efficient execution. These schemes can be determined before the execution of the solver by analysing the sparsity pattern of the matrix and can be exploited in each iteration. Moreover, the schemes are applicable as long as the symbolic structure of the matrix which is determined by the discretization mesh does not change, i.e. they can be used in each time step of a time dependent problem or in each iterative step of a nonlinear problem which is solved by linearization. In this paper, a data distribution and a communication scheme are presented which are based on the analysis of the column indices of the non-zero matrix elements.

Storage schemes for large sparse matrices depend on the sparsity pattern of the matrix, the considered algorithm, and the architecture of the employed computer system. The storage scheme considered in this paper is often used in FE programs and suitable for regular as well as for irregular discretization meshes. It can be found in similar form in e.g. [14]. The non-zeros of the matrix are stored row-wise in three one-dimensional arrays a^w, a^s, and a^z. a^w contains the values of the non-zeros, a^s the corresponding column indices. In a^z, the position of the beginning of each row in a^w and a^s is stored.

First, the matrix is distributed row-wise to each processor, the vector components accordingly. By analyzing the column indices, each processor determines which matrix elements result in computations with local data and which ones with non-local data. Communication and local computations are performed overlapped to reduce waiting times.

The parallelization strategy has been applied to the method of conjugate gradients with preconditioning [11] [15] and the symmetric Lanczos algorithm [8] [9] [16] [19] [20].

The method of conjugate gradients (CG) is a frequently used iterative solver for systems of linear equations $Ax = b$, particularly for sparse coefficient matrices A. The method converges for matrices which are symmetric and positive definite. In these investigations, the modified CG algorithm suggested by Aykanat e.a. [4] has been applied since it has better parallelization properties than the original method. Preconditioning is done by simple diagonal scaling [15] which hardly contributes to the total execution time but usually accelerates the convergence considerably. Polynomial preconditioning [3] is subject to current investigations; this class of preconditioners allows using the same parallelization strategies as for the pure CG algorithm since the essential additional operations are sparse matrix times dense vector computations. Furthermore, the applicability of these strategies to the QMR algorithm for solving non-hermitian systems of linear equations [10] will be investigated.

Lanczos methods are most commonly used to approximate a small number of extreme eigenvalues and eigenvectors for a real symmetric large sparse $n \times n$ matrix A [8] [9] [12] [16] [19] [20]. The principle of the Lanczos methods is described in the following. Starting with A and an initial vector q_1, a vector sequence $q_m, \quad m = 2, 3, \ldots$, and a sequence of $m \times m$ symmetric tridiagonal matrices $T_m, \quad m = 1, 2, 3, \ldots$, are generated by an iterative process. Essentially, the iteration consists of a matrix-vector product and vector-vector operations. This property makes it easy to exploit the sparsity of A. Certain eigenvalues of the matrices $T_m, \quad m = 1, 2, 3, \ldots$, may be good approximations of eigenvalues of A. The eigenvalues of the tridiagonal matrices may be determined by the parallel bisection technique described in [6] or other methods. One result of the approximation theory for Lanczos methods is that if the extreme eigenvalues of A are well seperated from the rest of the spectrum then these are usually the first to be well approximated by eigenvalues of T_m [16]. If some good approximations of eigenvalues of A have been found the corresponding eigenvectors of T_m may be computed by inverse iteration and transformed to eigenvectors of A using the Lanczos vectors $q_m, \quad m = 1, 2, 3, \ldots$ [8]. Criteria to decide which eigenvalues of T_m are good approximations of eigenvalues of A are described in e.g. [8] and [16]. Here, the two variants of the Lanczos tridiagonalization described in [5] and [13] have been applied; the latter method has better parallelization properties than the former.

Numerical and performance tests of the developed parallel variants of the considered solvers for systems of equations and eigenproblems have been carried out on the distributed-memory system INTEL iPSC/860 of the Research Centre Jülich with sparse matrices from two FE models. The first FE model comes from environmental science; it simulates the behaviour of pollutants in geological systems [1] [18]. In the second FE model from structural mechanics, stresses in materials induced by thermal expansion are calculated by applying the FE program SMART [2]. The parallel algorithms have been shown to be well suited for the considered large sparse matrices. The parallel CG method is employed in both projects. On a distributed-memory system, the developed data distribution and communication scheme result in flexible algorithms. These algorithms perform well for large sparse matrices of very different sparsity patterns.

References

[1] 3DFEMWATER: a three-dimensional finite element model of water flow through saturated-unsaturated media. Oak Ridge National Laboratory. *ORNL-6386*, 1987

[2] SMART, Benutzerhandbücher. Institut für Statik und Dynamik der Luft- und Raumfahrtkonstruktionen der Universität Stuttgart. *ISD-Berichte*, 1976-1992.

[3] S.F. Ashby. Minimax polynomial preconditioning for hermitian linear systems. *SIAM J. Matrix Anal. Appl.*, 12:766–789, 1991.

[4] C. Aykanat, F. Özgüner, D.S. Scott. Vectorization and parallelization of the conjugate gradient algorithm on hypercube-connected vector processors. *Microprocessing and Microprogramming*, 29:67–82, 1990.

[5] V.A. Barker, C. Yingqun. LANSYM: a Fortran subroutine for computing eigensolutions of symmetric sparse matrices on the Connection Machine. Institute for Numerical Analysis, Technical University of Denmark, Lyngby, *Report NI-92-12*, December 1992.

[6] A. Basermann, P. Weidner. A parallel algorithm for determining all eigenvalues of large real symmetric tridiagonal matrices. *Parallel Computing*, 18:1129–1141, 1992.

[7] L. Collatz. *Eigenwertaufgaben mit technischen Anwendungen.* Akademische Verlagsgesellschaft Geest & Portig K.-G., Leipzig, 1963.

[8] J.K. Cullum, R.A. Willoughby. *Lanczos Algorithms for Large Symmetric Eigenvalue Computations.* Volume I: Theory, Birkhäuser, Boston Basel Stuttgart, 1985.

[9] J.K. Cullum, R.A. Willoughby. *Lanczos Algorithms for Large Symmetric Eigenvalue Computations.* Volume II: Programs, Birkhäuser, Boston Basel Stuttgart, 1985.

[10] R.W. Freund, N.M. Nachtigal. QMR: a quasi-minimal residual method for non-Hermitian linear systems. *Numerische Mathematik*, 60:315–339, 1991.

[11] M.R. Hestenes, E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49:409–436, 1952.

[12] T.J.R. Hughes. *The Finite Element Method.* Prentice-Hall, Englewood Cliffs, 1987.

[13] S.K. Kim, A.T. Chronopoulos. A class of Lanczos-like algorithms implemented on parallel computers. *Parallel Computing*, 17:763–778, 1991.

[14] C.P. Kruskal, L. Rudolph, M. Snir. Techniques for parallel manipulation of sparse matrices. *Theoretical Computer Science*, 64:135–157, 1989.

[15] J.M. Ortega. *Introduction to Parallel and Vector Solution of Linear Systems.* Plenum Press, New York London, 1988.

[16] B.N. Parlett. *The Symmetric Eigenvalue Problem.* Prentice-Hall, Englewood Cliffs, 1980.

[17] Y. Saad. *Numerical Methods for Large Eigenvalue Problems.* Manchester University Press, Manchester, 1992.

[18] H. Vereecken, G. Lindenmayr, A.Kuhr, D.H. Welte, A. Basermann. Numerical modelling of field scale transport in heterogeneous variably saturated porous media. *KFA/ICG-4 Internal Report No. 500393*, January 1993.

[19] J.H. Wilkinson. *The Algebraic Eigenvalue Problem.* Clarendon Press, Oxford, 1965.

[20] J.H. Wilkinson, C. Reinsch. *Handbook for Automatic Computation.* Volume II: Linear Algebra, Springer-Verlag, Berlin Heidelberg New York, 1971.

A Parallel Implementation of an EBE Solver for the Finite Element Method

R. P. Silva* E. B. Las Casas† M. L. B. Carvalho‡
Federal University of Minas Gerais
Av. Contorno, 842 – 2º andar
30110–060 Belo Horizonte MG
BRAZIL

A parallel implementation using PVM on a cluster of workstations of an Element By Element (EBE) solver using the Preconditioned Conjugate Gradient (PCG) method is described, along with an application in the solution of the linear systems generated from finite element analysis of a problem in three dimensional linear elasticity.

The PVM (*Parallel Virtual Machine*) system [?], developed at the Oak Ridge Laboratory, allows the construction of a parallel MIMD machine by connecting heterogeneous computers linked through a network. In this implementation, version 3.1 of PVM is used, and 11 SLC Sun workstations and a Sun SPARC-2 model are connected through Ethernet.

The finite element program is based on SDP, *System for Finite Element Based Software Development* [?], developed at the Brazilian National Laboratory for Scientific Computation (LNCC). SDP provides the basic routines for a finite element application program, as well as a standard for programming and documentation, intended to allow exchanges between research groups in different centers.

In the finite element method, the linear system to be solved is $Kp = f$, where K is the "stiffness" matrix, p the unknowns to be calculated and f

*Structural Engineering Department — ramon@dcc.ufmg.br
†Structural Engineering Department — lauraclc@brufmg.bitnet
‡Computer Science Department — mlbc@dcc.ufmg.br

the "force" vector. As matrix K and vector p are the result of a Boolean sum —assembly— of individual contributions of each element, the problem can be written as:

$$Kp = \left(\sum_{e=1}^{nel} K^e\right) p = \sum_{e=1}^{nel} K^e p^e \tag{1}$$

where nel is the number of elements, and K^e and p^e are the element contributions to the global system. To avoid assembling the global stiffness matrix, the solution is obtained at element level, using the PCG method. This allows large savings in storage, making the program useful in the solution of very large models. In an analysis of the PCG method [?], using Jacobi as the preconditioner due to its simplicity, the product $K^e p^e$ defines the order of the algorithm, and can be run in parallel for different groups of elements. Details on the data structure are described in [?]. To implement the parallel version of the solver, it is only necessary to have at each processor the data related to elements contained at one group of elements. Communication can be kept low by restricting it to nodes belonging to elements at the group boundaries. The efficiency of the algorithm is then conditioned by the partition in groups, which can be made automatically trying to minimize the number of elements in common boundaries. As an application of the described implementation, a cube subjected to concentrated load at one corner was analyzed, with different levels of discretization. The number of equations varied from 2361 to 39390, and the obtained efficiency, given by (2), is shown in Fig. 1.

$$E = \left(\frac{t^{slc}}{t//}\right)\left(\frac{1}{ngr}\right) \tag{2}$$

where t^{slc} is the time running sequentially in one SLC processor, $t^{//}$ is the time spent in the parallel processing, and ngr is the number of processors.

It was not possible to compute the efficiency for the finer mesh (39390 unknowns), as the problem is too large to be processed sequentially in the SLC in its present configuration with 8 Mbytes. The average execution time for this case was 3963 and 3414 seconds respectively for 9 and 12 processors, corresponding to 511 iterations. The performance of the algorithm is affected by the system load during execution, forcing the measures to be obtained from averaged values.

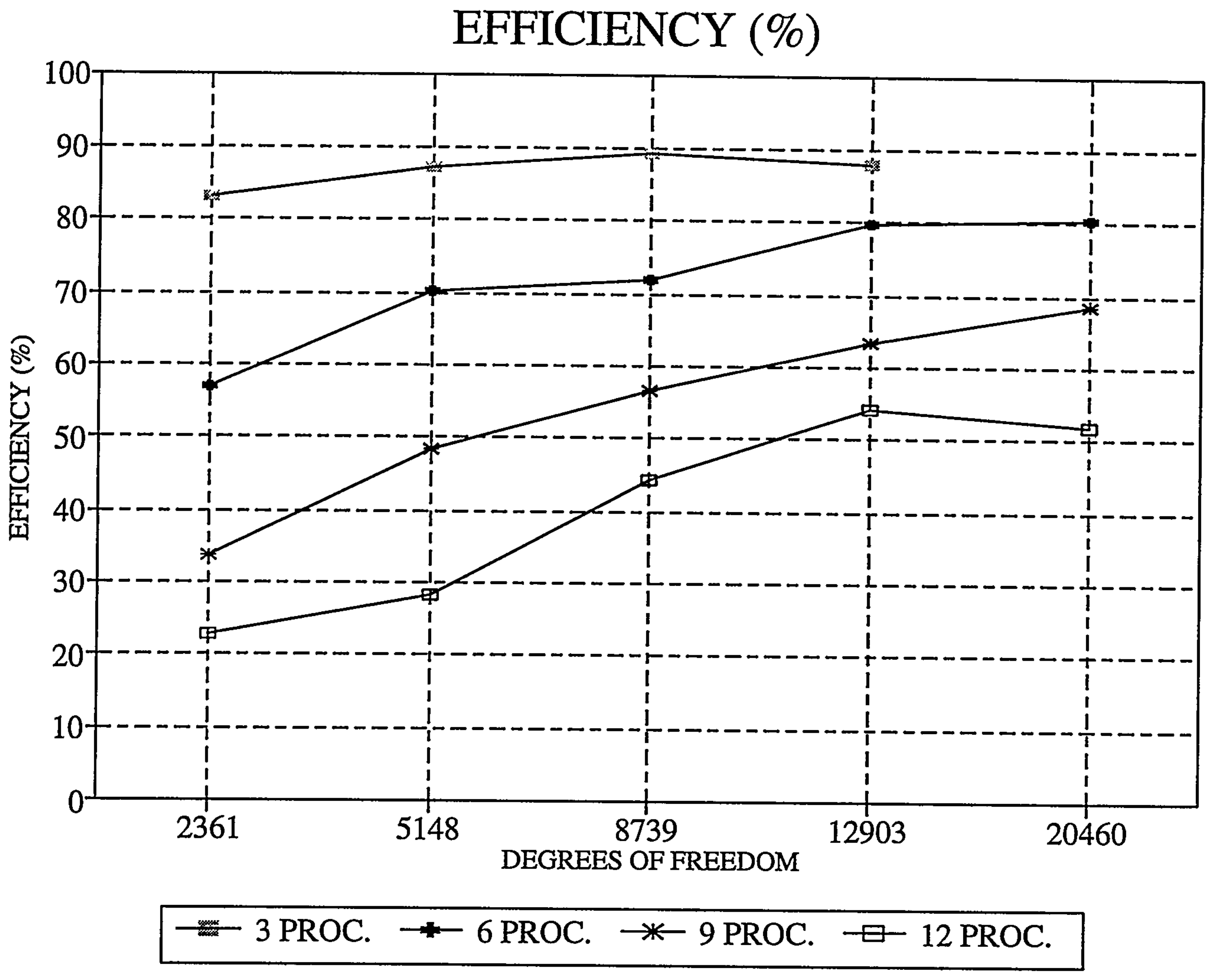


Figure 1: Computational Efficiency (in %) for Different Discrete Models

An Implementation of the TFQMR–Algorithm on a Distributed Memory Machine

Martin Bücker *
Technical University of Aachen
email: m.buecker@kfa-juelich.de

January 3, 1994

1 Introduction

A fundamental task of numerical computing is to solve linear systems

$$A\vec{x} = \vec{b}. \tag{1}$$

Such systems are essential parts of many scientific problems, e.g. finite element methods contain linear systems to approximate partial differential equations. Linear systems can be solved by direct as well as iterative methods.

Using a direct method means to factorize the coefficient matrix. A typical and well known direct method is the Gaussian elimination. Direct methods work well as long as the systems remain small. Unfortunately, there is need for the solution of large linear systems [8]. When systems get large direct methods result in enormous computing time because of their complexity. As an example take (1) with a $N \times N$ matrix A. Then the solution of (1) can be computed by Gaussian elimination requiring $O(N^3)$ operations. Additionally, an implementation of a direct method has to take care of the high storage requirement.

In contrast to direct methods, iterative methods use successive approximations to obtain more accurate solutions to a linear system at each step. The iteration process generates a sequence of iterates $\vec{x}_n$ converging to the solution $\vec{x}$ of (1). The iteration process ends if either $\vec{x}_n$ fulfills a chosen convergence criterion or *breakdowns*, i.e. division by zero, occur. Possible breakdowns can be avoided by look–ahead techniques [4, 10].

Often, linear systems are sparse meaning that the coefficient matrix contains only a few nonzero entries. A powerful iterative method to solve large sparse linear systems with Hermitian positive definite coefficient matrices is the conjugate gradient method (CG) [6]. Each step of CG involves the coefficient matrix only in the form of one matrix–vector product with A. Frequently, computation times of iterative methods are almost entirely dominated by the time to calculate these products. So an important aspect of choosing a specific algorithm among several possible ones is the number of matrix–vector products.

While the repository for Hermitian systems is quite rich there exist only a few methods for non–Hermitian systems. An extension of CG to linear systems with general non–Hermitian nonsingular matrices leads to the biconjugate gradient method (BCG) [9]. BCG generates

*This work was done at the Central Institute for Applied Mathematics, Jülich Research Centre, Germany two CG–like iterates. One sequence is based on matrix–vector products with A, the other one with A^T. So each step of BCG needs the computation of two matrix–vector products. BCG shows an irregular convergence behavior and breakdowns may occur.

Both disadvantages of BCG — with the exception of incurable breakdowns — can be overcome by the quasi–minimal residual method (QMR) [5]. QMR produces smooth convergence curves and can be implemented with look–ahead. Like BCG each step of QMR is built of two matrix–vector products with A as well as with A^T. On parallel machines with distributed memory, efficient transposition is a difficult task, see e.g. [7]. Just recently, Freund [3] introduced an algorithm eliminating the matrix–vector product with A^T. This transpose–free quasi–minimal residual method (TFQMR) uses two matrix–vector products with A in each iteration step. Like QMR, TFQMR demonstrates a regular convergence behavior but breakdowns are possible. Here we will focus on an implementation of TFQMR on a machine with distributed memory.

The rest of the paper is organized as follows. In section 2, we start with the TFQMR method. The parallel implementation is described in section 3. Timing results are given in section 4.

Throughout the paper, the vector norm $\|\vec{x}\|$ is the Euclidean norm. Inner products are denoted by $\langle\vec{x},\vec{y}\rangle$. The notation

$$\mathcal{K}_m(\vec{h},D) := \mathrm{span}\{\vec{h}, D\vec{h}, \ldots, D^{m-1}\vec{h}\}$$

is used for the mth Krylov subspace of $\mathbb{C}^N$ generated by $\vec{h} \in \mathbb{C}^N$ and the $N \times N$ matrix D. To obtain a matrix by concatenation of the vectors $\vec{y}_1, \vec{y}_2, \ldots, \vec{y}_n$ we write $[\vec{y}_1\ \vec{y}_2 \cdots \vec{y}_n]$.

2 TFQMR–Algorithm

The solution of linear systems (1) with general non–Hermitian nonsingular $N \times N$ coefficient matrices can be computed by the transpose–free quasi–minimal residual method (TFQMR). This section shortly presents the algorithm. Details can be found in [3]. TFQMR is derived via the standard conjugate gradient squared algorithm (CGS) without look–ahead [11]. CGS was the first transpose–free BCG–type method and produces iterates $\vec{x}_{2n}$ by an updating scheme of the form

$$\vec{x}_{2n} = \vec{x}_{2n-2} + \alpha_{n-1}(\vec{y}_{2n-1} + \vec{y}_{2n}). \qquad (2)$$

The vectors $\vec{y}_1, \vec{y}_2, \ldots, \vec{y}_{2n}$ of (2) span the following Krylov subspace:

$$\mathrm{span}\{\vec{y}_1, \vec{y}_2, \ldots, \vec{y}_m\} = \mathcal{K}_m(\vec{r}_0, A), \qquad (3)$$

where $m = 1, 2, \ldots, 2n$ and $\vec{r}_0 = \vec{b} - A\vec{x}_0$ is the residual vector obtained for some initial guess $\vec{x}_0$. Each step of CGS generates two search directions $\vec{y}_{2n-1}$ and $\vec{y}_{2n}$. But the actual iterate only makes use of the sum of both directions. In contrast, TFQMR creates two iterates $\vec{x}_{2n-1}$ and $\vec{x}_{2n}$ per step exploiting all available search directions. The TFQMR–iterates have the form:

$$\vec{x}_m = \vec{x}_0 + [\vec{y}_1\ \vec{y}_2 \cdots \vec{y}_m]\vec{z}_m, \quad \vec{z}_m \in \mathbb{C}^m, \quad (4)$$

with $m = 2n-1, 2n$. The free parameter vector $\vec{z}_m$ in (4) can be chosen such that the iterates satisfy a condition which is similar to the quasi–minimization property of the QMR method. The resulting iteration process is shown in Figure 1.

3 Parallel Implementation

In this section we describe a message–passing based implementation of TFQMR. To our knowledge, this is the first such implementation on a massively parallel processor like Intel's iPSC/860.

We begin by pointing out the data structure used and show how the data is distributed among the processors. The basic operations, matrix–vector products, inner products and vector updates, respectively, are discussed with regard to parallelism. Finally, we make some remarks concerning the stopping criterion.

3.1 Data Structure

As mentioned above, the dominant time–consuming building blocks of iterative methods are the matrix–vector products. Consequently, these products have to be implemented carefully. We use a sparse storage scheme to reach our goal: The solution of large sparse systems. So only nonzero elements of A are stored. We choose the Compressed Row Storage [1] format because we don't assume more than the sparsity structure of the matrix. Other storage schemes should be taken into consideration if specific properties of the coefficient matrices are known. E.g., the Compressed Diagonal Storage [1] format could be appropriate for banded matrices with a constant bandwidth.

The Compressed Row Storage format stores subsequent nonzero entries of the matrix rows in contiguous memory cells. The matrix is represented by three different arrays. The data structure of a $N \times N$ matrix with t real nonzeros is shown in figure 2. The array `value` contains the data entries of the matrix A. The corresponding column indices are stored in `col_ind`. The contents of the array `row_ptr` points to the locations in the array `value` that

```
Choose $\vec{x}_0 \in \mathbb{C}^N$
$\vec{w}_1 = \vec{b} - A\vec{x}_0$
$\vec{y}_1 = \vec{b} - A\vec{x}_0$
$\vec{r}_0 = \vec{b} - A\vec{x}_0$
$\vec{v}_0 = A\vec{y}_1$
$\vec{d}_0 = \vec{0}$
$\tau_0 = ||\vec{r}_0||$
$\vartheta_0 = 0$
$\eta_0 = 0$
Choose $\vec{\tilde{r}}_0$ such that $\rho_o = \langle \vec{\tilde{r}}_0, \vec{r}_0 \rangle \neq 0$
for $n = 1, 2, 3, \ldots$ do
    $\sigma_{n-1} = \langle \vec{\tilde{r}}_0, \vec{v}_{n-1} \rangle$
    $\alpha_{n-1} = \frac{\rho_{n-1}}{\sigma_{n-1}}$
    $\vec{y}_{2n} = \vec{y}_{2n-1} - \alpha_{n-1}\vec{v}_{n-1}$
    for $m = 2n-1, 2n$ do
        $\vec{w}_{m+1} = \vec{w}_m - \alpha_{n-1}A\vec{y}_m$
        $\vartheta_m = \frac{||\vec{w}_{m+1}||}{\tau_{m-1}}$
        $c_m = \frac{1}{\sqrt{1+\vartheta_m^2}}$
        $\tau_m = \tau_{m-1}\vartheta_m c_m$
        $\eta_m = c_m^2 \alpha_{n-1}$
        $\vec{d}_m = \vec{y}_m + \frac{\vartheta_{m-1}^2 \eta_{m-1}}{\alpha_{n-1}} \vec{d}_{m-1}$
        $\vec{x}_m = \vec{x}_{m-1} + \eta_m \vec{d}_m$
        if $\vec{x}_m$ has converged then
            stop
        end if
    end for
    $\rho_n = \langle \vec{\tilde{r}}_0, \vec{w}_{2n+1} \rangle$
    $\beta_n = \frac{\rho_n}{\rho_{n-1}}$
    $\vec{y}_{2n+1} = \vec{w}_{2n+1} + \beta_n \vec{y}_{2n}$
    $\vec{v}_n = A\vec{y}_{2n+1} + \beta_n (A\vec{y}_{2n} + \beta_n \vec{v}_{n-1})$
end for
```

Figure 1: TFQMR algorithm of [3]

starts a row of A. If a matrix element a_{ij} satisfies

$$\texttt{value}(s) = a_{ij} \tag{5}$$

then the following two equations hold simultaneously

$$\texttt{col\_ind}(s) = j \tag{6}$$

$$i = \max\{r \mid \texttt{row\_ptr}(r) \leq s\}. \tag{7}$$

Accessing — reading as well as writing — a specific matrix element a_{ij} is an expensive operation using this data structure as (7) reveals. But fortunately, TFQMR only makes use of A in the form of a matrix–vector product. Figure 3 shows how a matrix–vector product in Compressed Row Storage format is computed sequentially.

As usual, we implement vectors as one–dimensional arrays.

3.2 Data Distribution

On a distributed memory machine, each processor stores part of the data in its local memory. The matrix A can be distributed among the processors based on several criteria. We choose to assign the rows of A to different processors by not splitting up any row. Any two processors are allowed to store a distinct number of rows. The number of rows each processor holds is driven by a parameter. This parameter tries to balance the number of arithmetic operations equally. A good choice of this parameter is machine–dependent as well as specific for the algorithm. Our parameter matches the parameter of [2] which uses the same data distribution for an implementation of CG. We know that this is an inadequate choice, but nevertheless it is preliminary acceptable as the results demonstrate.

Distribution of the vectors is done by breaking up the components. It corresponds to the distribution of the rows of A.

3.3 Matrix–Vector Products

Looking at figure 1 the nth iteration seems to contain three matrix–vector products $A\vec{y}_{2n-1}, A\vec{y}_{2n}$ and $A\vec{y}_{2n+1}$. But an analysis shows that each iteration step only involves two matrix–vector products. The reason for this is that the value of $A\vec{y}_j$, with odd j, can be stored and reused in the following iteration step. A matrix–vector product is executed based on a communication scheme presented in [2]. At the beginning of the algorithm, an investigation is carried out to determine which processors have to communicate with one another. Finally, these results are made global. The investigation is done only once and this information is used in every matrix–vector product. For a more detailed description of the communication scheme, the reader is refered to [2].

The communication time of a matrix–vector

```
MATRIX = record
        value  : array [1..t] of REAL
        col_ind: array [1..t] of INTEGER
        row_ptr: array [1..N] of INTEGER
end record
```

Figure 2: $N \times N$ Matrix with t nonzeros in Compressed Row Storage format

```
for $i = 1, \ldots, N$ do
    $y(i) = 0$
    for $k = \texttt{row\_ptr}(i), \ldots, \texttt{row\_ptr}(i+1) - 1$ do
        $y(i) = y(i) + \texttt{value}(k) * b(\texttt{col\_ind}(k))$
    end for
end for
```

Figure 3: Sequential matrix–vector product in Compressed Row Storage format, $\vec{y} = A\vec{b}$

product can be overlapped with local computations. Currently, our implementation does not support such overlapping.

3.4 Inner Products

To compute an inner product, each processor calculates the inner product of the local segments. Then the global inner product is computed by applying a global system call making available the sum of all local inner products to all processors.

Computation of inner products enforces synchronization. An efficient implementation tries to minimize these synchronization points. In figure 1, each iteration step of TFQMR consists of four inner products $\langle \vec{\tilde{r}}_0, \vec{v}_{n-1} \rangle$, $\|\vec{w}_{2n}\|$, $\|\vec{w}_{2n+1}\|$ and $\langle \vec{\tilde{r}}_0, \vec{w}_{2n+1} \rangle$, respectively. The two convergence tests add more synchronization, see 3.6.

As with the matrix–vector products, we do not overlap any communication with computation. Furthermore, the number of synchronization points is not minimized at all.

3.5 Vector Updates

Vector updates are done locally. Each processor updates its own segment. Synchronization and communication are not necessary.

3.6 Stopping Criterion

Usually, the decision whether an iterate $\vec{x}_m$ is close enough to the solution is based on the residual $\vec{r}_m = \vec{b} - A\vec{x}_m$. In TFQMR, the values of $\vec{r}_m$ or $\|\vec{r}_m\|$ are not readily available. Freund [3] proposed a stopping criterion by using the inequality

$$\|\vec{r}_m\| \leq \sqrt{m+1}\, \tau_m. \tag{8}$$

So the actual value of $\|\vec{r}_m\|$ only needs to be computed in the final stages of the iteration process.

We use the difference of two subsequent iterates as the stopping criterion. More precisely, we calculate $| \vec{x}_m - \vec{x}_{m-1} |$ and take the maximum over all components with the cost of adding a further synchronization each time a convergence test is executed.

4 Timings

The implementation is done on the Intel iPSC/860 of the Research Centre Jülich, Germany. This site provides the user with up to 32 nodes each of 16 MB storage. The code is written in FORTRAN. We use the Portland–Group compiler, Rel. 4.0, with optimization level 3. All results are taken while other users share the computer so that the operating system may influence the measurements.

We run the code with two examples from environmental sciences [12]. Example I is a system of dimension $N = 17368$ with $t = 304000$ nonzeros. The corresponding values for example II are $N = 49392$, $t = 1242814$. We emphasize that both systems are symmetric ! This means that results with non–Hermitian systems still have to be run.

Table 1 demonstrates the results using the systems mentioned above. As is easily verified, TFQMR is well parallelizable.

		Example	
		I	II
	2	370	—
	4	206	560
p	8	123	306
	16	81	176
	32	63	111

Table 1: Time pro iteration step with p processors in 10^{-3} sec

5 Conclusions

The TFQMR algorithm to solve linear systems with general non–Hermitian nonsingular coefficient matrices is shown to work efficiently on a parallel machine with distributed memory. As described, the current implemenation is far from being optimal but shows satisfactory results concerning parallelism.

Future work will focus on the parameter to adjust the data distribution. Examples will be extended to non–Hermitian systems. We will try other stopping criteria including the one proposed by Freund (8). Our goal is to use fewer synchronization points by packaging communication which is needed to calculate the inner products. Communication and computation will be overlapped and the code will be put to a Paragon system.

6 Acknowledgements

I'd like to thank F. Hoßfeld and P. Weidner of Jülich Research Centre for creating a stimulating environment. My honest appreciation goes to A. Basermann for providing me with his communication scheme.

References

[1] 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, 1993.

[2] A. Basermann. Datenverteilungs– und Kommunikationsmodelle für parallele CG–Verfahren zur Lösung von Gleichungssystemen mit dünnbesetzter Koeffizientenmatrix aus FE–Anwendungen. Aachener Informatik–Berichte Nr. 93–7, 51–67, 1993.

[3] R.W. Freund. A Transpose–Free Quasi–Minimal Residual Algorithm for Non–Hermitian Linear Systems. *SIAM J. Sci. Comput.*, 14:470–482, 1993.

[4] 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.*, 14:137–158, 1993.

[5] R.W. Freund and N.M. Nachtigal. QMR: A Quasi–Minimal Risidual Method for Non–Hermitian Linear Systems. *Numer. Math.*, 60:315–339, 1991.

[6] M. Hestenes and E. Stiefel. Methods of Conjugate Gradients for Solving Linear Systems. *J. Res. Nat. Bur. Standards*, 49:409–436, 1952.

[7] C.–T. Ho and S.L. Johnsson. Matrix Transposition on Boolean n-cube Configured Ensemble Architectures. Technical Report YALEU/DCS/TR–494, Yale University, 1986.

[8] T.J.R. Hughes. *The Finite Element Method.* Prentice–Hall, 1987.

[9] C. Lanczos. Solutions of Systems of Linear Equations by Minimized Iterations. *J. Res. Nat. Bur. Standards*, 49:33–53, 1952.

[10] B.N. Parlett, D.R. Taylor, and Z.A. Liu. A Look–Ahead Lanczos Algorithm for Unsymmetric Matrices. *Math. Comp.*, 44:105–124, 1985.

[11] P. Sonneveld. CGS, a Fast Lanczos–Type Solver for Nonsymmetric Linear Systems. *SIAM J. Sci. Stat. Comput.*, 10:36–52, 1989.

[12] H. Vereecken, G. Lindenmayr, A. Kuhr, D. H. Welte, and A. Basermann. Numerical Modelling of Field Scale Transport in Heterogeneous Variably Saturated Porous Media. Internal Report KFA/ICG–4 No. 500393, Research Centre Jülich, 1993.

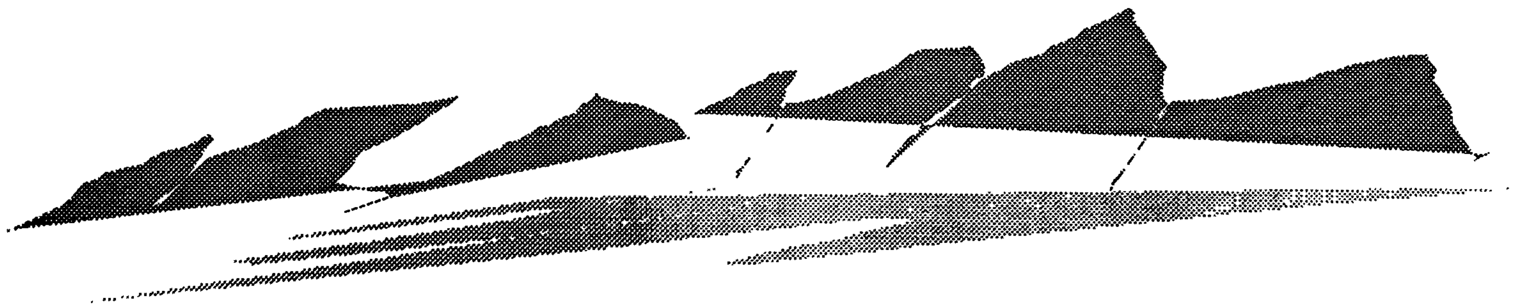
Thursday, April 7

Student Paper Winners
Chair: Tom Manteuffel and Steve McCormick
Room A

4:45 - 5:10 Qing He
Parallel Algorithms for Unconstrained Optimizations by Multisplitting

5:10 - 5:35 Lina Hemmingsson
Analysis of Semi-Toeplitz Preconditioners for First-Order PDEs

5:35 - 6:00 Johannes Tausch
Equivalent Preconditioners for Boundary Element Methods



PARALLEL ALGORITHMS FOR UNCONSTRAINED OPTIMIZATIONS BY MULTISPLITTING [*]

QING HE [†]

Abstract. In this paper a new parallel iterative algorithm for unconstrained optimization using the idea of multisplitting is proposed. This algorithm uses the existing sequential algorithms without any parallelization. Some convergence and numerical results for this algorithm are presented. The experiments are performed on an Intel iPSC/860 Hyper Cube with 64 nodes. It is interesting that the sequential implementation on one node shows that if the problem is split properly, the algorithm converges much faster than the one without splitting.

Key words. parallel algorithms, multisplitting, unconstrained optimization, QR decompositon, Householder QR decomposition, hypercube computer, variable metric method, BFGS update.

AMS subject classifications. 65H10, 65K05, 65K10, 90C30

1. Introduction.

It has long been believed that to speedup nonlinear optmization algorithms, the key is to consider, e.g., the QR decomposition, or the system solver, which comprise the major part of most optimization algorithms. Hence in the parallel processing community many researchers have been concentrating on parallelizing the linear algebra routines [3], and other routines for solving system of equations [2]. Huang and O'leary [12] used multisplitting algorithms to attack the nonlinear optimization problem by finding the root of the gradient of the objective functions.

We take a different approach whereby we split the original problem into a set of small subproblems, and obtain the solution to the original problem by solving the subproblems using existing sequential algothims. This approach forces all the awkward linear algebra operations, such as matrix factorization, to a serial environment and hence avoids the need to consider efficient parallel implementation of such operations.

Before we proceed we give a brief review of the existing *multisplitting* parallel algorithms for linear and nonlinear systems of equations. These are relevant here because the splitting we propose is very similar but occurs prior to the minimization, whereas these algorithms use splitting within the minimization to split the relevant linear systems of equations. In section 2 our new algorithms are given, with proof of convergence in section 3. The parallel convergence criteria are discussed in section 4, while section 5 shows some numerical results. Ongoing and future work are discussed in section 6.

Throughout this paper $||.||$ is assumed to be the 2-norm.

1.1. Linear Systems of Equations.

DEFINITION 1. *[14] Let be given a matrix $A \in R^{n\times n}$ and a collection of matrices $M^j, N^j, E^j \in R^{n\times n}$, $j = 1, \ldots, J$ satisfying*

1. $A = M^j - N^j$ *for* $j = 1, \ldots, J$,
2. M^j *is nonsingular for* $j = 1, \ldots, J$,
3. E^j *is a nonnegative diagonal matrix for* $j = 1, \ldots, J$ *and* $\sum_{j=1}^{J} E^j = I$.

[*] This research was performed in part using the Intel Gamma System operated by Caltech on behalf of the Concurrent Supercomputing Consortium. Access to this facility was provided by Caltech.

[†] Department of Mathematcs, Arizona State Univeristy, Tempe, AZ 85287-1804, and Department of Computer Science and Engineering, Arizona State Univeristy, Tempe, AZ 85287-5406 (qhe@enuxmp1.eas.asu.edu).

Then the collection of triples (M^j, N^j, E^j), $j = 1, \ldots, J$, is called a multisplitting of A. The corresponding linear multisplitting method for solving $Ax = b$ is defined by the iteration

$$x^{k+1} = \sum_{j=1}^{J} E^j (M^j)^{-1}(N^j x^k + b), k = 1, \ldots,$$

Most of the convergence results are summarized in [6]. Further convergence results for specific situations are given in [5], [8], [7], [11], [12], [13], [14], [15], [16], [17], [18], [21], [22].

1.2. Nonlinear Systems of Equations. The problem is to solve

$$F(x) = 0, \qquad where \qquad F : R^n \to R^n,$$

$$F(x_1, \ldots, x_n) = (F_1(x_1, \ldots, x_n), \ldots, F_n(x_1, \ldots, x_n))^T.$$

DEFINITION 2. *[5] For $j = 1, \ldots, J$ let $F^j : R^n \times R^n$ be such that $F^j(x, x) = F(x)$ for all $x \in R^n$. Moreover, for $j = 1, \ldots, J$ let E^j be a non-negative diagonal $n \times n$-matrix with $\sum_{j=1}^{J} E^j = I$. Then the collection of pairs (F^j, E^j), $j = 1, \ldots, J$, is called a nonlinear multisplitting of F. The corresponding nonlinear multisplitting method (NM-method) is defined by the iteration*

$$x^{k+1} = \sum_{j=1}^{J} E^j y^{k,j}, k = 0, 1, \ldots$$

where $y^{k,j}$ solves $F^j(x^k, y^{k,j}) = 0$.

The local convergence results are found in [5], and R. E. White [19] presents additional convergence results for nonlinear equations with special structures.

2. The Algorithms.

2.1. Linear Least Squares. Suppose $A \in R^{m \times n}$, $x \in R^n$, and $b \in R^m$ and that the least squares solution is required for

$$f(x) = \|Ax - b\|. \tag{2.1}$$

A partition of A can be defined, $A = (A_1, A_2, \ldots, A_r)$ where each A_i is an $m \times n_i$ submatrix of A , and $\sum_{i=1}^{r} n_i = n$ so that

$$f(x) = \|\sum_{i=1}^{r} A_i X_i - b\|$$

where $x = (X_1, X_2, \ldots, X_r)^T$ is partitioned consistently with A. Let $y = (Y_1, Y_2, \ldots, Y_r)^T$, where Y_i is an $n_i \times 1$ submatrix of y, and $Z^{j,k} = (Z_1^{j,k}, Z_2^{j,k}, \ldots, Z_r^{j,k})^T$, where $Z_i^{j,k}$ is an $n_i \times 1$ submatrix of $Z^{j,k}$. For $1 \le j \le r$, define $b_j(x) = b - \sum_{i \ne j}^{r} A_i X_i$, and take positive scalars α_i^k, $\lim_{k \to \infty} \alpha_i^k = \alpha_i > 0$, where k is the iteration number, such that $\sum_{i=1}^{r} \alpha_i^k = 1$. Then the solution to 2.1 can be found in parallel by solving the subproblems

$$minimize \|A_j X_j - b_j(x^k)\|, \qquad 1 \le j \le r.$$

Let B_i^k be an $m \times 1$ vector, for $1 \le i \le r$ and $k \ge 0$, and x^* the solution of 2.1.

ALGORITHM 1. *(Serial Version)*

```
1. $k =: 0$, and guess an $x^k$;
2. do while not converged
    2.1 calculate $B_i^k = A_i X_i^k$, $1 \le i \le r$
    2.2 calculate $b_i(x^k) = b - \sum_{j \ne i}^r B_j^k$, $1 \le i \le r$
    2.3 For $1 \le j \le r$, find $Y_j^{k+1}$ to minimize $\|A_j X_j - b_j(x^k)\|$
    2.4 For $1 \le i \le r$ and $1 \le j \le r$, set
```

$$Z_i^{j,k+1} = \begin{cases} Y_i^{k+1} & \text{if } i = j \\ X_i^k & \text{otherwise} \end{cases}$$

```
    2.5 set $x^{k+1} = \sum_{j=1}^r \alpha_j^{k+1} Z^{j,k+1}$
    2.4 test for convergence
    2.5 $k = k + 1$
   end do
3. $x^* = x^k$
```

Now notice that, for $1 \le i \le r$,

$$\begin{aligned} X_i^{k+1} &= \sum_{j=1}^r \alpha_j^{k+1} Z_i^{j,k+1} \\ &= \alpha_i^{k+1} Y_i^{k+1} + \sum_{j \ne i}^r \alpha_j^{k+1} X_i^k \\ &= \alpha_i^{k+1} Y_i^{k+1} + (\sum_{j \ne i}^r \alpha_j^{k+1}) X_i^k \\ &= \alpha_i^{k+1} Y_i^{k+1} + (1 - \alpha_i^{k+1}) X_i^k. \end{aligned} \tag{2.2}$$

This equality is employed in the parallel version of the algorithm to void communication.

ALGORITHM 2. *(Parallel Version) For all processors i do*

```
Begin
 1. $k =: 0$
 2. guess an $X_i^k$
 3. calculate $B_i^k = A_i X_i^k$
 4. do while not converged
     4.1 get all $B_j^k$ for $j \ne i$
     4.2 calculate $b_i(x^k) = b - \sum_{j \ne i}^r B_j^k$
     4.3 find $Y_i^{k+1}$ to minimize $\|A_i X_i - b_i(x^k)\|$
     4.4 calculate $B_i^{k+1} = \alpha_i^{k+1} A_i Y_i^{k+1} + (1 - \alpha_i^{k+1}) B_i^k$
     4.5 test for convergence
     4.6 $k = k + 1$
    end do
 5. $x^* = y^k = (Y_1^k, Y_2^k, \cdots, Y_r^k)$
End
```

2.2. Nonlinear Minimization. Let f: $R^n \to R^1$ have positive semidefinite Hessian and only one stationary point x^*. Then the problem is to minimize f(x) in the bounded neighbourhood D of x^*.

For $1 \leq j \leq r$, let $\overline{x_j} = (\overline{X_1}^j, \overline{X_2}^j, ..., \overline{X_r}^j)^T$, where, for $1 \leq i \leq r$

$$\overline{X_i}^j = \begin{cases} Y_i & \text{if i = j} \\ X_i & \text{otherwise} \end{cases} . \tag{2.3}$$

Let $f_j(x, Y_j) = f(\overline{x_j})$, then we obtain the following algorithm:

ALGORITHM 3. *(Serial Version)*

```
1. $k =: 0$
2. guess an $x^k$
3. do while not converged
    3.1 For $1 \leq j \leq r$, find $Y_j^{k+1}$ to minimize $f_j(x^k, Y_j)$
    3.2 For $1 \leq i \leq r$ and $1 \leq j \leq r$,
        $Z_i^{j,k+1} = \begin{cases} Y_i^{k+1} & \text{if } i = j \\ X_i^k & \text{otherwise} \end{cases}$
    3.3 $x^{k+1} = \sum_{j=1}^r \alpha_j^{k+1} Z^{j,k+1}$
    3.4 test for convergence
    3.5 $k = k + 1$
   end do
4. $x^* = x^k$
```

By taking advantage of 2.2, we obtain the parallel version of Algorithm 3:

ALGORITHM 4. *(Parallel Version) For all processors i do*

```
Begin
1. $k =: 0$;
2. guess an $X_i^k$
3. do while not converged
    3.1 get all $X_j^k$ for $j \neq i$
    3.2 find $Y_i^{k+1}$ to minimize $f_i(x^k, Y_i)$
    3.3 calculate $X_i^{k+1} = \alpha_i^{k+1} Y_i^{k+1} + (1 - \alpha_i^{k+1}) X_i^k$
    3.4 test for convergence
    3.5 $k = k + 1$
   end do
4. $x^* = y^k$
End
```

3. Theoretical Results.

3.1. Preliminaries.

LEMMA 1. *Let $f : R^n \to R^1$ be a strictly convex function, for $1 \leq j \leq r$, $\alpha_j > 0$, and $\sum_{j=1}^r \alpha_j = 1$. For $x_0 \in R^n$ and $Z_{j,*} \in R^n$ $(1 \leq j \leq r)$, if $f(x_0) = f(Z_{j,*})$ for all j, and $x_0 = \sum_{j=1}^r \alpha_j Z_{j,*}$, then $x_0 = Z_{j,*}$ for all j.*

Proof. In fact, let $S_e = \{j | Z_{j,*} = x_0\}$, then since $x_0 = \sum_{j=1}^r \alpha_j Z_{j,*}$ we have

$$(1 - \sum_{j \in S_e} \alpha_j) x_0 = \sum_{j \notin S_e} \alpha_j Z_{j,*}.$$

Notice that

$$1 - \sum_{j \in S_e} \alpha_j = \sum_{i \notin S_e} \alpha_i$$

so,

$$x_0 = \sum_{j \notin S_c} \frac{\alpha_j}{\sum_{i \notin S_c} \alpha_i} Z_{j,*}$$

thus, by $\alpha_i > 0$ we have

$$\begin{aligned} f(x_0) &= f(\sum_{j \notin S_c} \frac{\alpha_j}{\sum_{i \notin S_c} \alpha_i} Z_{j,*}) \\ &< \sum_{j \notin S_c} \frac{\alpha_j}{\sum_{i \notin S_c} \alpha_i} f(Z_{j,*}) \\ &= \sum_{j \notin S_c} \frac{\alpha_j}{\sum_{i \notin S_c} \alpha_i} f(x_0) \\ &= \frac{\sum_{j \notin S_c} \alpha_j}{\sum_{i \notin S_c} \alpha_i} f(x_0) \\ &= f(x_0), \end{aligned}$$

which is a contradiction. □

LEMMA 2. *Let $\{x_j^k\}$ $(1 \le j \le m)$ be a set of bounded sequences, then there exists an integer sequence $\{k_h\}$, where $k_h \to \infty$ when $h \to \infty$, such that $\{x_j^{k_h}\}$ converges for all j.*

Proof. $\{x_1^k\}$ has a subsequence, say, $\{x_1^{k_{h^1}}\}$ which converges, then $\{x_2^{k_{h^1}}\}$ has a subsequence $\{x_2^{k_{h^2}}\}$ which converges, ..., and finally, $\{x_m^{k_{h^{m-1}}}\}$ has a subsequence $\{x_m^{k_h}\}$ which converges. By the construction, $\{x_j^{k_h}\}$ converges for all j since

$$\{k_{h^1}\} \supset \{k_{h^2}\} \supset \cdots \supset \{k_{h^{m-1}}\} \supset \{k_h\}.$$

□

LEMMA 3. *Let $\{x^k\}, \{Z^{j,k}\}$ $(1 \le j \le r)$ be as in Algorithm 3, then $\{f(x^k)\}$ is a monotonely decreasing sequence, and*

$$\lim_{k\to\infty} f(x^k) = \lim_{k\to\infty} f(Z^{j,k}) = f_0,$$

for $1 \le j \le r$, and for some real number f_0.

Proof. f(x) is a convex function, so

$$\begin{aligned} f(x^{k+1}) &= f(\sum_{j=1}^{r} \alpha_j^{k+1} Z^{j,k+1}) \\ &\le \sum_{j=1}^{r} \alpha_j^{k+1} f(Z^{j,k+1}) \\ &\le \sum_{j=1}^{r} \alpha_j^{k+1} f(x^k) \\ &= f(x^k) \end{aligned}$$

since $f(Z^{j,k+1}) \le f(x^k)$. So $\{f(x^k)\}$ is monotonely decreasing, and thus has limits. Let $f_0 = \lim_{k\to\infty} f(x^k)$, and notice that

$$0 \leq \sum_{j=1}^{r} \alpha_j^{k+1}(f(x^k) - f(Z^{j,k+1})) \leq f(x^k) - f(x^{k+1})$$

so { $f(Z^{j,k})$ } converges for all j and $f_0 = \lim_{k\to\infty} f(Z^{j,k})$ since $lim_{k\to\infty}\alpha_j^k = \alpha_j > 0$. □

Finally, we quote the following theorem without proof.

THEOREM 1. *[1] If f is a convex function defined on the convex set C and attains its maximum over C at an interior point x^0 of C, then f is constant on C.*

3.2. Convergence Results. Let f'_{Y_j} be the Jacobi with respect to the subvector Y_j of y.

THEOREM 2. *The f_0 in LEMMA 3 is the minimum value of f.*

Proof. By LEMMA 3, $f_0 = \lim_{k\to\infty} f(x^k) = \lim_{k\to\infty} f(Z^{j,k})$, for all j. Notice that $\{Z^{j,k}\}$ is bounded, so, by LEMMA 2, has a subsequence $\{Z^{j,k_h}\}$, where $k_h \to \infty$, when $h \to \infty$, such that $Z_{j,*} = \lim_{h\to\infty} Z^{j,k_h}$, $1 \leq j \leq r$, for some $Z_{j,*}$. Since f(x) is continuous $f_0 = \lim_{h\to\infty} f(Z^{j,k_h}) = f(Z_{j,*})$, for all j. By the definition of x^{k_h} in step 3.3 of Algorithm 3,

$$\begin{aligned}
\sum_{j=1}^{r} \alpha_j Z_{j,*} &= \sum_{j=1}^{r} \lim_{h\to\infty} (\alpha_j^{k_h} Z^{j,k_h}) \\
&= \lim_{h\to\infty} \sum_{j=1}^{r} \alpha_j^{k_h} Z^{j,k_h} \\
&= \lim_{h\to\infty} x^{k_h} \\
&= x_0
\end{aligned}$$

for some x_0, and

$$f_0 = \lim_{h\to\infty} f(x^{k_h}) = f(x_0).$$

Case 1, f is strictly convex.

By LEMMA 1 $x_0 = Z_{j,*}$ for all j. And since $f'_{Y_j}(Z^{j,k_h}) = 0$ for all h, we have

$$f'_{Y_j}(Z_{j,*}) = \lim_{h\to\infty} f'_{Y_j}(Z^{j,k_h}) = 0. \tag{3.1}$$

Therefore $f'_{Y_j}(x_0) = f'_{Y_j}(Z_{j,*}) = 0$, i.e., $\nabla f(x_0) = 0$, and thus x_0 minimizes f(x), and f_0 is the minimum value of f.

Case 2, f convex.

Let C be the convex hull of $\{Z_{j,*}\}$, and $Z_{j,*} \neq x_0$ for all j, then x_0 is in the interior of C since $\alpha_j > 0$ $(1 \leq j \leq r)$. Furthermore for $x \in C$ there are $\lambda_j > 0$ $(1 \leq j \leq r)$ such that

$$x = \sum_{j=1}^{r} \lambda_j Z_{j,*}, \qquad \textit{and} \qquad \sum_{j=1}^{r} \lambda_j = 1.$$

Since f(x) is a convex function,

$$f(x) = f(\sum_{j=1}^{r} \lambda_j Z_{j,*})$$

$$\begin{aligned}
&\leq \sum_{j=1}^{r} \lambda_j f(Z^{j,*}) \\
&= \sum_{j=1}^{r} \lambda_j f_0 \\
&= f_0.
\end{aligned}$$

But then f attains its maximum at an interior point of C and is therefore constant on C by THEOREM 1, so $\nabla f(x_0) = 0$, and thus x_0 minimizes f(x), and f_0 is the minimum value of f. When $Z^{j,*} = x_0$ for some j the proof follows in a similar manner. □

For strictly convex functions we have

THEOREM 3. *If f is strictly convex then the sequences $\{x^k\}, \{Z^{j,k}\} (1 \leq j \leq r)$ in Algorithm 3, and hence Algorihtm 4 , all converge to the minimum point of f.*

Proof. By LEMMA 3, and THEOREM 2,

$$f_0 = \lim_{k \to \infty} f(x^k) = \lim_{k \to \infty} f(Z^{j,k})$$

for all j. Now we prove that $\{x^k\}$ converges. Let $D^k = \{y | f(y) \leq f(x^k)\}$, then obviously $D^{k+1} \subset D^k$, and $x_k \in D^k$. Since the minimum point is unique we have $\bigcap_{k=1}^{\infty} D^k = \{x_0\}$. Therefore $x_0 = \lim_{k \to \infty} x^k$. Similarly, $Z^{j,k+1} \in D^k$, and thus $x_0 = \lim_{k \to \infty} Z^{j,k}$, for $1 \leq j \leq r$. □

Observations:

1. We may solve the problem by using the known results for the systems of nonlinear equations on the function ∇f, but if we solve it directly we have a wider choice of the methods, such as, Descent, Conjugate-Direction, Gauss-Newton, and Davidon-Fletcher-Powell Methods;

2. The choice of methods used on local processors may be crucial as is the case in the Jacobi Block method for the eigenvalue problem for symmetric matrices;

3. Our experiments show that, when choosing x^{k+1}, if we add the following step, then the performance is improved significantly:
Let $y^{k+1} = (Y_1^{k+1}, Y_2^{k+1}, \cdots, Y_r^{k+1})$, if $f(y^{k+1}) < f(x^{k+1})$, then set $x^{k+1} = y^{k+1}$.
The proof of convergence for this variation of our algorithm is similar. Our timing results are for the algorithms with this improvement.

4. Convergence Checking. Given $\varepsilon > 0$, we can check either $\|x^{k+1} - x^k\| < \varepsilon$, or $\|f'(x^k)\| < \varepsilon$, or $\|f(x^{k+1}) - f(x^k)\| < \varepsilon$.

In parallel, for all j, processor j checks either $\|X_j^{k+1} - X_j^k\| < \varepsilon$, or $\|f'_{Y_j}(x^k)\| < \varepsilon$, since we have $\|x^{k+1} - x^k\| \leq \sum_{j=1}^{r} \|X_j^{k+1} - X_j^k\|$, and a similar inequality holds for $\|f'(x^k)\|$. We test $\|f(x^{k+1}) - f(x^k)\| < \varepsilon$ in our programs on the Hyper Cube since, from the above Observation 3, $f(x^k)$ is always available.

5. Numerical Results. For all our tests we choose $\alpha_i^k = \alpha = 1/r$, $k \geq 1$. $1 \leq i \leq r$, where r is the number of splittings. Since the splittings are also non-overlapping, each subproblem has the same size. If ε is the machine precision then the tolerance is set to $\sqrt{\varepsilon}$. The MFLOPS for our linear least squares algorithms is $itr * n * (3n + 2n^2/r - 31n/6r - 3/2)/t$, where the matrix has size $n \times n$, r is the number of splittings, t is the time in second, and itr is the number of iterations. For nonlinear algorithms the MFLOPS depends on the function chosen.

Table 5.1 presents timings for Algorithm 1, using Householder QR decomposition [9] for the subproblems, on one node. In the first row is the timing without splitting.

The test problem is an $m \times m$ nonsingular linear least squares problem, where m = 1024. The problem is test problem 1 from [3], for which

$$A_{ij} = \begin{cases} 1 - 2/m & \text{if } i = j \\ -2/m & \text{otherwise} \end{cases} .$$

Iteration starts with $x_i = 0.0$ for $1 \leq i \leq n$. The converged solution is $x_i = -1.0$ for $1 \leq i \leq n$.

TABLE 5.1

Timing on one node with splittings using Householder QR decompostion for all the subproblems.

# of splitting	α	# of iteration	timing (sec)	MFLOPS
1	N/A	1	741.56	2.892834
2	1/2	3	416.132	7.744013
4	1/4	2	213.168	5.053860
8	1/8	3	121.618	6.682460
16	1/16	3	74.183	5.541246
32	1/32	3	53.86	3.903662
64	1/64	3	47.652	2.305085
128	1/128	3	63.491	0.939302
256	1/256	3	177.632	0.194418

Except for the cases of splitting into 1024 and 512 subproblems, when the machine ran out of memory, the program is faster than without splitting and in the case for splitting into 64 subproblems it is 15.56 times faster than without splitting. Hence there is an optimal splitting. The test results of Algorithm 3, using the variable metric method [4] on the subproblems, on one node have similar behaviour.

Table 5.2 is the timing of Algorithm 2 for the same problem on the Hypercube. The speedup increases with problem size.

TABLE 5.2

Timing on 64 node hypercube for linear least squares using Householder QR decomposition for all the subproblems. The problem size is $n \times n$, r is the # of splittings which equals the # of processors used, itr is the # of iterations, t is the elapsed time in second, c is the communication time, cp is the percentage of the total communication time v.s. total run time, the speedup sp is with respect to the timing on one node.

r	α	itr	n	t	sp	c	cp %	MFLOPS
1	N/A	1	512	79.336	1	0	0	3.376358
4	1/4	2	512	7.165	11.072714	0.8	11.165388	18.857202
8	1/8	3	512	3.194	24.839073	0.10	3.130870	32.095298
16	1/16	3	512	1.489	53.281397	0.11	7.387509	35.214700
32	1/32	4	512	1.526	51.989513	0.18	11.795544	23.936952
64	1/64	2	512	2.675	29.658318	0.93	34.76635	5.561277
1	N/A	1	1024	741.56	1	0	0	2.892834
4	1/4	2	1024	56.126	13.212415	0.24	0.427609	19.194694
8	1/8	3	1024	16.476	45.008498	0.45	2.731245	49.326740
16	1/16	3	1024	5.494	134.976338	0.48	8.736804	74.821616
32	1/32	3	1024	3.326	222.958510	0.57	17.137703	63.214452
64	1/64	3	1024	3.59	206.562678	0.117	3.259053	30.596636

Table 5.3 presents timings for Algorithm 4 applied to the minimization of the following function [3]:

$$f(x_1, x_2, \cdots, x_n) = \sum_{i=1}^{n} (e^{i*x_i/n} - e^{-i^2/n})^2.$$

Initial values are $x_i = -(i + 0.1), (1 \leq i \leq n)$, and the real solution has $x_i = -i, (1 \leq i \leq n)$. We use the variable metric method with BFGS updating formula [4] on all subproblems. From the table we can see that the speedup is greater than the number of processors used. One reason for this is partly due to cache, or memory size. When the problem is split onto a number of processors they are nearly all in the main memory, or cache, but if it is run on one processor then the main memory is not enough for holding a big problem and so at run time doing expensive swapping takes up a large proportion of the time. But, more importantly, the speedup does show the potential of our algorithms for parallel machines.

TABLE 5.3

Timing on 64 node hypercube for nonlinear minimization using Variable Metric Method, with BFGS updating Formula, for the subproblems. The problem size is n, r is the # of splittings which equals the # of processors used, itr is the # of iterations, t is the elapsed time in second, c is the communication time, cp is the percentage of the total communication time v.s. total run time, the speedup sp is with respect to the timing on one node.

r	α	itr	n	t	sp	c	cp %
1	N/A	1	512	209.23	1	0	0
4	1/4	1	512	2.363	88.579995	0.66	27.93060
8	1/8	1	512	2.76	75.804349	0.38	13.768116
16	1/16	2	512	1.659	126.112153	0.54	3.254973
32	1/32	7	512	2.682	78.008951	0.127	4.735272
64	1/64	5	512	2.688	77.834823	0.92	34.226192
1	N/A	1	1024	1442.475	1	0	0
4	1/4	1	1024	49.261	29.282291	0.111	0.225330
8	1/8	1	1024	2.261	637.980994	0.80	3.538257
16	1/16	2	1024	2.446	589.728094	0.116	4.74236
32	1/32	3	1024	2.499	577.220862	0.147	5.882353
64	1/64	2	1024	2.789	517.201491	0.97	34.779491

6. Current and Future Work. Currently we have generalized the results to the singular case, and for overlapped splitting, e.g., $Y_i^k \cap Y_j^k \neq \phi$ for $i \neq j$. Further the splitting can be dynamically changed, e.g., we can add or delete components from Y_i^k at any iteration. Furthermore we are testing the algorithms with an inexact subspace search, by which we mean that we do not find the minimum for the subproblem in each iteration, as in the inexact line search algorithm. We are also testing the effect on the rate of convergence of the choice of methods for the subproblems. Moreover we need theoretical results for the convergence rate. Notice that our speedup is with respect to the timing on one node using the algorithm without splitting. But if, for example, in Table 5.2, for $n = 1024$, we use the timing on one node using the algorithm with optimal splitting, e.g., in Table 5.1, $r = 64$, then the speedup is not so great. However, we have not been able to prove that there is always an optimal splitting for Algorithm 1, or Algorithm 3, such that the algorithm converges much faster than the one without splitting, even though our test results, so far, support this. More details can be found in [10].

7. Acknowledgements. This result would not have been possible without my advisor, Dr. Rosemary Renaut, introducing to me the idea of multisplitting for the

system of equations in the first place, and her guidance throughout the whole process of theoretical and experimental work.

REFERENCES

[1] Mordecai Avriel, Walter E. Diewert, Siegfried Schaible, and Israel Zang, *Generalized Concavity*, Plenum Press, New York, 1988.

[2] Thomas F. Coleman and Guangye Li, *Solving Systems of Nonlinear Equations on A Message-Passing Multiprocessor*, SIAM J. SCI. STAT. COMPUT. Vol. 11, No. 6, pp. 1116-1135, November 1990.

[3] Thomas F. Coleman and Paul E. Plassmann, *A Parallel Nonlinear Least-Sugares Solver: Theoretical Analysis and Numerical Results*, SIAM J. SCI. STAT. COMPUT. Vol. 13, No. 3, pp. 771-793, May 1992.

[4] R. Fletcher, *Practical Methods of Optimization*, A Wiley-Interscience Publication.

[5] A. Frommer, *Parallel Nonlinear Multisplitting Methods*, Numer. Math., 56 (1989), pp. 269-282.

[6] ———, *Lösung linearer Gleichungssysteme auf Parallelrechnern*, View Verlag, Braunschweig, Kapitel 9.

[7] Andreas Frommer, and Günter Mayer, *Convergence of Relaxed Parallel Multisplitting Methods*, Linear Algebra Appl. 119(1989), pp.141-152.

[8] ———, *Parallel interval multisplittings*, Numer. Math., Math. 56(89), pp. 255-267.

[9] Gene H. Golub, Charles F. Van Loan, *Matrix Computations*, Second Edition, The Joins Hopkins University Press.

[10] Qing He, *New Parallel Algorithms for Unconstrained Optimizations by Multisplitting*, Ph.D thesis in praparation.

[11] Chiou-Ming Huang and Dianne P. O'Leary, *A Krylov Multisplitting Algorithm for Solving Linear Systems of Equations*, Linear Algebra Appl. 194(1993), pp. 9.

[12] ———, *A Parallel Inexact Newton Method Using A Krylov Multisplitting Algorithm*, Computer Science Department Report CS-TR-3112, Institute for Advanced Computer Studies Report UMIACS-93-71, University of Maryland, July 1993.

[13] M. Neumann, and R. J. Plemmon, *Convergence of Parallel Multisplitting Iterative Methods for M-Matrices*, Linear Algebra Appl. 88-89(1987), pp. 559-574.

[14] Dianne P. O'Leary, and R. E. White, *Multi-Splitting of Matrices and Parallel Solution of Linear Systems*, SIAM J. Alg. Disc. Meth., 6(1985), pp. 630-640.

[15] Bert Pohl, *Ein Algorithms zur Lösung von Anfangswertproblemen auf Parallelrechnern*, Informatik-Dissertationen ETH Zürich NR. 37.

[16] Hans Schneider, *Theorems on M-Splittings of a Singular M-Matrix Which Depend on Graph Structure*, Linear Algebra Appl. 58(1984), pp. 407-424.

[17] Daniel B. Szyld, and Mark T. Jones, *Two-Stage and Multisplitting Methods for the Parallel Solution of Linear Systems*, SIAM J. Matrix Anal. Appl. Vol.13, No. 2, pp. 671-679, April 1992.

[18] Wang Deren, *On the Convergence of the Parallel Multisplitting AOR Algorithm*, Linear Algebra Appl. 154-156(1991), pp. 473-486.

[19] R. E. White, *Parallel Algorithms for Nonlinear Problems*, SIAM J. Alg. Disc. Meth., 7(1986), pp. 137-149.

[20] ———, *A Nonlinear Parallel Algorithm with Application to the Stefan Problem*, SIAM J. Numer. Anal., 23(1986), pp. 639-652.

[21] ———, *Multisplitting with Different Weighting Schemes*, SIAM J. Matrix Anal. Appl. Vol. 10, No. 4, pp. 481-493, October 1989.

[22] ———, *Multisplitting of a Symmetric Positive Definite Matrix*, SIAM J. Matrix Anal. Appl. Vol. 11, No. 1, pp. 69-82, January 1990.

Analysis of Semi-Toeplitz Preconditioners for First-order PDEs[*]

Lina Hemmingsson[†]

December 8, 1993

Abstract

A semi-Toeplitz preconditioner for nonsymmetric, nondiagonally dominant systems of equations is studied. The preconditioner solve is based on a Fast Modified Sine Transform. As a model problem we study a system of equations arising from an implicit time-discretization of a scalar hyperbolic PDE. Analytical formulas for the eigenvalues of the preconditioned system are derived. The convergence of a minimal residual iteration is shown to be dependent *only* on the grid ratio in space and *not* on the number of unknowns.

1 Introduction

We are interested in solving systems of first-order PDEs such as the Euler equations and the Maxwell equations. For the Euler equations, we are mainly interested in the case when there are different time-scales present in the problem and the fastest time-scale does not have to be resolved. An important application is almost incompressible flow [4], [5], where the sound waves in the medium are much faster than the motion of the fluid. For an explicit discretization in time the restriction on the time-step due to the CFL-criterion becomes too severe. For the Maxwell equations, we intend to consider complicated geometries such that the smallest grid-cell in the space-discretization restricts the time-step considerably in an explicit time-discretization. Therefore we use an implicit time-discretization. This leads to a system of equations

$$Au = b \tag{1}$$

which has to be solved for each time-step. A is nonsymmetric, and since the time-step is large compared to the space-step, it is also strongly nondiagonally dominant.
In this report we theoretically examine the convergence properties when we apply a minimal residual iteration to (1), utilizing a semi-Toeplitz preconditioner. Such preconditioners have shown to be well suited in a domain decomposition setting of the problem [8].

2 The model problem

2.1 The differential equation

In [7] we consider a scalar two-dimensional equation with variable coefficients. Here we will restrict our presentation to constant coefficients in order to be able to derive theoretical properties of the preconditioned system.

$$u_t + u_{x_1} + u_{x_2} = g \quad , \quad 0 < x_1 \leq 1 \quad , \quad 0 < x_2 \leq 1 \quad , \quad t > 0 \ . \tag{2}$$

(2) is well posed if we prescribe $u(x_1, 0, t)$, $u(0, x_2, t)$, and $u(x_1, x_2, 0)$.

[*]This work was supported by the Swedish National Board for Industrial and Technical Development, NUTEK, under contract No. 89-02539 P

[†]Department of Scientific Computing, Uppsala University, Box 120, S-751 04 Uppsala, Sweden

2.2 Discretization

Introduce a uniform grid as

$$x_{d,j} = jh_d \quad , \quad j = 0,\dots,m_d \quad , \quad h_d = 1/m_d \quad , \quad d = 1,2 \ .$$

Let $u_{j,k}$ denote the approximate solution at the point $(x_{1,j}, x_{2,k})$. Now we discretize equation (2) in time using the trapezoidal rule with time-step Δt

$$\frac{u^{n+1}-u^n}{\Delta t} + \frac{1}{2}\sum_{d=1}^{2}((\frac{\partial u}{\partial x_d})^{n+1} + (\frac{\partial u}{\partial x_d})^n) = \frac{1}{2}(g^{n+1}+g^n). \tag{3}$$

For the space derivatives we use second-order centered differences in the interior of the domain and one-sided first-order differences at the outflow boundaries. We define the following quantities

$$\kappa_d = \frac{\Delta t}{h_d} \quad , \quad d = 1,2 \ ,$$

$$u^n = (\, u^n_{1,1} \quad u^n_{2,1} \quad \cdots \quad u^n_{m_1,1} \quad u^n_{1,2} \quad \cdots \quad u^n_{m_1,m_2})^T ,$$

and finally write the equations as

$$Au^{n+1} \equiv (I_{m_2} \otimes A_1 + \kappa_2 A_2 \otimes I_{m_1})u^{n+1} = b \tag{4}$$

where b contains known quantities and A_1 and A_2 are defined by

$$A_1 = \begin{pmatrix} 4 & \kappa_1 & & & \\ -\kappa_1 & 4 & \kappa_1 & & \\ & \ddots & \ddots & \ddots & \\ & & -\kappa_1 & 4 & \kappa_1 \\ & & & -2\kappa_1 & 4+2\kappa_1 \end{pmatrix} \quad , \quad A_2 = \begin{pmatrix} 0 & 1 & & & \\ -1 & 0 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 0 & 1 \\ & & & -2 & 2 \end{pmatrix}. \tag{5}$$

3 Iterative methods

The matrix A defined in (4) is extremely sparse, with five nonzero diagonals and semi-bandwidth m_1. Modern CG-like iterative methods are well suited for nonsymmetric systems of equations. In this report we consider minimal residual iterations, [3], which in each iteration fulfills

$$||r^{(i)}||_2 = \min_{p_i \in \mathcal{P}_i, p_i(0)=1} ||p_i(M^{-1}A)r^{(0)}||_2$$

where $\mathcal{P}_i$ is the set of all polynomials of degree i, M is the left preconditioner, $r^{(i)} = M^{-1}(Au^{(i)} - b)$ and $u^{(i)}$ is the approximation of u obtained in iteration i. If $M^{-1}A$ is diagonalizable we obtain

$$\frac{||r^{(i)}||_2}{||r^{(0)}||_2} \leq \text{cond}_2(W_{M^{-1}A}) \min_{p_i \in \mathcal{P}_i, p_i(0)=1} \max_{1\leq \ell \leq n} |p_i(\lambda_\ell)| \equiv \text{cond}_2(W_{M^{-1}A}) \cdot \varepsilon_i \tag{6}$$

where $W_{M^{-1}A}$ is the eigenvector matrix and λ_ℓ are the eigenvalues of $M^{-1}A$. From (6) we conclude that we will have finite termination in n iterations where n is the number of distinct eigenvalues to the preconditioned system. Moreover, if we can precondition our system such that the eigenvalues of $M^{-1}A$ are contained in k dense clusters we have a good approximation to the solution in k iterations. Clustering of the eigenvalues may be even more important than a condition number improvement, [1], [2], and [12]. In Section 4 we define a preconditioner that yields a highly clustered spectrum. Since the iterative method includes the solution of

$$Mx = y, \tag{7}$$

in each step we must have a fast solver for this system. The preconditioner solve defined in this report is based on a Fast Modified Sine Transform developed in [6].

4 The preconditioner

In this report we consider the semi-Toeplitz preconditioner M defined by

$$M = I_{m_2} \otimes \hat{A}_1 + \kappa_2 A_2 \otimes I_{m_1} \quad , \quad \hat{A}_1 = \begin{pmatrix} 4 & \kappa_1 & & \\ -\kappa_1 & \ddots & \ddots & \\ & \ddots & \ddots & \kappa_1 \\ & & -\kappa_1 & 4 \end{pmatrix} \tag{8}$$

In [7] it is shown that M can be decomposed as

$$M = (I_{m_2} \otimes S_{m_1}) T (I_{m_2} \otimes S_{m_1}^H) \tag{9}$$

where T is block-tridiagonal with diagonal blocks as

$$T = I_{m_2} \otimes \Lambda_1 + \kappa_2 A_2 \otimes I_{m_1} \tag{10}$$

where

$$\Lambda_1 = \text{diag}(\lambda_{1,1}, \ldots, \lambda_{1,m_1}) \quad , \quad \lambda_{1,j} = 4 + 2i\kappa_1 \cos\left(\frac{j\pi}{m_1+1}\right) \tag{11}$$

S_{m_1} is the modified sine matrix defined by

$$S_{m_1}(j,k) = \sqrt{\frac{2}{m_1+1}}\, i^{j+k+1} \sin\left(\frac{jk\pi}{m_1+1}\right) \quad j,k = 1, \ldots, m_1 \tag{12}$$

By rearranging the unknowns the solution of

$$Tx = y,$$

decouples into m_1 independent tridiagonal systems of order m_2. In [6] it is shown how the modified sine transforms can be evaluated using Fourier transforms of vectors $y \in C^{\frac{m_1+1}{2}}$. By symmetry we can also reduce the intermediate solution of the tridiagonal systems to the solution of only the first $\frac{m_1+1}{2}$ systems.
To sum up this section we present the different steps in the solution of (7):

- m_2 Fourier transforms of vectors $y \in C^{\frac{m_1+1}{2}}$
- $\frac{m_1+1}{2}$ solutions of tridiagonal systems of order m_2
- m_2 Fourier transforms of vectors $y \in C^{\frac{m_1+1}{2}}$

5 Spectrum of the preconditioned system

By defining

$$E = A - M = I_{m_2} \otimes E_1$$

where E_1 is defined by

$$E_1 = \begin{pmatrix} 0 & \cdots & 0 & 0 \\ \vdots & & \vdots & \vdots \\ 0 & \cdots & 0 & 0 \\ 0 & \cdots & -\kappa_1 & 2\kappa_1 \end{pmatrix}, \tag{13}$$

we get

$$M^{-1}A = I + M^{-1}(I_{m_2} \otimes E_1) \tag{14}$$

Otto [9] shows how to compute the eigenvalues of (14) when M is a semicirculant preconditioner, i.e., each block is circulant. We use the same technique to determine the eigenvalues of (14). We will make use of a lemma in [9] which we restate here for readability. Let $\mathcal{E}(a,b)$ denote the closed ellipse centered at the origin with semimajor axis b oriented along the imaginary axis and semiminor axis a. Also let $\mathcal{E}^+(a,b)$ denote the region $\{z | z \in \mathcal{E}(a,b) \text{ and } \Re e(z) \geq 0\}$.

Lemma 5.1 *The eigenvalues $\lambda_{2,k}$, $k = 1,\dots,m_2$, of A_2 satisfy:*

- $\lambda_{2,k} \neq \lambda_{2,j}$, $k \neq j$
- $\lambda_{2,k} \in \mathcal{E}^+(4m_2^{-3/4}, 2 + 4m_2^{-3/2})$

From Lemma 5.1 we conclude that there exists a nonsingular matrix V such that

$$V^{-1}A_2V = \Lambda_2 = \operatorname{diag}(\lambda_{2,1},\dots,\lambda_{2,m_2})$$

By defining

$$D = (V^{-1} \otimes S_{m_1}^H)M(V \otimes S_{m_1}) \tag{15}$$

where S_{m_1} is the modified sine matrix of order m_1 defined in (12) we get

$$\begin{aligned} D &= (V^{-1} \otimes S_{m_1}^H)(I_{m_2} \otimes \hat{A}_1 + \kappa_2 A_2 \otimes I_{m_1})(V \otimes S_{m_1}) = \\ &= I_{m_2} \otimes S_{m_1}^H \hat{A}_1 S_{m_1} + \kappa_2 V^{-1}A_2V \otimes I_{m_1} = I_{m_2} \otimes \Lambda_1 + \kappa_2\Lambda_2 \otimes I_{m_1} \end{aligned}$$

where Λ_1 is defined in (11). From Lemma 5.1 we get that

$$\Re e(\lambda_{1,j} + \kappa_2\lambda_{2,k}) = 4 + \kappa_2 \Re e(\lambda_{2,k}) \geq 4$$

which implies that $I_{m_2} \otimes \Lambda_1 + \kappa_2\Lambda_2 \otimes I_{m_1}$ is nonsingular, i.e., M^{-1} exists. Hence

$$D = I_{m_2} \otimes \Lambda_1 + \kappa_2\Lambda_2 \otimes I_{m_1} = \operatorname{diag}(D_1,\dots,D_{m_2})$$

where

$$D_k = \operatorname{diag}(d_\nu) \quad , \quad d_\nu = 4 + 2i\kappa_1 \cos\left(\tfrac{j\pi}{m_1+1}\right) + \kappa_2\lambda_{2,k} \quad , \quad \nu{=}(k-1)m_1+j \quad , \quad k{=}1,\dots,m_2 \quad , \quad j{=}1,\dots,m_1 \ , \tag{16}$$

and the spectral decomposition of M^{-1} becomes

$$M^{-1} = (V \otimes S_{m_1})D^{-1}(V^{-1} \otimes S_{m_1}^H).$$

Now consider the matrix W given by

$$\begin{aligned} W &= (V \otimes I_{m_1})^{-1}M^{-1}E(V \otimes I_{m_1}) = (I_{m_2} \otimes S_{m_1})\operatorname{diag}(D_1^{-1},\dots,D_{m_2}^{-1})(I_{m_2} \otimes S_{m_1}^H E_1) = \\ &= \operatorname{diag}(S_{m_1}D_1^{-1}S_{m_1}^H E_1,\dots,S_{m_1}D_{m_2}^{-1}S_{m_1}^H E_1) \equiv \operatorname{diag}(W_1,\dots,W_{m_2}) \end{aligned}$$

As W is a similarity transformation of $M^{-1}E$ and W is block-diagonal, the eigenvalues of $M^{-1}E$ equal the eigenvalues of W_k , $k = 1,\dots,m_2$. Due to the sparse structure of E_1, W_k only have 2 nonzero columns

$$W_k = \begin{pmatrix} 0 & \cdots & 0 & W_k(1,m_1-1) & W_k(1,m_1) \\ 0 & \cdots & 0 & W_k(2,m_1-1) & W_k(2,m_1) \\ \vdots & & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & W_k(m_1,m_1-1) & W_k(m_1,m_1) \end{pmatrix} \tag{17}$$

where

$$\left.\begin{aligned} W_k(\ell,m_1-1) &= -\kappa_1 \textstyle\sum_{j=1}^{m_1} S_{m_1}(\ell,j)d_\nu^{-1}S_{m_1}^H(j,m_1) = -\kappa_1 i^{\ell-m_1}\Phi_{m_1,\ell}(\frac{4+\kappa_2\lambda_{2,k}}{2},\kappa_1) \\ W_k(\ell,m_1) &= 2\kappa_1 \textstyle\sum_{j=1}^{m_1} S_{m_1}(\ell,j)d_\nu^{-1}S_{m_1}^H(j,m_1) = 2\kappa_1 i^{\ell-m_1}\Phi_{m_1,\ell}(\frac{4+\kappa_2\lambda_{2,k}}{2},\kappa_1) \end{aligned}\right. \quad , \quad \ell = 1,\dots,m_1 \tag{18}$$

where $\nu{=}(k-1)m_1+j$, d_ν is defined in (16) and

$$\Phi_{k,\ell}(\alpha,\beta) = \frac{2}{m_1+1}\sum_{j=1}^{m_1} F_{k,\ell}\left(\frac{j\pi}{m_1+1}\right). \tag{19}$$

where

$$F_{k,\ell}(\theta) = \tfrac{\sin(k\theta)\sin(\ell\theta)}{2\alpha+2i\beta\cos(\theta)} \quad , \quad k,\ell = 1,\dots,m_1 \ , \tag{20}$$

where α is a complex number with $\Re e\ \alpha > 0$, $\beta > 0$ is real and $0 \leq \theta \leq \pi$. We now show how to evaluate the sums in (19) using the technique presented in [9]. Using this result we can then compute the sums defined in (18).

Theorem 5.2 *$\Phi_{k,\ell}(\alpha,\beta)$ defined in (19) can be evaluated using residue calculus, yielding*

$$\Phi_{k,\ell}(\alpha,\beta) = \frac{i}{\beta}\left(\frac{z^{k+\ell} - z^{|k-\ell|}}{z - z^{-1}} + \frac{(z^{-k} - z^{k})(z^{-\ell} - z^{\ell})}{z - z^{-1}}\frac{z^{2m_1+2}}{1 - z^{2m_1+2}}\right)$$

where

$$z = i\left(\frac{\alpha}{\beta} - \sqrt{1 + \left(\frac{\alpha}{\beta}\right)^2}\right)$$

and the principal branch of the square root function is employed.

Proof:
By periodicity we get

$$\Phi_{k,\ell}(\alpha,\beta) = \frac{1}{m_1+1}\sum_{j=0}^{2m_1+1} F_{k,\ell}\left(\frac{j\pi}{m_1+1}\right) \quad , \quad k,\ell = 1,\ldots,m_1$$

$F_{k,\ell}$ is C^1 and 2π-periodic $\Rightarrow$

$$F_{k,\ell}(\theta) = \sum_{q=-\infty}^{\infty} c_q e^{iq\theta} \quad , \quad c_q = \frac{1}{2\pi}\int_0^{2\pi} e^{-iq\theta}F_{k,\ell}(\theta)d\theta$$

The Poisson summation formula yields

$$\Phi_{k,\ell}(\alpha,\beta) = \sum_{q=-\infty}^{\infty} 2c_{q(2m_1+2)} = 2c_0 + \sum_{q=1}^{\infty}\left(2c_{q(2m_1+2)} + 2c_{-q(2m_1+2)}\right)$$

Using the Euler identities in (20) gives

$$F_{k,\ell}(\theta) = -\frac{1}{4}\frac{(e^{ik\theta} - e^{-ik\theta})(e^{i\ell\theta} - e^{-i\ell\theta})}{2\alpha + i\beta(e^{i\theta} + e^{-i\theta})} = \frac{e^{i(k-\ell)\theta} + e^{i(\ell-k)\theta} - e^{i(k+\ell)\theta} - e^{-i(k+\ell)\theta}}{8\alpha + 4i\beta(e^{i\theta} + e^{-i\theta})}$$

By making the substitution $z = e^{i\theta}$ we get

$$2c_q = -\frac{i}{2\beta}\frac{1}{2\pi i}\oint_C \frac{z^{k-\ell-q} + z^{\ell-k-q} - z^{k+\ell-q} - z^{-\ell-k-q}}{(z-z_1)(z-z_2)}dz \quad , \quad \forall q \tag{21}$$

where C is the positively oriented unit circle and

$$z_1 = i\left(\frac{\alpha}{\beta} - \sqrt{1 + \left(\frac{\alpha}{\beta}\right)^2}\right) \quad , \quad z_2 = i\left(\frac{\alpha}{\beta} + \sqrt{1 + \left(\frac{\alpha}{\beta}\right)^2}\right) \tag{22}$$

where $|z_1| < 1$ and $|z_2| > 1$. Hence z_1 is within C while z_2 is not. Note that $z_2 = z_1^{-1}$.
Two types of integrals arise which are calculated using the residue theorem.

- $p \geq 0$ $\; I_1 = \frac{1}{2\pi i}\oint_C \frac{z^p}{(z-z_1)(z-z_2)}dz = \frac{z_1^p}{z_1 - z_2} = \frac{z_1^p}{z_1 - z_1^{-1}}$
- $p \geq 1$ $\; I_2 = \frac{1}{2\pi i}\oint_C \frac{z^{-p}}{(z-z_1)(z-z_2)}dz = \frac{z_1^{-p}}{z_1 - z_2} + \frac{1}{(p-1)!}\varphi^{(p-1)}(0)$

  $\varphi^{(p-1)}(0)$ denotes the $(p-1)$th derivative of $\varphi(z)$ at $z = 0$, where $\varphi(z)$ is defined by

  $\varphi(z) = \frac{1}{(z-z_1)(z-z_2)} = \frac{1}{z_1 - z_2}((z - z_1)^{-1} - (z - z_2)^{-1})$ yielding

  $\varphi^{(p-1)}(z) = \frac{1}{z_1 - z_2}(p-1)!(-1)^{p-1}((z - z_1)^{-p} - (z - z_2)^{-p})$, and hence

  $I_2 = \frac{z_1^{-p}}{z_1 - z_1^{-1}} + \frac{(-1)^{p-1}}{z_1 - z_1^{-1}}((-z_1)^{-p} - (-z_2)^{-p}) = \frac{z_1^p}{z_1 - z_1^{-1}}.$

Using this result in (21) yields

$$\begin{cases} c_0 = \frac{i}{2\beta}\left(\frac{z_1^{k+\ell}-z_1^{|k-\ell|}}{z_1-z_1^{-1}}\right) \\ c_{q(2m_1+2)} = -\frac{iz_1^{q(2m_1+2)}}{4\beta}\left(\frac{z_1^{\ell-k}+z_1^{k-\ell}-z_1^{-k-\ell}-z_1^{k+\ell}}{z_1-z_1^{-1}}\right) \\ c_{-q(2m_1+2)} = c_{q(2m_1+2)} \end{cases} \quad , \quad k,\ell = 1,\ldots,m_1$$

Since $|z_1| < 1$ we get

$$\sum_{q=1}^{\infty}(z_1^{2m_1+2})^q = \frac{z_1^{2m_1+2}}{1-z_1^{2m_1+2}}$$

which gives

$$\sum_{q=1}^{\infty}\left(2c_{q(2m_1+2)} + 2c_{-q(2m_1+2)}\right) = -\frac{i}{\beta}\frac{1}{z_1-z_1^{-1}}\left(z_1^{k-\ell}+z_1^{\ell-k}-z_1^{-k-\ell}-z_1^{k+\ell}\right)\frac{z_1^{2m_1+2}}{1-z_1^{2m_1+2}}$$

Summing up we conclude that

$$\Phi_{k,\ell}(\alpha,\beta) = \frac{i}{\beta}\left(\frac{z_1^{k+\ell}-z_1^{|k-\ell|}}{z_1-z_1^{-1}} + \frac{(z_1^{-k}-z_1^{k})(z_1^{-\ell}-z_1^{\ell})}{z_1-z_1^{-1}}\frac{z_1^{2m_1+2}}{1-z_1^{2m_1+2}}\right)$$

where z_1 is defined in (22) which completes the proof. $\square$

Thus, by Theorem 5.2 and using that $\Re e(4+\kappa_2\lambda_{2,k}) > 0$ and $\kappa_1 > 0$ we can find analytical expressions for $W_k(\ell, m_1-1)$ and $W_k(\ell, m_1)$ defined in (18). The characteristic equation for W_k defined in (17) is

$$\begin{aligned} 0 &= \det(\lambda I_{m_1} - W_k) = \lambda^{m_1-2}\begin{vmatrix} \lambda - W_k(m_1-1,m_1-1) & -W_k(m_1-1,m_1) \\ -W_k(m_1,m_1-1) & \lambda - W_k(m_1,m_1) \end{vmatrix} = \\ &= \lambda^{m_1-1}(\lambda - i\kappa_1\Phi_{m_1,m_1-1}(\tfrac{4+\kappa_2\lambda_{2,k}}{2},\kappa_1) - 2\kappa_1\Phi_{m_1,m_1}(\tfrac{4+\kappa_2\lambda_{2,k}}{2},\kappa_1)) \end{aligned}$$

Hence W_k only has one nonzero eigenvalue λ_k given by

$$\lambda_k = i\kappa_1\Phi_{m_1,m_1-1}(\tfrac{4+\kappa_2\lambda_{2,k}}{2},\kappa_1) + 2\kappa_1\Phi_{m_1,m_1}(\tfrac{4+\kappa_2\lambda_{2,k}}{2},\kappa_1) = -z_k^2\frac{1-z_k^{2m_1-2}}{1-z_k^{2m_1+2}} + 2iz_k\frac{1-z_k^{2m_1}}{1-z_k^{2m_1+2}}$$

where

$$z_k = i\left(\frac{4+\kappa_2\lambda_{2,k}}{2\kappa_1} - \sqrt{1+\left(\frac{4+\kappa_2\lambda_{2,k}}{2\kappa_1}\right)^2}\right) \quad , \quad k = 1,\ldots,m_2 \ . \tag{23}$$

We summarize this result in a theorem.

Theorem 5.3 *The matrix $M^{-1}A$ where M is defined in (8) and A in (4), has $(m_1-1)m_2$ eigenvalues that are identically one. The remaining m_2 eigenvalues μ_k are given by*

$$\mu_k = 1 - z_k^2\frac{1-z_k^{2m_1-2}}{1-z_k^{2m_1+2}} + 2iz_k\frac{1-z_k^{2m_1}}{1-z_k^{2m_1+2}} \quad , \quad k = 1,\ldots,m_2 \tag{24}$$

where z_k is defined in (23).

As the dimension of the nullspace of E is $(m_1-1)m_2$, it is clear that the eigenvalue 0 to $M^{-1}E$ has $(m_1-1)m_2$ linearly independent eigenvectors. Hence we know that a minimal residual iteration will converge in at most m_2+1 iterations.

From Theorem 5.3 we can derive

Theorem 5.4 *Assume that*

$$\Delta t = ch_1^{\alpha} \quad , \quad 0 < \alpha < 1 \quad , \quad c > 0 \tag{25}$$

and

$$m_1 = \lceil \frac{m_2(1+2m_2^{-\frac{3}{2}})}{\phi} \rceil \quad , \quad 0 < \phi < 1 \tag{26}$$

where $\lceil \frac{a}{b} \rceil$ denotes the closest integer greater than $\frac{a}{b}$. In the limit $m_1 \to \infty$, the m_2 eigenvalues of $M^{-1}A$ that are different from one all reside on a curve-segment $\mu(\gamma)$

$$\mu(\gamma) = 2 - 2\gamma^2 + 2\sqrt{1-\gamma^2} - 2i\gamma(\sqrt{1-\gamma^2}+1) \quad , \quad -\phi \le \gamma \le \phi \tag{27}$$

Proof:
Define ζ_k as

$$\zeta_k = \frac{4+\kappa_2\lambda_{2,k}}{2\kappa_1} \quad , \quad k = 1,\dots,m_2$$

By (25) we get

$$\zeta_k = \frac{2h_1}{ch_1^\alpha} + \frac{h_1}{2h_2}\lambda_{2,k} = 2c^{-1}m_1^{\alpha-1} + w_k \quad , \quad k = 1,\dots,m_2 \tag{28}$$

where

$$w_k = \frac{m_2}{2m_1}\lambda_{2,k} \quad , \quad k = 1,\dots,m_2 \tag{29}$$

Lemma 5.1 together with (29) and (28) yields

$$w_k \in \mathcal{E}^+(2\frac{m_2^{\frac{1}{4}}}{m_1}, \frac{m_2}{m_1}(1+2m_2^{-\frac{3}{4}})) \quad , \quad k = 1,\dots,m_2$$

By (26) $m_2 < m_1$ and $\frac{m_2}{m_1}(1+2m_2^{-\frac{3}{4}}) \le \phi$ yielding

$$w_k \in \mathcal{E}^+(2\frac{m_2^{\frac{1}{4}}}{m_1}, \phi) \in \mathcal{E}^+(2m_1^{-\frac{3}{4}}, \phi)$$

Now define

$$\epsilon = c^{-1}m_1^{\alpha-1} + \delta_k m_1^{-\frac{3}{4}} \quad , \quad 0 \le \delta_k \le 1, \tag{30}$$

which gives

$$\zeta_k = 2\epsilon + i\gamma_k \quad , \quad -\phi \le \gamma_k \le \phi. \tag{31}$$

For $m_1 \gg 1$ we have $\epsilon \ll 1$ and (31) together with (30) in (23) and a Taylor expansion yields

$$z_k = i(\zeta_k - (1+\zeta_k^2)^{1/2}) = 2i\epsilon - \gamma_k - i\sqrt{1-\gamma_k^2} + \frac{2i\epsilon\gamma_k}{\sqrt{1-\gamma_k^2}} + \mathcal{O}(\epsilon^2)$$

We obtain

$$z_{k,\infty} \equiv \lim_{m_1\to\infty} z_k = -\gamma_k - i\sqrt{1-\gamma_k^2} \quad , \quad -\phi \le \gamma_k \le \phi \tag{32}$$

For m_1 sufficiently large there is a positive constant c_k such that

$$|z_k|^2 = 1 - \frac{4\epsilon\gamma_k}{\sqrt{1-\gamma_k^2}} + \mathcal{O}(\epsilon^2) < 1 - c_k m_1^{\alpha-1}.$$

Thus, we get $|z_k|^{2m_1} \to 0$, which together with equation (24) gives

$$\lim_{m_1\to\infty} \mu_k = 1 - z_{k,\infty}^2 + 2iz_{k,\infty} \quad , \quad k = 1,\dots,m_2 \; . \tag{33}$$

By inserting (32) in (33) and exploiting that $-\phi \le \gamma_k \le \phi$, $\forall k$, we conclude that the eigenvalues of $M^{-1}A$ that are different from one all reside upon the curve-segment $\mu(\gamma)$ defined by (27). □
From Theorem 5.4 we see that when Δt is defined by (25) the eigenvalues stay bounded and are well separated from the origin when the problem-size increases. In Figure 1 we show how well the asymptotic formula (27) agrees with the eigenvalues for a large problem-size.

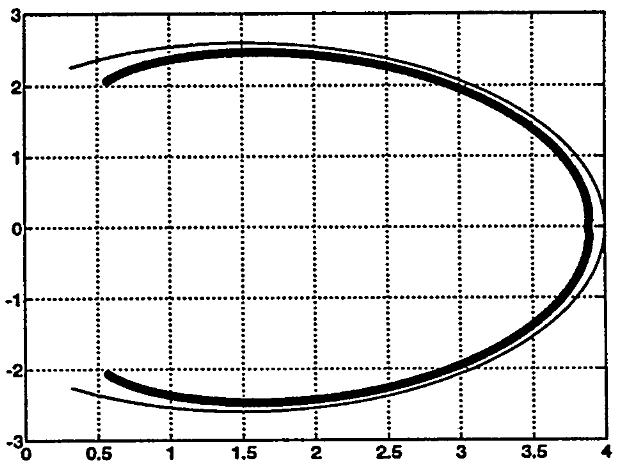


Figure 1: Asymptotic spectrum solid line and eigenvalues marked with circles for $\alpha = 0.99$, $c = 100$, $\phi = 0.99$, and $m_2 = 500$.

6 Asymptotic convergence factor

In this section we determine the *asymptotic convergence factor* ρ defined by

$$\rho \equiv \lim_{i\to\infty} \varepsilon_i^{1/i} \tag{34}$$

where ε_i is defined in (6). We enclose the asymptotic spectrum μ defined in (27) in a circle $\mathcal{C}(4, R)$ and use the general result from [10]. Here $\mathcal{C}(c, R)$ denotes the circle with center c and radius R.

Lemma 6.1 *The circle $\mathcal{C}(4, R)$ where R is defined as*

$$R = |\mu(\pm\phi) - 4| = \sqrt{8 + 12\phi^2 - 8\sqrt{1-\phi^2}} \tag{35}$$

encloses the asymptotic spectrum μ defined in (27).

Proof:
Define $r(\gamma)$ as the distance between 4 and $\mu(\gamma)$. Then

$$r^2 = (-2 - 2\gamma^2 + 2\sqrt{1-\gamma^2})^2 + (2\gamma(\sqrt{1-\gamma^2} + 1))^2 = 8 + 12\gamma^2 - 8\sqrt{1-\gamma^2}$$

and

$$\frac{d^2r^2}{d\gamma^2} = 24 + \frac{8}{(1-\gamma^2)^{1/2}} + \frac{8\gamma^2}{(1-\gamma^2)^{3/2}} > 0$$

Hence r^2 obtains its maximum value at the endpoints yielding $r(\gamma) \leq R$ which proves the lemma $\square$

Theorem 6.2 *For $0 < \phi < \frac{\sqrt{8}}{3}$ the asymptotic convergence factor ρ fulfills*

$$\rho \leq \frac{\sqrt{2 + 3\phi^2 - 2\sqrt{1-\phi^2}}}{2} < 1$$

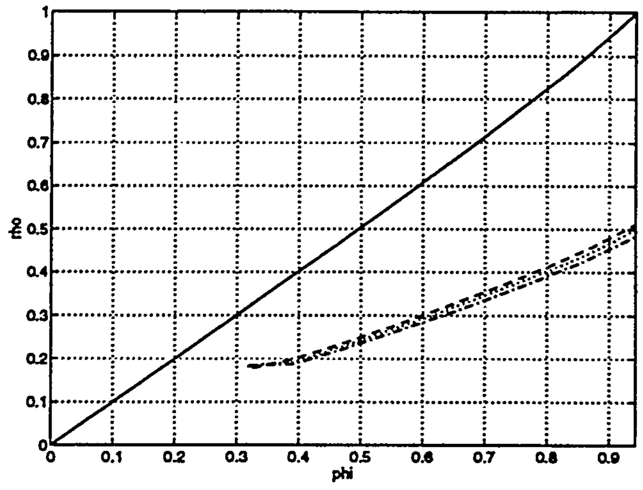


Figure 2: The solid line represents the asymptotic convergence factor for $\alpha = 0.99$ and $c = 100$. The residual reduction is plotted for $m_1 = 127$ dashed line, $m_1 = 255$ dotted line and $m_1 = 511$ dashed-dotted line.

Proof:
For $0 < \phi < \frac{\sqrt{8}}{3}$ we obtain $8 + 12\phi^2 < 8 + 12 \cdot \frac{8}{9} = 8\sqrt{1 - \frac{8}{9}} + 16 < 8\sqrt{1-\phi^2} + 16$, and consequently $\sqrt{8 + 12\phi^2 - 8\sqrt{1-\phi^2}} < 4$. Using Lemma 6.1 and the general result in [10] yields

$$\rho \leq \frac{R}{4} = \frac{\sqrt{2 + 3\phi^2 - 2\sqrt{1-\phi^2}}}{2} < 1.$$

□

In Figure 2 we show the asymptotic convergence factor for varying ϕ. In the same figure we show the residual reduction $\tilde{\rho}$ defined by

$$\tilde{\rho} = \left(\frac{\|r_i\|_2}{\|r_0\|_2} \right)^{1/i}$$

for different problem-sizes using GMRES [11] and $i = 20$.
Finally in Figure 3 we show the actual number of iterations obtained from GMRES(20) for different problem-sizes. The iteration has converged when $\|M^{-1}r^{(i)}\|_2 / \|M^{-1}b\|_2 < 10^{-6}$.

From Figure 2 and 3 we see that the convergence seems to depend only on the ratio between the number of grid-points in the different space-directions and not on the number of unknowns. In Figure 3 we see that the number of iterations actually goes down when we increase the size of the problem.

7 Conclusions

In this report we have studied semi-Toeplitz preconditioners and minimal residual iterations to solve block-tridiagonal systems of equations. Analytical formulas for the eigenvalues of the preconditioned system are derived. We have also shown that in the limit $m_d \to \infty$ the eigenvalues of the preconditioned system all lie on a curve-segment that can be easily computed. The eigenvalues stay bounded and are well separated from the origin independently of the problem-size. From this eigenvalue distribution we can derive upper bounds to the asymptotic convergence factor. These bounds have been verified numerically. Finally we show empirically that the number of iterations does not grow when we increase the number of unknowns.

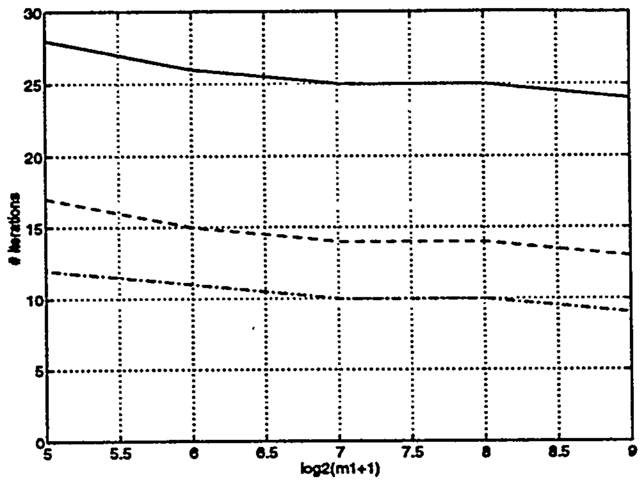


Figure 3: Number of iterations for $\alpha = 0.99$ and $c = 100$. Solid line represents $\phi = 0.99$, dashed line $\phi = 0.75$, and dashed-dotted line $\phi = 0.5$.

References

[1] O. Axelsson, *A restarted version of a generalized preconditioned preconditioned conjugate gradient method*, Report No. 8710, Department of Mathematics, University of Nijmegen, Nijmegen, the Netherlands, 1987.

[2] O. Axelsson and G. Lindskog, *On the eigenvalue distribution of a class of preconditioning methods*, Numer. Math., 48 (1986), pp. 479-498.

[3] R. W. Freund, G. H. Golub, and N. M. Nachtigal, *Iterative solution of linear systems*, Acta Numerica (1991), pp. 57-100.

[4] J. Guerra and B. Gustafsson, *A semi-implicit method for hyperbolic problems with different time-scales*, SIAM J. Numer. Anal., 23 (1986), pp. 734-749.

[5] B. Gustafsson and H. Stoor, *Navier-Stokes equations for almost incompressible flow*, SIAM J. Numer. Anal., 28 (1991), pp. 1523-1547.

[6] L. Hemmingsson, *A Fast Modified Sine Transform for Solving Block-Tridiagonal Systems with Toeplitz Blocks*, Submitted to Num. Alg.

[7] L. Hemmingsson, *Toeplitz preconditioners with Block Structure for First-order PDEs*, Report No. 156, Dept. of Scientific Computing, Uppsala University, Uppsala, Sweden, 1993.

[8] L. Hemmingsson, *Domain Decomposition Methods for Hyperbolic Problems in 2D*, Draft report, Dept. of Scientific Computing, Uppsala University, Uppsala, Sweden.

[9] K. Otto, *Analysis of preconditioners for hyperbolic PDE*, Report No. 147, Dept. of Scientific Computing, Uppsala University, Uppsala, Sweden, 1992.

[10] Y. Saad, *Krylov subspace methods for solving large unsymmetric linear systems*, Math. Comp., 37 (1981), pp. 105-126.

[11] Y. Saad and M. Schultz, *GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856-869.

[12] H. A. van der Vorst, *Preconditioning by incomplete decompositions*, Ph.D. thesis, Rijksuniversiteit Utrecht, Utrecht, the Netherlands, 1982.

EQUIVARIANT PRECONDITIONERS FOR BOUNDARY ELEMENT METHODS[1]

JOHANNES TAUSCH[2,3]

Abstract. In this paper we propose and discuss two preconditioners for boundary integral equations on domains which are nearly symmetric. The preconditioners under consideration are equivariant, that is, they commute with a group of permutation matrices. Numerical experiments demonstrate their efficiency for the GMRES method.

1. Introduction. In the past few years symmetry exploiting methods have become a popular topic in numerical linear algebra. Of special interest are linear systems that come from discretizing an integral equation defined on a domain with geometrical symmetries. These problems inherit the structure of the underlying group of symmetry transformations when the discretization is done in an appropriate way. By making use of this structure, it is possible to significantly reduce the amount of work involved in solving these systems.

Of course, this method will fail when the symmetry is destroyed by perturbing the domain slightly. However, we expect that the symmetric and the perturbed problems are somewhat close to each other and that it is possible to take advantage of this situation as well.

Here we present an approach how this can be achieved. We propose two preconditioners for the iterative solution of the discretized equation which have the structure of the related symmetry.

Now let us outline this paper. After a brief overview of boundary element methods and iterative methods for the linear system associated with the integral equation in sections §2 and §3, we discuss in §4 symmetry reduction methods. The following two sections describe the preconditioners used. In section §5 a preconditioner is constructed by discretizing the integral equation on a nearby symmetric surface. The following section §6 contains a preconditioner which is the solution of a minimization problem. Some results of numerical experiments are presented in section §7.

2. Boundary Integral Methods. Consider the linear integral equation

$$\lambda\rho(x) + \mathcal{K}\rho(x) = g(x) \qquad x \in \mathcal{B} \tag{1}$$

with the boundary integral operator

$$\mathcal{K}\rho(x) = \int_{\mathcal{B}} k(x,y)\rho(y)d\mathcal{B}(y),$$

defined on a linear space of functions on $\mathcal{B}$ which is denoted by X.

We are interested in the case when $\mathcal{B}$ is a closed and compact surface in the three-space $\mathcal{B} \subset \mathbb{R}^3$. Equations like (1) arise from solving Laplace's equation on a domain with boundary $\mathcal{B}$, in which case the kernel $k(x,y)$ is weakly singular.

One of the standard approaches for solving (1) numerically is the application of a collocation method. The idea here is to seek an approximation of the solution

[1] Submitted for the student paper competition of the Colorado Conference on Iterative Methods.

[2] Partially supported by the National Science Foundation under grant number DMS-9104058

[3] Department of Mathematics, Colorado State University, Ft. Collins, Colorado 80523, USA, e-mail: Tausch@Math.Colostate.edu

ρ in a finite dimensional linear subspace $X_n \subset X$ spanned by the basis functions $\{\phi_1, \dots, \phi_n\}$. Moreover, consider points on the surface $\{p_1, \dots, p_n\} \subset B$ chosen in a way such that the interpolation problem

$$\text{find } \rho_n \in X_n \text{ such that } \rho(p_i) = \rho_n(p_i) \quad \text{for } i = 1, \dots, n$$

is uniquely solvable for all functions $\rho \in X$. Typically, the subspace X_n consists of functions that are piecewise polynomial on a triangulation of B, this has been extensively studied by Atkinson, see [3].

In general, the solution of the integral equation (1) will not be in X_n, and therefore a function in this subspace cannot solve the equation at all points on the surface. Instead, one forces the unknown function ρ_n to satisfy (1) at least at the collocation points. If ρ_n is written as a linear combination of the basis functions $\rho_n = \sum c_i\phi_i$, then this yields the following linear system for the coefficients $x = (c_1, \dots, c_n)^T$

$$Ax = b \tag{2}$$

with matrix entries

$$A(i,j) = \lambda\phi_j(p_i) + (\mathcal{K}\phi_j)(p_i) \tag{3}$$

and right hand side

$$b(i) = g(p_i)\,.$$

For estimates of the discretization error $|\rho_n - \rho|$ we refer to the vast literature on numerical methods for integral equations, see e.g. [4] or [13].

3. GMRES. The matrix of the collocation method in (2) is in general dense and non-symmetric. A standard approach for solving (2) is to apply a multi grid technique, see for instance [12] or [13]. These methods work well when the operator $\mathcal{K}$ in the integral equation (1) is compact. If the boundary is only piecewise smooth, then this assumption is often violated and the multi grid iteration may perform poorly or may even diverge [5].

Only recently, other iterative methods, like conjugate gradients and Krylov subspace methods have been studied in connection with boundary integral equations, see e.g. [8] and [18]. One of these methods is the GMRES method of Saad and Schultz [14].

A well-known technique to improve the convergence of iterative methods is preconditioning, see e.g. [11]. The idea here is to multiply the linear system $Ax = b$ with the inverse of a nonsingular matrix L, the preconditioner, and to solve the equivalent linear system $L^{-1}Ax = L^{-1}b$. Alternatively, one can transform the unknown x to $Lx = y$ and solve $AL^{-1}y = b$. Note that the product $L^{-1}A$ or AL^{-1} is actually never formed - this would be too costly. One step in an iteration method involves – among other operations – the multiplication of a vector with the matrix A. The extra expense of the preconditioned method is to solve a linear system with matrix L in each step.

A good choice for a preconditioner is a matrix that factors A in the form $L^{-1}A = I + B$ where the norm of the remainder $\|B\|_2$ is small. Then the eigenvalues of the preconditioned matrix $L^{-1}A$ cluster around unity and yield a small condition number.

Another suitable preconditioner is a matrix L that factors A to a low rank perturbation of the identity, i.e. $L^{-1}A = I + R$. Here R denotes a low rank matrix: $rk(R) = p \ll n$ and no assumption must be made about the norm of R. In this case the preconditioned GMRES iteration terminates after at most p steps with the exact solution.

4. Equivariance. We are interested in the structure of the system matrix in (2) when the surface $\mathcal{B}$, on which the integral equation (1) is to be solved, has symmetries. That is, $\mathcal{B}$ is left invariant under a group Γ of isometries such as rotations and reflections. Each isometry $\gamma \in \Gamma$ gives rise to a linear operator $\Pi_\gamma : X \to X$ mapping the function $f \in X$ on $f \circ \gamma^{-1}$. The operators $\{\Pi_\gamma | \gamma \in \Gamma\}$ defined in this way form a group under the usual multiplication of operators which is isomorphic to the group of isometries. Henceforth we will use the same symbol for both groups.

Moreover, suppose the kernel of the integral operator $\mathcal{K}$ depends only on the distance of the two points x and y, i.e.

$$k(x,y) = \tilde{k}(|x-y|)\,.$$

This situation is typical when equation (1) comes from an integral reformulation of a boundary value problem. Using a change of variables, it is straightforward to see that such an integral operator commutes with the action of an element in Γ, i.e.

$$\mathcal{K}\Pi_\gamma = \Pi_\gamma \mathcal{K}\,. \tag{4}$$

In general, an operator $\mathcal{K}$ with the property (4) is called *equivariant with respect to the group action* Γ, or simply Γ-*equivariant.*

In order to be able to exploit the structure of the continuous problem, the discretization must not destroy the symmetry. This is the content of our assumption on the basis functions and the collocation points:

Assumption: The group Γ defines a group of permutations on the indices by

$$[\Pi_\gamma\phi_1, \dots, \Pi_\gamma\phi_n] \;=\; [\phi_{\gamma 1}, \dots, \phi_{\gamma n}]$$

and

$$[\gamma(p_1), \dots, \gamma(p_n)] \;=\; [p_{\gamma 1}, \dots, p_{\gamma n}]\,.$$

and the permutation group is isomorphic to Γ.

The permutations on the indices in turn induce permutation matrices Π_γ by setting:

$$(\Pi_\gamma b)(i) = b(\gamma^{-1} i)$$

for a vector $b \in \mathbb{R}^n$. Using the equivariance of $\mathcal{K}$ and the above defined group actions we obtain for the matrix entries in (2):

$$A(\gamma^{-1}i, j) = (\mathcal{K}\phi_j)(p_{\gamma^{-1}i}) = (\Pi_\gamma \mathcal{K}\phi_j)(p_i) = (\mathcal{K}\Pi_\gamma\phi_j)(p_i) = (\mathcal{K}\phi_{\gamma j})(p_i) = A(i, \gamma j)\,.$$

Hence we see that the matrix A has the three equivalent properties:

$$A(\gamma^{-1}i, j) = A(i, \gamma j) \qquad \forall i, j, \gamma \tag{5}$$

$$A(i,j) = A(\gamma i, \gamma j) \qquad \forall i, j, \gamma \tag{6}$$

$$\Pi_\gamma A = A \Pi_\gamma \qquad \forall \gamma\,. \tag{7}$$

Equation (7) is the discrete analogue to (4). From equation (6) it is clear that the matrix entries are constant on the orbits of Γ, consequently, the matrix is determined by only $n^2/\#\Gamma$ elements, if there are no fixed points of the group action on the index set.

It is possible to exploit this structure to save computational work when solving the linear system (2). The key here lies in the irreducible representations of the group Γ – for an introduction into representation theory see e.g. [15]. The irreducible representations determine a sparse and unitary matrix F – the generalized Fourier matrix – which factors L in the form $L = F^H \hat{L} F$ with $\hat{L}$ being a block diagonal matrix. Since F is a unitary matrix, its inverse is given by the Hermitian transpose, i.e. $F^{-1} = F^H$. Instead of the linear system $Lx = b$, the equivalent system $\hat{L}\hat{x} = \hat{b}$ with right hand side $\hat{b} = Fb$ and unknown $\hat{x} = Fx$ is solved and then the solution x is recovered from $\hat{x}$ via the inverse transformation $x = F^H \hat{x}$. We call this method the Generalized Fourier Transformation (GFT), because it extends the idea of using the discrete Fourier transformation for circulant matrices to equivariant matrices [7].

The diagonal blocks of $\hat{L}$ can be computed with n^2 floating point operations, the transformations $\hat{b} = Fb$ and $x = F^H \hat{x}$ can be done in $(\#\Gamma)n$ flops. This shows that the overhead is negligible, considering that A is a full matrix. The major savings in computational effort comes from the fact that a number of small systems are solved as opposed to one big system in the un-reduced case. For more details about the GFT we refer to [16]. An earlier description of symmetry reduction using projections can be found in [1] and [10]. The approach presented there yields equivalent subsystems. Applications to boundary element methods with numerical results are in [2] and in [19].

5. Preconditioner derived from a nearby symmetric surface. In this and the next section we describe two preconditioners to handle problems defined on domains that are close to a domain with geometrical symmetries. The basic idea of our first preconditioner is to discretize the integral equation on the symmetric surface. As was pointed out in the previous section, this yields an equivariant matrix which can be inverted efficiently. Let us briefly discuss how the preconditioner is constructed. Usually surfaces are represented via parameterizations. Since we want to include piecewise smooth surfaces, it is convenient to work with piecewise linear (PL) manifolds. Here we recall the definition found in Georg [9]: A *PL-manifold* is a finite collection $\mathcal{S}_{PL}$ of closed triangles in $\mathbb{R}^3$, each having affinely independent vertices $T_i = [v_0^i, v_1^i, v_2^i]$, $i = 1, \ldots, J$. In addition, we require

1. The intersection of two triangles in $\mathcal{S}_{PL}$ is either empty or a vertex or an edge.
2. Each edge is common to exactly two triangles.

We assume that it is possible to parameterize the symmetric as well as the perturbed surface with the same PL-manifold. Then our discretization scheme can be described as follows:

1. find a symmetry respecting PL-manifold $\mathcal{S}_{PL}$ and parameterizations (i.e. piecewise smooth isomorphisms) $m : \mathcal{S}_{PL} \to \mathcal{B}$ and $\tilde{m} : \mathcal{S}_{PL} \to \tilde{\mathcal{B}}$ of the symmetric and the perturbed surface respectively.
2. define a set of basis functions $\{\psi_1, \ldots, \psi_n\}$ on $\mathcal{S}_{PL}$. Lifting them to the surfaces $\mathcal{B}$ and $\tilde{\mathcal{B}}$ via $\psi_i = \phi_i \circ m$ and $\psi_i = \tilde{\phi}_i \circ \tilde{m}$ produces basis functions $\{\phi_1, \ldots, \phi_n\}$ and $\{\tilde{\phi}_1, \ldots, \tilde{\phi}_n\}$. Note that the basis on $\mathcal{B}$ must respect the symmetry in the sense of our assumption.
3. define collocation points $\{q_1, \ldots, q_n\}$ on $\mathcal{S}_{PL}$ and map them to the two surfaces: $p_i = m(q_i)$ and $\tilde{p}_i = \tilde{m}(q_i)$.

Applying the collocation method, we obtain the two nonsingular matrices L and A arising from the symmetric and the un-symmetric problem, respectively. The matrix L is a good preconditioner when the quantity $||A - L||$ is small. It follows from [17] that this is in fact true when the parameterizations and their first few derivatives are close to each other.

The use of the above preconditioner has a significant drawback. Usually the assembly of the system matrix is the most costly part of a boundary element technique. Our method requires two matrices – one for the symmetric and one for the perturbed problem. Even though only a fraction of matrix entries have to be determined in the equivariant case, the savings in the iteration may not justify the extra calculation of the preconditioner.

This objection however does not apply for situations where the perturbed surface differs from the symmetric surface only on a small piece. In this case the respective basis functions and the collocation points are identical except for a few, which implies that only a few rows and columns of A have to be updated to obtain the preconditioner. In other words, A is a low rank perturbation of L, or $L^{-1}A$ is a low rank perturbation of the identity, which in turn implies that the number of steps in the GMRES iteration of $L^{-1}A$ is bounded by this rank.

In the next section we introduce a preconditioner which does not require additional surface integrations.

6. An Optimal Preconditioner. Our next goal is to find an equivariant matrix L so that the product $L^{-1}A$ is as close to the identity as possible. In other words, the preconditioner has to minimize the quantity $\|L^{-1}(A-L)\|$ in a consistent matrix norm. An upper bound for this number can be obtained easily, setting $B = A - L$ we estimate:

$$\|L^{-1}B\| = \|(A-B)^{-1}B\| = \|(I - A^{-1}B)^{-1}A^{-1}B\| \leq \frac{\|A^{-1}B\|}{1 - \|A^{-1}B\|} .$$

Since the right hand side of this inequality is monotonically increasing with $\|A^{-1}B\|$, it is straight forward to minimize $\|B\| = \|A - L\|$. Thus we require that our preconditioner solves the following minimization problem:

$$L \equiv \min\{\|A - L\| : L \text{ is } \Gamma\text{-equivariant}\} . \tag{8}$$

We will show that this problem is trivially solvable in the Frobenius norm: To obtain the optimal preconditioner, one has to average over the orbits of the group action on the indices. This construction can be viewed as a generalization of T. Chan's circulant preconditioner for Toeplitz systems [6], since a circulant matrix of order n is equivariant with respect to a cyclic group of order n. Here this idea is extended to any finite group. Before we formally describe the preconditioner in the following theorem, we need some simple notions on a group acting on a set of integers:

Consider the index i in the index set $N := \{1, \ldots, n\}$. We denote the set $\mathrm{Orb}(i) := \{\gamma i : \gamma \in \Gamma\}$ the orbit of the group action on the index i. Note, that the cardinality of this set is not always equal to the group order $\#\Gamma$, since the group action might have fixed points, i.e. $\gamma i = i$ for some $i \in N$ and some $\gamma \neq e$. We denote by $\bar{\Gamma}_i$ a minimal subset of the group that generates the orbit of i, that is $\mathrm{Orb}(i) = \{\gamma i : \gamma \in \bar{\Gamma}_i\}$. A subset $S \subset N$ that contains exactly one element from each orbit is called a selection of indices. Clearly $\{\gamma i\}_{i \in S, \gamma \in \bar{\Gamma}_i}$ is a list of all elements in N without repetitions.

Now we are in a position to state the theorem:

THEOREM 1. *Let $S \subset N$ be a selection of indices and let $\bar{\Gamma}_i$ be a minimal generator of the orbit Orb(i) for an index $i \in S$. Then the matrix $L = \{L(i,j)\}_{i,j}$ defined by*

$$L(i,j) = \frac{1}{\#\bar{\Gamma}_i} \sum_{\gamma \in \bar{\Gamma}_i} A(\gamma i, \gamma j) \qquad \forall i \in S, j \in N \tag{9}$$

and extended by

(10)
$$L(\gamma i,j) \;=\; L(i,\gamma^{-1}j) \qquad \forall i\in S, j\in N, \gamma\in\Gamma\,,$$

is the optimal solution of the minimization problem (8) in the Frobenius norm.

Proof. Let L be an arbitrary Γ-equivariant matrix. The main idea of the proof is to rearrange the summation in the Frobenius norm $\|A-L\|_F^2$ so that the orbits of $L(i,j)$ stand together.

$$\begin{aligned}
\|A-L\|_F^2 &= \sum_{i'\in N}\sum_{j'\in N}\left(A(i',j')-L(i',j')\right)^2\\
&= \sum_{i\in S}\sum_{\gamma\in\bar\Gamma_i}\sum_{j'\in N}\left(A(\gamma i,j')-L(\gamma i,j')\right)^2\\
&= \sum_{i\in S}\sum_{\gamma\in\bar\Gamma_i}\sum_{j\in N}\left(A(\gamma i,\gamma j)-L(\gamma i,\gamma j)\right)^2 \qquad \text{setting } j=\gamma^{-1}j'\\
&= \sum_{i\in S}\sum_{j\in N}\sum_{\gamma\in\bar\Gamma_i}\left(A(\gamma i,\gamma j)-L(i,j)\right)^2 \qquad \text{using the equivariance of L}
\end{aligned}$$

The last expression consists of un-coupled optimization problems for the $L(i,j)$. The optimal solution can be obtained by minimizing each term, i.e.

$$\sum_{\gamma\in\bar\Gamma_i}\left(A(\gamma i,\gamma j)-L(i,j)\right)^2$$

for each $i\in N, j\in S$. This yields

$$L(i,j)=\frac{1}{\#\bar\Gamma_i}\sum_{\gamma\in\bar\Gamma_i}A(\gamma i,\gamma j)\,.$$

It is easy to see that L is equivariant and independent of the choice of S or $\bar\Gamma_i$. ∎

The computation of $L(i,j)$ for $i\in S$ and $j\in N$ by equation (9) requires $\#\bar\Gamma_i$ additions, yielding n^2 floating point operations for the whole matrix L. Thus the calculation of the optimal preconditioner costs approximately about as much as one single matrix - vector multiplication. This is clearly an advantage over the preconditioner of the previous section, because no surface integrations are necessary.

Note, that the matrix constructed in Theorem 1 is not always nonsingular. However, one matrix in the feasible set of the optimization problem (8) is the preconditioner derived from the symmetric surface, which is nonsingular. Thus, the optimal preconditioner is nonsingular if the perturbation of the surface is small enough.

7. Some Numerical Results.

7.1. Second Kind Equations.

In the following we present some numerical results using the optimal preconditioner described above. The first examples deal with the integral equation

$$2\pi\rho(x)+\int_B\frac{\partial}{\partial n_y}\left(\frac{1}{|x-y|}\right)\rho(y)dB(y)+(2\pi-\Omega(x))\rho(x)=g(x)\,,$$

which comes from solving Laplace's equation on a domain with boundary surface B. Here $\Omega(x)$ denotes the solid angle of B at the point x, which is 2π when the surface

N	*its*	δ	*its*	δ
48	11	.050	5	.0055
192	10	.042	5	.0095
768	11	.048	6	.0083

TABLE 1
Iteration results for the ellipsoid

N	*its*	δ	*its*	δ
48	19	.176	6	.014
192	22	.219	7	.028
768	22	.235	7	.033

TABLE 2
Iteration results for the perturbed cube

is smooth at x. The linear systems come from collocating with functions that are piecewise constant on a triangluation of the parameter space.

The GMRES iteration was continued until the 2-norm of the residual was reduced by the factor 10^{-15}. The tables show the number of iterations (*its*) as well as the average reduction factor of the residual in each step (δ) for various refinements of the grid (N denotes the number of triangles). The parameters *its* and δ are compared for the original and the preconditioned system.

The domain in the first example is the ellipsoid $(x/1.1)^2 + (y/1.05)^2 + z^2 = 1$. The parameterizing $\mathcal{S}_{PL}$-manifold is the unit cube, which induces a group action of order 48 on the indices. This group action was used to construct the preconditioner as in (9). The results of the experiments are shown in Table 1. In the middle column are the results of the iteration applied on the original system, in the right column are the respective numbers for the preconditioned system.

The second domain is a cube, with one side perturbed by a quadratic surface, see Figure 1. Due to the edges, the number of steps of the un-preconditioned iteration is higher than in the previous example with a smooth surface. Note however, that the increase of steps in the preconditioned method is not so significant. The numerical results are displayed in Table 2.

In the third example (a cube with a small cube removed in one corner, c.f. Figure 2) the preconditioner performed poorly. This is due to the fact, that the derivatives of the parameterizations are not nearby, yielding boundary integral operators that are not close in some norm. The results are shown in Table 3.

The same experments with a preconditioner that comes from a surface with symmetries did not reveal noticable differences. The optimal preconditioner performed only slightly better.

7.2. First Kind Equations. In the above numerical examples the preconditioner reduced the number of steps in the iteration significantly, however the un-preconditioned iteration converges reasonably fast as well, especially in the case of the ellipsoid. This is due to the well-posed nature of second kind equations (1) with compact operators.

This picture changes when first kind equations are to be solved. Equations of this type are of great importance for boundary element methods. We experimented with

N	*its*	δ	*its*	δ
384	22	.24	16	.20
1152	23	.26	18	.23

TABLE 3
Iteration results for the domain of Figure 2

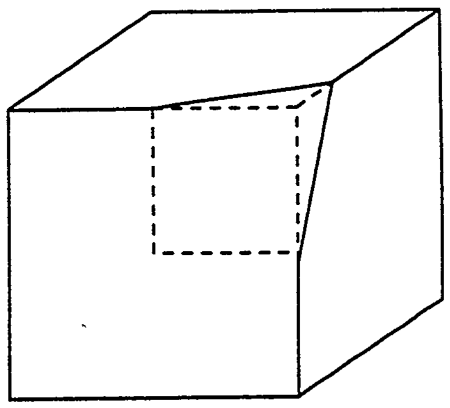


FIG. 1. *The perturbed cube*

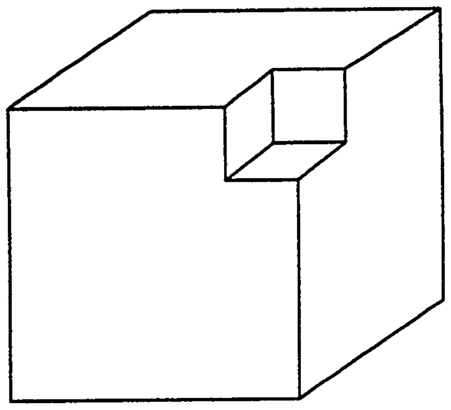


FIG. 2. *The cube with a small cube removed in one corner*

N	*its*	δ	*its*	δ
48	23	.250	8	.039
192	33	.373	10	.068
768	44	.495	10	.084

TABLE 4
Results for a first kind equation on the pertubed cube

N	*its*	δ	*its*	δ
384	39	.450	9	.064
1152	49	.541	10	.087

TABLE 5
Results for a first kind equation on the domain of Figure 2

the single layer operator from potential theory, which is defined by

$$\mathcal{S}\rho(x) = \int_{\mathcal{B}} \frac{1}{|x-y|}\,\rho(y)\,d\mathcal{B}(y)\,.$$

Since this operator is compact, its inverse is not bounded and the equation $\mathcal{S}\rho = g$ is ill posed. When a discretization technique is applied then the number of GMRES iterations will increase with the refinement of the mesh.

The experiments suggest that our preconditioners work well especially for first kind equations. Compare with the results of Table 4, which were obtained by discretizing the perturbed cube of Figure 1 in the same way as for the double layer equation.

As expected, the number of iterations increases as the grid is refined. This is also the case when the preconditioner is used, however the increase is much slower. The two preconditioners that we have discussed perform almost equally well as in the case of the double layer equation.

Suprisingly, we obtained good performance for the single layer equation even on the domain of Figure 2, where preconditioning of the double layer equation failed, see Table 5. This behavior may be attributed to the higher sensitivity of the double layer operator to perturbations of the surface. We will investigate this in our future work [17].

REFERENCES

[1] E. ALLGOWER, K. BÖHMER, K. GEORG, AND R. MIRANDA, *Exploiting symmetry in boundary element methods*, SIAM J. Numer. Anal., 29 (1992), pp. 534–552.

[2] E. ALLGOWER, K. GEORG, AND J. WALKER, *Exploiting symmetry in 3-D boundary element methods*, in Contributions in Numerical Mathematics, R. Agarwal, ed., vol. 2, Singapore, 1992, World Scientific Publ. Comp., pp. 15–25.

[3] K. E. ATKINSON, *Piecewise polynomial collocation for integral equations on surfaces in three dimensions*, Journal of Integral Equations, 9 (1984), pp. 25–48.

[4] ———, *A survey of boundary integral equations for the numerical solution of Laplace's equation in three dimensions*, in Numerical Solution of Integral Equations, M. Goldberg, ed., New York, 1990, Plenum Press, pp. 1–34.

[5] K. E. ATKINSON AND I. G. GRAHAM, *Iterative solution of linear systems arising from the boundary integral method*, SIAM J. Sci. Stat. Comput., 13 (1992), pp. 694–722.

[6] T. CHAN, *An optimal circulant preconditioner for Toeplitz systems*, SIAM J. Sci. Stat. Comput., 9 (1988), pp. 766–771.

[7] P. DAVIS, *Circulant Matrices*, John Wiley and Sons, New York, 1979.

[8] J. D. Flores, *The conjugate gradient method for solving Fredholm integral equations of the second kind*, Intern. J. Computer Math., 48 (1993), pp. 77–94.

[9] K. Georg, *Approximation of integrals for boundary element methods*, SIAM J. Sci. Stat. Comput., 12 (1991), pp. 443–453.

[10] K. Georg and R. Miranda, *Exploiting symmetry in solving linear equations*, in ISNM, E. L. A. K. M. Golubitsky, ed., vol. 104, Basel, Switzerland, 1992, Birkhäuser Verlag.

[11] G. Golub and C. van Loan, *Matrix Computations*, The John Hopkins University Press, Baltimore and London, second ed., 1989.

[12] W. Hackbusch, *Multigrid Methods and Applications*, Springer Verlag, Berlin, 1985.

[13] ———, *Integralgleichungen*, vol. 68 of Leitfäden der angewandten Mathematik und Mechanik, B.G. Teubner, Stuttgart, 1989.

[14] 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. 105–126.

[15] J.-P. Serre, *Linear Representations of Finite Groups*, vol. 42 of Graduate Texts in Mathematics, Springer Verlag, Berlin, Heidelberg, New York, 1977.

[16] J. Tausch, *A generalization of the discrete Fourier transformation*, in Exploiting Symmetry in Applied and Numerical Analysis, E. L. Allgower, K. Georg, and R. Miranda, eds., vol. 29 of Lectures in Applied Mathematics, Providence, RI, 1993, American Mathematical Society.

[17] ———, *Perturbation analysis for some linear boundary integral operators*. Preprint, Colorado State University, October 1993.

[18] S. Vavasis, *Preconditioning for boundary integral equations*, SIAM J. Matrix Anal., 13 (1992), pp. 905–925.

[19] J. Walker, *Numerical experience with exploiting symmetry groups for boundary element methods*, in Exploiting Symmetry in Applied and Numerical Analysis, E. Allgower, K. Georg, and R. Miranda, eds., vol. 29 of Lectures in Applied Mathematics, Providence, RI, 1993, American Mathematical Society.

Thursday, April 7

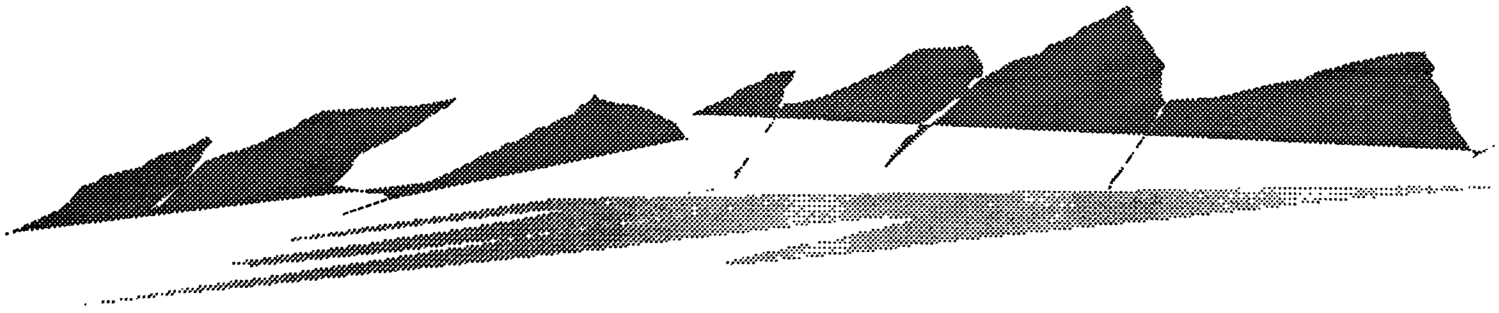
ODE Solvers
Chair: Paul Saylor
Room B

4:45 - 5:10 Vladimir Druskin
Explicit and Implicit ODE Solvers Using Krylov Subspace Optimization: Application to the Diffusion Equation and Parabolic Maxwell's System

5:10 - 5:35 A. Lorber
On the Relationship Between ODE Solvers and Iterative Solvers for Linear Equations

5:35 - 6:00 Andrew Lumsdaine
Krylov-Subspace Acceleration of Time Periodic Waveform Relaxation

6:00 - 6:25 Yimin Kang
Convergence Analysis of Combinations of Different Methods



EXPLICIT AND IMPLICIT ODE SOLVERS USING KRYLOV SUBSPACE OPTIMIZATION: APPLICATION TO THE DIFFUSION EQUATION AND PARABOLIC MAXWELL'S SYSTEM

Vladimir Druskin and Leonid Knizhnerman

Introduction

We solve the Cauchy problem for an ODE system

$$Au + \frac{\partial u}{\partial t} = 0, \qquad u|_{t=0} = \varphi, \tag{1}$$

where A is a square real nonnegative definite symmetric matrix of the order N, φ is a vector from $\mathbf{R}^N$. The stiffness matrix A is obtained due to semi-discretization of a parabolic equation or system with time-independent coefficients. We are particularly interested in large stiff 3-D problems for the scalar diffusion and vectorial Maxwell's equations.

First we consider an explicit method in which the solution of (1) on a whole time interval is projected on a Krylov subspace originated by A. Then we suggest another Krylov subspace with better approximating properties using powers of an implicit transition operator. These Krylov subspace methods generate optimal in a spectral sense polynomial approximations for solution of (1), similar to CG for SLE.

Lanczos Spectral Decomposition of the matrix exponential

Let us perform m steps of the Lanczos method with A and φ. The approximate solution is then [1, 2, 9-11, 12]

$$u = \exp(-tA)\varphi \approx \|\varphi\| Q \exp(-tH)e_1, \tag{2}$$

where Q is the $N \times m$ matrix of the m first Lanczos vectors and H is the $m \times m$ tridiagonal symmetric matrix of the coefficients of the Lanczos recurrence, $e_1 = (1, 0, \ldots, 0)^T$. To reach convergence, one can increase the dimension m just by adding new columns to Q and extending H. The main arithmetical work in (2), connected with obtaining the matrices Q and H, is approximately equal to the one the of m steps of an explicit time-stepping method (m multiplications of A by vectors and $5mN$ additional scalar multiplications). But in spite of strict stability limitations on explicit methods, approximation (2) is unconditionally stable [1,3,7] and requires $O(\sqrt{a \log a})$ steps to converge, where $a \equiv t\|A\|/2$ is the Courant number [1,2,6].

Lanczos decomposition of time-stepping schemes

Eq. (2) is similar to unpreconditioned CG. Consider an approach to preconditioning the Lanczos decomposition.

Suppose $\exp(-\tau A) \approx P_\tau$, $\tau > 0$, then

$$u(\tau i) \approx P_\tau^i \varphi. \tag{3}$$

The transition operator P_τ can be obtained due to implicit time-stepping, for stability reasons its symmetry and the condition $\|P_\tau\| \leq 1$ are required. We perform m steps of the Lanczos method with P_τ and φ and approximate (3) similarly to (2):

$$u(\tau i) \approx \|\varphi\| Q H^i e_1. \tag{4}$$

Approximation of $u(\tau i)$ with (4) requires $m = O(\sqrt{i \log i})$ [4]. This reduction compared with i steps of direct computing (3) is due to good Lanczos approximations for a few well-propagating modes of (3).

To construct a computationally efficient transition operator for multidimensional PDE, we use exponential splitting. Suppose $A = \sum_{l=1}^{k} A_l$, where A_l are nonnegative symmetric operators, then we can take

$$P_\tau \equiv \prod_{l=1}^{k} \exp(-0.5\tau A_l) \prod_{l=1}^{k} \exp(-0.5\tau A_{k+1-l}). \tag{5}$$

Scheme (3, 5) has error $O(\tau^2)$ [11], and after proper selection of τ combination (3–5) requires $m = O(a^{1/4}\sqrt{\log a})$.

Splitting the scalar diffusion equation and 3-D quasistationary Maxwell's system

The multidimensional diffusion equation is split into 1-D problems [Peaceman and Rachford, 1956]:

$$A_i \equiv -\frac{\partial}{\partial x_i}\left(d \frac{\partial}{\partial x_i}\right), \tag{6}$$

where $d(x)$ is heat conduction. Then $\exp(-0.5\tau A_l)$ can be easily computed by means of a suitable Pade approximation, say [1/1] or [1/2].

The spatial operator of quasistationary (eddy current) Maxwell's equations is defined on 3-D vector-functions of the electrical field E:

$$AE \equiv \frac{1}{\sigma}\nabla \times \left(\frac{1}{\mu}\nabla \times E\right), \tag{7}$$

where σ and μ are variable electrical conductivity end magnetic permeability. Because of cross-coupling terms, (7) can not be presented as a sum of 1-D differential operators, similarly to (6). However, we have found that if

$$A_l E \equiv \frac{1}{\sigma}\nabla \times \left(D^l \frac{1}{\mu}\nabla \times E\right),$$

where D^l is a 3×3 matrix, $D^l = \operatorname{diag}(\delta_l^j)$, $l = 1, 2, 3$, then

$$\exp(-0.5\tau A_l)E = I - \frac{1}{\sigma}\nabla \times \left(R^l \frac{1}{\mu}\nabla \times E\right), \qquad (8)$$

where R^l is a 3×3 block operator,

$$R^l \equiv \operatorname{diag}\{\delta_l^j M_l^{-1}[I - \exp(-0.5\tau M_l)]\},$$

and

$$M_l \equiv -\sum_{n \neq l} \frac{1}{\mu}\frac{\partial}{\partial x_i}\left(\frac{1}{\sigma}\frac{\partial}{\partial x_i}\right), \qquad 1 \leq n \leq 3.$$

So, 3-D vectorial problem (8) is reduced to computation of a function of the scalar 2-D elliptic operator M_l. The latter can be done due to second level splitting: combination of 1-D spitting (3, 5, 6) and Lanczos decomposition (4).

Numerical example

For a problem for Maxwell's equations with $N \approx 10^6$ and $a \approx 10^8$ the value of m and CPU time on IBM R6000 have been reached $7 \cdot 10^3$ and 5 h. respectively using the explicit variant of Lanczos decomposition (2). Combination of the splitting and Lanczos decomposition (4) has reduced m to 500 and the CPU time to 0.5 h. We have finally gained speed up of a factor of 1000 compared with conventional time-stepping methods.

References

[1] V.L.Druskin, L.A.Knizhnerman, Dep. at VINITI 02.03.1987, No. 1535 − $B87$(*in Russian*).

[2] V.L.Druskin, L.A.Knizhnerman, U.S.S.R. Comput. Maths. Math. Phys. **29** (1989) No.6, 112-121 *Pergamon Press*

[3] V.Druskin, L.Knizhnerman, U.S.S.R. Comput. Maths. Math. Phys. **31** No.7,(1991) 20-30 *Pergamon Press*

[4] V.Druskin, L.Knizhnerman, CAM , **42** (1992) 221-231.

[6] E.Galloupulos, unpublished manuscript, (1991).

[7] A.Greenbaum, Linear Algebra and Applic. **113** (1989) 7-63.

[8] A.Nauts, R.E.Wyatt, **51** (1983) 2238-2241.

[9] B.Nour-Omid, R.W.Clough, Earthquake Eng. and Structur. Dynamics **12** (1984) 565-577.

[10] T.J.Park, J.C.Light, J. Chem. Phys. **85** (1986) 5870-5876.
[11] Strang SIAM J.Numer.Anal. **5** 507-517, (1968)
[12] H.A. van der Vorst, CAM, **18** (1987) 249-263.

On the Relationship Between ODE Solvers and Iterative Solvers for Linear Equations

A. Lorber, W. Joubert, and G. F. Carey
Computational Fluid Dynamics Laboratory
The University of Texas at Austin

Abstract

The connection between the solution of linear systems of equations by both iterative methods and explicit time stepping techniques is investigated. Based on the similarities, a suite of Runge-Kutta time integration schemes with extended stability domains are developed using Chebyshev iteration polynomials. These Runge-Kutta schemes are applied to linear and non-linear systems arising from the numerical solution of PDE's containing either physical or artificial transient terms. Specifically, the solutions of model linear convection and convection-diffusion equations are presented, as well as the solution of a representative non-linear Navier-Stokes fluid flow problem. Included are results of parallel computations.

KRYLOV-SUBSPACE ACCELERATION OF TIME PERIODIC WAVEFORM RELAXATION*

ANDREW LUMSDAINE†

Abstract. In this paper we use Krylov-subspace techniques to accelerate the convergence of waveform relaxation applied to solving systems of first order time periodic ordinary differential equations. We consider the problem in the frequency domain and present frequency dependent waveform GMRES (FDWGMRES), a member of a new class of frequency dependent Krylov-subspace techniques. FDWGMRES exhibits many desirable properties, including finite termination independent of the number of timesteps and, for certain problems, a convergence rate which is bounded from above by the convergence rate of GMRES applied to the static matrix problem corresponding to the linear time-invariant ODE.

1. Introduction. Consider the problem of numerically solving the linear time periodic boundary value problem for a system of linear time-invariant first order ordinary differential equations:

$$\begin{aligned} \dot{x}(t) + Ax(t) &= f(t) \\ x(0) &= x(T) \end{aligned} \tag{1.1}$$

Here, $A \in \mathbb{R}^{N\times N}$, $f(t) \in \mathbb{R}^N$ is a given right-hand side, and $x(t) \in \mathbb{R}^N$ is the unknown vector to be computed over the simulation interval $t \in [0,T]$. In [7, 8], the authors describe and analyze the waveform relaxation (WR) method applied to solving (1.1) and demonstrate that multigrid techniques can be used effectively to accelerate the convergence of WR.

In this paper we address the question of using Krylov-subspace techniques to accelerate the convergence of WR applied to solving (1.1). We also consider the problem in the frequency domain and present a new class of frequency dependent Krylov-subspace techniques.

2. Waveform Relaxation. The iterates produced by continuous time waveform relaxation based on the splitting (M,N) (i.e., $A = M - N$) satisfy

$$\begin{aligned} \dot{x} + Mx^{k+1} &= Nx^k + f \\ x(0) &= x(T) \end{aligned} \tag{2.1}$$

for $t \in [0,T]$. Equivalently, (2.1) can be expressed in operator form as the iteration

$$x^{k+1} = \mathcal{K}x^k + \psi$$

defined on the space $\mathbb{H} = \mathbb{L}_2([0,T],\mathbb{R}^N)$, with $\mathcal{K} : \mathbb{H} \to \mathbb{H}$ given by

$$(\mathcal{K}x)(t) = e^{-tM}\left(I - e^{-TM}\right)^{-1}\int_0^T e^{(s-T)M}Nx(s)ds + \int_0^t e^{(s-t)M}Nx(s)ds$$

and $\psi \in \mathbb{H}$ given by

$$\psi(t) = e^{-tM}\left(I - e^{-TM}\right)^{-1}\int_0^T e^{(s-T)M}f(s)ds + \int_0^t e^{(s-t)M}f(s)ds$$

The solution x to (1.1) is thus a fixed point of the WR iteration and satisfies the integral operator equation

$$(I - \mathcal{K})x = \psi. \tag{2.2}$$

The operator $\mathcal{K}$ is well defined if $i\omega n \notin \sigma(M)$, where $\omega = \frac{2\pi}{T}$ [8]. Integral operator equations can be similarly defined for the initial-value problem on a finite interval $[0,T]$ and on the half-infinite interval $[0,\infty)$[2]. We will respectively refer to the integral operators so defined as $\mathcal{K}_T$ and $\mathcal{K}_\infty$.

* This work was supported in part by National Science Foundation grant CCR92-09815.

† Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556; `Andrew.Lumsdaine@nd.edu`.

3. Krylov-Subspace Acceleration. In [1], Krylov-subspace techniques are used to accelerate the convergence of WR for solving initial-value problems. The same approach can be used for time periodic problems. As with initial-value problems, the operator $\mathcal{K}$ for time periodic problems is not self-adjoint so only methods suitable for non-self-adjoint operators can be applied. One such method is waveform GMRES (WGMRES), an extension of the generalized minimum residual algorithm (GMRES) [5] to the space $\mathbb{H}$.

Algorithm 3.1 (Waveform GMRES).

```
1. Start: Set $r^0 = \psi - (I - \mathcal{K})x^0$, $v^1 = r^0/\|r^0\|$
2. Iterate: For $k = 1, 2, \ldots$, until satisfied do:
    • $h_{j,k} = \langle (I - \mathcal{K})v^k, v^j \rangle, j = 1, 2, \ldots, k$
    • $w^{k+1} = (I - \mathcal{K})v^k - \sum_{j=1}^{k} h_{j,k}v^j$
    • $h_{k+1,k} = \|w^{k+1}\|$
    • $v^{k+1} = w^{k+1}/h_{k+1,k}$
3. Form approximate solution:
    • $x^k = x^0 + V^k y^k$, where $y^k$ minimizes $\|\beta e_1 - \bar{H}^k y^k\|$
```

As shown in [1], WGMRES applied to WR for the linear initial-value problem converges and the same results can be applied to the time periodic problem. Quantitative convergence estimates for the finite interval initial-value problem are very difficult to obtain. Although the spectrum of $\mathcal{K}_T$ is well defined (the singleton $\{0\}$), the operator is very non-normal and the behavior of an iterative method based on the Krylov subspace generated by $\mathcal{K}_T$ will not be determined solely by the spectrum. (Pseudospectral analysis might be helpful in this regard, however [3, 6].)

On the other hand, the operator $\mathcal{K}$ may actually be normal for cases of interest, and typical convergence results for GMRES applied to normal operators will hold for WGMRES applied to the time periodic problem. Such cases of interest include Jacobi WR with constant diagonal matrix. As shown in [8], the spectrum of $\mathcal{K}$ is given by

$$\sigma(\mathcal{K}) = \bigcup_{n \in \mathbb{Z}} \sigma((in\omega I + M)^{-1}N) \bigcup \{0\}.$$

This is in comparison to the spectrum of $\mathcal{K}_\infty$ which consists of the curve defined by

$$\sigma((i\xi I + M)^{-1}N)$$

and the enclosed points [2, 3]. Note that $\sigma(M^{-1}N) \subset \sigma(\mathcal{K}) \subset \sigma(\mathcal{K}_\infty)$ so we might expect that the convergence rate of WGMRES for the time periodic problem to be better than WGMRES applied to the initial-value problem on the half-infinite interval (or on very long finite intervals) but to be worse than GMRES applied to the matrix problem involving $M^{-1}N$.

4. Frequency Dependent Waveform Methods. The waveform GMRES algorithm is based on minimization of the residual norm $\|r\|$. Instead of seeking to solve (2.2) by minimizing $\|r\|$, one might instead seek to minimize another quantity. In particular, one approach is to independently minimize the contribution to the residual norm made by each of the member functions of some basis set of $\mathbb{H}$. This approach gives rise to the so-called frequency dependent waveform methods, the first example of which, frequency dependent waveform SOR (FDWSOR), is described in [4].

For the periodic boundary value problem, a natural choice of basis is the set of complex exponentials $e^{in\omega t}$. For linear time-invariant problems, one can express (2.2) in terms of its Fourier coefficients

$$\left(I - \hat{\mathcal{K}}[n]\right)\hat{x}[n] = \hat{\psi}[n]. \tag{4.1}$$

where

$$\hat{\mathcal{K}}[n] = (in\omega I + M)^{-1}N.$$

Here, the nth Fourier coefficient of a periodic integrable function x is given by

$$\hat{x}[n] = (\mathcal{F}x)[n] = \frac{1}{T}\int_0^T x(t)e^{-in\omega t}dt$$

Note that (4.1) is a complex matrix problem at each frequency $n\omega$. One could therefore propose frequency dependent WGMRES (FDWGMRES) to solve (4.1).

Algorithm 4.1 (Frequency Dependent WGMRES).

```
1. Start:
    • Calculate Fourier coefficients $\hat{\psi} = \mathcal{F}\psi$, $\hat{x}^0 = \mathcal{F}x^0$
    • Set $\hat{r}^0 = \hat{\psi} - (I - \hat{\mathcal{K}})\hat{x}^0$, $\hat{\beta} = \|\hat{r}^0\|$, $\hat{v}^1 = \hat{r}^0/\hat{\beta}$
2. Iterate: For $k = 1, 2, \ldots$, until satisfied do:
    • $\hat{h}_{j,k} = \langle (I - \hat{\mathcal{K}})\hat{v}^k, \hat{v}^j \rangle$, $j = 1, 2, \ldots, k$
    • $\hat{w}^{k+1} = (I - \hat{\mathcal{K}})\hat{v}^k - \sum_{j=1}^k \hat{h}_{j,k}\hat{v}^j$
    • $\hat{h}_{k+1,k} = \|\hat{w}^{k+1}\|$
    • $\hat{v}^{k+1} = \hat{w}^{k+1}/\hat{h}_{k+1,k}$
3. Form approximate solution:
    • $\hat{x}^k = \hat{x}^0 + \hat{V}^k\hat{y}^k$, where, for each $n$, $\hat{y}^k[n]$ minimizes $\|\hat{\beta}[n]e_1 - \widehat{\bar{H}}^k[n]\hat{y}^k[n]\|$
    • Calculate $x = \mathcal{F}^{-1}\hat{x}$
```

Note that Algorithm 4.1 has many characteristics of a waveform algorithm — in this case, instead of waveform functions of t, the algorithm is operating on sequence space functions of n. Note also that Algorithm 4.1 is not the Fourier transform version of WGMRES. Whereas the projection of $(I - \mathcal{K})$ onto the Krylov subspace generated by WGMRES is a $(k+1) \times k$ real matrix, the projection of $(I - \hat{\mathcal{K}})$ onto the Krylov subspace generated by Algorithm 4.1 is a $(k+1) \times k$ matrix function of the complex quantity $in\omega$.

Since Algorithm 4.1 is essentially the parallel application of GMRES to independent matrix problems, the convergence of the algorithm as a whole will be determined by the slowest converging block. Thus, the discrete time version of the algorithm will exhibit termination in N steps for an $N \times N$ matrix problem, regardless of the number of timesteps. Moreover, for many problems of interest, the worst case convergence will be dictated by the matrix block corresponding to $n = 0$. That is, the convergence will be bounded from above by the worst-case convergence of GMRES applied to the matrix problem with $M^{-1}N$. This is in contrast to WGMRES which does not exhibit finite termination (independent of the number of timesteps) and which has convergence that is in some sense bounded from below by the convergence of GMRES applied to $M^{-1}N$.

5. Experimental Results. In this section, we present the results from numerical experiments using discrete-time versions of WR, WGMRES, and FDWGMRES (discretized in this case with first order BDF on 64 timesteps). Space does not permit description of the discrete time versions of WGMRES and FDWGMRES, but their development is straightfoward. In the experiments, we solve

$$\frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} = f \qquad (5.1)$$
$$u(0,x) = u(T,x)$$

with $T = 2048$, using a spatial discretization of $N = 32$ points, and using a random function for f. For all methods, we take M to be the diagonal of A. For FDWGMRES, the necessary Fourier coefficients are calculated using the fast Fourier transform (FFT) and inverse FFT. Note that since A is linear time-invariant, it is only necessary to calculate the Fourier coefficients associated with the integration formula in order to compute $\hat{\mathcal{K}}$.

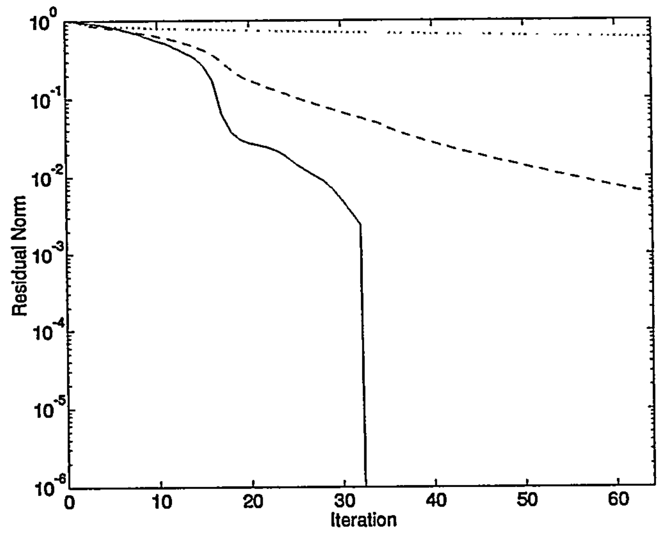


FIG. 5.1. *Convergence comparison between WR (dotted), WGMRES (dashed), and FDWGMRES (solid) applied to solving (5.1)*

For this example, the matrix A is symmetric, positive-definite, and has constant diagonal (so that $M = \alpha I$). It is easy to show that for such a choice of M,

$$\sigma(\hat{\mathcal{K}}[n]) = (\frac{\alpha}{i\omega n + \alpha})\sigma(\hat{\mathcal{K}}[0]) = (\frac{\alpha}{i\omega n + \alpha})\sigma(M^{-1}N).$$

For any $n \neq 0$, the spectrum of $\hat{\mathcal{K}}[n]$ is the spectrum of $M^{-1}N$ with a rotation and a scaling of less than unity magnitude. Since for all n, the eigenvalues of $\hat{\mathcal{K}}[n]$ lie in a straight line in the complex plane, the upper bound on the convergence of FDWGMRES is determined by the spectrum of $\hat{\mathcal{K}}[n]$ when $n = 0$, i.e., by the spectrum of $M^{-1}N$.

Figure 5.1 shows a comparison of the convergence of the residual norm for WR, WGMRES, and FDWGMRES applied to solving (5.1). Note that, as anticipated, WGMRES and FDWGMRES converge much more quickly than WR (with FDWGMRES being better than WGMRES) and that FDWGMRES terminates in N steps.

Figure 5.2 shows a comparison of the convergence of the residual norm for GMRES applied to the static matrix problem corresponding to (5.1), WGMRES applied to the initial-value problem corresponding to (5.1) (we take zero initial condition) and WGMRES and FDWGMRES applied to solving (5.1). As expected, the convergence of GMRES for the static problem lies between FDWGMRES and WGMRES and that of WGMRES for the time periodic problem lies slightly below that for the initial-value problem.

6. Conclusion.

FDWGMRES exhibits many desirable properties, including finite termination independent of the number of timesteps and, for certain problems, a convergence rate which is bounded from above by the convergence rate of GMRES applied to the static matrix problem corresponding to the linear time-invariant ODE. The results shown here are preliminary, but very encouraging, and current work includes the study of the parallel implementation of FDWGMRES as well as its extension to initial-value problems.

In [4], frequency dependent techniques were developed for accelerating waveform SOR applied to finite-interval initial-value problems. The optimal over-relaxation parameter for SOR in the infinite interval case is a function of the spectral radius (as opposed to the entire spectrum) of the waveform iteration operator. Since this spectral radius is a function only of $i\xi$ (for $\xi \in \mathbb{R}$), one can use Fourier transform techniques to

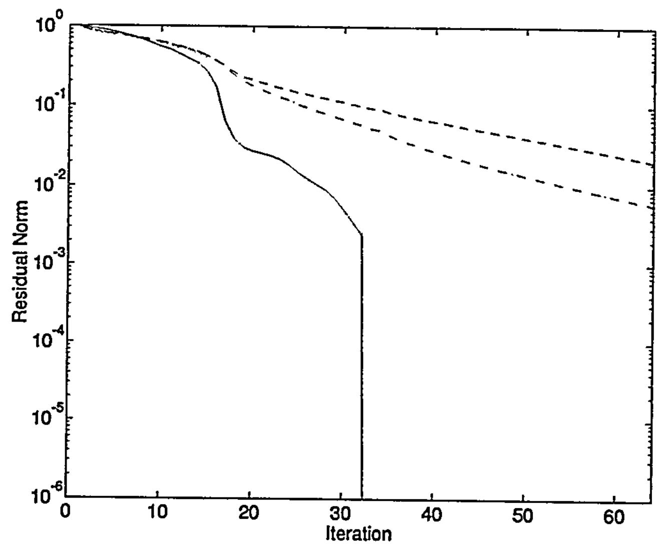


FIG. 5.2. *Convergence comparison between GMRES (dotted) applied to the static matrix problem corresponding to (5.1), WGMRES (dashed) applied to the initial-value problem corresponding to (5.1), and WGMRES (dash-dotted) and FDWGMRES (solid) applied to solving (5.1)*

determine optimal values of the over-relaxation parameter for each ξ. The resulting over-relaxation kernel can be appropriately truncated and applied to finite-interval problems. Applying frequency dependent Krylov-subspace techniques to finite-interval initial-value problems is not as straightforward, primarily because it does not seem to be sufficient in the initial-value problem case to restrict attention only to the boundary of the spectrum of the infinite interval operator (as can be done with FDWSOR). To do so implies periodicity and results in precisely the FDWGMRES algorithm presented here. We are presently investigating combining frequency-dependent WSOR with WGMRES, however, to produce an equivalent to FDWGMRES for initial-value problems.

Acknowledgments. The author would like to acknowledge many helpful discussions with Ken Jackson, Mark Reichelt, and Jacob White.

REFERENCES

[1] A. Lumsdaine, *Theoretical and Practical Aspects of Parallel Numerical Algorithms for Initial Value Problems, with Applications*, PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1992.
[2] U. Miekkala and O. Nevanlinna, *Convergence of dynamic iteration methods for initial value problems*, SIAM J. Sci. Stat. Comp., 8 (1987), pp. 459–467.
[3] S. C. Reddy, *Pseudospectra of operators and discretization matrices and an application to stability of the method of lines*, Numerical Analysis Reports 91-4, MIT, Cambridge, MA, 1991.
[4] M. W. Reichelt, *Optimal frequency-dependent SOR acceleration of waveform relaxation with application to semiconductor device simulation*, in Copper Mountain Conference on Multigrid Methods, Copper Mountain, Colorado, 1993.
[5] Y. Saad and M. Schultz, *GMRES: A generalized minimum residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.
[6] L. N. Trefethen, *Pseudospectra of matrices*, in Proc. 14th Dundee Biennial Conf. on Numer. Anal., 1991.
[7] S. Vandewalle and R. Piessens, *Efficient parallel algorithms for solving initial-boundary value and time-periodic parabolic partial differential equations*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 1330–1346.
[8] ———, *On dynamic iteration methods for solving time-periodic differential equations*, SIAM J. Numer. Anal., 30 (1993), pp. 286–303.

CONVERGENCE ANALYSIS OF COMBINATIONS OF DIFFERENT METHODS

Yimin Kang
Department of Mathematics and Computer Science
Clarkson University
Potsdam, NY

SUMMARY

This paper provides a convergence analysis for combinations of different numerical methods for solving systems of differential equations. We shall prove that combinations of two convergent linear multistep methods or Runge-Kutta methods produce a new convergent method of which the order is equal to the smaller order of the two original methods.

1. INTRODUCTION

The motivation of this work is to study the convergence and stability of semi-implicit time differencing arising from many concrete applications (e.g., [4-5], etc.). For example, consider a time evolution problem in the form

$$y'(t) = a(t, y(t)) + b(t, y(t)).$$

Suppose that the solution of this equation can be decomposed into two time scales: a fast motion and a slow motion, the term $a(t, y(t))$ is most responsible to the fast motion and the term $b(t, y(t))$ makes little contribution to the fast motion. When solving such a system by numerical methods, explicit methods will require extremely small time steps for stability reasons. Implicit methods can remove this restrictive requirement, but need to invert the whole system. Semi-implicit methods, on the other hand, circumvent restrictive stability conditions by treating implicitly the term $a(t, y(t))$ and avoid inverting the whole system by treating explicitly the remaining term $b(t, y(t))$.

The semi-implicit time differencing was first proposed by Robert in [6]. Since then, people have applied the method to different applications. The technique, however, has been only analyzed for specific problems and particular methods. In this paper, we are going to proof a stronger convergence analysis that includes semi-implicit methods as a special case. Namely, we will prove that combination of any two convergent methods of the same type (linear multistep methods or Runge-Kutta methods) yields a new convergent method.

Section 2 proves the convergence for the case of linear multistep methods, section 3 analyzes Runge-Kutta methods. We will follow the notations in [2].

Consider the equation

$$y'(x) = a(x, y(x)) + b(x, y(x)). \tag{1}$$

Throughout the paper, $y(x)$ will always refers to the solution of (1). Whenever necessary, we will use function $f = a + b$. To solve the equation (1) numerically, the term $a(x, y(x))$ and the term $b(x, y(x))$ are treated by two different standard linear multistep methods defined by

Method ($\mathcal{A}$):

$$y_{n+1} = \sum_{j=0}^{m} c_j\, y_{n-j} + h \sum_{j=-1}^{m} a_j\, f(x_{n-j}, y_{n-j}), \tag{2}$$

Method ($\mathcal{B}$):

$$y_{n+1} = \sum_{j=0}^{m} c_j\, y_{n-j} + h \sum_{j=-1}^{m} b_j\, f(x_{n-j}, y_{n-j}), \tag{3}$$

The combination of ($\mathcal{A}$) and ($\mathcal{B}$) leads to a new method

Method ($\mathcal{AB}$):

$$y_{n+1} = \sum_{j=0}^{m} c_j\, y_{n-j} + h \left[\sum_{j=-1}^{m} a_j\, a(x_{n-j}, y_{n-j}) + \sum_{j=-1}^{m} b_j\, b(x_{n-j}, y_{n-j}) \right]. \tag{4}$$

At first, we want to show that the equation (1) can be rewritten in the autonomous form. Let

$$z(x) = \begin{bmatrix} x \\ y(x) \end{bmatrix}, \qquad \tilde{a}\left(\begin{bmatrix} u \\ v \end{bmatrix}\right) = \begin{bmatrix} 1 \\ a(u, v) \end{bmatrix}, \qquad \tilde{b}\left(\begin{bmatrix} u \\ v \end{bmatrix}\right) = \begin{bmatrix} 0 \\ b(u, v) \end{bmatrix},$$

then the equation (1) can be written as

$$z'(x) = \tilde{a}(z(x)) + \tilde{b}(z(x)). \tag{5}$$

If we apply the linear multistep method ($\mathcal{AB}$) to equation (5), we end up with

$$x_{n+\beta_i} = \sum_{j=0}^{m} c_j x_n + h \sum_{j=1}^{s} a_j \tag{6a}$$

$$y_{n+\beta_i} = \sum_{j=0}^{m} c_j\, y_{n-j} + h \left[\sum_{j=-1}^{m} a_j\, a(x_{n-j}, y_{n-j}) + \sum_{j=-1}^{m} b_j\, b(x_{n-j}, y_{n-j}) \right]. \tag{6b}$$

Equation (6a) is exactly the same as (4). It turns out that equation (6b) is equivalent to the requirement that method ($\mathcal{A}$) has order 1 (or it is true for $y(x) = x$), which is under our assumptions anyway. Therefore, applying the linear multistep method ($\mathcal{AB}$) to the autonomous problem (5) is equivalent to applying the method to the original problem (1). Hereafter in this section, we will use the autonomous equation (5). For simplicity, we still use notations y, a and b, so the equation becomes

$$y'(x) = a(y(x)) + b(y(x)), \tag{7}$$

and the terms $a(x_{n-j}, y_{n-j})$, $b(x_{n-j}, y_{n-j})$ and $f(x_{n-j}, y_{n-j})$ in (2)–(4) are replaced by $a(y_{n-j})$, $b(y_{n-j})$ and $f(y_{n-j})$, respectively.

THEOREM 2A For the linear multistep methods $(\mathcal{A})$, $(\mathcal{B})$ and $(\mathcal{AB})$ defined in (2)–(4), if $(\mathcal{A})$ is convergent and of order $p \geq 1$, $(\mathcal{B})$ is also convergent and of order $q \geq 1$, then $(\mathcal{AB})$ is convergent and of order $\min(p, q)$.

Proof. (1). Consistency and order of convergence

In [2], the Taylor expansion of the solution $y(x)$ is given by

$$y(x) = y(x_0) + \sum_{r(t)=1}^{k} \frac{\beta(t)(x-x_0)^{r(t)} F(f,t)(y(x_0))}{[r(t)-1]!} + O\left((x-x_0)^{k+1}\right)$$

where t is a directed tree in graph theory, $r(t)$ is the number of vertex in t, $\beta(t)$ denotes the number of ways of labelling a tree t with $r(t) - 1$ distinct labels on condition that every vertex except the root is labelled, $F(f, t)$ is the *elementary differential* which is defined for $t \in T$ (T denotes the set of directed trees) by

$$F(f, \tau)(y) = f(y)$$

where τ is the tree with a single vertex and by

$$F(f,t)(y) = f^{(s)}(y)(F(f,t_1)(y), F(f,t_2)(y), \ldots, F(f,t_s)(y)),$$

where $t = [t_1, t_2, \ldots, t_s]$, for all $y \in X$. $F(f, t)$ is one of the terms in $y^{r(t)}(x) = \frac{d^{r(t)-1}}{dx^{r(t)-1}} f(y(x))$. (See [2] for details).

We now introduce a new concept *fundamental differential.* For a general function $g : X \to X$, the fundamental differential $D(g, t) : X \to X$, corresponding to $t \in T$ is defined by

$$D(g, \tau)(y) = g(y)$$

where τ is the tree with a single vertex and by

$$D(g,t)(y) = g^{(s)}(y)(F(f,t_1)(y), F(f,t_2)(y), \ldots, F(f,t_s)(y)),$$

where $t = [t_1, t_2, \ldots, t_s]$, for all $y \in X$. $D(f, t)$ is one of the terms in $\frac{d^{r(t)-1}}{dx^{r(t)-1}} g(y(x))$. Clearly, elementary differential is a special kind of fundamental differential when $g \equiv f$. From the definition, one can prove

$$F(f,t) = D(f,t) = D(a+b,t) = D(a,t) + D(b,t), \quad \forall\, t \in T. \tag{8}$$

Follow the arguments in [2], one can prove that the Taylor expansion for the function $g(y(x))$, where $g : X \to X$ is a general function, is given by

$$g(y(x)) = \sum_{r(t)=1}^{k} \frac{\beta(t)(x-x_0)^{r(t)-1} D(g,t)(y(x_0))}{[r(t)-1]!} + O\left((x-x_0)^{k+1}\right)$$

Without losing generality, assume $p \le q$. Therefore, the method $(\mathcal{A})$ and the method $(\mathcal{B})$ are both of order p. According to order conditions for standard linear multistep methods, one has

$$k\sum_{j=-1}^{m}(-j)^{k-1}a_j \;=\; 1-\sum_{j=0}^{m}(-j)^k c_j \;=\; k\sum_{j=-1}^{m}(-j)^{k-1}b_j. \qquad ,k=0,1,2,\ldots,p.$$

For the method $(\mathcal{AB})$, the local truncation error at point x_{n+1} is

$$\begin{aligned}
h\tau_{n+1} &= y(x_{n+1}) - \left[\sum_{j=0}^{m} c_j\, y(x_{n-j}) + h\sum_{j=-1}^{m} a_j\, a(y(x_{n-j})) + h\sum_{j=-1}^{m} b_j\, b(y(x_{n-j}))\right] \\
&= y(x_n) + \sum_{r(t)=1}^{p} \frac{\beta(t)F(f,t)(y(x_n))}{[r(t)-1]!} h^{r(t)} + O(h^{p+1}) \\
&\quad - \sum_{j=0}^{m} c_j \left(y(x_n) + \sum_{r(t)=1}^{p} \frac{\beta(t)F(f,t)(y(x_n))}{[r(t)-1]!}(-jh)^{r(t)} + O(h^{p+1})\right) \\
&\quad - h\sum_{j=-1}^{m} a_j \left(\sum_{r(t)=1}^{p-1} \frac{\beta(t)D(a,t)(y(x_0))}{[r(t)-1]!}(-jh)^{r(t)-1} + O(h^{p})\right) \\
&\quad - h\sum_{j=-1}^{m} b_j \left(\sum_{r(t)=1}^{p-1} \frac{\beta(t)D(b,t)(y(x_0))}{[r(t)-1]!}(-jh)^{r(t)-1} + O(h^{p})\right) \\
&= \sum_{k=0}^{p} h^k \left(1-\sum_{j=0}^{m}(-j)^k - k\sum_{j=-1}^{m} a_j(-j)^k\right) \sum_{r(t)=k} \frac{\beta(t)F(f,t)(y(x_n))}{k!} + O(h^{p+1}) \\
&= O(h^{p+1}).
\end{aligned}$$

Therefore, $\tau_{n+1} = O(h^p)$.

(2). Stability

The stability of linear multistep methods are determined by root conditions of the *characteristic polynomial*

$$\rho(r) = r^{m+1} - \sum_{j=0}^{m} c_j r^{m-j}.$$

Either the method $(\mathcal{A})$ or the method $(\mathcal{B})$ being stable leads to the stability of the linear multistep method $(\mathcal{AB})$. That concludes our proof of the theorem. ♠

Again, the aim is to solve the autonomous equation (7) by numerical methods. (Non-autonomous equations can be rewritten into autonomous form, provided one of the standard Runge-Kutta methods ($\mathcal{A}$) or ($\mathcal{B}$) defined below is of order at least one.) The term $a(y(x))$ and the term $(b(y(x))$ will be treated by two different standard s-stage Runge-Kutta methods defined by

Method ($\mathcal{A}$):

$$y_{n+\beta_i} = y_n + h\sum_{j=1}^{s} a_{ij}\, f(y_{n+\beta_j}), \qquad s = 1, 2, \ldots, s+1, \tag{9}$$

Method ($\mathcal{B}$):

$$y_{n+\beta_i} = y_n + h\sum_{j=1}^{s} b_{ij}\, f(y_{n+\beta_j}), \qquad s = 1, 2, \ldots, s+1. \tag{10}$$

The combination of ($\mathcal{A}$) and ($\mathcal{B}$) is given by

Method ($\mathcal{AB}$):

$$\begin{aligned} y_{n+\beta_i} &= y_n \\ &+ h\left[\sum_{j=1}^{s} a_{ij}\, a(y_{n+\beta_j}) + \sum_{j=1}^{s} b_{ij}\, b(y_{n+\beta_j})\right], s = 1, 2, \ldots, s+1, \end{aligned} \tag{11}$$

THEOREM 3A For the Runge-Kutta methods ($\mathcal{A}$), ($\mathcal{B}$) and ($\mathcal{AB}$) defined in (9)–(11), if ($\mathcal{A}$) is convergent and of order $p \geq 1$, ($\mathcal{B}$) is also convergent and of order $q \geq 1$, then ($\mathcal{AB}$) is convergent and of order $\min(p, q)$.

Before we prove the theorem, let us develop some preliminary lemmas. At a fixed point x_n, define functions

$$Y_i(x) = y(x_n) + (x - x_n)\left[\sum_{j=1}^{s} a_{ij}\, a(Y_j(x)) + \sum_{j=1}^{s} b_{ij}\, b(Y_j(x))\right], 1 \leq i \leq s+1.$$

LEMMA 3B The Runge-Kutta method (11) has order p if and only if $Y_{s+1}^{(m)}(x_n) = y^{(m)}(x_n)$, for $1 \leq m \leq p$, $\forall x_n$.

Proof. This is because the local truncation error

$$\begin{aligned}\tau_{n+1} &= \frac{1}{h}\left\{y(x_{n+1}) - y(x_n) - h\left[\sum_{j=1}^{s} a_{s+1,j}\, a(Y_j(x)) + \sum_{j=1}^{s} b_{s+1,j}\, b(Y_j(x))\right]\right\}\\ &= \frac{1}{h}\left(y(x_{n+1}) - Y_{s+1}(x_{n+1})\right)\\ &= \frac{1}{h}\left(\sum_{m=0}^{p} \frac{y^{(m)}(x_n)}{m!}h^m + O(h^{p+1}) - \sum_{m=0}^{p} \frac{Y_{s+1}^{(m)}(x_n)}{m!}h^m + O(h^{p+1})\right)\\ &= \sum_{m=0}^{p} \frac{y^{(m)}(x_n) - Y_{s+1}^{(m)}(x_n)}{m!}h^{m-1} + O(h^p). \qquad \spadesuit\end{aligned}$$

According to lemma $(3B)$, in order to prove that (11) has order p, it suffices to show that for $1 \le m \le p$, $\forall x_n$. The formula of $y^{(m)}(x)$ is derived in [2]:

$$y^{(m)}(x) \;=\; \sum_{r(t)=m} \alpha(t)F(f,t)(y(x)). \tag{12}$$

where $\alpha(t)$ is the number of ways of labelling t with a given totally ordered set V with $\sharp V = r(t)$. Next, we are going to develop formula for $Y_{s+1}^{(m)}(x)$. Define

$$\tilde{y} \;=\; [Y_1(x), Y_2(x), \ldots, Y_{s+1}(x)]^T$$

so that

$$\tilde{y} \;=\; \tilde{y}(x_n) + (x - x_n)(\tilde{a}(\tilde{y}(x)) + \tilde{b}(\tilde{y}(x)), \tag{13}$$

where $\tilde{y}(x_n) = [y(x_n), y(x_n), \ldots, y(x_n)]^T$ and $\tilde{a} : X^{s+1} \to X^{s+1}$, $\tilde{b} : X^{s+1} \to X^{s+1}$ are defined by

$$\tilde{a}([z_1, z_2, \ldots, z_{s+1}]^T) = [\sum_j a_{1j}a(z_j), \sum_j a_{2j}a(z_j), \ldots, \sum_j a_{s+1,j}a(z_j)], \tag{14}$$

$$\tilde{b}([z_1, z_2, \ldots, z_{s+1}]^T) = [\sum_j b_{1j}b(z_j), \sum_j b_{2j}b(z_j), \ldots, \sum_j b_{s+1,j}b(z_j)]. \tag{15}$$

Define the restriction mapping $R_i : X^{s+1} \to X$ by

$$R_i([z_1, z_2, \ldots, z_{s+1}]^T) \;=\; z_i,$$

then $\tilde{a}$, $\tilde{b}$ are related to a, b by

$$R_i(\tilde{a}(z)) \;=\; \sum_j a_{ij}\, a(R_i(z)), \qquad 1 \le i \le s+1,$$

$$R_i(\tilde{b}(z)) \;=\; \sum_j b_{ij}\, b(R_i(z)), \qquad 1 \le i \le s+1,$$

LEMMA 3C If function a is m times differentiable at each of $R_1z, R_2z, \ldots, R_{s+1}z \in X^{s+1}$, then $\tilde{a}$ defined by (14) is m times differentiable at z and for all $u_1, u_2, \ldots, u_n \in X^{s+1}$ and for $i = 1, 2, \ldots, s+1$,

$$R_i(\tilde{a}^m(u_1, u_2, \ldots, u_n)) = \sum_{j=1}^{s} a_{ij} a^{(m)}(R_j z)(R_j u_1, R_j u_2, \ldots, R_j u_n). \tag{16}$$

Proof: Use induction on m. When $m = 0$, (16) reduces to (14). Assume the result holds for orders of derivatives less than $m > 0$. For fixed $u_1, u_2, \ldots, u_{m-1} \in X$ and $u \neq 0$ we compute

$$\begin{aligned}
&\frac{1}{\| u \|} \| R_i\tilde{a}^{m-1}(z+u)(u_1, u_2, \ldots, u_{m-1}) - R_i\tilde{a}^{m-1}(z)(u_1, u_2, \ldots, u_{m-1}) \\
&\quad - \sum_{j=1}^{s} a_{ij} a^{(m)}(R_j z)(R_j u_1, R_j u_2, \ldots, R_j u_{n-1} R_j u) \| \\
&= \frac{1}{\| u \|} \| \sum_{j=1}^{s} a_{ij} [a^{(m-1)}(R_j(z+u))(R_j u_1, R_j u_2, \ldots, R_j u_{n-1}) \\
&\quad - a^{(m-1)}(R_j(z)(R_j u_1, R_j u_2, \ldots, R_j u_{n-1}) \\
&\quad - a^{(m)}(R_j(z)(R_j u_1, R_j u_2, \ldots, R_j u_{n-1} R_j u) \|
\end{aligned}$$

which tends to zero as $\| \; u \; \| \to 0$ because function a is m times differentiable. Thus $\tilde{a}$ is m times differentiable at z and the derivative is given by (16). ♠

LEMMA 3D If $D(a,t)$ exists for some $t \in T$, and if $z \in X^{s+1}$ is such that $R_i z = y_0$ for $i = 1, 2, \ldots, s+1$, then $D(\tilde{a}, t)$ exists and

$$R_i D(\tilde{a}, t)(z) = \Phi_i(\mathcal{A}, t) D(a, t)(y_0), \quad i = 1, 2, \ldots, s+1. \tag{17}$$

where Φ_i is the so-called *elementary weight* corresponding to the Runge-Kutta method $(\mathcal{A})$. It is defined by

$$\Phi_i(\mathcal{A}, \tau) = \sum_{j=1}^{s} a_{ij},$$

$$\Phi_i(\mathcal{A}, [t_1 t_2 \ldots t_m]) = \sum_{j=1}^{s} a_{ij} \Phi_j(\mathcal{A}, t_1) \Phi_j(\mathcal{A}, t_2) \ldots \Phi_j(\mathcal{A}, t_{s+1}).$$

Proof: By induction. At first, when $\lambda t = \tau$, (17) reduces to (14). For $t = [t_1 t_2 \dots t_m]$, assume that (17) holds with for each of $t_1, t_2, \dots, t_m$, and we have

$$\begin{aligned}
R_i D(\tilde{a}, t)(z) &= R_i \tilde{a}^{(m)}(z)(F(\tilde{f}, t_1)(z), F(\tilde{f}, t_2)(z), \dots, F(\tilde{f}, t_1)(z)) \\
&= \sum_{j=1}^{s} a_{ij} a^{(m)}(y_0)(R_j F(\tilde{f}, t_1)(z), R_j F(\tilde{f}, t_2)(z), \dots, R_j F(\tilde{f}, t_1)(z)) \\
&= \sum_{j=1}^{s} a_{ij} a^{(m)}(y_0)(\Phi_j(\mathcal{A}, t_1) F(f, t_1)(y_0), \dots, \Phi_j(t_m) F(f, t_1)(y_0)) \\
&= \sum_{j=1}^{s} a_{ij} \Phi_j(\mathcal{A}, t_1) \Phi_j(\mathcal{A}, t_2) \dots \Phi_j(\mathcal{A}, t_m) \times \\
&\qquad a^{(m)}(y_0)(F(f, t_1)(y_0), F(f, t_2)(y_0), \dots, F(f, t_m)(y_0)) \\
&= \Phi_i(\mathcal{A}, t) D(a, t)(y_0). \qquad ♠
\end{aligned}$$

Clearly, lemma $(3C)$ and $(3D)$ also hold for function b together with Runge-Kutta method $(\mathcal{B})$.

LEMMA 3E If functions a and b are $n-1$ times differentiable at $y(x_n)$ then

$$Y_{s+1}^{(m)}(x_n) = \sum_{r(t)=m} r(t)\beta(t)(\Phi_{s+1}(\mathcal{A}, t) D(a, t)(y(x_n)) + \Phi_{s+1}(\mathcal{B}, t) D(a, t)(y(x_n)))$$

Proof: Using (13) and the derivative formula for the backward Euler method derived in [2], one has

$$\tilde{y}^{(m)}(x_n) = \sum_{r(t)=m} r(t)\beta_m(t) F(\tilde{f}, t)(y(x_n)),$$

so that, by lemma $(3D)$,

$$\begin{aligned}
Y_{s+1}^{(m)} &= R_{s+1} \tilde{y}^{(m)}(x_n) \\
&= \sum_{r(t)=m} r(t)\beta_m(t) R_{s+1} F(\tilde{f}, t)(y(x_n)) \\
&= \sum_{r(t)=m} r(t)\beta(t)(R_{s+1} D(\tilde{a}, t)(y(x_n)) + +R_{s+1} D(\tilde{b}, t)(y(x_n))) \\
&= \sum_{r(t)=m} r(t)\beta(t)(\Phi_{s+1}(\mathcal{A}, t) D(a, t)(x_n) + +\Phi_{s+1}(\mathcal{B}, t) D(b, t)(x_n)). \qquad ♠
\end{aligned}$$

Proof of Theorem 3A. (1). Consistency and order of convergence

Without losing generality, assume $p \leq q$. Therefore, the method $(\mathcal{A})$ and the method $(\mathcal{B})$ are both of order p. According to order conditions for standard Runge-Kutta methods ([2]), one has

$$\Phi_{s+1}(\mathcal{A}, t) = \Phi_{s+1}(\mathcal{B}, t) = \frac{\alpha(t)}{r(t)\beta(t)} \tag{18}$$

for $r(t) \leq \min(p,q)$. Substitute (18) into lemma $(3E)$

$$\begin{aligned} Y_{s+1}^{(m)}(x_n) &= \sum_{r(t)=m} \alpha(t)(D(a,t)(y(x_n)) + D(a,t)(y(x_n))) \\ &= \sum_{r(t)=m} \alpha(t)F(f,t)(y(x_n)) \\ &= y^{(m)}(x_n) \end{aligned}$$

for all $m \leq \min(p,q)$. Therefore, the Runge-Kutta method $(\mathcal{AB})$ has order $\min(p,q)$, and it is consistent because $\min(p,q) \geq 1$.

(2). Stability

The stability of the Runge-Kutta method $(\mathcal{AB})$ comes from the fact that the method is of at least order 1. According to lemma $(3B)$, $Y'_{s+1}(x_n) = y'(x_n) = f(y(x_n))$. Therefore, a Lipschitz condition holds at x_n as long as function $f(y(x))$ is bounded in the solution region. That leads to the stability of the method. ♠

4. CONCLUDING REMARKS

In this paper, we have proved that applying two different convergent schemes to different terms of the right hand side of a differential equation yields a new convergent numerical method and the order of the new method is the smaller order of the original two methods. Obviously, this idea can be generalized to apply more than two different numerical schemes to different terms in one equation, the combination is of course a convergent method.

As the further work, we will study how semi-implicit methods improve the absolute stability.

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation under Grant No. ATM-9118966.

Special thanks to Professor Scott R. Fulton, who introduced me this interesting problem and did a lot of preliminary investigations on semi-implicit methods.

REFERENCES

1. K. E. Atkinson: *An Introduction to Numerical Analysis*, John, Wiley & Sons, 1989.
2. J. C. Butcher: *The Numerical Analysis of Ordinary Differential Equations — Runge-Kutta and General Linear Methods*, John, Wiley & Sons, 1987.
3. S. R. Fulton: A Semi-Implicit Spectral Method for the Anelastic Equations, *J. of Comput. Phys.*, vol 106, No. 2, 1993, pp. 299-305.
4. J. J. Hack, and R. Jakob: Description of a Global Shallow Water Model Based on the Spectral Transform Method, NCAR Technical Note NCAR/TN-343+STR, Feb 1992, pp. 39.
5. M. Kwizak, and A. J. Robert: A Semi-Implicit Scheme for the Grid Point Atmospheric Models of the Primitive Equations, *Monthly Weather Review*, vol 99, No. 1 Jan. 1971, pp 32-36.
6. A. J. Robert: The integration of a spectral model of the Atmosphere by the implicit method, Proc. WMO/IUGG Symp. on Numerical Weather Prediction, Tokyo. Meteor. Soc. Japan. VII-19-VII-24.

Friday, April 8

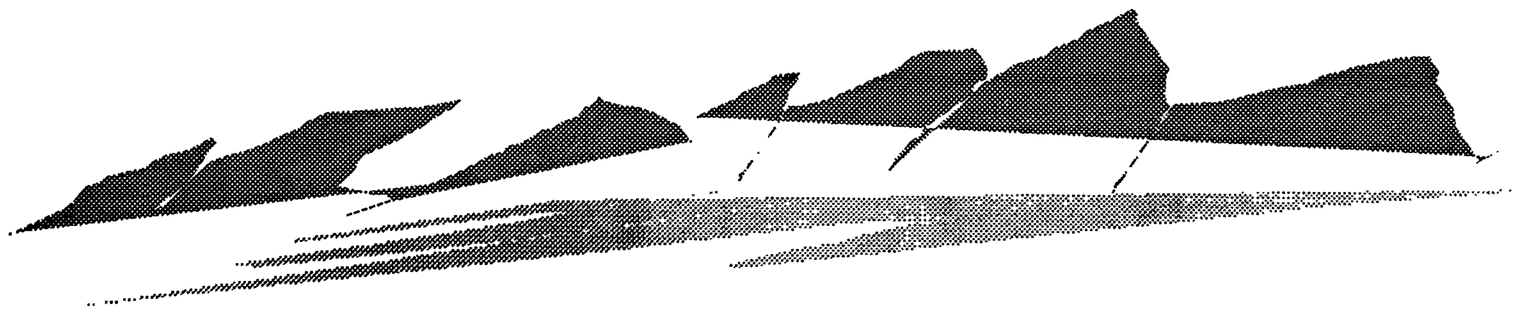
Multigrid and Multilevel Methods I
Chair: Steve McCormick
Room A

8:00 - 8:25 Jian Shen
Implementations of the Optimal Multigrid Algorithm for the Cell-Centered Finite Difference on Equilateral Triangular Grids

8:25 - 8:50 Craig C. Douglas
Constructive Interference II: Semi-Chaotic Multigrid Methods

8:50 - 9:15 Jan Janssen
Multigrid Waveform Relaxation on Spatial Finite Element Meshes

9:15 - 9:40 Stefan Vandewalle
Time-Parallel~Iterative Methods for Parabolic PDEs: Multigrid Waveform Relaxation and Time-Parallel Multigrid



Implimentations of the Optimal Multigrid Algorithm for the Cell-centered Finite Difference on Equilateral Triangular Grids

Richard E. Ewing, Ove Sævareid and Jian Shen

Institute for Scientific Computation, Texas A & M University

College Station, TX 77843-3404

December, 1993

24

Abstract

A multigrid algorithm for the cell-centered finite difference on equilateral triangular grids for solving second-order elliptic problems is proposed. This finite difference is a four-point star stencil in a two-dimensional domain and a five-point star stencil in a three dimensional domain. According to our analysis, the advantages of this finite difference are that it is an $O(h^2)$-order accurate numerical scheme for both the solution and derivatives on equilateral triangular grids, the structure of the scheme is perhaps the simplest, and its corresponding multigrid algorithm is easily constructed with an optimal convergence rate.

We are interested in relaxation of the equilateral triangular grid condition to certain general triangular grids and the application of this multigrid algorithm as a numerically reasonable preconditioner for the lowest-order Raviart-Thomas mixed triangular finite element method. Numerical test results are presented to demonstrate our analytical results and to investigate the applications of this multigrid algorithm on general triangular grids.

For Colorado Conference on Iterative Methods, Breckenridge, Colorado, April 5-9,1994

CONSTRUCTIVE INTERFERENCE II: SEMI–CHAOTIC MULTIGRID METHODS

CRAIG C. DOUGLAS*

Abstract. Parallel computer vendors have mostly decided to move towards multi-user, multi-tasking per node machines. A number of these machines already exist today. Self load balancing on these machines is not an option to the users except when the user can convince someone to boot the entire machine in single user mode, which may have to be done node by node.

Chaotic relaxation schemes were considered for situations like this as far back as the middle 1960's. However, very little convergence theory exists. Further, what exists indicates that this is not really a good method.

Besides chaotic relaxation, chaotic conjugate direction and minimum residual methods are explored as smoothers for symmetric and nonsymmetric problems. While having each processor potentially going off in a different direction from the rest is not what one would strive for in a unigrid situation, the change of grid procedures in multigrid provide a natural way of aiming all of the processors in the right direction.

We present some new results for multigrid methods in which synchronization of the calculations on one or more levels is not assumed. However, we assume that we know how far out of synch neighboring subdomains are with respect to each other. We can show that the combination of a limited chaotic smoother and coarse level corrections produces a better algorithm than would be expected.

* Mathematical Sciences Department, IBM Research Division, Thomas J. Watson Research Center, P. O. Box 218, Yorktown Heights, NY 10598 and Department of Computer Science, Yale University, P. O. Box 208285 New Haven, CT 06520–8285 E-mail: *na.cdouglas@na-net.ornl.gov*

Multigrid Waveform Relaxation on Spatial Finite Element Meshes.

Jan Janssen (janj@cs.kuleuven.ac.be)
Dept. of Computing Science, Katholieke Universiteit Leuven
Celestijnenlaan 200A, B-3001 Leuven, Belgium
and
Stefan Vandewalle (stefan@ama.caltech.edu)
Applied Mathematics 217-50, Caltech
Pasadena, CA 91125, USA

We shall discuss the numerical solution of a parabolic partial differential equation

$$\frac{\partial \mathrm{u}}{\partial t}(x,t) = \mathcal{L}\mathrm{u}(x,t) + \mathrm{f}(x,t) \ , \quad x \in \Omega \ , \quad t > 0 \ , \tag{1}$$

supplied with a boundary condition and given initial values.

The spatial finite element discretization of (1) on a discrete grid Ω_h leads to an initial value problem of the form

$$B\dot{u} + Au = f \ , \quad u(0) = u_0 \ , \quad t > 0 \ , \tag{2}$$

with B a non-singular matrix.

The waveform relaxation method is a method for solving ordinary differential equations. It differs from most standard iterative techniques in that it is a continuous-time method, iterating with functions in time, and thereby well-suited for parallel computation. For systems of the form (2), the method can be defined by the splittings $B = M_B - N_B$, $A = M_A - N_A$, and the iteration scheme

$$M_B\dot{u}^{(\nu)} + M_A u^{(\nu)} = N_B\dot{u}^{(\nu-1)} + N_A u^{(\nu-1)} + f \ , \tag{3}$$

with $u^{(\nu)}(0) = u_0$.

A discrete-time variant can be obtained by discretizing the former scheme in time using a general linear multistep method. Its defining expression for solving the ODE $\dot{y} = f(t,y)$, $y(0) = y_0$ is

$$\frac{1}{\tau}\sum_{j=0}^{k} \alpha_j y_{n+j} = \sum_{j=0}^{k} \beta_j f_{n+j} \ ,$$

with τ a constant time-step and y_i an approximation of the solution y at time-level $t = i\tau$. We obtain the discrete-time equivalent of (3),

$$\frac{1}{\tau}\sum_{j=0}^{k}\alpha_j M_B u_{n+j}^{(\nu)} + \sum_{j=0}^{k}\beta_j M_A u_{n+j}^{(\nu)} = \\ \frac{1}{\tau}\sum_{j=0}^{k}\alpha_j N_B u_{n+j}^{(\nu-1)} + \sum_{j=0}^{k}\beta_j N_A u_{n+j}^{(\nu-1)} + \sum_{j=0}^{k}\beta_j f_{n+j}\ , \quad n \geq 0\ . \tag{4}$$

Both iteration schemes (3) and (4) can be rewritten as explicit relations,

$$u^{(\nu)} = \mathcal{K}u^{(\nu-1)} + \varphi \quad \text{and} \quad u_\tau^{(\nu)} = \mathcal{K}_\tau u_\tau^{(\nu-1)} + \varphi_\tau\ ,$$

where the 'subscript τ' notation denotes sequences of values associated with successive time-levels, i.e., $u_\tau = \{u_i\}_{i=0}^{N_t}$, with N_t the number of time-steps. In terms of the error $e^{(\nu)} = u^{(\nu)} - u$ and its discrete-time variant $e_\tau^{(\nu)}$, we obtain

$$e^{(\nu)} = \mathcal{K}e^{(\nu-1)} \quad \text{and} \quad e_\tau^{(\nu)} = \mathcal{K}_\tau e_\tau^{(\nu-1)}\ . \tag{5}$$

The operators $\mathcal{K}$ and $\mathcal{K}_\tau$ are called the *continuous-time* and the *discrete-time waveform relaxation operators* respectively. The convergence behaviour of both operators is studied by (discrete) Laplace-transformation of the relations (5). In particular, we can proof that

$$\rho(\mathcal{K}) = \sup_{\xi\in I\!R} \rho(\mathbf{K}(i\xi)) \quad \text{and} \quad \rho(\mathcal{K}_\tau) = \sup_{|\xi|=1} \rho\left(\mathbf{K}\left(\frac{1}{\tau}\frac{a}{b}(\xi)\right)\right)\ ,$$

with a and b the characteristic polynomials of the linear multistep formula, and $\mathbf{K}(z) = (zM_B + M_A)^{-1}(zN_B + N_A)$ the waveform relaxation matrix.

The talk will focus on numerical results for a model problem, i.e., the one-dimensional heat equation. We compare the obtained results with the derived theoretical properties for different linear multistep methods and different finite element basis functions. In general, for Gauss-Seidel splittings of both matrices B and A, we observe that $\rho(\mathcal{K}) = 1 - O(h^2)$, i.e., convergence gets slower as the problem size gets larger.

The convergence of the standard waveform relaxation method can be accelerated by the multigrid idea, which is known to be very efficient for solving elliptic PDEs. We will extend this idea to time-dependent problems by choosing all the operations in the multigrid algorithm as operations on functions in time.

We shall first describe a two-grid cycle for problem (2), obtained by finite element discretization of (1) on a discrete grid Ω_h. We need two nested grids $\Omega_H \subset \Omega_h$, a restriction operator r to transform fine-grid functions into coarse-grid functions, and a prolongation operator p to do the opposite. A two-grid cycle for the initial value problem (2) calculates a new fine-grid iterate $u_h^{(k)}$ out of the former iterate $u_h^{(k-1)}$ in three stages: pre-smoothing, coarse-grid correction and post-smoothing.

In the pre-smoothing part, one applies ν_1 standard waveform relaxation steps on the iterate $u_h^{(k-1)}$ to obtain a new approximate $\bar{u}_h$. These waveform relaxation steps turn out to be very efficient to reduce the high-frequency error components. However, they fail in reducing the low-frequency error components, which is the reason why the standard waveform relaxation method is slowly converging.

In a two-grid cycle, we reduce the low-frequency error components by the so-called coarse-grid correction. The correction $e_h = \bar{u}_h - u_h$ satisfies the defect equation $B_h \dot{e}_h + A_h e_h = d_h$. We solve the coarse-grid equivalent of this defect equation, $B_H \dot{e}_H + A_H e_H = r d_h$, interpolate the solution e_H to the fine grid and correct: $\bar{\bar{u}}_h = \bar{u}_h - p e_H$.

Finally, the post-smoothing part applies ν_2 more standard waveform relaxation steps on $\bar{\bar{u}}_h$ to obtain the new fine-grid iterate $u_h^{(k)}$.

By introducing the error $e^{(k)} = u^{(k)} - u$ and its discrete-time variant $e_\tau^{(k)}$, we have the following relations between two successive two-grid errors,

$$e^{(k)} = \mathcal{M} e^{(k-1)} \quad \text{and} \quad e_\tau^{(k)} = \mathcal{M}_\tau e_\tau^{(k-1)} .$$

The operators $\mathcal{M}$ and $\mathcal{M}_\tau$ are called the *continuous-time* and the *discrete-time two-grid waveform relaxation operators* respectively.

Convergence results can be proved, in complete analogy with the standard waveform relaxation case, by (discrete) Laplace-transformation. We obtain

$$\rho(\mathcal{M}) = \sup_{\xi \in I\!R} \rho(\mathrm{M}(i\xi)) \quad \text{and} \quad \rho(\mathcal{M}_\tau) = \sup_{|\xi|=1} \rho\left(\mathrm{M}\left(\frac{1}{\tau}\frac{a}{b}(\xi)\right)\right) .$$

The matrix $\mathrm{M}(z)$ is called the two-grid waveform relaxation matrix, and is given by

$$\begin{aligned}\mathrm{M}(z) &= \mathrm{K}^{(\nu_2)}(z)(I - p(zB_H + A_H)^{-1} r(zB_h + A_h))\mathrm{K}^{(\nu_1)}(z)\\ \mathrm{K}(z) &= (zM_{B_h} + M_{A_h})^{-1}(zN_{B_h} + N_{A_h})\end{aligned}$$

Again, these theoretical results are illustrated by means of our model problem. For a finite element discretization of the one-dimensional heat equation with linear basis functions, we can proof that $\rho(\mathcal{M}) \leq c$, where c is a h-independent constant. This means that the convergence rate of the two-grid waveform relaxation method is independent of the problem size. This result is confirmed by numerical experiments, with several combinations of different basis functions and linear multistep formulae.

Time-parallel iterative methods for parabolic PDES: multigrid waveform relaxation and time-parallel multigrid

Stefan Vandewalle
Caltech, Applied Mathematics 217-50, Pasadena, CA 91125

Extended Abstract
1994 Colorado Conference on Iterative Methods

1 Introduction.

Time-stepping methods for parabolic partial differential equations are essentially sequential. This prohibits the use of massively parallel computers unless the problem on each time-level is very large. This observation has led to the development of algorithms that operate on more than one time-level simultaneously; that is to say, on grids extending in space and in time. The so-called *parabolic multigrid methods* solve the time-dependent parabolic PDE as if it were a stationary PDE discretized on a space-time grid.

In [6, 7], we have investigated the use of *multigrid waveform relaxation*, an algorithm developed by Lubich and Ostermann [1]. The algorithm is based on a multigrid acceleration of waveform relaxation, a highly concurrent technique for solving large systems of ordinary differential equations. Another method of this class is the *time-parallel multigrid method.* This method was developed by Hackbusch in [2], and was recently subject of further study by Horton [3, 4]. It extends the elliptic multigrid idea to the set of equations that is derived by discretizing a parabolic problem in space and in time.

2 Convergence analysis

Although both methods are very closely related, and although both have been used successfully for solving a variety of problems, their convergence properties are very different. In this talk I shall first review previously published convergence results. I will then present some new insights obtained recently by a two-level exponential

Fourier mode analysis. The latter results assist in understanding some observations reported earlier in the literature.

I shall consider in particular the robustness of both methods with respect to the mesh aspect ratio $\Delta t/(\Delta x)^2$, where Δt is the time-increment and Δx is the spatial mesh width. It will be shown that the waveform method is very robust, and attains typical multigrid convergence rates independent of the spatial mesh size, the time-increment or the number of time-steps computed simultaneously. The time-parallel multigrid method, however, is much less robust. Its use is restricted to grids where the time-increment is large compared to the fine grid spatial mesh size.

I will show that the convergence of the time-parallel method can be improved considerably by choosing a different coarsening strategy whenever $\Delta t/(\Delta x)^2$ is below a critical value. A two-level Fourier mode analysis and results of some numerical computations with this method are presented in [5].

3 Parallel implementation and complexity

These methods can be implemented on a message passing multicomputer by using a straightforward grid partitioning. Each process is assigned to a block of unknowns in the combined space-time grid. Extensive timing results illustrate a significant performance gain obtainable with the parabolic multigrid methods when compared to concurrent implementations of a variety of classic time-stepping solvers. In particular, speed-ups over 300 have been obtained on the Intel Delta. For a similar problem, solved on the same space-time grid to a similar accuracy, the speed-up of a standard time-stepping method was limited to about 20.

These methods are also very well suited for implementation on massively parallel systems of SIMD type. Assigning one processing element per grid point in the combined space-time domain leads to algorithms with extremely low parallel complexities. The methods have been implemented on a 32K Connection Machine. Timing results illustrate and confirm the theoretically derived complexity estimates.

References

[1] C. Lubich and A. Ostermann. Multigrid dynamic iteration for parabolic equations. *BIT*, 27:216–234, 1987.

[2] W. Hackbusch. Parabolic multi-grid methods. In R. Glowinski and J.-L. Lions, editors, *Computing Methods in Applied Sciences and Engineering VI*, pages 189–197, Amsterdam, 1984. North Holland.

[3] G. Horton. Time-parallel multigrid solution of the Navier-Stokes equations. In C. Brebbia, editor, *Applications of Supercomputers in Engineering.* Elsevier, August 1991.

[4] G. Horton. The time-parallel multigrid method. *Communic. in Appl.-Num. Meth.*, 8:585–595, 1992.

[5] G. Horton and S. Vandewalle. A space-time multigrid method for parabolic P.D.E.s. Technical Report IMMD 3, 6/93, Universität Erlangen-Nürnberg, Martensstrasse 3, D-91058 Erlangen, Germany, July 1993.

[6] S. Vandewalle and R. Piessens. Numerical experiments with nonlinear multigrid waveform relaxation on a parallel processor. *Applied Numerical Mathematics*, 8(2):149–161, 1991.

[7] S. Vandewalle and R. Piessens. Efficient parallel algorithms for solving initial-boundary value and time-periodic parabolic partial differential equations. *SIAM J. Sci. Stat. Comput.*, Vol 13 (6), Nov. 1992.

Friday, April 8

Applications I
Chair: Jim Morel
Room B

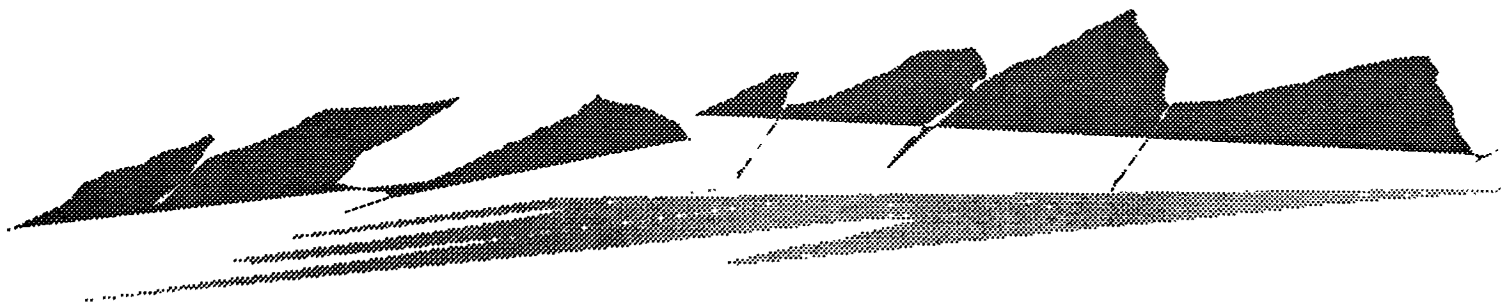
8:00 - 8:25 S.F. Ashby
Modeling Groundwater Flow on Massively Parallel Computers

8:25 - 8:50 M.J. Hagger
Two Grid Iteration With a Conjugate Gradient Fine Grid Smoother Applied to a Groundwater Flow Model

8:50 - 9:15 Jussi Rahola
Solution of Dense-Systems of Linear Equations in Electromagnetic Scattering Calculations

9:15 - 9:40 Tom Cwik
An Iterative Parallel Sparse Matrix Equation Solver with Application to Finite Element Modeling of Electromagnetic Scattering

9:40 - 10:15 Coffee Break



Modeling Groundwater Flow on Massively Parallel Computers

S. F. Ashby, R. D. Falgout, T. W. Fogwell, A. F. B. Tompson

Abstract

We will explore the numerical simulation of groundwater flow in three-dimensional heterogeneous porous media. An interdisciplinary team of mathematicians, computer scientists, hydrologists, and environmental engineers is developing a sophisticated simulation code for use on workstation clusters and MPPs. To date, we have concentrated on modeling flow in the saturated zone (single phase), which requires the solution of a large linear system. We will discuss our implementation of preconditioned conjugate gradient solvers. The preconditioners under consideration include simple diagonal scaling, *s*-step Jacobi, adaptive Chebyshev polynomial preconditioning, and multigrid. We will present some preliminary numerical results, including simulations of groundwater flow at the LLNL site. We also will demonstrate the code's scalability.

Motivation

Groundwater contamination is a major environmental problem at numerous governmental and industrial sites. The Department of Energy, for instance, is currently cleaning up several of its contaminated sites, including Lawrence Livermore National Laboratory (LLNL). Mathematical modeling plays an important role in the design and management of remediation procedures. For example, simulations are used to predict flow behavior and to help engineers determine the best location for pumping wells (in a pump-and-treat scheme).

To enable realistic modeling of large sites, one must take into account the three-dimensional and heterogeneous nature of the subsurface materials. The physical scale of the site to be modeled (several square kilometers), and the need to resolve heterogeneities (to within meters), leads to computational grids with upwards of one billion mesh points. To solve such problems in a reasonable amount of time, one must employ massively parallel computing power and advanced numerical methods.

Two grid iteration with a conjugate gradient fine grid smoother applied to a groundwater flow model

M.J. Hagger[*], K.A. Cliffe[†]and A. Spence[‡]

January 20, 1994

Abstract

This talk is concerned with the efficient solution of $A\mathbf{x} = \mathbf{b}$, where A is a large, sparse, symmetric positive definite matrix arising from a standard finite element discretisation of the groundwater flow problem

$$\nabla.(k\nabla p) = 0$$

Here k is the coefficient of rock permeability in applications is highly discontinuous. The discretisation is carried out using the Harwell NAMMU finite element package, using, for 2D, 9 node biquadratic rectangular elements, and 27 node biquadratics for 3D. The aim is to develop a robust technique for iterative solutions of 3D problems based on a regional groundwater flow model of a geological area with sharply varying hydrogeological properties. Numerical experiments with polynomial preconditioned conjugate gradient methods on a 2D groundwater flow model were found to yield very poor results, converging very slowly. In order to utilise the fact that A comes from the discretisation of a PDE we try the two grid method as is well analysed from studies of multigrid methods, see for example "Multi-Grid

[*]School of Mathematical Sciences, University of Bath, Claverton Down, Bath
[†]Theoretical Studies Department, Harwell Laboratories, Didcot
[‡]School of Mathematical Sciences, University of Bath, Claverton Down, Bath

Methods and Applications" by W. Hackbusch. Specifically we consider two discretisations resulting in stiffness matrices A_N and A_n, of size N and n respectively, where $N > n$, for both a model problem and the geological model. We perform a number of conjugate gradient steps on the fine grid, ie using A_N, followed by an exact coarse grid solve, using A_n, and then update the fine grid solution, the exact coarse grid solve being done using a frontal method factorisation of A_n. Note that in the context of the standard two grid method this is equivalent to using conjugate gradients as a fine grid smoothing step. See, for example, "Analysis and Comparison of Relaxation schemes in Robust Multigrid and Preconditioned Conjugate Gradient Methods" by R. Kettler, appearing in "Multigrid Methods" edited by W. Hackbusch and U. Trottenberg. Experimental results are presented to show the superiority of the two grid iteration method over the polynomial preconditioned conjugate gradient method.

Solution of dense systems of linear equations in electromagnetic scattering calculations

Jussi Rahola
Center for Scientific Computing
P.O. Box 405
FIN-02101 Espoo Finland
rahola@csc.fi

February 15, 1994

Abstract

The discrete-dipole approximation (DDA) is a method for calculating the scattering of light by an irregular particle. The DDA has been used for example in calculations of optical properties of cosmic dust. In this method the particle is approximated by interacting electromagnetic dipoles. Computationally the DDA method includes the solution of large dense systems of linear equations where the coefficient matrix is complex symmetric. In our work, the linear systems of equations are solved by various iterative methods such as the conjugate gradient method applied to the normal equations and QMR. The linear systems have rather low condition numbers due to which many iterative methods perform quite well even without any preconditioning. Some possible preconditioning strategies are discussed. Finally, some fast special methods for computing the matrix-vector product in the iterative methods are considered. In some cases, the matrix-vector product can be computed with the fast Fourier transform, which enables us to solve dense linear systems of hundreds of thousands of unknowns.

References

[1] E.M. Purcell and C.R. Pennypacker, Scattering and absorption of light by nonspherical dielectric grains, *Astrophys. J.*, **186**, 705-714, 1973.

[2] B.T. Draine, The discrete-dipole appriximation and its application to interstellar graphite grains, *Astrophys. J.*, **333**, 848-872, 1988.

[3] J.J. Goodman, B.T. Draine, and P.J. Flatau, Application of fast-Fourier-transform techniques to the discrete-dipole approximation, *Optics Letters*, **16**, No 15, 1198-1200, 1991.

[4] K. Lumme and J. Rahola, Light scattering by porous dust particles in the discrete-dipole approximation, *Astrophys. J.*, to appear.

An Iterative Parallel Sparse Matrix Equation Solver With Application to Finite Element Modeling of Electromagnetic Scattering

Tom Cwik, Vahraz Jamnejad and Cinzia Zuffada
Jet Propulsion Laboratory
California Institute of Technology
Pasadena CA 91109

I. Introduction

The usefulness of finite element modeling follows from the ability to accurately simulate the geometry and three-dimensional fields on the scale of a fraction of a wavelength. To make this modeling practical for engineering design, it is necessary to integrate the stages of geometry modeling and mesh generation, numerical solution of the fields–a stage heavily dependent on the efficient use of a sparse matrix equation solver, and display of field information. The stages of geometry modeling, mesh generation, and field display are commonly completed using commercially available software packages. Algorithms for the numerical solution of the fields need to be written for the specific class of problems considered. Interior problems, i.e. simulating fields in waveguides and cavities, have been successfully solved using finite element methods. Exterior problems, i.e. simulating fields scattered or radiated from structures, are more difficult to model because of the need to numerically truncate the finite element mesh. To practically compute a solution to exterior problems, the domain must be truncated at some finite surface where the Sommerfeld radiation condition is enforced, either approximately or exactly. Approximate methods attempt to truncate the mesh using only local field information at each grid point, whereas exact methods are global, needing information from the entire mesh boundary. In this work, a method that couples three-dimensional finite element (FE) solutions interior to the bounding surface, with an efficient integral equation (IE) solution that exactly enforces the Sommerfeld radiation condition is developed [1]. The bounding surface is taken to be a surface of revolution (SOR) to greatly reduce computational expense in the IE portion of the modeling.

Essential to an efficient solution of the system of equations resulting from the model is a sparse matrix equation solution algorithm. A partitioned system of equations results from combining the finite element modeling of the vector wave equation with the boundary integral equation. The system has a large sparse block resulting from the finite element model, block diagonal components resulting from the integral equation developed on the SOR, and sparse rectangular blocks resulting from the coupling of the two representations. This system is solved for a specific excitation (right-hand-side) using iterative methods appropriate for the general non-symmetric, complex valued nature of the equations. Moreover, the system is intended to be solved on coarse-grained, distributed memory machines.

II. The System of Equations

The scatterer and surrounding space are broken into two regions--an interior part containing the scatterer and freespace region out to a defined surface, and the exterior homogenous part (Figure 1). To efficiently model fields in the exterior region, the surface bounding the interior is prescribed to be a surface of revolution. In this interior region, the weak form of the wave equation is used to model the geometry and fields. The discretization of this equation results in a complex valued, symmetric matrix representing the fields on the mesh. To truncate the mesh and model fields both on the surface and everywhere in the exterior medium an integral equation is used. Fields on the surface are obtained from equivalent tangential currents via an integral over the boundary using the freespace Green's function kernel. Because the symmetry of the SOR is used in this equation, the resulting discretized matrix consists of block diagonal pieces–each block corresponding to the Fourier modes used in the representation of the induced surface currents. It is only necessary to match boundary conditions on the SOR to complete the model. Tangential components of the fields found from the finite element representation are matched with those of the integral equation representation in a weak integral sense to complete this coupling of the fields.

The resulting system of equations is written as

$$\begin{vmatrix} K & C & 0 \\ C^{\dagger} & 0 & Z_0 \\ 0 & Z_M & Z_J \end{vmatrix} \begin{vmatrix} H \\ M \\ J \end{vmatrix} = \begin{vmatrix} 0 \\ 0 \\ V_i \end{vmatrix} \tag{1}$$

where K is the complex valued symmetric finite element matrix, Z_o, Z_m, and Z_j are the complex valued block diagonal matrices resulting from the integral equation modeling, and C and $C^{\dagger}$ are the sparse, complex valued coupling matrices. The dagger represents the Hermitian matrix. H, M and J are the magnetic fields, magnetic and electric currents respectively, and V_i is the excitation vector.

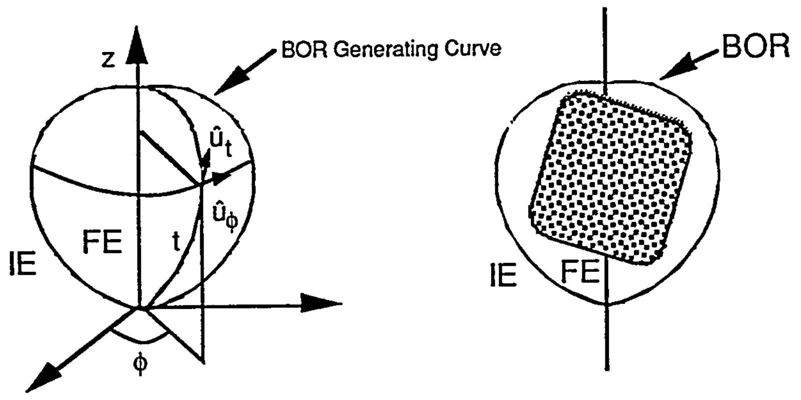


Figure 1. Geometry of scatterer showing interior and exterior regions.

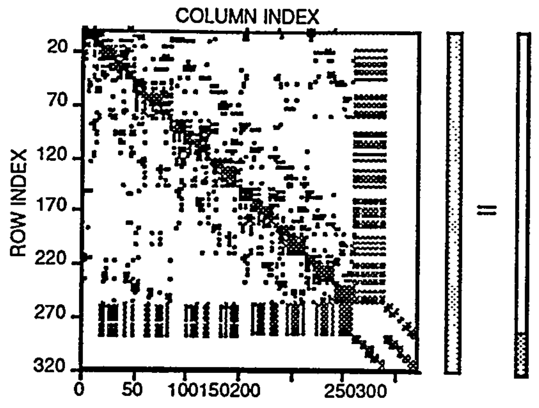


Figure 2. Scatter plot figuratively showing structure of system of equations. Darkened spaces indicate non-zero matrix entries.

Figure 2 shows a representative graphical plot of the sparse equations for a very small system. Dark areas represent non-zero elements of the matrix equation.

III. Solution of the System of Equations

Because three-dimensional modeling is undertaken, it is necessary to not allow fill-in of the matrix as done in direct LU methods. Even skyline storage methods grow exceedingly memory intensive due to the bandwidth of a three-dimensional mesh that is generated around the scatterer. Therefore it is essential to apply iterative methods that only require storage for the matrix and a few extra vectors. The specific partitioned nature of the system leads to two natural methods of solution

• Solve the entire system in one step using the non-symmetric iterative quasi minimum residual (QMR) algorithm [2].

• Solve the system in two steps as follows
1) eliminate H by computing $Z_K = C^\dagger K^{-1} C$
2) solve the system

$$\begin{aligned} -Z_K M + Z_0 J &= 0 \\ Z_M M + Z_J J &= V_i \end{aligned} \quad (2)$$

The first step involves performing the operation $KX = C$, which is accomplished by applying a symmetric QMR iterative algorithm since K is symmetric. The resulting overall matrix (2) is treated as dense, and the solution of this second problem is accomplished via a direct LU decomposition since its size is relatively small.

The first solution requires computing the global matrix-dense vector multiply and transpose matrix-dense vector multiply necessary at each step of the QMR algorithm. This is accomplished block-by-block, multiplying the sparse symmetric K matrix, stored in compressed form, by the appropriate segment of the dense vector; similarly the other components of the system are multiplied in the proper order.

The second solution again requires a sparse symmetric matrix–dense vector multiply in the symmetric QMR algorithm for the initial stage of the solution. The dense matrix Z_k must then be stored and combined with the other blocks for the final LU decomposition stage of the algorithm. Variants of this stage can be made to exploit the properties of the partitioned matrix in (2). It is noted that when solutions for multiple right-hand-side excitations are required, the one-step solution can be prohibitive if the solutions must be performed one at a time. The second method is much more amenable to this case since the excitation only enters in when a the LU decomposition is formed in the significantly smaller system in (2). We also note that the computation in this solution is dominated by the calculation of Z_k since C will have hundreds of columns.

IV. Parallel Implementation

Previously, much experience has been gained in the parallel solution of dense matrix equations resulting from the solution of electromagnetic scattering problems [3,4]. Using coarse-grained, distributed memory machines, the overlap of communication and computation, as well as efficient communication structures for matrix factorization were developed. This work led to high performance algorithms for the dense systems. The extensions to the sparse system outlined above is not as clear. The sparse K matrix has about 15 non-zero entries per row, independent of the order of that matrix–an order that easily reaches into several hundred thousand. For efficient solutions, preconditioners are necessary to reduce the number of iterations, and hopefully wall clock time of the solution.

The first solution outlined above is amenable to parallel implementation on machines such as the Intel Paragon. A scheme to perform the parallel matrix-vector multiplies can be arrived at, allowing solutions that scale with the machine. The K matrix is decomposed onto the processors using a recursive inertial partitioning algorithm [5]; the other blocks are more easily decomposed due to their structure. It appears that preconditioning of this system can not be successfully accomplished.

The second solution is not so easily parallelized due to the two-step nature of the algorithm. The initial stage can be completed as a subset of the first solution outlined above. It is then necessary to rearrange data forming the reduced system (2). This step can require a possible large amount of communication and data storage. In either solution, the performance found in the dense methods mentioned above will not be found in the sparse solvers due to inefficient cache usage resulting from the sparse data. The second solution method can reduce this inefficiency if blocks of the C matrix are operated on simultaneously in the first stage of this algorithm–completing sparse matrix-dense matrix multiplies in the iterative algorithm rather than sparse matrix-dense vector operations. The second solution is also more amenable to preconditioning due to the nature of the K matrix.

This talk will focus on the iterative solvers developed for both solution methods outlined above. Material specific to parallel implementation issues currently underway will also be discussed.

V. References

[1] T. Cwik, V. Jamnejad, and C. Zuffada, "Coupling finite element and integral equation representations to efficiently model large three-dimensional scattering objects," *1993 IEEE International APS Symposium and URSI Radio Science Meeting Digest*, pp. 1756-1759, Ann Arbor, MI, June 27-July 2, 1993.

[2] Freund, R. W. and Nachtigal, N. M., " QMR: a Quasi-minimal Residual Method for non-Hermitian Linear Systems, " Numerische Mathematik, 60, pp. 315-339, 1991.

[3] T. Cwik, J. Patterson, and D. Scott, "Electromagnetic Scattering Calculations on the Intel Touchstone Delta," *Proc. IEEE Supercomputing 92,* Minneapolis MN, pp 538-542, Nov 16-20 1993. (This Gordon Bell Prize Finalist paper is also summarized in *IEEE Computer,* Jan 1993.)

[4] T. Cwik, R. van de Geijn, and J. Patterson, "The Application Of Massively Parallel Computation To Integral Equation Models Of Electromagnetic Scattering, (invited paper)" *J Opt Soc Am B*, accepted for publication, Nov 1993.

[5] R. Calalo, T. Cwik, R. Ferraro, W. Imbriale, N. Jacobi, P. Liewer, T. Lockhart, G. Lyzenga, S. Mulligan, J. Parker, J. Patterson, "Hypercube Matrix Computation Task--Research in Parallel Computational Electromagnetics, JPL Report for 1988-1989," 1989.

Friday, April 8

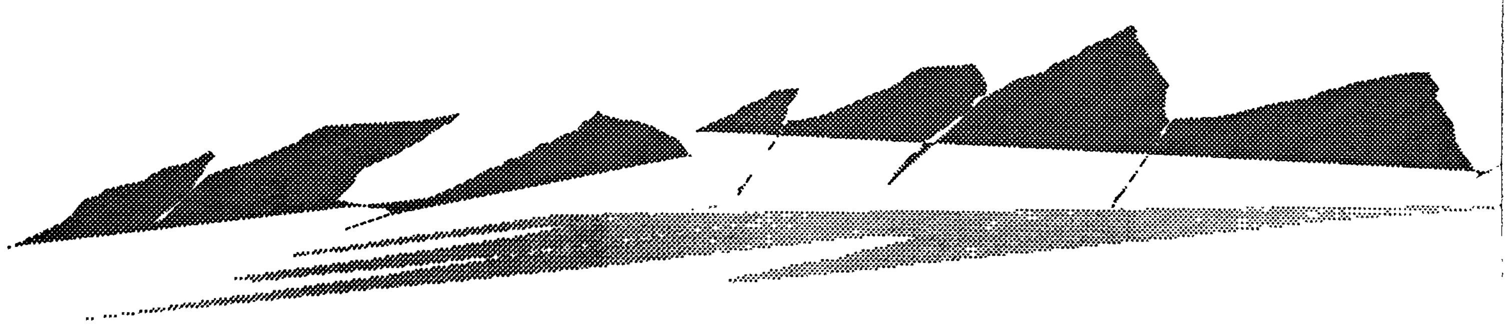
Multigrid and Multilevel Methods II
Chair: Steve McCormick
Room A

10:15 - 10:40 Michael Griebel
On the Relation Between Traditional Iterative Methods and Modern Multilevel/Domain Decomposition Methods

10:40 - 11:05 Irad Yavneh
Multigrid with Red Black SOR Revisited

11:05 - 11:30 Michael Jung
Implicit Extrapolation Methods for Multilevel Finite Element Computations

11:30 - 11:55 Steve McCormick
Multilevel First-Order System Least Squares for PDE'S



[illegible]

ON THE RELATION BETWEEN TRADITIONAL ITERATIVE METHODS AND MODERN MULTILEVEL/DOMAIN DECOMPOSITION METHODS

MICHAEL GRIEBEL
INSTITUT FÜR INFORMATIK, TECHNISCHE UNIVERSITÄT MÜNCHEN
D-80290 MÜNCHEN, GERMANY

Abstract. In recent years, it has turned out that many modern iterative algorithms (multigrid schemes, multilevel preconditioners, domain decomposition methods etc.) for solving problems resulting from the discretization of PDEs can be interpreted as additive (Jacobi-like) or multiplicative (Gauss-Seidel-like) subspace correction methods. The key to their analysis is the study of certain metric properties of the underlying splitting of the discretization space V into a sum of subspaces $V_j, j = 1.., J$ resp. of the variational problem on V into auxiliary problems on these subspaces.

Here, we propose a modified approach to the abstract convergence theory of these additive and multiplicative Schwarz iterative methods, that makes the relation to traditional iteration methods more explicit. To this end we introduce the enlarged Hilbert space $\tilde{V} = V_0 \times \ldots \times V_J$ which is nothing else but the usual construction of the cartesian product of the Hilbert spaces V_j and use it now in the discretization process. This results in an enlarged, semidefinite linear system to be solved instead of the usual definite system.

Then, modern multilevel methods as well as domain decomposition methods simplify to just traditional (block-) iteration methods. Now, the convergence analysis can be carried out directly for these traditional iterations on the enlarged system, making convergence proofs of multilevel and domain decomposition methods more clear, or, at least, more classical. The terms that enter the convergence proofs are exactly the ones of the classical iterative methods. It remains to estimate them properly. The convergence proofs itself follow basically line by line the old proofs of the respective traditional iterative methods. Additionally, new multilevel/domain decomposition methods are constructed straightforwardly by now applying just other old and well known traditional iterative methods to the enlarged system.

Thus, this approach closes the gap between traditional iterative methods and modern multilevel/domain decomposition algorithms. These results may be viewed as an extension of the recent surveys [X, Y] and [GO].

REFERENCES

[X] J.Xu, Iterative methods by space decomposition and subspace correction, SIAM Review 34 (1992) 581-613.
[Y] H.Yserentant, Old and new convergence proofs for multigrid methods. Acta Numerica 1993.
[GO] M.Griebel and P.Oswald, On the abstract theory of additive and multiplicative Schwarz algorithms, submitted to Math. Comp.

Multigrid with Red Black SOR Revisited

Irad Yavneh
Faculty of Computer Science
Technion—Israel Institute of Technology
Haifa 32000, Israel

December 1993

Abstract

Optimal relaxation parameters are obtained for red-black point Gauss-Seidel relaxation in multigrid solvers of a family of elliptic equations. The resulting relaxation schemes are found to retain high efficiency over an appreciable range of coefficients of the elliptic operator, yielding simple, inexpensive and fully parallelizable smoothers in many situations where more complicated and less cost-effective block-relaxation and/or partial coarsening are commonly used.

[illegible]

IMPLICIT EXTRAPOLATION METHODS FOR MULTILEVEL FINITE ELEMENT COMPUTATIONS

MICHAEL JUNG AND ULRICH RÜDE

FACHBEREICH MATHEMATIK
TECHNISCHE UNIVERSITÄT CHEMNITZ-ZWICKAU
D-09009 CHEMNITZ
GERMANY
E-MAIL: DR.MICHAEL.JUNG@MATHEMATIK.TU-CHEMNITZ.DE
E-MAIL: RUEDE@MATHEMATIK.TU-CHEMNITZ.DE

Abstract. The finite element package FEMGP has been developed to solve elliptic and parabolic problems arising in the computation of magnetic and thermomechanical fields. FEMGP implements various methods for the construction of hierarchical finite element meshes, a variety of efficient multilevel solvers, including multigrid and preconditioned conjugate gradient iterations, as well as pre- and postprocessing software. Within FEMGP, multigrid τ-extrapolation can be employed to improve the finite element solution iteratively to higher order. This algorithm is based on an *implicit* extrapolation, so that the algorithm differs from a regular multigrid algorithm only by a slightly modified computation of the residuals on the finest mesh. Another advantage of this technique is, that in contrast to explicit extrapolation methods, it does not rely on the existence of global error expansions, and therefore neither requires uniform meshes nor global regularity assumptions. In the paper we will analyse the τ-extrapolation algorithm and present experimental results in the context of the FEMGP package. Furthermore, the τ-extrapolation results will be compared to higher order finite element solutions.

1. Introduction.

Multigrid methods have been shown to be very efficient solvers for elliptic partial differential equations (PDE). In this paper we are concerned with the so-called τ-extrapolation method, see Brandt [1] and Hackbusch [2]. The τ-extrapolation algorithm is an extension of conventional multigrid that can improve the accuracy of the numerical result by implicitly using higher order approximations. In particular we will show that one step of multigrid τ-extrapolation for piecewise linear C^0 finite elements is equivalent to using quadratic elements.

In this paper we focus on self-adjoint second order linear elliptic partial differential equations of the form

$$Lu = f \quad \text{in} \quad \Omega, \tag{1}$$

where Ω is a polygonal domain in $\mathbb{R}^2$, and suitable boundary conditions. In the examples in Section 3 we will use the equations of elasticity as a typical model problem.

In order to discretize (1) we employ the standard finite element (FE) approach. Consider a family of triangulations $\mathcal{T}_h$ of Ω. The approximation of the solution space by piecewise linear continuous functions leads to a family of discrete systems

$$L_h u_h = f_h. \tag{2}$$

To define a multilevel scheme, we introduce the operator I_{2h}^h to be a linear interpolation from the approximation space on $\mathcal{T}_{2h}$ to $\mathcal{T}_h$. Besides the *weighted restriction* $I_h^{2h} = (I_{2h}^h)^T$ we also use the *injection* operator $\mathbf{I}_h^2 h$ defined by simply dropping the nodal values of the refined mesh. Further note, that the finite element discretization process implies the *Galerkin condition* $L_{2h} = I_h^{2h} L_h I_{2h}^h$.

Our experimental framework is the *Finite Element Multi-Grid Package* FEMGP developed at the Technische Universität Chemnitz-Zwickau for the solution of elliptic and parabolic problems arising in the computation of magnetic and thermomechanical fields. The equivalence of τ-extrapolation to using higher order finite elements justifies its application for unstructured meshes as produced within FEMGP.

2. τ-extrapolation. The classical *full approximation storage multigrid algorithm* (FAS) for the solution of (2) consists of the following steps.

$$\begin{array}{lrcl}
(3) & \text{Smooth} & & L_h u_h = f_h \\
(4) & \tau_{2h}^h & = & L_{2h} \mathbf{I}_h^{2h} u_h - I_h^{2h} L_h u_h \\
(5) & \tilde{f}_{2h} & = & I_h^{2h} f_h + \omega \tau_{2h}^h \\
(6) & \text{Solve} & & L_{2h} u_{2h} = \tilde{f}_{2h} \\
(7) & u_h & = & u_h + I_{2h}^h \left(u_{2h} - \mathbf{I}_h^{2h} u_h \right) \\
(8) & \text{Smooth} & & L_h u_h = f_h
\end{array}$$

For $\omega = 1$ and linear operators L_h and L_{2h} these steps are equivalent to the conventional *correction storage* (CS) multigrid method (see Brandt [1]). The FAS scheme is commonly used as the basis of nonlinear multigrid methods. For $\omega \neq 1$, the FAS method performs an implic[illegible] extrapolation of the equations. In contrast to direct extrapolation methods, this so-called τ-extrapolation method does not require global error expansions, but can be applied even when global expansions are not available.

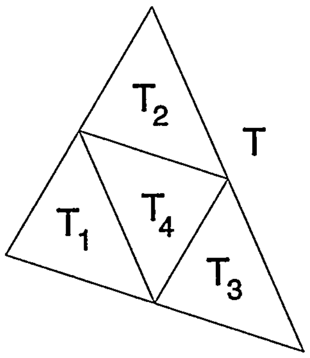


FIG. 1. *Refined element*

τ-extrapolation for these situations can be justified by analysing the structure of the finite element stiffness matrices. Consider a finite element mesh $\mathcal{T}_{2h}$, and a refined mesh $\mathcal{T}_h$. Fig. 1 depicts an element $T \in \mathcal{T}_{2h}$, and $T_1, T_2, T_3, T_4 \in \mathcal{T}_h$. Let $L_{2h}(T)$ denote the element stiffness matrix for linear trial functions in T, and $L_h(T)$ denote the stiffness matrix assembled from the four linear element stiffness matrices $L_h(T_1), L_h(T_2), L_h(T_3), L_h(T_4)$. For a scalar equation $L_{2h}(T) \in \mathbb{R}^{3\times3}$ and $L_h(T) \in \mathbb{R}^{6\times6}$. The refined mesh defines nodal points in $\mathcal{T}_{2h}$ that can also be used as a nodal basis of *quadratic* elements. Denote the corresponding quadratic element stiffness matrix for $\mathcal{T}_{2h}$ by $Q_{2h}(T) \in \mathbb{R}^{6\times6}$. Furthermore, the *injection operator* $\mathbf{I}_h^{2h} : \mathbb{R}^6 \longrightarrow \mathbb{R}^3$ can be used formally to map the nodal values for the refined element to the unrefined one.

For symmetric elliptic operators with constant coefficients in $\mathcal{T}_{2h}$ we now have the identity

$$Q_{2h}(T) = \frac{4}{3} L_h(T) - \frac{1}{3} (\mathbf{I}_h^{2h})^T L_{2h}(T) \mathbf{I}_h^{2h}. \tag{9}$$

Because of the linearity of the assembly process this identity also holds for the global stiffness matrix. Equation (9) can be proved by an elementary, but tedious analysis based on explicitly setting up the stiffness matrices. A complete proof will be given in the full version of this paper.

Another approach for proving (9) is based on asymptotic expansions for quadrature rules over the triangle, see Rüde [9]. This more general analysis shows that the method can be used when the coefficients are not constant. In this case the linear combination constitutes an appropriate numerical quadrature formula for the quadratic element stiffness matrix. This analysis also opens the possibility to generalize this technique to higher order. Some preliminary results in this direction are contained in Rüde [7].

When (9) has been established, it is easy to prove that with $\omega = 4/3$, (4–7) is an iteration with a fixed point u_h defined by

$$Q_{2h}u_h = f_h. \tag{10}$$

To see this, we note that a fixed point of (4–7) satisfies

$$I_{2h}^{h}(u_{2h} - \mathbf{I}_h^{2h}u_h) = 0.$$

This is implied by

$$\tilde{f}_{2h} - L_{2h}\mathbf{I}_h^{2h}u_h = 0$$

and

$$I_h^{2h}\left(f_h - \frac{4}{3}L_h u_h + \frac{1}{3}(\mathbf{I}_h^{2h})^T L_{2h}\mathbf{I}_h^{2h}u_h\right) = 0,$$

which shows (10).

Note that the iteration (4–7) has infinitely many fixed points only one of which is given by Lemma (10). If iteration (4–7) is used in isolation, the limit depends on the initial value with which the iteration is started. If used in combination with the smoothing steps (3) and (8), the limit value becomes uniquely determined, however, the overall iteration converges to a different limit, because now two iterations with different fixed points are combined. A perturbation analysis as in Hackbusch [2] or Rüde [6] can be used to show that the accuracy of the quadratic approximation is only perturbed by negligible higher order terms. Alternatively, smoothers may be developed that are consistent with the higher order stiffness matrix, see McCormick and Rüde [3] and [8].

Finally note that a consistent higher order representation of the right hand side f can be obtained by a further straightforward modification of (5).

3. Numerical Example. The example in this section shows that the FE discretization using linear triangular elements combined with an extrapolation step in the multigrid algorithm leads to the same results as the FE discretization with quadratic elements.

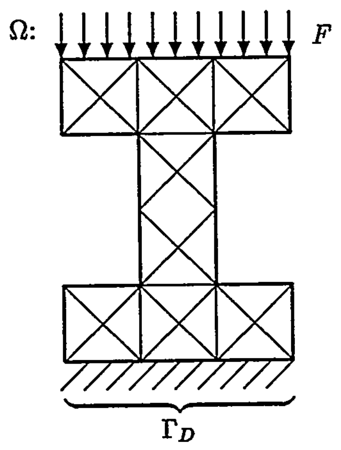
We consider a problem of linear elasticity in the state of plane stress. The starting point for a FE discretization of such a problem is the variational formulation, i.e.

Find the displacement field $u = (u_1, u_2)^T \in V_0$, such that

$$\frac{E}{1+\nu}\int\limits_{\Omega}\left[\frac{\partial u_1}{\partial x_1}\frac{\partial v_1}{\partial x_1}+\frac{\partial u_2}{\partial x_2}\frac{\partial v_2}{\partial x_2}+\frac{\nu}{1-\nu}\operatorname{div}u\,\operatorname{div}v+\right.$$
$$\left.\frac{1}{2}\left(\frac{\partial u_1}{\partial x_2}\frac{\partial u_2}{\partial x_1}+\frac{\partial v_1}{\partial x_2}\frac{\partial v_2}{\partial x_1}\right)\right]dx=\int\limits_{\Gamma_N} g_{2,1}v_1+g_{2,2}v_2\,ds \tag{11}$$

holds for all test functions $v \in V_0$.

Here $g_2 = (g_{2,1}, g_{2,2})^T$ denotes the surface tractions, E is Young's elasticity modulus, and ν is the Poisson ratio. The space V_0 is defined by $V_0 = \{v \in [H^1(\Omega)]^2 : v_1(x) = v_2(x) = 0 \text{ on } \Gamma_D\}$ and $\partial\Omega = \Gamma_D \cup \Gamma_N$. The geometry of Ω, the initial triangulation, and the coefficients used are depicted in Fig. 2.



$$E = 196\ GPa$$
$$\nu = 0.3$$
$$g_{2,1} = 0$$
$$g_{2,2} = \begin{cases} F = 1000\ N & \text{on the upper part of the boundary} \\ 0 & \text{otherwise} \end{cases}$$

FIG. 2. *Shape of domain and data for test problem.*

Starting with the coarsest triangulation which is shown in the figure we generate a sequence of nested triangular meshes $\mathcal{T}_q$, $q = 1, 2, \ldots, l$. Corresponding to each triangulation $\mathcal{T}_q$ we define the FE space V_q spanned by the usual continuous piecewise linear functions. As result of the discretization process we obtain a sequence of systems

$$L_q u_q = f_q, \quad q = 1, 2, \ldots, l,$$

of the algebraic FE equations.

For solving the system $L_l u_l = f_l$ we use a multigrid algorithm that gives us a FE solution with the same discretization error as in the case of a FE discretization with piecewise quadratic function. As mentioned in Section 2 the FE stiffness matrix Q_{l-1} obtained by a FE discretization on the triangulation $\mathcal{T}_{l-1}$ with p–hierarchically quadratic functions is equivalent to the matrix

$$\hat{L}_l = \begin{pmatrix} L_{l-1} & \frac{4}{3}L_{vm} \\ \frac{4}{3}L_{mv} & \frac{4}{3}L_{mm} \end{pmatrix},$$

where the blocks L_{vm}, L_{mv}, and L_{mm} are the parts of the stiffness matrix

$$\begin{pmatrix} L_{l-1} & L_{vm} \\ L_{mv} & L_{mm} \end{pmatrix}$$

which we obtain by means of the two-level hierarchical basis ($(l-1,l)$ hierarchical basis) with piecewise linear functions.

We perform the following steps in our multigrid algorithm.

Algorithm MGEX

```
Let be given the initial guess $u_l^{(1,0)}$. Set $k = 1$.
  1. Transform the vector $u_l^{(1,0)}$ in the two-level hierarchical basis.
  2. For $k = 1, 2, \dots, k_e$:
     (a) Perform $\nu_1$ iteration steps of the Gauss-Seidel method for solving the
         system
             $L_{mm} u_{l,m} = f_{l,m} - L_{mv} u_{l,v}^{(k,0)}$
         such that we get an approximate solution $\tilde{u}_{l,m}$ and $u_l^{(k,1)} = (u_{l,v}^{(k,0)}, \tilde{u}_{l,m})^T$.
     (b) Compute the defect
             $d_{l-1} = f_{l-1} - L_{l-1} u_{l,v}^{(k,1)} - \frac{4}{3} L_{lm} u_{l,m}^{(k,1)}$.
     (c) Solve the system
             $L_{l-1} w_{l-1} = f_{l-1}$
         by means of a usual multigrid ($(l-1)$-grid) algorithm (see. e.g. [2]).
     (d) Compute the new approximation $u_l^{(k,2)}$
             $u_l^{(k,2)} = (u_{l,v}^{(k,1)} + w_{l-1}, u_{l,m}^{(k,1)})^T$.
     (e) Perform $\nu_2$ iteration steps of the Gauss-Seidel method for solving the
         system
             $L_{mm} u_{l,m} = f_{l,m} - L_{mv} u_{l,v}^{(k,2)}$
         such that we get a approximate solution $\tilde{u}_{l,m}$ and $u_l^{(k,1)} = (u_l^{(k,2)}, \tilde{u}_{l,m})^T$.
     (f) Set
             $u_l^{(k+1,0)} = u_l^{(k,3)}$.
  3. Transform the vector $u_l^{(k_e,0)}$ into the usual nodal basis
```

We use this algorithm without steps 1 and 3 for solving the preconditioning systems within the preconditioned conjugate gradient method applied to the system

$$\hat{L}_l u_l = \hat{f}_l, \qquad \hat{f}_l = (f_{l-1}, \frac{4}{3} f_{lm})^T \tag{12}$$

of algebraic linear equations. The resulting algorithm is equivalent to a preconditioned conjugate gradient (PCCG) algorithm for solving the system

$$Q_{l-1} u_l = \hat{f}_l \tag{13}$$

obtained by a FE discretization with p-hierarchical quadratic functions. Table 1 demonstrates this equivalence by showing the computed energies for the extrapolation algorithm compared to the explicit use of a quadratic finite element approximation. The CPU-time needed to obtain the solution of the systems (12) and (13) are almost the same.

TABLE 1
Comparison of τ-extrapolation and quadratic elements

l	energy norm of the solution u_l d ,termined by	
	the algorithm MGEX	the (PCCG) method for the system (13)
3	6.343895	6.343875
4	6.419340	6.419311
5	6.453745	6.453784

4. Conclusion. Starting form the multigrid τ-extrapolation algorithm, we have constructed an iterative method that implicitly generates a higher order approximation. The method is equivalent to using higher order finite elements, but is cheaper because no complicated assembly process is required and the defect evaluation requires only computations with simple low order stiffness matrices. The practical usefulness of the method is shown for a problem in linear elasticity.

REFERENCES

[1] A. BRANDT, *Multigrid techniques: 1984 guide with applications to fluid dynamics*, GMD Studien, 85 (1984).

[2] W. HACKBUSCH, *Multigrid Methods and Applications*, Springer Verlag, Berlin, 1985.

[3] S. MCCORMICK AND U. RÜDE, *On local refinement higher order methods for elliptic partial differential equations*, International Journal of High Speed Computing, 2 (1990), pp. 311–334. Also available as TU-Bericht I-9034.

[4] M.JUNG AND U. LANGER, *Applications of multilevel methods to practical problems*, Surv. Math. Ind., 1 (1991), pp. 217–257.

[5] W. QUECK, *The Finite-Element-Multigrid-Package FEMGP A software tool for solving boundary value problems on personal computers*, in Proceedings of the GAMM-Seminar on Multigrid Methods, Sept. 21 – 25, 1992 in Gosen, Germany, Berlin, 1993, Institut für Angewandte Analysis und Stochastik, pp. 39–48. Report 5, ISSN 0942-9077.

[6] U. RÜDE, *Multiple tau-extrapolation for multigrid methods*, Bericht I-8701, Institut für Informatik, TU München, January 1987.

[7] ———, *Extrapolation and related techniques for solving elliptic equations*, Bericht I-9135, Institut für Informatik, TU München, September 1991.

[8] ———, *The hierarchical basis extrapolation method*, SIAM J. Sci. Stat. Comput., 13 (1992), pp. 307–318. Proceedings of the First Copper Mountain Conference on Iterative Methods, April 1-5, 1990, T. Manteuffel ed.

[9] ———, *Extrapolation techniques for constructing higher order finite element methods*, Bericht I-9304, Institut für Informatik, TU München, 1993.

[10] T. STEIDTEN AND M. JUNG, *Das Multigrid-Programmsystem FEMGPM zur Lösung elliptischer und parabolischer Differentialgleichungen einschließlich mechanisch-thermisch gekoppelter Probleme (Version 06.90)*, Programmdokumentation, Technische Universität Karl-Marx-Stadt, Sektion Mathematik, 1990.

Multilevel First-Order System Least Squares for PDEs

by
Steve McCormick

The purpose of this talk is to analyze the least-squares finite element method for second-order convection-diffusion equations written as a first-order system. In general, standard Galerkin finite element methods applied to non-self-adjoint elliptic equations with significant convection terms exhibit a variety of deficiencies, including oscillations or nonmonotonicity of the solution and poor approximation of its derivatives. A variety of stabilization techniques, such as up-winding, Petrov-Galerkin, and stream-line diffusion approximations, have been introduced to eliminate these and other drawbacks of standard Galerkin methods. Yet, although significant progress has been made, convection-diffusion problems remain among the more difficult problems to solve numerically. The first-order system least-squares approach promises to overcome these deficiencies.

This talk develops ellipticity estimates and discretization error bounds for elliptic equations (with lower order terms) that are reformulated as a least-squares problem for an equivalent first-order system. The main results are the proofs of ellipticity and optimal convergence of multiplicative and additive solvers of the discrete systems.

Friday, April 8

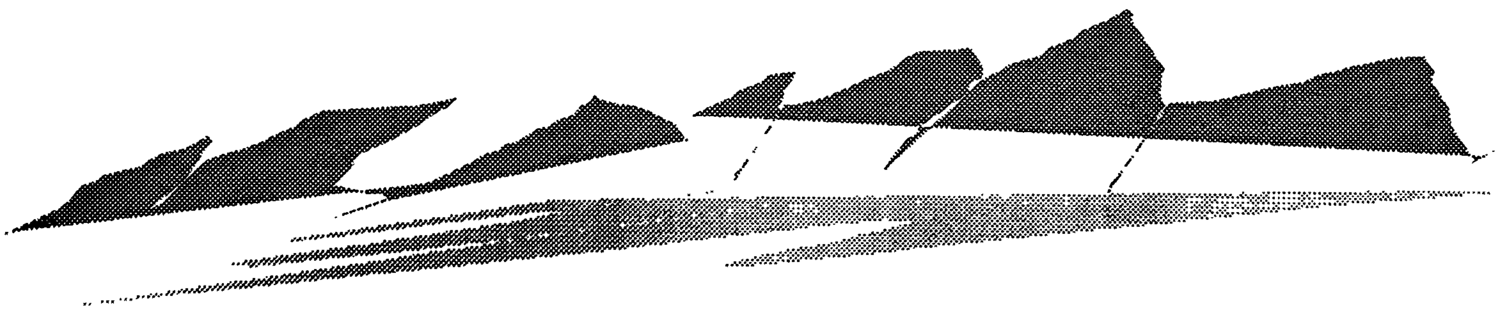
Applications II
Chair: Jim Morel
Room B

10:15 - 10:40 Ray S. Tuminaro
A Multigrid Preconditioner for the Semiconductor Equations

10:40 - 11:05 R. Bauer
Preconditioned CG-Solvers and Finite Element Grids

11:05 - 11:30 Karen R. Baker
Modeling the Diffusion of Phosphorus in Silicon in 3-D

12:00 - 4:30 Informal Discussion



A MULTIGRID PRECONDITIONER FOR THE SEMICONDUCTOR EQUATIONS

JUAN C. MEZA * AND RAY S. TUMINARO †

Currently, integrated circuits are primarily designed in a 'trial and error' fashion. That is, prototypes are built and improved via experimentation and testing. In the near future, however, it may be possible to significantly reduce the time and cost of designing new devices by using computer simulations. To accurately perform these complex simulations in three dimensions, however, new algorithms and high performance computers are necessary.

In this paper we discuss the use of multigrid preconditioning inside a semiconductor device modeling code, DANCIR. The DANCIR code is a full three-dimensional simulator capable of computing steady-state solutions of the drift-diffusion equations for a single semiconductor device and has been used to simulate a wide variety of different devices. At the inner core of DANCIR is a solver for the nonlinear equations that arise from the spatial discretization of the drift-diffusion equations on a rectangular grid. These nonlinear equations are resolved using Gummel's method which requires three symmetric linear systems to be solved within each Gummel iteration. It is the resolution of these linear systems which comprises the dominant computational cost of this code. The original version of DANCIR uses a Cholesky preconditioned conjugate gradient algorithm to solve these linear systems. Unfortunately, this algorithm has a number of disadvantages: 1) it takes many iterations to converge (if it converges), 2) it can require a significant amount of computing time, and 3) it is not very parallelizable.

To improve the situation, we consider a multigrid preconditioner. The multigrid method uses iterations on a hierarchy of grids to accelerate the convergence on the finest grid. To adapt the multigrid method to the drift-diffusion equations, interpolation, projection, and coarse grid discretization operators need to be developed. Developing these operators in the context of the drift-diffusion equations requires some care due to the presence of greatly varying physical phenomena including wide scale variation in PDE coefficients and small scale phenomena such as contact points. In both cases, the development of operator dependent interpolation and projection is essential to improving the performance. Further, the presence of the oxide layer requires some care to insure that the different operators and the grid hierarchy adequately approximate the PDE on all levels. Finally, the presence of a severely stretched grid gives rise to anisotropic phenomena which requires a suitable relaxation procedure.

A two-dimensional version of a multigrid scheme was developed and incorporated into the DANCIR code. The multigrid method can be used with highly variable PDE

* This work was supported by the Applied Mathematical Science Program, U.S. Department of Energy, Office of Energy Research and was performed at Sandia National Laboratories operated for the U.S. Department of Energy under contract No. DE-AC04-94AL85000.

† Centre Européen de Recherche et de Formation Avancée en Calcul Scientifique (CERFACS), Toulouse, France

coefficients, an oxide layer, small contact regions, and anisotropic behavior. The resulting scheme is fast, parallel, and requires many fewer iterations than the ILU scheme that it replaced. On several sample problems ranging from 1,617 to 82,937 unknowns, we improved the performance by a factor of between 2 and 4 over the ILU preconditioner. Extensions to the three-dimensional case are straight-forward and planned for the future.

Finally, we note that while we have used a multigrid solver for the linear equations that arise within Gummel's method, there are potentially much greater savings if the Gummel technique can be replaced by a nonlinear multigrid iteration. We have not pursued this, but we hope that this study will give insight into this possibility.

Preconditioned CG–Solvers and Finite Element Grids

R. Bauer and S. Selberherr
Institute for Microelectronics, Technical University of Vienna
Gusshausstrasse 27–29, A-1040 Vienna, Austria
Phone +43/1/58801-3854, FAX +43/1/5059224, E-mail: bauer@iue.tuwien.ac.at

Introduction

To extract parasitic capacitances in wiring structures of integrated circuits we developed the two– and three–dimensional finite element program SCAP (Smart Capacitance Analaysis Program). The program computes the task of the electrostatic field from a solution of Posson's equation via finite elements and calculates the energies from which the capacitance matrix is extracted. The unknown potential vector, which has for three–dimensional applications 5000 – 50000 unknowns, is computed by a ICCG solver. Currently three– and six–node triangular, four– and ten–node tetrahedronal elements are supported.
The capacitance matrix for a charge balanced n–conductor problem has $n\ (n-1)/2$ entries and can be extracted by $n\ (n-1)/2$ energy runs. For each run it is necessary to apply a linearly independent potential vector to the contacts and calculate the electrostatical energy. Since the Poisson equation is linear it is only necessary to compute n potential vectors and build up the missing potential vectors from old vectors by superposition.
The variational formulation of the solution of the Euler equation

$$\text{div } \varepsilon(x,y,z) \text{ grad } \psi(x,y,z) = 0 \tag{1}$$

is an equivalent formulation of the minimization of the functional

$$I = \varepsilon_0 \int_G \varepsilon_r(x,y,z) \left(\left(\frac{\partial\psi}{\partial x}\right)^2 + \left(\frac{\partial\psi}{\partial y}\right)^2 + \left(\frac{\partial\psi}{\partial z}\right)^2 \right) dV \quad \rightarrow min\ , \tag{2}$$

which represents exactly twice the electrostatic field energy E_{pot}.

Matrix Assembling

For linear shape functions (only for these) we obtain for the assembled stiffness Matrix an M – Matrix (**S** is an M–Matrix if $s_{ii} > 0, s_{ij} \leq 0$ for $i \neq j$, **S** is nonsingular and $\mathbf{S}^{-1} \geq 0$) based on a Delaunay grid in two dimensions. In [6] it is shown by a counterexample that a three–dimensional Delaunay triangulation does not in general satisfy the condition. Additional boundary nodes have to be inserted to achieve a M–Matrix.
For the following discretization on a triangular partitioning of the domain, using linear shape functions, some criterions have to be satisfied in order to obtain a positive interior connection value between two nodes.
The unknown function ψ is approximated by a combination of linear shape functions for the three triangle corner nodes 1 to 3.

$$\psi(\xi,\eta) \;=\; \psi_1\ N_1 + \psi_1\ N_2 + \psi_1\ N_3 \qquad N_1 = 1 - \xi - \eta \qquad N_2 = \xi \qquad N_3 = \eta \tag{3}$$

The transformed formulation for a normalized tirangular element in ξ, η coordinates with inserted shape functions reads

$$I = A\,\psi^T \int_G \frac{\partial \mathbf{N}}{\partial \xi}\,\frac{\partial \mathbf{N}^T}{\partial \xi}\,d\xi d\eta\;\psi + B\,\psi^T \int_G \left(\frac{\partial \mathbf{N}}{\partial \xi}\,\frac{\partial \mathbf{N}^T}{\partial \eta} + \frac{\partial \mathbf{N}}{\partial \eta}\,\frac{\partial \mathbf{N}^T}{\partial \xi}\right) d\xi d\eta\;\psi +$$
$$C\,\psi^T \int_G \frac{\partial \mathbf{N}}{\partial \eta}\,\frac{\partial \mathbf{N}^T}{\partial \eta}\,d\xi d\eta\;\psi\,, \tag{4}$$

with the geometric coefficients

$$\begin{aligned} A &= ((y_3 - y_1)^2 + (x_3 - x_1)^2)/J \\ B &= -((y_3 - y_1)\,(y_2 - y_1) + (x_3 - x_1)\,(x_2 - x_1))/J \\ C &= ((y_2 - y_1)^2 + (x_2 - x_1)^2)/J \\ J &= (x_2 - x_1)\,(y_3 - y_1) - (x_3 - x_1)\,(y_2 - y_1) \end{aligned} \tag{5}$$

integration over the elements yields

$$\mathbf{S}_{\text{el}} = \frac{1}{2}\begin{pmatrix} A + 2\,B + C & -A - B & -B - C \\ -A - B & A & B \\ -B - C & B & C \end{pmatrix}. \tag{6}$$

(7)

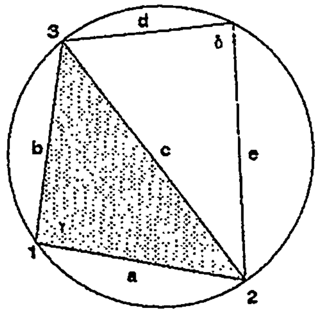


Figure 1: Delaunay criterion

All entries $s_{el\;ii} > 0$ of the stiffness matrix of this element fullfill the M–Matrix criterion. For the off–diagonals, for instance s_{23} the matrix entry in the element matrix becomes positive if the angle opposite two nodes (in this example γ becomes obtuse. For the corresponding in the global stiffness matrix we have to examine the edge c which is shared amongst two triangles. To obtain an M–Matrix the sum of the entries in the global stiffness matrix at each connection has to be negative! Assuming the vertex numbering shown in Fig. 1, we obtain the following criterion for a positive connectivity between nodes 2 and 3.

$$\frac{\mathbf{a}\cdot\mathbf{b}}{|\mathbf{a}\times\mathbf{b}|} + \frac{\mathbf{d}\cdot\mathbf{e}}{|\mathbf{d}\times\mathbf{e}|} > 0 \qquad \cot\,\gamma + \cot\,\delta > 0 \qquad \text{or} \qquad \gamma + \delta < \pi \tag{8}$$

This equation assumes that we have the same material for the two adjacent triangles. In the case of different permittivities in different segments we have to rewrite the equation to

$$\varepsilon_1\,\cot\,\gamma + \varepsilon_2\,\cot\,\delta > 0\,. \tag{9}$$

As a conclusion we can see that for the general case even the Delaunay criterion or sphere criterion in two dimensions does not guarantee an M–Matrix as stiffness matrix. For higher order shape functions, the element stiffness matrix have positive and negative off–diagonal entries and therefore are no M–Matrices.

Preconditioner

For preconditioning the stiffness matrix a CG algorithm an incomplete Cholesky factorization in conjunction with the Eisenstat trick [3] is used. If **S** is not an M–Matrix the Cholesky decomposition will certainly not give a regular splitting. As consequence the factorization of **S** is not always stable.
This has observed for some examples with quadratic shape functions and bad grid quality. To overcome this non–M–Matrix problem Manteuffel [5] introduced a damping factor for the off diagonal elements for the preconditioner. For the diagonally scaled stiffness matrix

$$\mathbf{S} \;=\; \mathbf{I}-\mathbf{B}\,, \qquad \mathbf{S}_d = \mathbf{I} - \frac{1}{1+\alpha}\,\mathbf{B} \tag{10}$$

with a sufficiently large α, $\mathbf{S}_d$ will be diagonally dominant. Experiments have shown that an α of 1.5 is an appropriate value for our problems. On the other hand it is evident that the solver looses efficiency by this method.
We observed that only for some nodes in the factorization the recursion starts to get negative

$$\bar{\mathbf{D}} = \mathbf{D} - diag\,(\mathbf{L}\cdot\bar{\mathbf{D}}^{-1}\cdot\mathbf{U}) \tag{11}$$

and the CG–Solver diverges. Therefore a simple idea is to eliminate these negative entries which arise from positive off–diagonal entries and to revert the sign of negative $diag\,(\mathbf{L}\;\bar{\mathbf{D}}^{-1}\;\mathbf{U})$ contributions and proceed. For our examples with the discretized Laplace operator we achieved always convergence. Node ordering for bandwidth reduction improves the solver speed but decreases the overall performance.

Example

The capacitance matrix for the following three–dimensional crosstalk example of two bitlines of a DRAM–Cell can be extracted by three runs with different applied contact potentials. To reduce the crosstalk to other bitlines the trick of twisted pairs will be used here. The disadvantage of the method is that the parasitic capacitances are increased and therefore more line driver current is required. Fig. 2 shows the significant parts of the arrangement: a ground plane (1), the bitline with contact pads (2) and a second bitline (3). The conductors are assumed to have infinite conductivity and are representing Dirichlet boundary conditions. Fig. 3 shows the whole discretized domain. On the outside boundary homogenous Neumann boundary conditions are assumed except from the bottom plane which is a Dirichlet contact.

On an HP755 we got the following results:

	17673 Tetrahedrons		141384 Tetrahedrons
Shape Functions	linear	quadratic	linear
Matrix size:	(3664, 7.23)	(26480, 13.93)	(26480, 7.56)
Cap_{11}	$5.81 \cdot 10^{-16} F$	$5.29 \cdot 10^{-16} F$	$5.43 \cdot 10^{-16} F$
Cap_{13}	$1.64 \cdot 10^{-16} F$	$1.60 \cdot 10^{-16} F$	$1.61 \cdot 10^{-16} F$
Cap_{23}	$4.11 \cdot 10^{-16} F$	$3.39 \cdot 10^{-16} F$	$3.58 \cdot 10^{-16} F$
CG			
num. iterations	80	186 (186) [1]	162
used time	1.40s	40s (40s)	23.0s
ICCG			
num. iterations	38	92 (77)	83
used time	1.50s	24.0s (21s)	15.0s

References

[1] Bauer, R., and Selberherr, S. Calculating Coupling Capacitances of Three–Dimensional Interconnections. ICSICT '92 Beijing, China, pp. 697–702.

[2] Bauer, R., and Selberherr, S. Capacitance Calculation of VLSI Multilevel Wiring Structures. VPAD '93, Nara, Japan, pp. 142–143.

[3] Eisenstat, S. C. Efficient Implementation of a Class of Preconditioned Conjugate Gradient Methods. SIAM J. SCI. STAT. COMPUT. Vol. 2, No.1, 1981, pp. 1 – 4.

[4] Bank, R. E. PLTMG: A Software Package for Solving Elliptic Partial Differntial Equations. Users' Guide 6.0 SIAM, ISBN 0898712556.

[5] Manteuffel T. A. Shifted Incomplete Cholesky–Factorization. Sparse Matrix Proceedings 1978 SIAM, pp. 41–46.

[6] Letniowski, F. W. Three–Dimensional Delaunay Triangulations for Finite Element Approximations to a Second–order Diffusion Operator SIAM J. SCI. STAT. COMPUT.,Vol. 13, No.3, 1992, pp. 765–770.

[1] CutHill–McKee node ordering

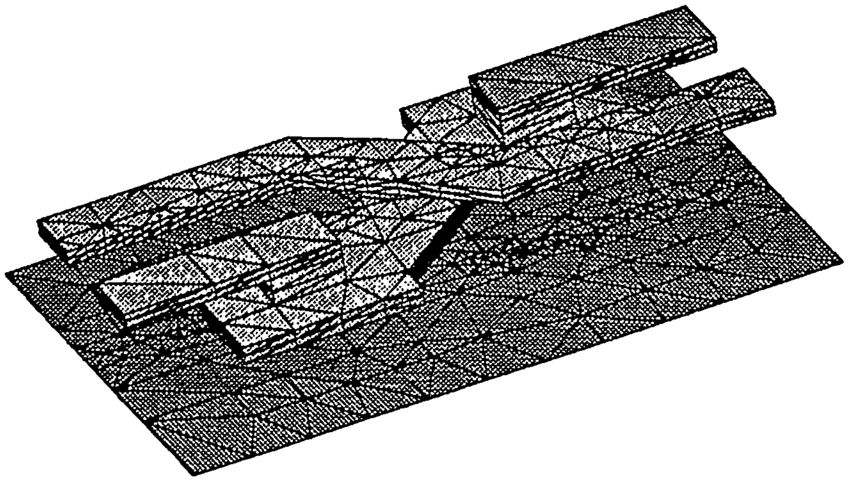


Figure 2: Crossed bitlines with contact pads

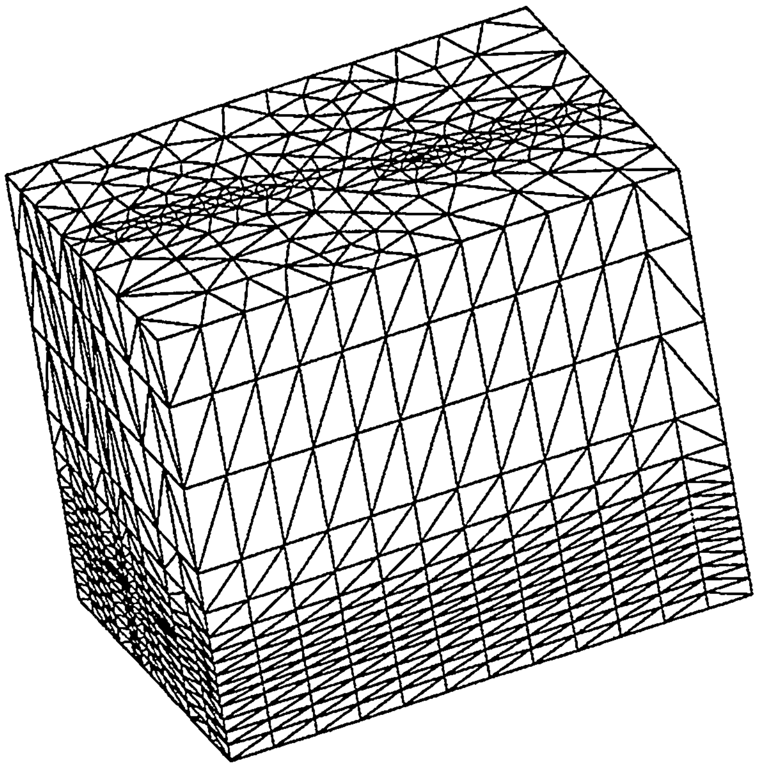


Figure 3: Tetrahedronal grid

Modeling the Diffusion of Phosphorus in Silicon in 3-D

KAREN R. BAKER
The University of Texas at Austin
Department of Mathematics
Austin, Texas 78713 U.S.A.

December 23, 1993

Abstract: **The use of matrix preconditioning in semiconductor process simulation is examined. The simplified nonlinear single-species model for the diffusion of phosphorus into silicon is considered. The experimental three-dimensional simulator, PEPPER3, which uses finite differences and the numerical method of lines to implement the reaction-diffusion equation is modified to allow NSPCG to be called to solve the linear system in the inner Newton loop. Use of NSPCG allowed various accelerators such as Generalized Minimal Residual (GMRES) and Conjugate Gradient (CG) to be used in conjunction with preconditioners such as Richardson, Jacobi, and Incomplete Cholesky.**

I. INTRODUCTION

Silicon is presently the most important semiconductor in the electronics industry. One of the first terms encountered in the study of semiconductor process simulation is VLSI. This is an abbreviation for very large scale integration and refers to integrated circuits containing tens of thousands to several million transistors on a silicon chip with device critical dimensions less than a half micron. Device dimensions have continued to shrink over the past few years increasing the need for accurate simulation tools for process modeling. Process simulation consists of implementing a mathematical model of a physical process on a computer and then using the implementation to represent the actual physical procedure. This is very cost effective for the industry since the number of wafer split-runs necessary during the fabrication phase of the integrated circuit is greatly reduced [3]. It is through a sequence of process steps that the resistors, capacitors, and transistors are formed on a silicon wafer.

II. THE PHYSICAL MODEL

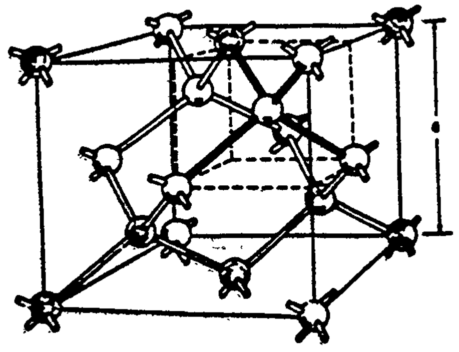
This paper addresses only doping profile simulation and, in particular, the doping profiles obtained while modeling

*Supported by the National Science Foundation, Grant 26-1002-0820, University of Texas at San Antonio, Advisor: Dr. Walter Richardson the diffusion of phosphorus into silicon in three dimensions. It is important to accurately model this type of diffusion since phosphorus is used as a dopant in the lightly doped extension region of LDD (Lightly Doped Drain) structures in NMOS transistors [3]. For intrinsic (low concentration) diffusion, the linear single-species diffusion equation, $\frac{\partial C}{\partial t} = D\Delta C$, is adequate. In this case, the impurity concentration is less than the intrinsic electron concentration, n_i, which is approximately 10^{18} cm^{-3} at 1000°C. The profile for this model is essentially Gaussian. For extrinsic (high concentration) diffusion, the profile exhibits anomalous behavior in the form of kinks, shoulders, and tails. If this unusual behavior is to be explained, it is necessary to model a nonlinear diffusion process.

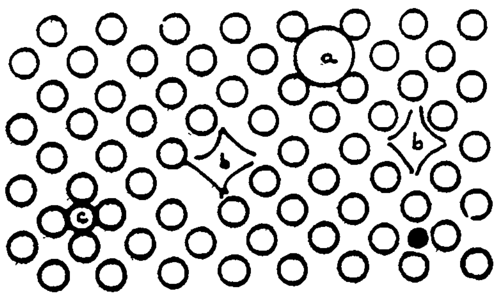
The starting silicon material (wafers) on which the circuits are fabricated have what is called a single crystal form which means there is a periodic arrangement of atoms throughout the entire solid. Its lattice is diamond cubic as shown in Figure 1(a). Suppose phosphorus, P, is diffusing into the silicon. As an illustration as to how the lattice is perturbed, consider the simplified 2-D slice shown in Figure 1(b). A substitutional phosphorus atom is in lattice site "a," site "b" is a vacancy, and site "c" is an interstitial, a silicon atom between lattice sites. All of these are examples of point defects.

Over the past few years, movement has been away from phenomenological models to point defect models. Yoshida, Fair, and others postulated that diffusion is more than a simple random-walk nearest neighbor interchange mechanism—it is vacancy assisted. Specifically, in

SILICON LATTICE



(a) Diamond Cubic Lattice



(b) 2-D Slice of Perturbed Lattice

Figure 1: (a) shows the silicon lattice. In (b), "a" is the site of a substitutional phosphorus atom, "b" is a vacancy, and "c" is an interstitial. ©Lattice Press, 1986 [4]

1971, Yoshida proposed that the diffusion mechanism was formation of phosphorus-vacancy pairs (E centers) which diffused through the lattice. His model incorporated the fact that vacancies exist in various charge states and in concentrations proportional to the Fermi level [5]:

$$[V^-] = [V^-]_i \exp\left(\frac{E_F - E_i}{kT}\right).$$

Thermodynamically, it is more likely for an impurity atom to bond with a vacancy than to exchange sites directly with a neighbor. Consequently, diffusivity is concentration-dependent via

$$n \gg n_i \;\Rightarrow\; [V^-] \gg [V^-]_i \;\Rightarrow\; D \gg D_i .$$

These assumptions result in a compound diffusion coefficient given by

$$D(n) = \qquad (1)$$

$$D_0 + D_- \left(\frac{n}{n_i}\right) + D_= \left(\frac{n}{n_i}\right)^2 + D_+ \left(\frac{n_i}{n}\right).$$

Thus, diffusivity is dependent on concentration of excess electrons or, more simply stated, on the concentration of the dopant. For phosphorus, D_+ is negligible. Under the assumptions of thermal equilibrium, $np = n_i^2$, and charge neutrality, $n + C_A = p + C_D$, the nonlinear single-species diffusion model for phosphorus becomes

$$\frac{\partial C}{\partial t} = \nabla \cdot \{D(C)[\nabla C + C\nabla(\ln n)]\} \quad (2)$$

where C is the impurity concentration and $n(C) = \frac{1}{2}\left(C + \sqrt{C^2 + 4n_i^2}\right)$ is the electron concentration. Since it is the least expensive to implement and run, Equation 2 is the most widely used model for nonlinear diffusion. This is the nonlinear single-species model implemented in PEPPER3, the experimental process simulator originally developed by Dr. Walter Richardson. PEPPER3 is the driver for the code which will be used to solve Equation 2 in 3-D. In Figure 2, the profile data obtained empirically by Yoshida is shown as well as the pseudo one-dimensional profile produced by PEPPER3 ([6], [2]). Note the plateau, kink, and tail in the phosphorus profile of the experimental data.

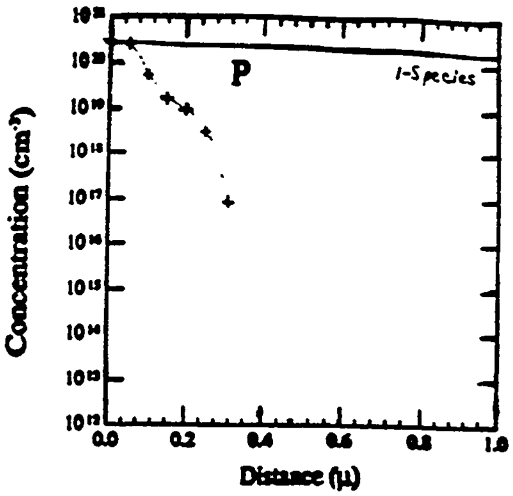


Figure 2: **A phosphorus predeposition simulation at 900°C for 10 minutes using the single-species model is shown. The experimental data is from [6].**

III. THE NUMERICAL ASPECTS

The partial differential equation of the nonlinear single-species is discretized using the Numerical Method of Lines

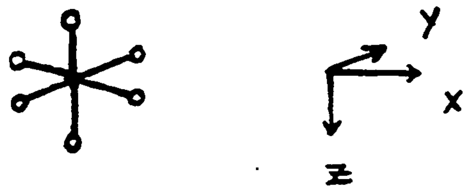


Figure 3: **Seven-point Finite Difference Stencil for the Spatial Discretization**

(NMOL). This method converts a partial differential equation such as

$$\frac{\partial C}{\partial t} = \nabla \cdot (D(C)\nabla C) + F(x,t,C)$$

to a large system of coupled ordinary differential equations which is continuous-in-time but discrete-in-space. A seven-point finite difference stencil is used for the spatial discretization. See Figure 3. Concentration dependent diffusivities are evaluated midway between adjacent grid points. For simplicity, assume that $\{k_i\}_{i=0}^{n}$ is the grid spacing in the x-direction. Then,

$$(D(C)C_x)_x \approx \frac{D(C_{i+\frac{1}{2}}) \cdot \frac{C_{i+1}-C_i}{k_i} - D(C_{i-\frac{1}{2}}) \cdot \frac{C_i - C_{i-1}}{k_{i-1}}}{\frac{k_i+k_{i-1}}{2}} .$$

A quasi-uniform mesh has an accuracy of $\mathcal{O}(k^2)$, and extension to the three-dimensional case is straightforward. The magnitude of the drift term in the nonlinear model of Equation 2 is small, so "upwinding" is not required. Under the assumption of charge neutrality, its effect may be accounted for by using the field enhancement factor,

$$\tilde{D}(C) = D(C)\left(1 + \frac{C}{\sqrt{C^2 + 4n_i^2}}\right).$$

This simplifies the implementation. NMOL is effective since there are no large first order spatial derivatives, i.e. convective or drift terms, present. Three types of test problems of increasing numerical difficulty are included in PEPPER3. Each test case represents a predeposition at 900°C in which phosphorus diffuses from the wafer surface. The initial concentration of phosphorus is zero in the single-species model. Reflecting boundary conditions are enforced on the bottom and the four sides and an appropriate Dirichlet condition is imposed on the top surface. In Problem A, phosphorus diffuses in from the entire surface. Since this is in effect a one-dimensional diffusion, PEPPER3 output can be checked against that of a commercial one-dimensional simulator such as FLASH. In Problem B, the phosphorus diffuses through two narrow strips on the top surface. This problem represents a 2-D diffusion solved numerically in 3-D. Again output can be checked against a commercial 2-D simulator such as SSUPREM4. Problem C requires solution in three dimensions since phosphorus is diffusing through an L-shaped region on the wafer surface and symmetry in the physical domain is lost. See Figure 4.

To solve the nonsymmetric linear system of the Quasi-Newton loop, it is advisable to use a projection method rather than a direct or relaxation method. The linear system that arises from the discretization is large and sparse; so, a di-

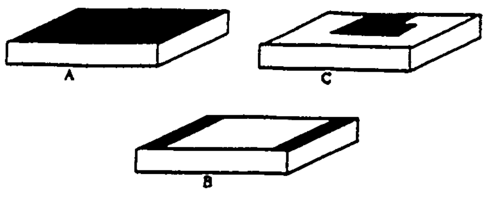


Figure 4: **(Geometry for the test problems, A, B, and C) Phosphorus diffuses through the indicated portion of the 2.0μm $\times$ 2.0μm $\times$ 5.0μm brick.**

rect method such as LU decomposition is not recommended due to fill-in, cost of peripheral storage, and round-off error necessitating multiple preceision arithmetic. While projection methods and relaxation methods are both iterative, they are distinguishable. If the system is of order n, then the projection method converges to the unique solution in at most n iterations and, in practice, far fewer. On the other hand, a relaxation method such as Jacobi, Gauss-Seidel, and SOR (Successive Over-Relaxation), is said to have converged to an approximate solution when a user-specified tolerance has been satisfied.

In the linear problem, $U_t = U_{xx}$, stiffness is exhibited since the Jacobian matrix $\mathbf{F_u}$ has eigenvalues whose real parts are large in magnitude and negative. As the spatial discretization is refined, these large negative eigenvalues increase in magnitude. Since the solution is still smooth, a finite difference method that doesn't require a very small step size is desirable. As will be seen later in this report, the nonlinear problem of Equation 2 also results in a stiff system. This is the motivation for examining BDF (Backward Differentiation Formulae) methods in conjunction with NMOL. In general, when multistep methods are applied to stiff problems, h can be of the same order of magnitude as the time scale of the problem, and the error remains bounded. Those BDF methods of practical use are given by

$$\mathbf{u}_n = \sum_{j=1}^{q} \alpha_j \mathbf{u}_{n-j} + h\beta_0 \dot{\mathbf{u}}_n \qquad 1 \leq q \leq 5.$$

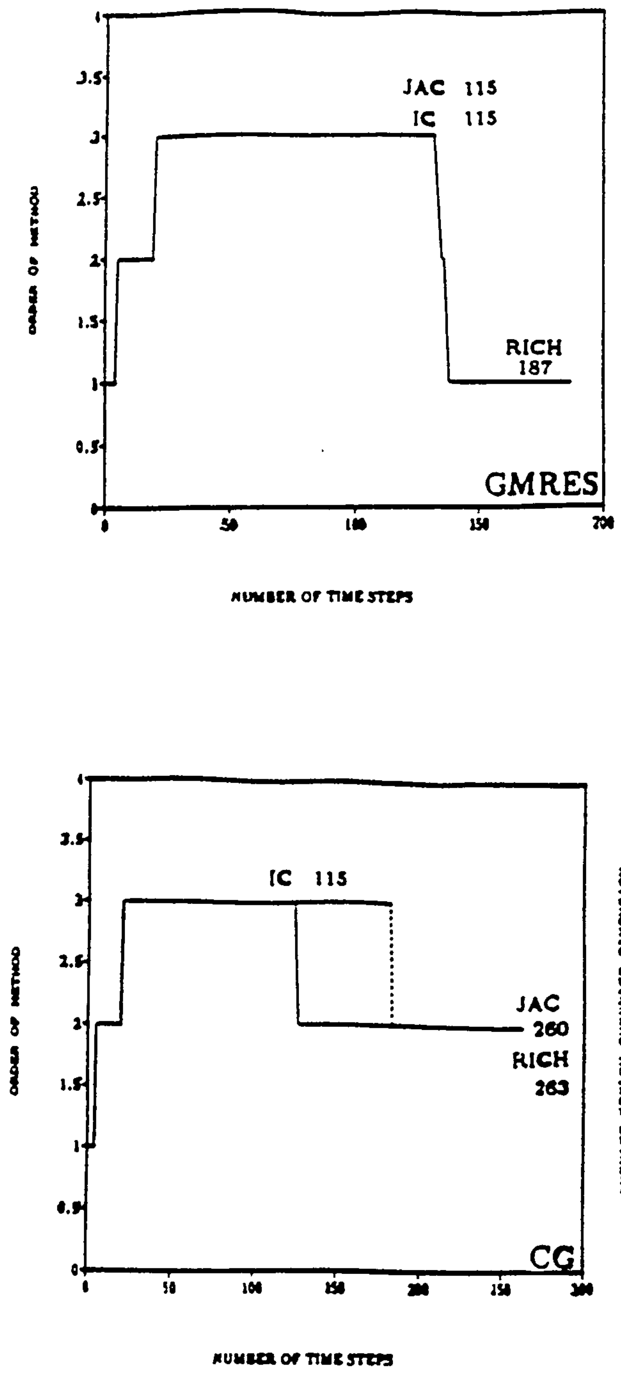
If $\beta_0 = 1$ and $\alpha_1 = 1$, then this reduces to implicit Euler. For graphs of the stability regions of these methods, see Gear's *Numerical Initial Value Problems in Ordinary Differential Equations* [1]. The eigenvalue analysis will confirm that LSODP (Livermore Solver for Ordinary Differential equations - Projection) is a good choice for the integrator for this model. Under stiff options, it uses BDF methods and solves the resulting nonlinear system by a Quasi-Newton technique.

Solving the linear system of the inner Newton loop is the most computationally intensive part of any diffusion code. When a second derivative operator is discretized using a finite difference scheme, the resultant matrix has a spectrum that is dependent upon the grid spacing. As the mesh is refined, the matrix becomes increasingly ill-conditioned. Because of this, the operator $\nabla \cdot (D(C)\nabla C)$ of the single-species model results in an increasingly stiff system of ODE's and a mildly nonsymmetric linear system. PEPPER3 was modified to call NSPCG (Nonsymmetric Preconditioned Conjugate Gradient) so that combinations of accelerators and preconditioners could be run on the model of Equation 2. Since PEPPER3 is written in the programming language C and LSODP and NSPCG are written in Fortran, care had to be exercised in the passing of parameters and pointers. After revision, the calling sequence within PEPPER3 was C calling Fortran calling C.

IV. RESULTS

All runs were on the Cray Y-MP8/864 at the Center for High Performance Computing (CHPC), Austin, Texas. To test the nonlinear single-species model of PEPPER3 with preconditioning, several combinations of accelerators and preconditioners were run on the Cray. Two of the more interesting results are contrasted. Problem C1 of size 20^3 was run using GMRES (Generalized Minimal Residual Method) as the accelerator with the matrix preconditioners, RICH3 (L. F. Richardson's Method), JAC3 (Jacobi Method), and IC3 (Incomplete Cholesky Method), where the 3 indicates that nonsymmetric diagonal storage of the seven nonzero diagonal, subdiagonals, and superdiagonals is being used. Then the same problem was rerun using CG (Conjugate Gradient) as the accelerator and the same three preconditioners. The splitting matrix, **Q**, for RICH3 is the identity matrix. Thus, it represents the null preconditioner. This "preconditioner" was used as a point of reference to assess the effect of preconditioning the Jacobian of the linear system of the Newton loop. For the preconditioner JAC3, the diagonal of the iteration matrix is the splitting matrix, and in IC3 an incomplete LU decomposition of the iteration matrix is used for **Q**. The level of fill-in allowed in the factorization is adjustable; but, in this case, no fill-in was allowed beyond the original matrix nonzero pattern. It is because of the finite difference scheme used in the discretization that the iteration matrix for the single-species model has exactly seven nonzero diagonals.

With IC preconditioning, GMRES and CG gave similar results when run on the nonlinear single-species problem. This is not too surprising since both are conjugate direction methods and under certain conditions GMRES reduces to CG. Three types of comparisons for each combination were done: number of time steps versus order of the method, number of time steps versus size of time step, and number of time steps versus the average Krylov subspace dimension. As shown in Figure 7, GMRES/JAC and GMRES/IC yielded the desired consistent increase in step size by the integrator, and CG/IC yielded a comparable result as well. Figure 6 shows that use of the preconditioner IC resulted in an estimated Krylov subspace of very low dimension. As shown in Figure 5, GMRES/JAC and GMRES/IC each steadily increased to an order of 3, which is optimum. CG/IC gave a similar result, but both CG/JAC and CG/RICH took over twice as many time steps with a decrease in method order. Overall, GMRES/IC gave the best results. There was a strong indication that the harder the



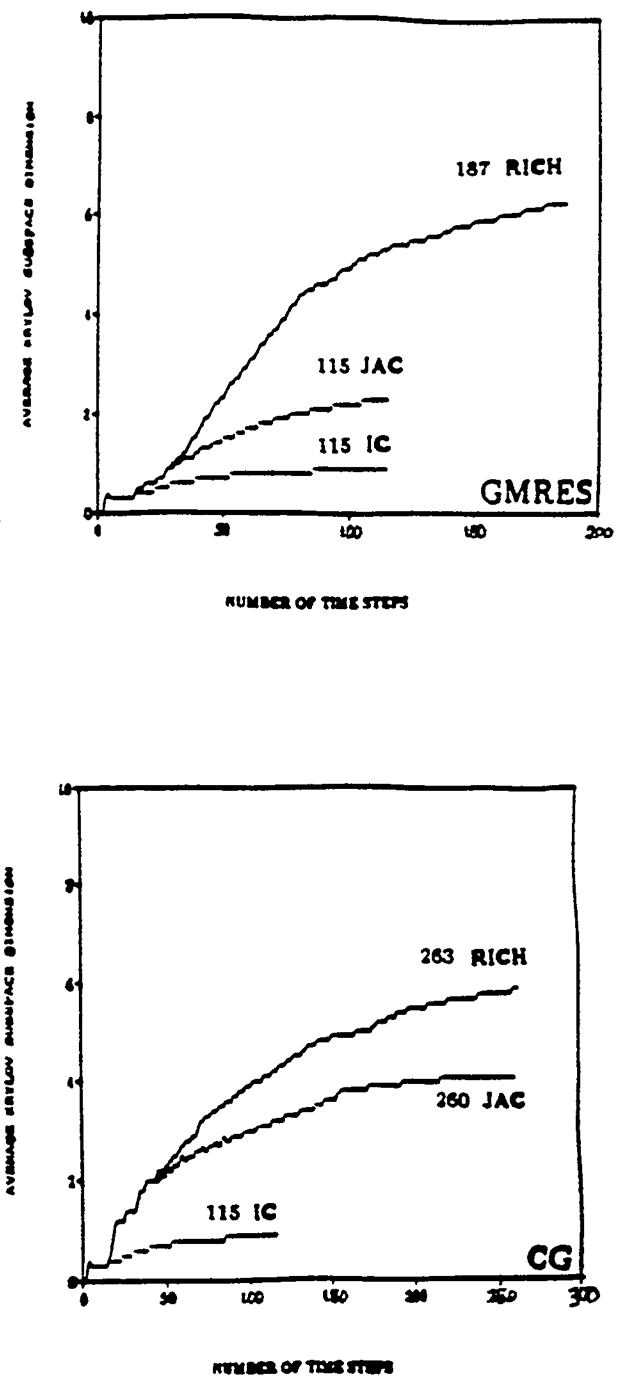
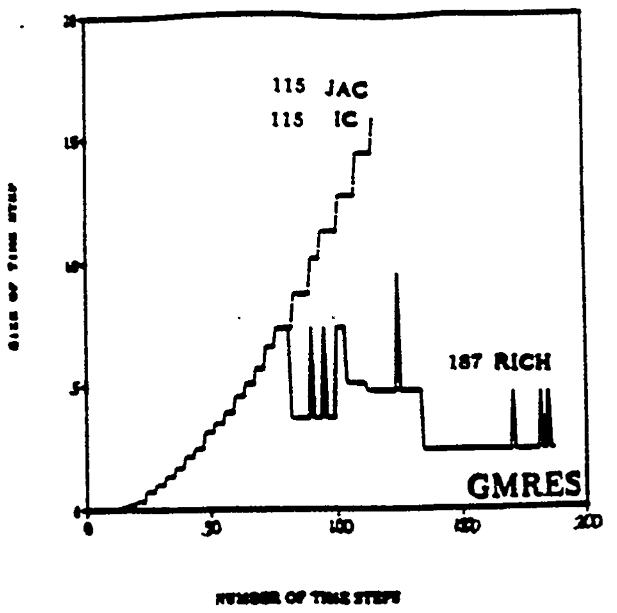


Figure 5: Number of Time Steps vs. Order of the Method, C1

Figure 6: Number of Time Steps vs. Average Krylov Subspace Dimension, C1



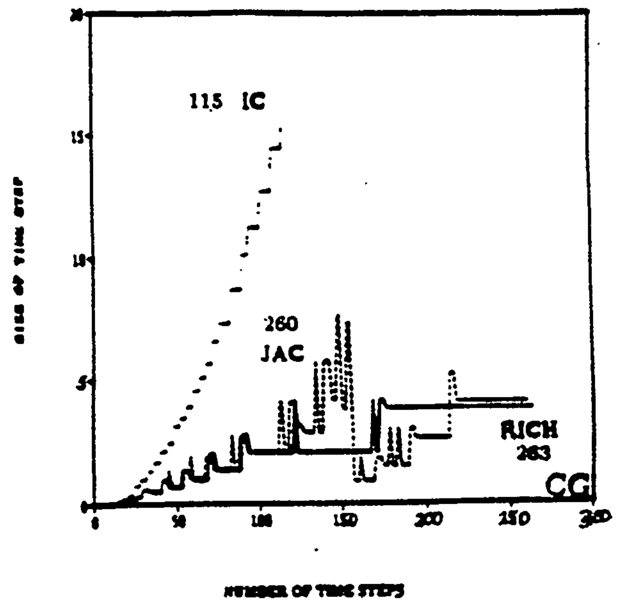


Figure 7: **Number of Time Steps vs. Size of Time Step, C1**

problem is to solve numerically and the larger the problem size, the more preconditioning may be expected to result in a lower CPU time and a better performance. The timed results using problem C1 of size 20^3 are shown in Figure 9. The chart clearly indicates that preconditioning in combination with the accelerator GMRES is effective. GMRES/IC was the only combination which processed uneventfully with no Code 1 warnings from NSPCG (i.e. failure to converge in `itmax` = 10 iterations). In contrast, CG/IC processed but with numerous such warnings from the solver. A comparison of the number of seconds of CPU time to solve the 10^3 problem for each of the test cases is shown in Figure 8. It will be noted that the problem size of 10^3 was not sufficiently large to demonstrate fully the benefits of preconditioning.

It can be shown that each eigenvalue of a positive definite matrix is a positive number. From this result it follows that each eigenvalue of a negative definite matrix is a negative number. In LSODP, the subroutine EWSET is called before each internal integration step to set an array of error weights, EWT. In the routine that loads the coefficient matrix needed by NSPCG it is possible to output the matrix $h * \texttt{elo} * \mathbf{D}^{-1}\mathbf{F}'\mathbf{D}$ where $\mathbf{F}'$ is the system Jacobian, h is the time step size, `elo` is the integration coefficient of the BDF method, and $\mathbf{D}$ is a diagonal matrix containing the reciprocal of the error weight entries. The transformation $\mathbf{F}' \rightarrow \mathbf{D}^{-1}\mathbf{F}'\mathbf{D}$ is a similarity transformation which preserves eigenvalues.

If a model yields a symmetric, positive definite Jacobian $\mathbf{F}'$ or even a nonsymmetric positive definite Jacobian, then the

	RICH3	JAC3	IC3	Matrix-Free
GMRES	29.0	29.8	31.8	70.4
CG	30.1	32.2	31.7	Failed to Run

(a) **Problem A1_10**

	RICH3	JAC3	IC3	Matrix-Free
GMRES	21.5	21.2	21.1	65.4
CG	20.5	20.9	21.2	54.2

(b) **Problem B1_10**

	RICH3	JAC3	IC3	Matrix-Free
GMRES	22.3	22.0	25.6	151.1
CG	27.4	54.0	26.5	Failed to Run

(c) **Problem C1_10**

Figure 8: **Number of Seconds of CPU Time to Solve the 10^3 Problem**

	RICH3	JAC3	IC3
GMRES	487.9	306.3	353.4*
CG	503.8	525.9	352.8

* No Code 1 Warnings From NSPCG

Figure 9: **Number of Seconds of CPU Time to Solve the 20^3 Problem**

eigenvalues for the Richardson iteration matrix will be real and negative. For the nonlinear single-species model it is nonsymmetric negative definite. The fact that this matrix is definite, guarantees convergence of the accelerator GMRES. The scaling done by LSODP does not necessarily preserve symmetry. The iteration matrix computed for the nonlinear A1 problem of size 5^3 had real eigenvalues concentrated on the interval $(-2.5, 0)$ and for size 10^3 the interval was $(-14, 0)$. Figure 10 shows the spectrum of the iteration matrix for Problem A1 of size 10^3 for the integration steps 60, 70, and 80, respectively. The eigenvalues are clearly negative with an increasing spectral radius as the integration proceeds, and they lie in the region of absolute convergence for the BDF methods.

V. CONCLUSIONS

Process simulation to aid in modeling the diffusion of phosphorus in silicon in three dimensions addresses a real world problem which is of vital interest to the semiconductor industry. A preconditioner and an accelerator appropriate for the solution of the single-species model have been identified. The eigenvalue analysis completed for this problem indicates that

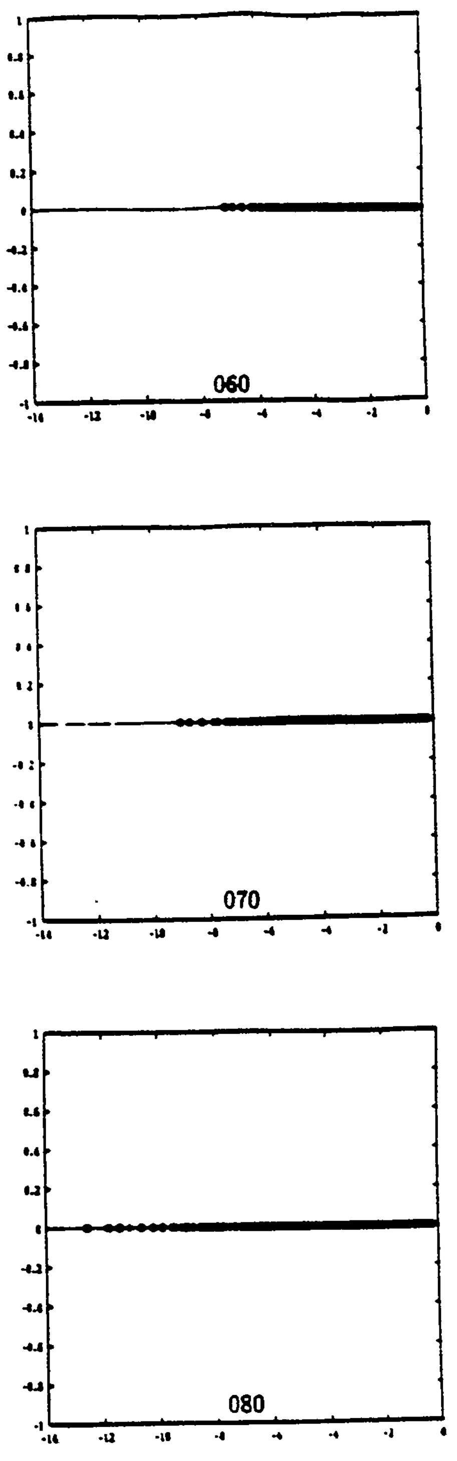


Figure 10: **Eigenvalue Plots for Problem A1, Steps 60, 70, and 80**

the computed results are in agreement with mathematical theory and also confirms the choice of an integrator such as LSODP which uses BDF methods. From what has been shown, as the problem becomes larger and stiffer, preconditioning will result in marked improvement in efficiency.

References

[1] C. William Gear. *Numerical Initial Value Problems in Ordinary Differential Equations.* Prentice-Hall, Inc., Englewood Cliffs, N. J., 1971.

[2] Walter B. Richardson, Graham F. Carey, and Brian J. Mulvaney. "Modeling Phosphorus Diffusion in Three Dimensions." *IEEE Transactions on Computer-aided Design*, 11, 1992.

[3] Stanley Wolf. *Silicon Processing for the VLSI Era, Volume 2: Process Integration.* Lattice Press, Sunset Beach, CA, 1990.

[4] Stanley Wolf and Richard N. Tauber. *Silicon Processing for the VLSI Era, Volume 1: Process Technology.* Lattice Press, Sunset Beach, CA, 1986.

[5] M. Yoshida. "Diffusion of Group V Impurity in Silicon." *Japan J. Appl. Phys.*, 10, June 1971.

[6] M. Yoshida, E. Arai, H. Nakamura, and Y. Terunuma. "Excess Vacancy Generation Mechanism at Phosphorus Diffusion into Silicon." *J. Applied Phys.*, 45, 1974.

Friday, April 8

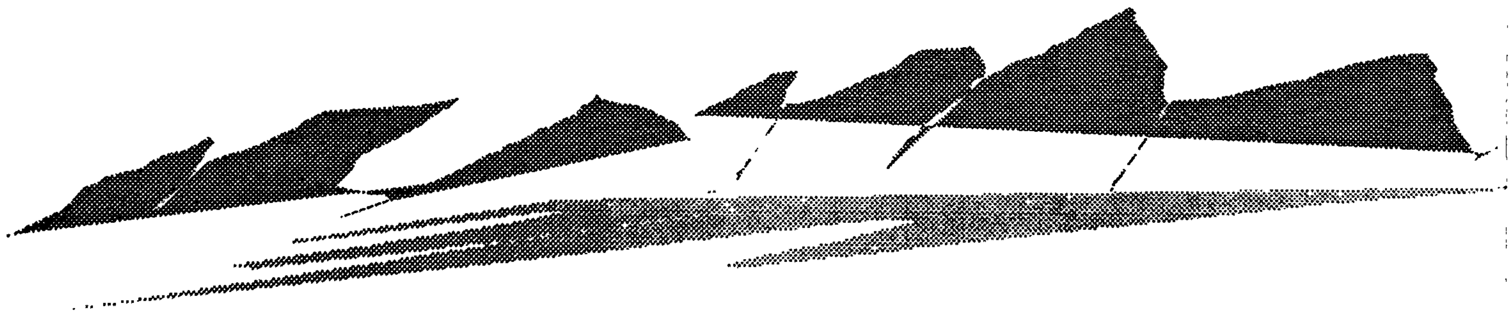
Multigrid and Multilevel Methods III
Chair: Joel Dendy
Room A

4:45 - 5:10 J.E. Dendy, Jr.
Grandchild of the Frequency Decomposition Multigrid Methods

5:10 - 5:35 Van Henson
On Multigrid Methods for Image Reconstruction from Projections

5:35 - 6:00 John Ruge
A Nonlinear Multigrid Solver for a Semi-Lagrangian Potential Vorticity-Based Barotropic Model on the Sphere

6:00 - 6:25 C. Liu
Multgrid Mapping and Box Relaxation for Numerical Simulation of the Whole Process of Flow Transition in 3-D Boundary Layers



GRANDCHILD OF THE FREQUENCY DECOMPOSITION MULTIGRID METHOD

J. E. Dendy, Jr.
Theoretical Division, MS-B284
Los Alamos National Laboratory
Los Alamos, N. M. 87545

and

C. C. Tazartes
Dept. of Mathematics, U. C. L. A.

Abstract. Previously we considered the frequency decomposition multigrid method and rejected it becuase it was not robust for problems with discontinuous coefficients. In this paper we show how to modify the method so as to obtain such robustness while retaining robustness for problems with anisotropic coefficients. We also discuss application of this method to a problem arising in global ocean modeling on the CM-5.

On Multigrid Methods for Image Reconstruction from Projections

Van Emden Henson
Naval Postgraduate School
Monterey, CA

Mark Limber
Simon Fraser University
Burnaby, B.C., Canada

Bruce T. Robinson
Naval Postgraduate School
Monterey, CA

The sampled Radon transform of a 2D function can be represented as a continuous linear map $\mathcal{R} : L^1 \to \mathbf{R}^N$. The image reconstruction problem is: given a vector $\mathbf{b} \in \mathbf{R}^N$, find an image (or density function) $u(x,y)$ such that $\mathcal{R}u = \mathbf{b}$. Since in general there are infinitely many solutions, we pick the solution with minimal 2-norm.

Numerous proposals have been made regarding how best to discretize this problem. One can, for example, select a set of functions ϕ_j that span a particular subspace $\Omega \subset L^1$, and model $\mathcal{R} : \Omega \to \mathbf{R}^N$. The subspace Ω may be chosen as a member of a sequence of subspaces whose limit is dense in L^1.

One approach to the choice of Ω gives rise to a *natural pixel* discretization of the image space. In this setting the equation $u(x,y) = \sum w_j \phi_j(x,y)$ can be written as $\mathbf{u} = \mathcal{R}^* \mathbf{w}$, where $\mathcal{R}^* : \mathbf{R}^N \to \Omega$ is the adjoint operator for $\mathcal{R}$. The problem then becomes that of finding a vector $\mathbf{w}$ satisfying $\mathcal{R}\mathcal{R}^* \mathbf{w} = \mathbf{b}$. This last may be written as $B\mathbf{w} = \mathbf{b}$ where B is a square matrix representing $\mathcal{R}\mathcal{R}^*$.

Two possible choices of the set $\{\phi_j\}$ are the set of characteristic functions of finite-width "strips" representing energy transmission paths and the set of intersections of such strips.

We have studied the eigenstructure of the matrices B resulting from these choices and the effect of applying a Gauss-Seidel iteration to the problem $B\mathbf{w} = \mathbf{b}$. There exists a *near null space* into which the error vectors migrate with iteration, after which Gauss-Seidel iteration stalls.

We attempt to accelerate convergence via a multilevel scheme, based on the principles of McCormick's Multilevel Projection Method (PML). Coarsening is achieved by thickening the rays which results in a much smaller discretization of an *optimal grid*, and a halving of the number of variables. This approach satisfies all the requirements of the PML scheme. We have observed that a multilevel approach based on this idea accelerates convergence at least to the point where noise in the data dominates.

A nonlinear multigrid solver for a semi-Lagrangian potential vorticity-based barotropic model on the sphere.

Ruge, J., Li, Y., McCormick, S. F., Brandt, A. and Bates, J. R.:

The formulation and time discretization of problems in meteorology are often tailored to the type of efficient solvers available for use on the discrete problems obtained. A common procedure is to formulate the problem so that a constant (or latitude-dependent) coefficient Poisson-like equation results at each time step, which is then solved using spectral methods. This both limits the scope of problems that can be handled and requires linearization by forward extrapolation of nonlinear terms, which, in turn, requires filtering to control noise.

Multigrid methods do not suffer these limitations, and can be applied directly to systems of nonlinear equations with variable coefficients. Here, a global barotropic semi-Lagrangian model, developed by the authors, is presented which results in a system of three coupled nonlinear equations to be solved at each time step. A multigrid method for the solution of these equations is described, and results are presented.

Multigrid Mapping and Box Relaxation for Simulation of the Whole Process of Flow Transition in 3-D Boundary Layers *

Chaoqun Liu and Zhining Liu
Center for Computational Mathematics
University of Colorado at Denver
Denver, Colorado 80217-3364

Abstract

A new multilevel technology was developed in this study which provides a successful numerical simulation for the whole process of flow transition in 3-D flat plate boundary layers, including linear growth, secondary instability, breakdown, and transition on a relatively coarse grid with low CPU cost. A fourth-order finite difference scheme on stretched and staggered grids, a fully implicit time-marching technique, a semi-coarsening multigrid based on the so-called approximate line-box relaxation, and a buffer domain for the outflow boundary conditions were all employed for high-order accuracy, good stability, and fast convergence. A new fine-coarse-fine grid mapping technique was developed to catch the large eddies and represent main roles of small eddies to keep the code running after the laminar flow breaks down. The computational results are in good agreement with linear stability theory, secondary instability theory, and some experiments. The computation also reproduced the K-type and C-type transition observed by laboratory experiments. The CPU cost for a typical case is around 2 – 9 CRAY-YMP hours.

*This work was supported by NASA Langley Research Center under grant number NAS1-10312 and by ICOMP/NASA Lewis Research Center under 93 turbulence modeling program.

Friday, April 8

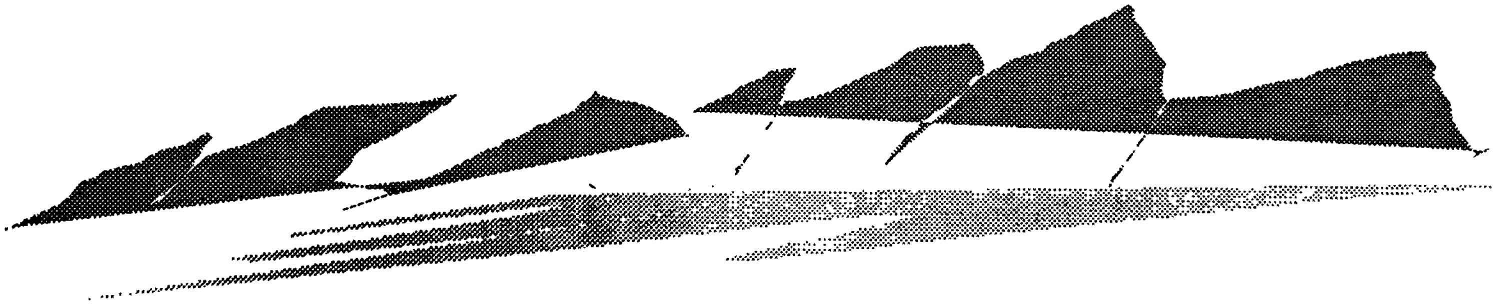
Applications III
Chair: Howard Elman
Room B

4:45 - 5:10 Thomas Hagstrom
Experimental and Theoretical Studies of Iterative Methods for Nonlinear, Nonsymmetric Systems Arising in Combustion

5:10 - 5:35 Colin Aro
Preconditioned Time-Difference Methods for Advection-Diffusion-Reaction Equations

5:35 - 6:00 D.Rh. Gwynllyw
Preconditioned Iterative Methods for Unsteady Non-Newtonian Flow Between Eccentrically Rotating Cylinders

6:00 - 6:25 David Silvester
Fast Non-Symmetric Iterations and Efficient Preconditioning for Navier-Stokes Equations



Experimental and Theoretical Studies of Iterative Methods for Nonlinear, Nonsymmetric Systems Arising in Combustion

Thomas Hagstrom
Department of Mathematics and Statistics
The University of New Mexico
Albuquerque, NM 87131

Krishnan Radhakrishnan
Sverdrup Technology - Lewis Research Group
Brook Park, OH 44135

We report on some iterative methods which we have tested for use in combustion simulations. In particular, we have developed a code to solve zero Mach number reacting flow equations with complex reaction and diffusion physics. These equations have the form of a nonlinear parabolic system coupled with constraints. In semi-discrete form, one obtains DAE's of index two or three depending on the number of spatial dimensions. We have implemented a fourth order (fully implicit) BDF method in time, coupled with a suite of fourth order explicit and implicit spatial difference approximations.

Most codes we know of for simulating reacting flows use a splitting strategy to march in time. This results in a sequence of nonlinear systems to solve, each of which has a simpler structure than the one we are faced with. The rapid and robust solution of the coupled system is the essential requirement for the success of our approach. We have implemented and analyzed nonlinear generalizations of conjugate gradient-like methods for nonsymmetric systems, including CGS and the quasi-Newton based method of Eirola and Nevanlinna. We develop a general framework for the nonlinearization of linear methods in terms of the acceleration of fixed-point iterations, where the latter is assumed to include the 'preconditioning'. Our preconditioning is a single step of a split method, using lower order spatial difference approximations as well as simplified (Fickian) approximations of the diffusion physics.

We also have developed a code for the computation of steadily propagating flames. This results in a nonlinear eigenvalue problem for the system of equations described above. Here we have implemented a version of the recursive projection method (RPM) of Schroff and Keller. This method involves the enhancement of a general fixed point iteration with a Newton iteration on a low-dimensional subspace. This subspace is adaptively modified based on the behavior of the iterations. Our basic fixed point iteration is in fact a time step of a split (or unsplit) method. Various constraint equations, such as normalizations to fix the phase of the traveling wave as well as equations relevant to mesh distribution, are solved via the Newton method in addition to equations identified by the RPM adaptations.

We display numerical experiments for both simplified models as well as complex, physically more accurate systems. Some theoretical results for model systems are also given.

Preconditioned Time-Difference Methods for Advection-Diffusion -Reaction Equations* (Abstract)

Colin Aro[†] Garry Rodrigue [†] Donald Wolitzer [‡]

Introduction

Explicit time differencing methods for solving differential equations are advantageous in that they are easy to implement on a computer and are intrinsically very parallel. The disadvantage of explicit methods is the severe restrictions placed on stepsize due to stability. Stabilty bounds for explicit time differencing methods on advection-diffusion-reaction problems are generally quite severe and implicit methods are used instead. The linear systems arising from these implicit methods are large and sparse so that iterative methods must be used to solve them. In this paper we develop a methodology for increasing the stability bounds of standard explicit finite differencing methods by combining explicit methods, implicit methods, and iterative methods in a novel way to generate new time-difference schemes, called **preconditioned time-difference methods.**

Background

We give a simple description of the method by considering the solution of a system of linear ordinary differential equations

$$\frac{d\mathbf{u}}{dt} = \mathbf{A}\mathbf{u}, \quad \mathbf{u}(0) = \mathbf{u}_0, \tag{0.1}$$

where $\mathbf{A}$ is an $n \times n$ matrix. Such differential equations are generally solved by explicit, implicit or hybrid predictor-corrector methods.

*This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

[†]Department of Applied Science, University of California, Davis, Calif. and Lawrence Livermore National Laboratory, Livermore, Calif., 94550

[‡]Department of Mathematics and Computer Science, Calif. State Univ., Hayward, Calif., 94542

General explicit multi-step methods are given by

$$\mathbf{u}^{n+1} = \sum_{j=0}^{\ell} P_j(\Delta t\mathbf{A})\mathbf{u}^{n-j}$$

where the P_j's are real polynomials. General implicit methods take the form

$$\mathbf{u}^{n+1} = \sum_{j=-1}^{\ell} Q_j(\Delta t\mathbf{A})\mathbf{u}^{n-j} \tag{0.2}$$

where the Q_j's are real polynomials. Note that implicit methods require a linear system to be solved:

$$[\mathbf{I} - Q_{-1}(\Delta t\mathbf{A})]\mathbf{u}^{n+1} = \sum_{j=0}^{\ell} Q_j(\Delta t\mathbf{A})\mathbf{u}^{n-j}.$$

Predictor-corrector methods attempt to get the best of both worlds by defining an approximation $\mathbf{u}^{n+1^{(0)}}$ to $\mathbf{u}^{n+1}$ by an explicit scheme

$$\mathbf{u}^{n+1^{(0)}} = \sum_{j=0}^{\ell} P_j(\Delta t\mathbf{A})\mathbf{u}^{n-j}$$

and then correcting this approximation by an implicit scheme

$$\mathbf{u}^{n+1^{(s+1)}} = Q_{-1}(\Delta t\mathbf{A})\mathbf{u}^{n+1^{(s)}} + \sum_{j=0}^{\ell} Q_j(\Delta t\mathbf{A})\mathbf{u}^{n-j},\ s = 1, 2, \ldots. \tag{0.3}$$

Convergence occurs if the spectral radius of $Q_{-1}(\Delta t\mathbf{A})$ is less than unity. This is generally true only if Δt is extremely small.

The linear systems in implicit methods can be very large and very sparse and often require iterative methods for their solution. The particular iterative method that is used depends significantly on the properties of the matrix and the type of computer that is to be used.

First order iterative methods are defined by first introducing a matrix splitting

$$\mathbf{M} - \mathbf{N} = [\mathbf{I} - Q_{-1}(\Delta t\mathbf{A})] \tag{0.4}$$

and then iterating in the following fashion:

$$\mathbf{M}\mathbf{u}^{n+1^{(s)}} = \mathbf{N}\mathbf{u}^{n+1^{(s-1)}} + \sum_{j=0}^{\ell} Q_j(\Delta t\mathbf{A})\mathbf{u}^{n-j},\ s = 1, 2, \ldots.$$

The splitting is chosen so that the spectral radius of $\mathbf{M}^{-1}\mathbf{N}$ is less than unity so that convergence of the iteration is assured. The iteration can be terminated when the norm of the residual falls below some threshhold or it can be terminated after a fixed number of iterations, say k, so that $\mathbf{u}^{n+1} = \mathbf{u}^{n+1^{(k)}}$ for each n.

Precondtioined Time Differencing

Preconditioned time differencing methods begin by using a classical explicit method to yield an initial guess for an iterative scheme. This iterative scheme is then used to solve an implicit equation where only a fixed number of iterations are taken. That is,

$$\mathbf{u}^{n+1^{(0)}} = \sum_{j=0}^{\ell} P_j(\Delta t\mathbf{A})\mathbf{u}^{n-j}$$

and

$$\mathbf{M}\mathbf{u}^{n+1^{(s)}} = \mathbf{N}\mathbf{u}^{n+1^{(s-1)}} + \sum_{j=0}^{\ell} Q_j(\Delta \mathbf{A})\mathbf{u}^{n-j}, \quad s = 1, 2, \ldots, m \tag{0.5}$$

for a splitting $\mathbf{M} - \mathbf{N}$ of $(\mathbf{I} - Q_{-1}(\Delta t\mathbf{A}))$. We then take $\mathbf{u}^{n+1} = \mathbf{u}^{n+1^{(m)}}$.

We note that the preconditioned time differencing method is related to the waveform relaxation method on linear problems where the explicit operator is the identity matrix.

An example of a preconditioned time-diferencing method is given by a Forward-Euler method as the explicit part and a Backward-Euler method as the implicit part. The preconditioner is the point-Jacobi matrix with only one iteration being taken. Specifically, this preconditioned time-differencing method is the following:

1. Initial Guess for Iterative Method: Forward Euler Method.

$$\mathbf{u}^{n+1^{(0)}} = (\mathbf{I} + \Delta t\mathbf{A})\mathbf{u}^n;$$

2. Overrelaxed Jacobi Matrix Splitting of the Backward Euler Matrix:

$$(\mathbf{I} - \Delta t\mathbf{A}) = \mathbf{M} - \mathbf{N},$$

$$\mathbf{M} = c\ \mathsf{diagonal}(\mathbf{I} - \Delta t\mathbf{A}), \quad c > 0.$$

3. Preconditioned Time Differencing Method.

$$\mathbf{u}^{n+1^{(s)}} = \mathbf{M}^{-1}\left\{\mathbf{N}\mathbf{u}^{n+1^{(s-1)}} + \mathbf{u}^n\right\}, \quad s = 1, 2, \ldots, m.$$

We take $m = 1$ so that this preconditioned scheme takes the following form.

$$\mathbf{M}\mathbf{u}^{n+1} = \{\mathbf{N}(\mathbf{I} + \Delta t\mathbf{A})\mathbf{u}^n + \mathbf{u}^n\} \tag{0.6}$$

or

$$\mathbf{u}^{n+1} = C(\Delta t)\mathbf{u}^n \tag{0.7}$$

where

$$C(\Delta t) = \Delta t^2 \mathbf{M}^{-1}\mathbf{A}^2 + \Delta t\mathbf{A} + \mathbf{I} \tag{0.8}$$

Matrix stability of (0.7) demands that $\rho(C(\Delta t)) < 1$.

Test Problems

We use the above preconditioned time differencing scheme (0.7) to solve the system of ordinary differential equations arising from a method of lines discretization of the K-dimensional diffusion equation

$$\frac{\partial u}{\partial t} = \alpha \sum_{i=1}^{K} \frac{\partial^2 u}{\partial x_i^2},\ \alpha > 0, \tag{0.9}$$

and the K-dimensional scalar advection-diffusion equation

$$\frac{\partial u}{\partial t} + \rho \cdot \nabla u = \epsilon \nabla \cdot \nabla u, \quad \rho, \epsilon = \text{constants}. \tag{0.10}$$

Convergence of the method for the above equations is established. Stabilty ranges for $\Delta t/\Delta x$ and $\Delta t/\Delta x^2$ are calculted by both a Von-Neumann stability analysis and a matrix stability analysis. Computations are given that verify these results. We then show experimentally that these stability ranges still hold when the precondtioned time differencing method (0.7) is used to solve the one-dimensional Burgers' equation

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \epsilon \frac{\partial^2 u}{\partial x^2}. \tag{0.11}$$

Finally, we test the preconditioned time differencing method using different explicit methods, preconditioners, and iteration counts for solving the following highly stiff, nonlinear reaction equations arising from an atomospheric model:

$$\begin{aligned}
\dot{y_1} &= -k_1y_1y_7 - k_2y_1y_2 + k_3y_9y_6 + k_4y_9y_7 - k_7y_1y_4 + 2k_{12}y_7 + k_{13}y_2 \\
\dot{y_2} &= k_1y_1y_7 - k_2y_1y_2 - k_6y_3y_2 - k_8y_4y_2 - k_{13}y_2 - k_{14}y_2 \\
\dot{y_3} &= 2k_5y_8y_9 - k_6y_2y_3 + k_7y_1y_4 + k_8y_2y_4 - k_9y_3y_4 - k_{11}y_3y_5 + 2k_{15}y_5 \\
\dot{y_4} &= k_6y_2y_3 - k_7y_1y_4 - k_8y_2y_4 - k_9y_3y_4 - 2k_{10}y_4^2 + k_{11}y_3y_5 \\
\dot{y_5} &= k_{10}y_4^2 - k_{11}y_3y_5 - k_{15}y_5
\end{aligned}$$

where the k_i are reaction constants and the y_6, y_7, y_8, y_9 represent chemical species in instantaneous equilibrium.

Preconditioned Iterative Methods for Unsteady Non-Newtonian Flow between Eccentrically Rotating Cylinders

D.Rh.Gwynllyw and T.N.Phillips
Department of Mathematics,
University of Wales,
Aberystwyth SY23 3BZ,
Wales, UK.

December 14, 1993

Abstract

The journal bearing is an essential part of all internal combustion engines as a means of transferring the energy from the piston rods to the rotating crankshaft. It consists essentially of an inner cylinder (the journal), which is part of the crankshaft, and an outer cylinder (the bearing), which is at the end of the piston rod. In general, the two cylinders are eccentric and there is a lubricating film of oil separating the two surfaces. The addition of polymers to mineral (Newtonian) oils to minimize the variation of viscosity with temperature has the added effect of introducing strain-dependent viscosity and elasticity.

The physical problem has many complicating features which need to be modelled. It is a fully three-dimensional problem which means that significant computational effort is required to solve the problem numerically. The system is subject to dynamic loading in which the journal is allowed to move under the forces the fluid imparts on it and also any other loads such as that imparted by the engine force. The centre of the journal traces out a nontrivial locus in space. In addition, there is significant deformation of the bearing and journal and extensive cavitation of the oil lubricant.

In the present study we restrict ourselves to the two-dimensional statically loaded problem. In previous work ([6], [7], [5]) a single domain spectral method was used which employed a bipolar coordinate transformation to map the region between the journal and the bearing onto a rectangle. The flow variables were then approximated on this rectangle using Fourier-Chebyshev expansions. However, to allow for future possible deformation of the journal and bearing surfaces due to increased load in the dynamically loaded case we have decided to use a more versatile spectral element formulation.

The governing equations for a generalized Newtonian fluid comprise the conservation of momentum

$$\rho\left(\frac{\partial \mathbf{v}}{\partial t}+\mathbf{v}.\nabla\mathbf{v}\right)=-\nabla p+\nabla.\mathbf{T}, \tag{1}$$

the conservation of mass

$$\nabla.\mathbf{v}=0, \tag{2}$$

and the constitutive equation

$$\mathbf{T} = 2\eta(\dot{\gamma})\mathbf{d}, \tag{3}$$

where ρ is the density, η is the variable viscosity, $\dot{\gamma} = \sqrt{2\, trace(\mathbf{d})^2}$ is the shear-rate. $\mathbf{T}$ is the extra-stress tensor and $\mathbf{d} = \frac{1}{2}(\nabla\mathbf{v}+(\nabla\mathbf{v})^T)$ is the rate of deformation tensor. The constitutive equation written in this form can easily be modified to incorporate viscoelasticity. The journal and bearing have radii R_J and R_B, respectively, with the distance between the centres of the cylinders given by e. The bearing is kept at rest, while the journal is rotated at an angular velocity Ω. The eccentricity is defined by $\epsilon = e/c$, where $c = R_B - R_J$ is known as the gap.

The region between the journal and bearing is partitioned into a number of spectral elements. Each physical element is mapped onto the parent element $[-1,1] \times [-1,1]$. A Legendre Gauss-Lobatto grid is used on the parent element. On the parent element the velocity and extra-stress components are approximated by interpolating polynomials whose nodes are the Gauss-Lobatto nodes, and the pressure by an interpolating polynomial whose nodes are the interior Gauss-Lobatto nodes. Therefore the pressure approximation is of degree $N-2$ if the velocity approximation is of degree N. The velocity and pressure approximations corresponding to element k are therefore

$$\mathbf{v}_N^k(\xi,\eta) = \sum_{i=0}^{N}\sum_{j=0}^{N} \mathbf{v}_{i,j}^k h_i(\xi)h_j(\eta), \tag{4}$$

$$p_N^k(\xi,\eta) = \sum_{i=1}^{N-1}\sum_{j=1}^{N-1} p_{i,j}^k h_i(\xi)h_j(\eta). \tag{5}$$

With the velocity and pressure approximation spaces thus chosen the Babuška-Brezzi compatibility condition is satisfied. There are no spurious pressure modes in the pressure approximation.

The solution of the steady problem is found by marching the time-dependent equations forward in time. The following time-splitting scheme is used:

$$\begin{aligned} \frac{\rho}{\Delta t}(\mathbf{v}^* - \mathbf{v}^n) &= 2\nabla\eta^n.\mathbf{d}^n - \frac{\rho}{\Delta t}\mathbf{v}^n.\nabla\mathbf{v}^n, & (6)\\ \frac{\rho}{\Delta t}(\mathbf{v}^{n+1} - \mathbf{v}^*) &= -\nabla p^{n+1} + \eta^n\nabla^2\mathbf{v}^{n+1}, & (7)\\ \nabla.\mathbf{v}^{n+1} &= 0. & (8)\end{aligned}$$

Equation (6) is an explicit equation for $\mathbf{v}^*$. The solution of (7) and (8) requires the inversion of an unsteady Stokes operator at each time step. This is the computationally intensive part of the algorithm.

The weak form of the unsteady Stokes problem is set up by multiplying each equation by an appropriate test function, integrating over each element and using Green's theorem. The corresponding discrete system is formed by replacing the integrals by appropriate quadrature rules whose nodes are the Gauss-Lobatto nodes points and choosing a sequence of test functions which span the appropriate approximation space. This leads to the system

$$\begin{aligned} A\mathbf{v} + \sigma B\mathbf{v} - D^T\mathbf{p} &= \mathbf{g}, & (9)\\ D\mathbf{v} &= \mathbf{h} & (10)\end{aligned}$$

where $\sigma = 2/\Delta t$. Block Gaussian elimination yields a symmetric positive definite system for the pressure

$$S\mathbf{p} = \mathbf{c} \tag{11}$$

where $S = D(A+\sigma B)^{-1}D^T$ and $\mathbf{c} = -D(A+\sigma B)^{-1}\mathbf{g} + \mathbf{h}$. The system (11) is solved for the vector of pressure unknowns. Since this system is solved using an iterative method, the preconditioned conjugate gradient method. the stiffness matrices do not need to be set up. Instead they are kept in elemental form.

It is documented in [2] and in other papers that, for the steady problem, the condition number of S, using the pressure mass matrix as preconditioner, is near unity and hence is well-conditioned for use in a preconditioned conjugate gradient algorithm. This result depends on the element aspect ratios. However, the geometry of the journal bearing is such that the elemental aspect ratios which are proportional to $\alpha \equiv c/2\pi R_J$ are very large. At large eccentricities the aspect ratios are much less than 1/100. Not even significantly redefining the elements would overcome this problem. As the aspect ratio increases the condition number of S increases with a small cluster of eigenvalues leaving the 'nice cluster' and tending towards zero. Using the usual pressure mass matrix as a preconditioner the preconditioned conjugate gradient method will fail to converge within the theoretical maximum N iterations indicating that round-off errors dominate.

Direct solution methods for (11) are undesirable because of the considerable time required in preprocessing S. Furthermore, due to the bad conditioning of S we find that Choleski decomposition gives inaccurate results.

Efficient preconditioners for S are therfore essential for the efficient solution of (11) using the preconditioned conjugate gradient method at each time step. The unsteady Stokes problem is notoriously difficult to precondition [4]. We considered two choices for the preconditioning matrix. The first was based on the diagonal of the pressure mass matrix. This matrix comprises nonzero entries which are the tensor products of the quadrature weights used in setting up the discrete variational formulation (9)-(10). This preconditioner, although cheap to form, was not as powerful as the second choice.

Note that the entries of S are dependent, among other things, on the eccentricity ϵ. We denote this dependence by $S(\epsilon)$. The second choice for preconditioner was to use the Choleski decomposition of $S(0)$ as the preconditioner for $S(\epsilon)$. Although the Choleski factorization of $S(0)$ is unsuitable if used to invert the system directly, it does, however, provide a very powerful preconditioner. For a fixed number of elements and a fixed degree of polynomial approximation this preconditioner yielded condition numbers independent of Δt. At very high eccentricities it is necessary to replace the preconditioner $S(0)$ by $S(\epsilon_0)$, where $\epsilon_0 > 0$.

For the dynamically loaded case the matrix S is independent of the orientation of the cylinders with respect to a fixed coordinate axis and hence the preconditioners mentioned above are equally valid for this problem.

The condition numbers of the preconditioned systems will be presented showing the dependence on the various parameters of the problem as well as the discretization parameters. Results will be presented which illustrate the reduction in the condition number and iterations when using preprocessed matrices as preconditioners and also the improvement in accuracy over supposedly exact direct methods. Accuracy is measured by comparing the loads and couples on the journal for the steady statically loaded case with those obtained in

[6] and also with lubrication theory [1]. Good agreement with previously published results is demonstrated.

References

[1] M.J.Davies and K.Walters. The behaviour of Non-Newtonian lubricants in journal bearings - a theoretical study. In: Rheology of Lubricants (T.C.Davenport, Ed.) Applied Science Publishers, 1972, 65-80.

[2] P.F.Fischer and A.T.Patera. Parallel spectral element solution of the Stokes problem. J.Comp.Phys. **92** (1991), 380-421.

[3] Y.Maday, D.Meiron, A.T.Patera and E.M.Rønquist Analysis of iterative methods for the steady and unsteady Stokes problem: application to spectral element discretizations. SIAM J. Sci. Comput. 14 (1993), 310-337.

[4] Y.Maday and A.T.Patera. Spectral element methods for the incompressible Navier-Stokes equations. In: State of the Art Surveys in Computational Mechanics (A.K.Noor and J.T.Oden, Eds.), 1989, 71-143.

[5] T.N.Phillips and G.W.Roberts. The treatment of spurious pressure modes in spectral incompressible flow calculations, J. Comput. Phys. **105** (1993), 150-164.

[6] G.W.Roberts, A.R.Davies and T.N.Phillips. Three-dimensional spectral approximations to Stokes flow between eccentrically rotating cylinders, Int. J. Num. Meth. Fluids **13** (1991), 217-233.

[7] G.W.Roberts and K.Walters. On viscoelastic effects in journal-bearing lubrication, Rheol. Acta **31** (1992), 55-62.

Fast non-Symmetric Iterations and Efficient Preconditioning for Navier-Stokes Equations

David Silvester † and Howard Elman ‡

Abstract.

Discretisation of the steady-state Navier-Stokes equations:

$$(1)\qquad \begin{aligned} (\mathbf{u}.\mathrm{grad})\mathbf{u} - \nu\nabla^2\mathbf{u} + \mathrm{grad}\, p &= 0 \\ \mathrm{div}\,\mathbf{u} &= 0. \end{aligned}$$

in some flow domain $\Omega \subset \mathrm{I\!R}^{\mathrm{d}}$, ($d = 2$ or 3), gives a system of non-linear algebraic equations for discretised variables $\mathbf{u}$ (the velocity), and p (the pressure). We assume that appropriate boundary conditions are imposed. The non-linear equation system can be linearised using a fixed-point (Picard) iteration to give a matrix system of the form:

$$(2)\qquad \begin{pmatrix} A & B^t \\ B & O \end{pmatrix} \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix},$$

which must be solved at every iteration. The matrix A is block diagonal, and consists of d convection-diffusion operators, one for each component of velocity. Two difficulties arise when solving (2). Firstly, the matrix A is not symmetric, although under certain conditions the symmetric part is positive definite. Secondly, the overall system is indefinite. This makes the design of fast and efficient iterative solvers for discretised Navier-Stokes operators an extremely challenging task.

Our objective is to analyse the convergence of preconditioned Krylov subspace iterations (for example, GMRES or QMR) applied to unsymmetric linear systems of the form (2). In practice, direct solution of (2) (via sparse elimimation) may be possible in the case $d = 2$, but direct solution is not feasible when $d = 3$. Note

† UMIST, Manchester M60 1QD, United Kingdom
‡ University of Maryland, College Park, MD 20742

that high accuracy solutions are not usually required, and a "good" initial solution estimate is readily available.

The approach that we adopt builds on our earlier work (cf. [1],[2],[3]) in the symmetric (Stokes) case. In particular, we concentrate on mixed finite element approximations of (1). Our analysis assumes that the preconditioning is of the form:

$$\begin{pmatrix} M_A & O \\ O & M_Q \end{pmatrix} \approx \begin{pmatrix} A & B^t \\ B & O \end{pmatrix}, \tag{3}$$

where M_A is some approximation of A, typically a multigrid or line relaxation preconditioner, and the operator M_Q approximates the Schur complement matrix $BA^{-1}B^t$. The analysis leads to bounds on the eigenvalue distribution of the preconditioned system in terms of two parameters: h (the characteristic mesh size) and ν (the viscosity). Our theoretical results are reinforced by numerical experiments, a selection of which will also be presented.

References.

1. Wathen, A. and Silvester, D. Fast iterative solution of stabilised Stokes systems part I: using simple diagonal preconditioners, SIAM J. Numer. Anal. **30**, pp. 630–649 (1993).
2. Silvester, D. and Wathen, A. Fast iterative solution of stabilised Stokes systems part II: using general block preconditioners, Report NA-92-06, Computer Science Department, Stanford University, (to appear in *SIAM J. Numer, Anal.*).
3. Elman H. and Golub, G. Inexact and preconditioned Uzawa algorithms for saddle point problems, Report CS-TR-3075, Computer Science Department, University of Maryland, 1993.

Friday, April 8
Workshop: Robust Iterative Methods
(Evening: 8:00p - 10:00p)
Chair: Youcef Saad
Room A

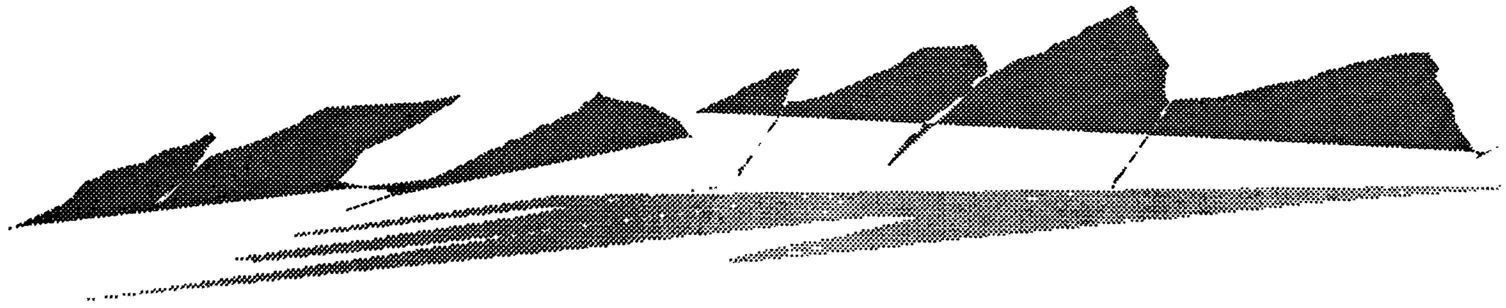
Youcef Saad
Iterative solvers in industrial applications: are we kidding ourselves?

Mike Heroux
Some current challenges for industrial CFD applications

Wei Pai Tang
Multi-stage ILU preconditioners for semiconductor device simulation

Larry Wigton
Experiences with Matrix-Iterative Solvers at Boeing

Alex Yeremin
Numerical experiences with advanced iterative solvers for industrial applications



Friday Evening's Workshop

Robust Iterative Methods

Organizer: Youcef Saad

Abstract

In spite of the tremendous progress achieved in recent years in the general area of iterative solution techniques, there are still a few obstacles to the acceptance of iterative methods in a number of applications. These applications give rise to very indefinite or highly ill-conditioned non Hermitian matrices. Trying to solve these systems with the simple-minded standard preconditioned Krylov subspace methods can be a frustrating experience. With the mathematical and physical models becoming more sophisticated, the typical linear systems which we encounter today are far more difficult to solve than those of just a few years ago. This trend is likely to accentuate. This workshop will discuss (1) these applications and the types of problems that they give rise to; and (2) recent progress in solving these problems with iterative methods. The workshop will end with a hopefully stimulating panel discussion with the speakers.

47

Speakers

- Youcef Saad, University of Minnesota
  "Iterative solvers in industrial applications: are we kidding ourselves?"
- Mike Heroux, Cray Research
  "Some current challenges for industrial CFD applications"
- Wei Pai Tang, University of Waterloo
  "Multi-stage ILU preconditioners for semiconductor device simulation"
  (in collaboration with Qing Fang, Peter Forsyth, John McMacken)
- Larry Wigton, Boeing Computer Services
  "Experiences with Matrix-Iterative Solvers at Boeing"
- Alex Yeremin, Russian Academy of Sciences and Elegant Mathematics, Inc.
  "Numerical experiences with advanced iterative solvers for industrial applications"

Saturday, April 9

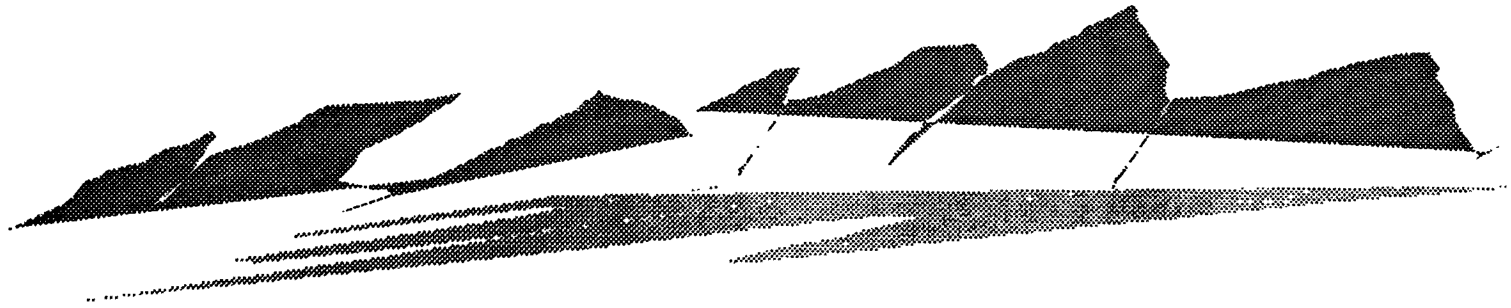
Preconditioners I
Chair: Steve Ashby
Room A

8:00 - 8:25 Edmond Chow
Approximate Inverse Preconditioners for General Sparse Matrices

8:25 - 8:50 Xiaoge Wang
CIMGS: An Incomplete Orthogonal Factorization Preconditioner

8:50 - 9:15 L. Kolotilina
Incomplete Block SSOR Preconditionings for High Order Discretizations

9:15 - 9:40 Fernando Alvarado
Block-Bordered Diagonalization and Parallel Iterative Solvers



Approximate inverse preconditioners for general sparse matrices

Edmond Chow and Youcef Saad
University of Minnesota
Department of Computer Science
Minneapolis MN 55455

Preconditioned Krylov subspace methods are often very efficient in solving sparse linear matrices that arise from the discretization of elliptic partial differential equations. However, for general sparse indifinite matrices, the usual ILU preconditioners fail, often because of the fact that the resulting factors *L* and *U* give rise to "unstable" forward and backward sweeps. In such cases, alternative preconditioners based on approximate inverses may be attractive. We are currently developing a number of such preconditioners based on iterating on each column to get the approximate inverse. For this approach to be efficient, the iteration must be done in sparse mode, i.e., we must use "sparse-matrix by sparse-vector" type operations. We will discuss a few options and compare their performance on standard problems from the Harwell-Boeing collection.

CIMGS: An Incomplete Orthogonal Factorization Preconditioner

Xiaoge Wang
Computer Science Department
Indiana University–Bloomington

Kyle Gallivan
ECE Department, University of Illinois–Urbana

Randall Bramley
Computer Science Department
Indiana University–Bloomington *

January 3, 1994

This paper introduces, analyzes, and tests a preconditioning method for conjugate gradient (CG) type iterative methods. We start by examining incomplete Gram–Schmidt factorization (IGS) methods in order to motivate the new preconditioner. We show that the IGS family is more stable than IC, and they successfully factor any full rank matrix. Furthermore, IGS preconditioners are at least as effective in accelerating convergence of CG type iterative methods as the incomplete Cholesky (IC) preconditioner. The drawback of IGS methods are their high cost of factorization. This motivates finding a new algorithm, CIMGS, which can generate the same factor in a more efficient way.

The new preconditioner (called CIMGS) is derived from and shown to be equivalent to IMGS, a member of the IGS family, and so for any full rank matrix it does not breakdown in exact arithmetic. In addition to preserv-

*Work supported by NSF grants CCR-9120105 and CDA-9309746 ing the effectiveness of IMGS, it also greatly reduces the cost of computing the preconditioner. Although originally designed for preconditioning least squares problems, it is also applicable to more general symmetric positive definite matrices. An error analysis shows that numerically CIMGS is less likely to be rank deficient than Cholesky factorization. This implies that CIMGS is more robust than either IC or complete Cholesky factorization.

For systems whose normal equation, are M–matrices, CIMGS induces a regular splitting. We also prove that under this assumption, CIMGS generates a factor R which is elementwise closer to the complete Cholesky factor than the factor R produced by IC. This shows that for M–matrices, CIMGS is better than IC in approximating the full Cholesky factor. Further results show that CIMGS better approximates the complete Cholesky factor R^c as the set of dropped positions gets smaller, and lies between complete Cholesky factorization and incomplete Cholesky factorization in its approximation properties. These properties usually hold numerically, even when A^TA is not an M–matrix. When the target sparsity pattern of the incomplete factor satisfies a mild and easily verified (or enforced) property, the upper triangular factor CIMGS generates is the same as the one incomplete Cholesky factorization does. This provides a radically different method of stabilizing IC factorization, based solely on the target sparsity pattern.

Numerical test results are presented that show

1. CIMGS in practice is robust. It succeeds for all 30 of our test problems, while Cholesky fails for 6 and IC fails for 18 problems.

2. CIMGS preconditioned CG is competitive with or superior to IC preconditioned CG in efficiency.

3. CIMGS preconditioned CG is competitive with complete Cholesky factorization applied to the normal equations in both robustness and efficiency.

Incomplete Block SSOR Preconditionings for High Order Discretizations

L.Kolotilina

Steklov Mathematical Institute
Fontanka 27, 191011 St.Petersburg, Russia
E-mail: liko @ lomi.spb.su

This talk considers the solution of linear algebraic systems $Ax = b$ resulting from the p-version of the Finite Element Method (FEM) using PCG iterations. Contrary to the h-version, the p-version ensures the desired accuracy of a discretization not by refining an original finite element mesh but by introducing higher degree polynomials as additional basis functions which permits to reduce the size of the resulting linear system as compared with the h-version.

The suggested preconditionings are the so-called Incomplete Block SSOR (IBSSOR) preconditionings. Assuming that $A = (A_{ij})$, $1 \le i,j \le m$, is a block partitioning of a symmetric positive definite matrix A an IBSSOR preconditioner B for A is constructed in the form

$$B = (\tilde{D} + L)\tilde{D}^{-1}(\tilde{D} + L^T),$$

where $\tilde{D} = Diag(\tilde{A}_{11}, \dots, \tilde{A}_{mm})$ and $\tilde{A}_{ii}$ is a s.p.d. approximation to $A_{ii}, i = 1, \dots, m$. In the context of the p-version of the FEM instead of the commonly used block partitionings based on a domain decomposition technique we consider the so-called hierarchical p-partitionings. These partitionings are constructed recursively and at each p-refinement step the stiffness matrix A is partitioned into a 2×2 block form

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix},$$

where the first block A_{11} corresponds to the "old" basis functions while A_{22} corresponds to the "new" basis functions.

Using hierarchical p-partitionings eusures that to construct an IBSSOR preconditioner after a p-refinement we nead only to construct an approximation to the new diagonal block of A while other diagonal blocks are already approximated. This is in contrast with the domain decomposition approach where at each p-refinement step one needs to construct the whole preconditioner a new.

The suggested preconditionings are analyzed theoretically and provided with the results of numerical experiments for $3D$ linear elasticity problems showing that the IBSSOR-CG algorithms based on the hierarchical p partitionings outperforms the IBSSOR-CG algorithms based on the domain decomposition approach. Moreover, sometimes the hierarchical p partitionings based IBSSOR-CG algorithms provide a unique way to compute the solution to highly ill-conditioned 3D FE systems.

BLOCK-BORDERED DIAGONALIZATION AND PARALLEL ITERATIVE SOLVERS

Fernando Alvarado **Hasan Dağ** **Monika ten Bruggencate**
Department of Electrical and Computer Engineering
The University of Wisconsin – Madison

Abstract

One of the most common techniques for enhancing parallelism in direct sparse matrix methods is the reorganization of a matrix into a blocked-bordered structure. Incomplete LDU factorization is a very good preconditioner for PCG in serial environments. However, the inherent sequential nature of the preconditioning step makes it less desirable in parallel environments. This paper explores the use of BBD (Blocked Bordered Diagonalization) in connection with ILU preconditioners. The paper shows that BBD-based ILU preconditioners are quite amenable to parallel processing. Neglecting entries from the entire border can result in a blocked diagonal matrix. The result is a great increase in parallelism at the expense of additional iterations. Experiments on the Sequent Symmetry shared memory machine using (mostly) power system type matrices indicate that the method is generally better than conventional ILU preconditioners and in many cases even better than partitioned inverse preconditioners, without the initial setup disadvantages of partitioned inverse preconditioners.

1 Introduction

A set of sparse linear system of equations

$$A\mathbf{x} = \mathbf{b}, \tag{1}$$

can be solved by either Gaussian elimination-based direct methods or iterative methods. Direct methods produce an exact solution to the machine precision, whereas iterative methods produce an approximate solution. The solution part in direct methods is preceded by an ordering and a symbolic factorization.

An iterative method to solve an $n \times n$ linear system starts with an initial approximation $\mathbf{x}^0$ to the solution $\mathbf{x}$ and generates a sequence of vectors $\{\mathbf{x}^i\}_{i=1}^{\infty}$ that converges to the solution $\mathbf{x}$. One of the most widely used iterative methods, when matrix A in (1) is symmetric positive definite, is the preconditioned conjugate gradient method. Below we present PCG algorithm for the sake of completeness. For more details see [8, 11, 17].

```
Initialize:
    Select $\mathbf{x}^0$
    Let $\mathbf{r}^0 = \mathbf{b} - A\mathbf{x}^0$
    Solve $M\tilde{\mathbf{r}}^0 \leftarrow \mathbf{r}^0$
    Let $\mathbf{p}^0 \leftarrow \tilde{\mathbf{r}}^0$
Iterate:
    $\alpha_k \leftarrow -(\tilde{\mathbf{r}}^k, \mathbf{r}^k)/(\mathbf{p}^k, A\mathbf{p}^k)$
    $\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k - \alpha_k \mathbf{p}^k$
    $\mathbf{r}^{k+1} \leftarrow \mathbf{r}^k + \alpha_k A\mathbf{p}^k$
    Solve $M\tilde{\mathbf{r}}^{k+1} = \mathbf{r}^{k+1}$
    $\beta_k \leftarrow (\tilde{\mathbf{r}}^{k+1}, \mathbf{r}^{k+1})/(\tilde{\mathbf{r}}^k, \mathbf{r}^k)$
    $\mathbf{p}^{k+1} \leftarrow \tilde{\mathbf{r}}^{k+1} + \beta_k \mathbf{p}^k$
```

The matrix M defines the preconditioner matrix implicitly. Every iteration requires one matrix vector product, two inner products, three saxpy operations, two scalar comparisons and the solution of a linear set of equations, $M\tilde{\mathbf{r}} = \mathbf{r}$. All operations except the solution of the linear equations can be done in parallel. This paper improves the parallelism of the linear equations solver.

1.1 Incomplete LU (ILU) Preconditioners

Given a symmetric positive definite matrix A, the Cholesky factorization of A is

$$A = LL^T \tag{2}$$

For large systems A is usually sparse. Due to fill-in, L can be considerably less sparse than A. Thus instead of using L, an approximation to L denoted by $\widetilde{L}$ can be used:

$$A = \widetilde{L}\widetilde{L}^T + B \tag{3}$$

where $B \neq 0$ is an error matrix, and $\widetilde{L}$ is a lower triangular matrix, which is more sparse than L. This matrix implicitly defines $M = \widetilde{L}\widetilde{L}^T$ (M is never computed explicitly) and it is called an *incomplete factorization* of A [11, 13, 17].

There are several ways of constructing and defining $\widetilde{L}$. One way is to construct L, and then discard those entries within L that correspond to zero positions in A. This approach is inefficient in that it requires the computation of the entire L matrix, which is often a costly step. A better way to perform this computation is to simply perform an ordinary factorization of a matrix, but preclude the creation of any new nonzeros. That is, all computation involving fills is suppressed during the factorization process. This simple departure from ordinary LU factorization is the "level 0" ILU algorithm.

The numerical performance of an ILU algorithm can be improved if some fill-in is permitted to occur. The simplest possibility is to permit the occurrence of fills that involve original matrix entries, but preclude the creation of fill entries that depend on prior fills. This is the "level 1" ILU algorithm. Further levels of fills based on prior fills may be permitted, defining higher level incomplete factorization algorithms. The more levels that are included, the closer $\widetilde{L}$ can be expected to be to L. However, more accurate also implies denser.

Other alterations to the basic ILU algorithm include the use of numerically-based rather than topology-based ILU [16], and the use of ordering prior to performing the ILU factorization [5].

1.2 Partitioned Inverse Preconditioners

This section provides a brief review of the partitioned inverse method for solving sparse linear equations [3, 6]. A sparse set of linear equations (1) is solved by direct methods by first ordering A to reduce fills during factorization [15], factoring A into the product of a lower triangular matrix L and an upper triangular matrix U, and then solving:

$$\begin{aligned} L\mathbf{y} &= \mathbf{b} &\quad (4)\\ U\mathbf{x} &= \mathbf{y} &\quad (5)\end{aligned}$$

where (4) is solved for $\mathbf{y}$ by forward substitution, and (5) is solved for $\mathbf{x}$ by backward substitution. If A is symmetric, then $U = L^T$. The remainder of the paper assumes that A is symmetric and positive definite. Define:

$$W = L^{-1} \quad (6)$$

The forward and back substitution steps (4) and (5) are replaced with the matrix-vector products:

$$\begin{aligned} \mathbf{y} &= W\mathbf{b} &\quad (7)\\ \mathbf{x} &= W^T\mathbf{y} &\quad (8)\end{aligned}$$

These products are quite amenable to parallel processing. The matrix L can also be expressed as the product of elementary matrices:

$$L = L_1 L_2 \cdots L_n \quad (9)$$

where the elementary matrix L_i is an identity matrix except for its i^{th} column, which contains column i of L. The inverse of L can be written as:

$$W = L^{-1} = W_n W_{n-1} \cdots W_1 \quad (10)$$

The matrix W is a lower triangular matrix that is usually considerably denser than L. Its graph is the transitive closure of the graph associated with L [3].

An alternate representation for W is based on grouping the elementary inverse factors W_i into m groups of adjacent factor:

$$W = \widehat{W}_m \widehat{W}_{m-1} \cdots \widehat{W}_1 \quad (11)$$

where each $\widehat{W}_k$ is the aggregate of several elementary inverse factor W_j. By aggregating the product in (10) into m factors rather than just one factor, the combined sparsity structure of these m factors of W can be the same as the structure of L itself. With suitable ordering and partitioning algorithms, it is usually possible to have $m \ll n$ [3, 4, 6, 12]. It is also possible to increase the sparsity of the W factors by discarding small entries of precluding the creation of new entries in W. This approximation defines a partitioned inverse preconditioner [1].

1.3 Block Bordered Preconditioners

One of the most common structures for enhancing parallelism in direct sparse matrix methods is the reorganization of a matrix into a blocked-bordered structure. Recent work by Zecevic et al [18] on the heuristics for bordered blocked methods based on structural decompositions ideas has resulted in a class of surprisingly effective recursive block-bordering methods which can be applied to arbitrary structure matrices. The methods are based loosely on the ideas of structural decomposition [18], and can also be related to earlier concept of diakoptics on sparse matrix applications [2, 9, 10]. For another reference on partitioning see also [14] for partitioning based on the Laplacian matrix.

The BBD algorithm is as follows:

1. Determine the valence of all nodes
2. Select a threshold valence value
3. Place all rows/columns with valences greater than the threshold in the lower right hand corner of the matrix.
4. Determine the blocked connected components of the matrix with the lower corner rows/columns removed
5. Selectively move rows from the lower corner back into the upper portion provided that their effect on density and/or block structure is not too significant
6. Group small blocks into single blocks
7. Repeat the bordering/blocking heuristics on the lower right hand corner block

The last step is surprising. Because no connectivity structure was required of the rows placed on the border (e.g., they did *not* have to form a separator or any other special structure), the recursion is quite effective. The tests that follow consider various BBD and approximation heuristics as preconditioners for the PCG method. Figure 1 shows a 118 × 118 matrix ordered to a BBD form (no fills added).

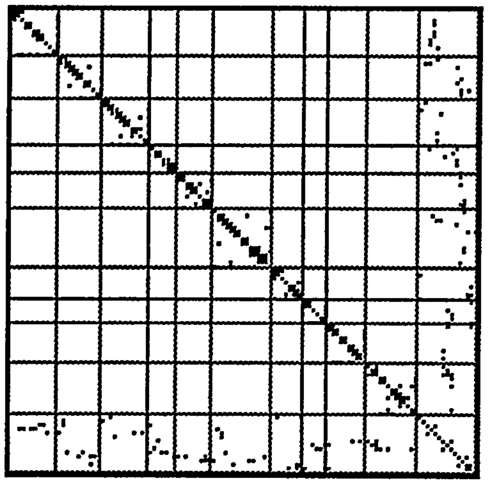


Fig. 1: *Topology of a* 118 × 118 *power system matrix after ordering with BBD prior to the addition of any fills (level 0 preconditioner).*

2 Experimental Results

The experiments reported in this section use power system sparse matrices. Power system problems are usually not well-structured [7] and the average number of nonzeros per row is usually very low (about four). Thus, the use of vector computers or parallel processors has not been very successful to date. Some of the best results have been obtained by ordering these matrices such that either denser rows or columns are formed or the profile of the matrix is reduced [5].

Figure 2 illustrates the solution times as a function of the number of processors using a Sequent Symmetry computer for three methods. The time required to setup the partitioned inverse preconditioner is not considered in this figure. The BBD method assumes that the matrix is block-bordered according to the BBD algorithm, and the resulting structure is used within a level 1 ILU preconditioner. The BD curve refers to the same case but where all entries in the border are also discarded. The number of iterations to convergence is sometimes greater but the total solution time is lower. Tables 1 and 2 give details about the number of iterations and other statistics for four different preconditioners and matrices. In Tables 1 and 2 p refers to number of processors, BBD refers to blocked border diagonal-based preconditioner, BD same as BBD but borders are discarded. The W indicates partitioned inverse preconditioner obtained from ILU1 by full inversion, and $\tilde{W}$ is same as W but no inversion fills allowed. When $\tilde{W}$ is used as preconditioner the number of iterations sometimes become much larger than that of W but less number of arithmetic operations makes it competitive sometimes better preconditioner than W. The row iter in tables refers to number of iterations for PCG.

Table 1: CPU times(s) for 1084x1084 matrix with four different preconditioners.

p	CPU time(s)				
	BBD	BD	W	$\tilde{W}$	ILU1
1	9.2	10.6	3.2	3.04	393
2	5.04	5.7	1.72	1.68	334
4	3.34	3.2	.9	.9	79
8	1.9	1.9	.58	.51	40
iter	9	34	4	7	4

Table 2: CPU times (sec) for 1993x1993 matrix with four different preconditioners.

p	CPU time(s)				
	BBD	BD	W	$\tilde{W}$	ILU1
1	20.1	8.3	14.5	6.8	1062
2	10.2	4.5	7.2	3.7	544
4	5.6	3.5	4.1	2	261
8	3.5	1.4	2.2	1.1	114
17	2.5	.9	1.5	0.8	49
iter	9	12	2	7	2

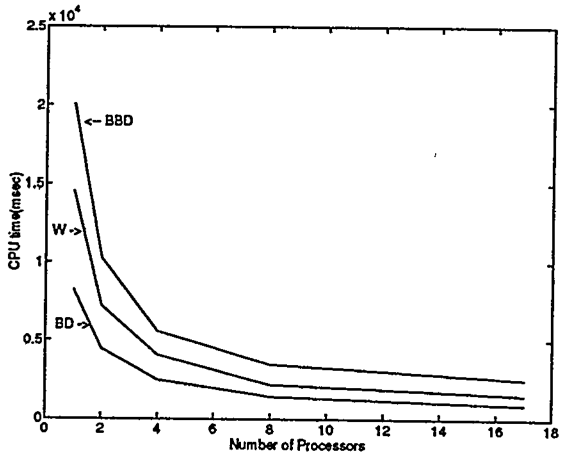


Fig. 2: *Cpu time(ms) for a* 1993 × 1993 *power system matrix. BBD refers to block-bordering without discarding entries, BD refers to block-bordering discarding all border entries, and W refers to the partitioned inverse preconditioner obtained by inversion of the ILU level 1 factors.*

3 Conclusions

Results on a limited number of experiments suggest that block-bordering methods (particularly approximate block-bordering methods after discarding the border) can be effective preconditioners in parallel environments.

REFERENCES

[1] F. L. Alvarado and H. Dağ. Sparsified and incomplete sparse factored inverse preconditioners. In *Copper Mountain Conference on Iterative Methods*, April 1992.

[2] F. L. Alvarado, D. K. Reitan, and M. Bahari-Kashani. Sparsity in diakoptic algorithms. *IEEE Transactions on Power Apparatus and Systems*, PAS-96(5):1450–1459, September/October 1977.

[3] F. L. Alvarado and R. Schreiber. Optimal parallel solution of sparse triangular systems. *SIAM Journal on Scientific and Statistical Computing*, pages 446–460, March 1993.

[4] F. L. Alvarado, D. C. Yu, and R. Betancourt. Partitioned sparse A^{-1} methods. *IEEE Transactions on Power Systems*, 5(2):452–459, May 1990.

[5] Iain S. Duff and Gerard A. Meurant. The effect of ordering on preconditioned conjugate gradients. *BIT*, 29:635–657, 1989.

[6] M. K. Enns, W. F. Tinney, and F. L. Alvarado. Sparse matrix inverse factors. *IEEE Transactions on Power Systems*, 5(2):466–473, May 1990.

[7] A. M. Erisman. Sparse matrix problems in electric power system analysis. In I. S. Duff, editor, *Sparse Matrices and Their Uses*, pages 31–56. New York, Academic, 1980.

[8] G. H. Golub and C. F. Van Loan. *Matrix Computations.* Johns Hopkins University Press, Baltimore, MD, second edition, 1989.

[9] H. H. Happ. *Diakoptics and Networks.* New York, Academic Press, 1971.

[10] H. H. Happ. Diakoptics-the solution of sytem problems by tearing. *Proceedings of the IEEE*, PAS-102(62):930–9, 1974.

[11] James M. Ortega. *Introduction to Parallel and Vector Solution of Linear Systems.* Plenum Press, 1988.

[12] A. Pothen and F. L. Alvarado. A fast reordering algorithm for parallel sparse triangular solution. *SIAM Journal on Scientific and Statistical Computing*, March 1992.

[13] Horst D. Simon. Incomplete *LU* preconditioners for conjugate gradient-type iterative methods. *SPE Reservoir Engineering*, pages 302–306, February 1988.

[14] Horst D. Simon. Partitioning of unstructured problems for parallel processing. Technical Report RNR-91-008, NASA Ames Research Center, Moffett Field, California, February 1991.

[15] W. F. Tinney and J. W. Walker. Direct solutions of sparse network equations by optimally ordered triangular factorization. *Proceedings of the IEEE*, 55(11):1801–1809, November 1967.

[16] David Young . Private Communication at the Copper Mountain Conference on Iterative Methods, April 8–12 1992.

[17] David M. Young. *Iterative Solution of Large Linear Systems.* Academic Press., 1971.

[18] A. I. Zečevič and D. D. Šiljak. Balanced decompositions of sparse systems for multilevel parallel processing. *IEEE Transactions on Circuits and Systems*, to apper.

Saturday, April 9

Applications IV
Chair:
Room B

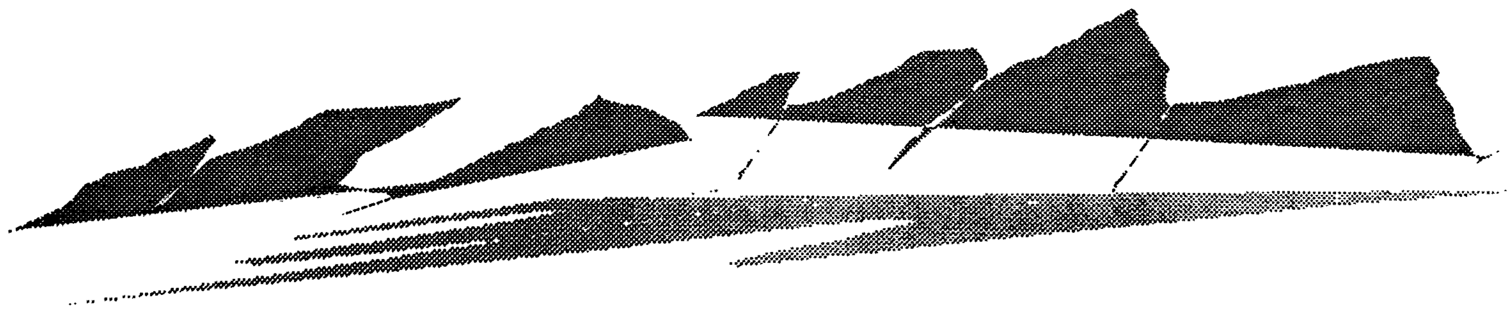
8:00 - 8:25 S.W. Bova
Iterative Methods for Stationary Convection-Dominated Transport Problems

8:25 - 8:50 A.J. Meir
Velocity-Vorticity Formulation of Three-Dimensional, Steady, Viscous, Incompressible Flows

8:50 - 9:15 Louis Howell
A Multilevel Approximate Projection for Incompresssible Flow Calculations

9:15 - 9:40 N.A. Hookey
Simulation of Viscous Flows Using a Multigrid-Control Volume Finite Element Method

9:40 - 10:15 Coffee Break



Iterative Methods for Stationary Convection-dominated Transport Problems

S.W. Bova
G.F. Carey
Computational Fluid Dynamics Laboratory
The University of Texas at Austin

Abstract

It is well known that many iterative methods fail when applied to nonlinear systems of convection-dominated transport equations. Most sucessful methods for obtaining steady-state solutions to such systems rely on time-stepping through an artificial transient, combined with careful construction of artificial dissipation operators. These operators provide control over spurious oscillations which pollute the steady state solutions, and, in the nonlinear case, may become amplified and lead to instability. In the present study, we investigate Taylor Galerkin and SUPG-type methods and compare results for steady-state solutions to the Euler equations of gas dynamics. In particular, we consider the efficiency of different iterative strategies and present results for representative two-dimensional calculations.

Velocity-Vorticity Formulation of Three-Dimensional, Steady, Viscous, Incompressible Flows

by

A. J. Meir
Department of Mathematics
Auburn University
Auburn, AL 36849-5310

In this work we discuss some aspects of the velocity-vorticity formulation of three-dimensional, steady, viscous, incompressible flows. We describe reasonable boundary conditions that should be imposed on the vorticity and a compatibility condition that the vorticity must satisfy.

This formulation may give rise to efficient numerical algorithms for approximating solutions of the Stokes problem, which in turn yields an iterative method for approximating solutions of the Navier-Stokes equations.

A Multilevel Approximate Projection for Incompressible Flow Calculations

Louis H. Howell
Lawrence Livermore National Laboratory
Livermore, CA 94550

An adaptive-mesh projection algorithm for unsteady, variable-density, incompressible flow at high Reynolds number has been developed in the Applied Mathematics Group at LLNL [1]. A grid-based refinement scheme combines the theoretical efficiencies of adaptive methods with the computational advantages of uniform grids [4], while a second-order Godunov method provides a robust and accurate treatment of advection in the presence of discontinuities without excessive dissipation [3]. This paper focuses on the work of the present author concerning the approximate projection itself, which involves the numerical inversion of the operator $\nabla \cdot (1/\rho)\nabla$ on various subsets of the adaptive grid hierarchy.

The projection is approximate in the sense that it does not enforce an exact discrete version of the divergence-free constraint, but it shares several valuable mathematical properties with exact projection methods. A single-grid version of this projection was introduced in [2]. I discuss the advantages of this method over exact formulations, and give a detailed discussion of the appropriate treatment of coarse-fine interfaces. Three basic variations are given: one based on bilinear elements which yields a 9-point 2D Laplacian stencil in uniform parts of the mesh, one based on linear triangular elements which yields a 5-point stencil, and an extension of the latter to a 7-point stencil in 3D.

Solution methods are discussed with an emphasis on efficiency for vector architectures. Since finer levels are advanced with smaller time steps than coarser levels, two distinct projection problems arise. In one, the projection must be performed on a single level, consisting of a union of rectangular patches. In the other, synchronization at the end of a coarse time step requires simultaneous solution on two or more levels, including special difference formulas at the coarse-fine interface. Interesting variable-density problems can involve abrupt density jumps of several orders of magnitude, e.g. the 800-to-1 discontinuity between water and air. A combination of multilevel, multigrid and conjugate-gradient techniques is required for efficient solution of these problems on the adaptive-grid hierarchy.

Some other multilevel methods, e.g. FAC [5], derive the relationships between coarse and fine grid data from the multigrid algorithm used to solve the system. In contrast, the present scheme operates with interface stencils determined by the structure of the discrete projection operator, on a grid structure determined by the requirements of the fluid simulation. Refinement ratios other than two between coarse and fine levels are supported. Four is a common choice.

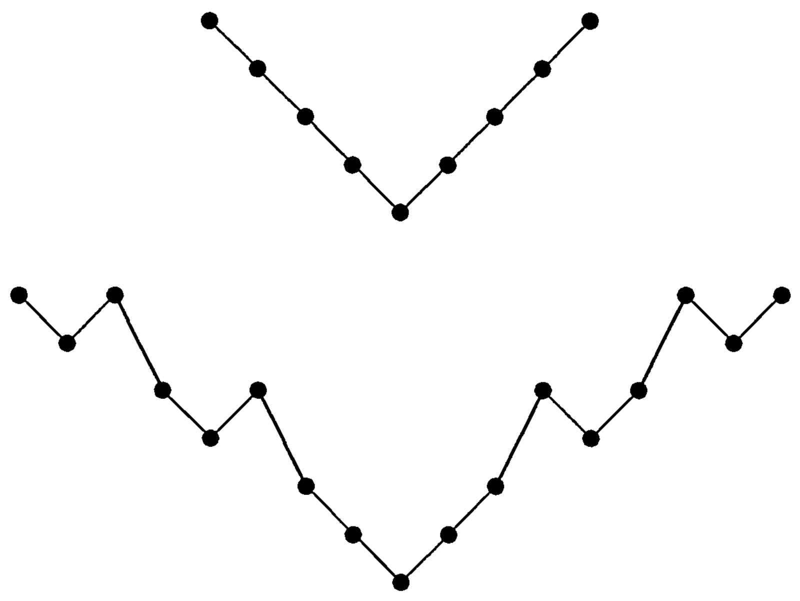


Figure 1: In the level-project multigrid cycle (top), operations apply only to interior points of a level. In the sync-project multilevel cycle (bottom) two operations are defined that cross the coarse-fine interface—computing the residual, and restricting it to the coarse grid.

Additional temporary levels of refinement may be created by the multilevel solver. Since coarse-fine discretizations are only defined between the main levels of the hierarchy, however, the addition of intermediate levels can make the multilevel iteration follow a more complicated pattern than a simple V-cycle. Figure 1 shows typical multilevel cycles for both the level-project and sync-project operations, while Figure 2 illustrates the coarse-fine interface stencils.

Most of the programming for this project has been done in C++, using a variety of data structures for representation of the adaptive mesh hierarchy, auxiliary geometric information, numerical data, boundary conditions, etc. Efficient execution, however, has required that most numerical work be delegated to kernel routines written in FORTRAN. Other efficiency issues range from fast evaluation of inner products on the adaptive mesh to reduction in subroutine call overhead.

Preliminary timings for the 2D level-project on a Cray YMP for a typical 19-grid test problem show 6.9 μsec/zone for the constant-density 5-point stencil, including divergence and gradient expense, with the residual reduced by a factor of 10^6 (6 V-cycles). The constant-density 9-point stencil required 7.9 μsec/zone, while the variable-density 9-point stencil took 17.4 μsec/zone. Additional timings will be presented, along with other numerical examples for problems in both two and three dimensions.

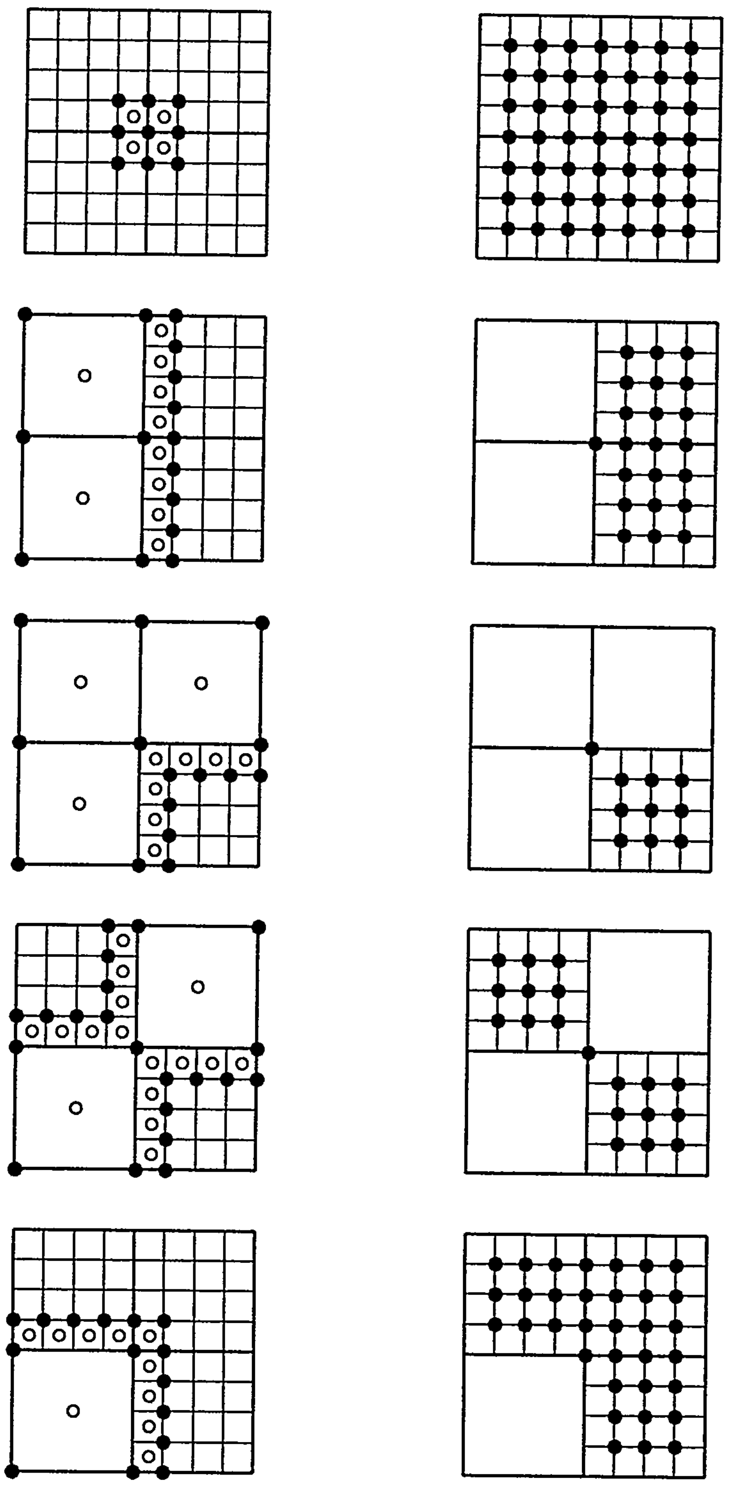


Figure 2: Stencils at grid edges and corners, shown for a refinement ratio of four. On the left, the stencil for $\nabla \cdot \sigma \nabla \phi$ uses ϕ values defined at nodes (solid circles) and σ values defined at cells (open circles). Also, the divergence stencil for $\nabla \cdot V$ uses V defined at these same (open) cell positions. On the right are the stencils for restricting residuals to the coarse grid.

References

[1] A. S. Almgren, J. B. Bell, P. Colella, and L. H. Howell, *An adaptive projection method for the incompressible Euler equations*, in 11th AIAA Computational Fluid Dynamics Conference, Orlando, July 6–9, 1993.

[2] A. S. Almgren, J. B. Bell, and W. G. Szymczak, *A numerical method for the incompressible Navier-Stokes equations based on an approximate projection*, Tech. Report UCRL-JC-112842, LLNL, January 1993.

[3] J. B. Bell, P. Colella, and H. M. Glaz, *A second-order projection method for the incompressible Navier-Stokes equations*, J. Comput. Phys., 85 (1989), pp. 257–283.

[4] M. J. Berger and J. Oliger, *Adaptive mesh refinement for hyperbolic partial differential equations*, J. Comput. Phys., 53 (1984), pp. 484–512.

[5] S. F. McCormick, *Multilevel Adaptive Methods for Partial Differential Equations*, SIAM, Philadelphia, 1989.

SIMULATION OF VISCOUS FLOWS USING A MULTIGRID–CONTROL VOLUME FINITE ELEMENT METHOD

N.A. Hookey
Memorial University of Newfoundland
Faculty of Engineering and Applied Science
St. John's, Newfoundland, Canada, A1B 3X5

Abstract

This paper discusses a multigrid–control volume finite element method (MG–CVFEM) for the simulation of viscous fluid flows. The CVFEM is an equal-order primitive variables formulation that avoids spurious solution fields by incorporating an appropriate pressure gradient in the velocity interpolation functions. The resulting set of discretized equations is solved using a coupled equation line solver (CELS) that solves the discretized momentum and continuity equations simultaneously along lines in the calculation domain. The CVFEM has been implemented in the context of both FMV- and V-cycle multigrid algorithms, and preliminary results indicate a five to ten fold reduction in execution times.

Numerical Method

The CVFEM presented in this paper is based on the equal-order methods proposed by Prakash [1], Hookey and Baliga [2], and Hookey [3]. These methods were successful, however, the author was not satisfied with the convergence rates of the CVFEM for compressible flows developed in [3]. Independently, McCormick [4] developed a finite volume element method (FVEM), which is similar to the CVFEM, and implemented this method in the context of both MG and fast adaptive composite (FAC) grids. This provided the impetus to improve the efficiency of the current CVFEM's by using multigrid techniques. The remainder of this paper presents a brief description of the CVFEM, the implementation details in the context of MG, and preliminary results from a benchmark problem.

Governing equations. The governing equations for steady, incompressible, two-dimensional, viscous fluid flow may be written in the following form [5]:

$$\vec{\nabla} \cdot \vec{J} = S \qquad ; \qquad \vec{\nabla} \cdot \vec{g} = 0 \tag{1}$$

where $\vec{g}$ is the mass flux vector $\rho\vec{v}$, and $\vec{J}$ is the combined convection diffusion flux vector for the appropriate dependent variable. For the x-momentum equation:

$$\vec{J} = \rho\vec{v}u - \mu\vec{\nabla}u \qquad ; \qquad S = S^u - \frac{\partial p}{\partial x} \tag{2}$$

with similar expressions for the y-momentum equation.

Applying the appropriate conservation principle to a control volume V, which is fixed in space, integral forms of Eq. (1) can be obtained:

$$\int_{\partial V} \vec{J} \cdot \vec{n} \, ds = \int_V S \, dV \qquad ; \qquad \int_{\partial V} \vec{g} \cdot \vec{n} \, ds = 0 \tag{3}$$

where ∂V is the surface of the control volume, and $\vec{n}$ is the unit outward normal to the differential area ds.

Domain discretization. In this equal-order CVFEM, the domain is discretized with three node triangular elements, and all dependent variables are stored at all of the nodes in the mesh. The discretization is based on a structured mesh to facilitate the implementation of the multigrid algorithm. Control volumes are constructed around each node by connecting the centroid of each element with the midpoints of its three sides. A sample discretization is shown in Fig. 1.

Interpolation functions. Values of ρ, μ and the appropriate source term S are evaluated at the centroid of an element, and these centroidal values are assumed to prevail over the corresponding element. Pressure is interpolated linearly within an element.

The velocity components are interpolated by functions similar to those proposed in [1, 2, 3]. These functions are defined with respect to a flow-oriented Cartesian coordinate (X, Y) coordinate system, specific to each element, as shown in Fig. 2. The origin of the coordinate system is located at the centroid of the element, and the X axis is aligned with the element average velocity vector, $\vec{v}_{av}$.

If it is assumed, only for deriving a suitable interpolation function for the velocity components, that $\rho\vec{v}_{av}$ prevails over an element, the resulting equation governing the steady-state transport of x momentum within the element, in the X, Y coordinate system is:

$$\rho U_{av}\frac{\partial u}{\partial X} = \mu\left(\frac{\partial^2 u}{\partial X^2} + \frac{\partial^2 u}{\partial Y^2}\right) + S^u \tag{4}$$

where U_{av} is the arithmetic mean of the nodal values of U. An element Peclet number, Pe_Δ, and an exponential variable, ξ, which responds to Pe_Δ and the direction of $\vec{v}_{av}$, can be defined as:

$$Pe_\Delta = \frac{\rho U_{av}}{\mu}\left(X_{max} - X_{min}\right) \quad ; \quad \xi = \frac{\mu}{\rho U_{av}}\left(exp\left(\frac{Pe_\Delta\left(X - X_{max}\right)}{X_{max} - X_{min}}\right) - 1\right) \tag{5}$$

where X_{max} and X_{min} are the maximum and minimum nodal values of X for an element, respectively. This new variable, ξ, is used to propose the following u interpolation function [3], which is a solution to Eq. (4):

$$u = A_u\xi + B_uY + C_u + \left(S^u - \frac{\partial p}{\partial x}\right)\left(\frac{X}{N\rho U_{av}} - \frac{(1 - 1/N)}{2\mu}Y^2\right) \tag{6}$$

where $N > 0$. An interpolation function for v may be derived in a similar manner. It should be noted that it is the inclusion of the pressure gradient in the velocity interpolation functions, and the resultant coupling of the velocity and pressure fields at the interpolation function level, which permits the development of this equal-order CVFEM.

Algebraic approximations. To provide an algebraic approximation to the integral conservation equations, Eq. (3), the interpolation function for u is substituted into Eq. (2), and within each element the convection diffusion flux across a control volume face k is integrated and separated into the following components:

$$\int_O^{M_k} \vec{J}\cdot\vec{n}_k\, ds = C_1^k u_1 + C_2^k u_2 + C_3^k u_3 + E_1^k p_1 + E_2^k p_2 + E_3^k p_3 + B^k \tag{7}$$

The volumetric integration of the source term in Eq. (3) results in additional pressure terms in the discretized momentum equation. An expression similar to Eq. (7) arises for the integration of the flux of y momentum across control volume face k.

The interpolation functions for u and v are used to provide an algebraic approximation to the integral of the mass flux across a control volume face k within an element. This integral may be written in the following form:

$$\int_{O}^{M_k} \vec{g}\cdot\vec{n}_k\, ds = c_1^k u_1 + c_2^k u_2 + c_3^k u_3 + d_1^k v_1 + d_2^k v_2 + d_3^k v_3 + e_1^k p_1 + e_2^k p_2 + e_3^k p_3 + b^k \quad (8)$$

Note that both the u and v interpolation functions contain an element-based pressure gradient, thus the appearance of pressure terms in this integral.

Discretized equations. When the integrated fluxes across the control volume faces in all of the elements are evaluated and added appropriately, the result is a set of simultaneous nonlinear algebraic equations representing the conservation of momentum and mass for each control volume in the mesh. These equations can be written in the following form:

$$a_i^u u_i = \sum_{j=1}^{n} a_j^u u_j + b_i^u p_i + \sum_{j=1}^{n} b_j^u p_j + d_i^u \quad (9)$$

$$a_i^v v_i = \sum_{j=1}^{n} a_j^v v_j + b_i^v p_i + \sum_{j=1}^{n} b_j^v p_j + d_i^v \quad (10)$$

$$a_i^p p_i = \sum_{j=1}^{n} a_j^p p_j + b_i^p u_i + \sum_{j=1}^{n} b_j^p u_j + c_i^p v_i + \sum_{j=1}^{n} c_j^p v_j + d_i^p \quad (11)$$

for the x,y momentum, and continuity equations, respectively. The number of neighbor nodes, n, depends on the orientation of the elements within the mesh, as shown in Fig. 1, and in this formulation $n \le 8$.

Solution of the discretized equations. Previous CVFEM's have used some form of the SIMPLE algorithm [1, 2, 7], however, the implementation of an equal-order CVFEM in the context of a SIMPLEC algorithm, although successful, was not satisfactory [2]. This led to the development of a CELS to solve the coupled discretization equations [3]. The solver used here is more complicated than a previous CELS for a staggered grid finite volume method [6], due to the equations that arise in CVFEM's. This solution technique has been shown to be more robust and efficient than SIMPLE-type algorithms [6], and it is perhaps a better choice for use in the context of multigrid.

Along a particular i or j line in the computational mesh, the discretized momentum and continuity equations, Eqs. (9)-(11) are tridiagonal in pressure and the appropriate velocity components, when the off-line terms are evaluated using currently available values and lumped into the d_i terms. This allows the derivation of a coupled solver which solves the discretized momentum and continuity equations simultaneously along a line in the calculation domain. It is similar in concept to a coupled TDMA or Thomas algorithm. The recursion relations are written in the following forms:

$$\begin{aligned} u_i &= Q_i^u u_{i+1} + R_i^u v_{i+1} + S_i^u p_{i+1} + Z_i^u \\ v_i &= Q_i^v u_{i+1} + R_i^v v_{i+1} + S_i^v p_{i+1} + Z_i^v \\ p_i &= Q_i^p u_{i+1} + R_i^p v_{i+1} + S_i^p p_{i+1} + Z_i^p \end{aligned} \quad (12)$$

and the R, Q, S, and Z coefficients are derived by appropriate substituion of Eq. (12) into Eqs. (9)-(11). The CELS solves for u, v and p simultaneously along a line using Eq. (12), and iteratively improves the overall solution by sweeping the calculation domain line-by-line. One iteration in the CELS consists of one sweep of all i lines followed by a sweep of all j lines. Iterations in the solver are terminated when the norms of the residuals from the three discretized equations in the current iteration are a desired fraction of the norms of the residuals on entering the solver.

Multigrid. The CVFEM has been implemented in FMV- and V-cycle multigrid algorithms. In both cases the coarse to fine grid transfers use linear interpolation, and fine to coarse grid transfers use full weighting, which is particularly well suited to control volume formulations. The velocity and pressure fields from the current finest grid solution are used to determine the coefficients in the discretized equations for the grid level on which the solution is currently proceeding. At each grid level, the coefficients in the discretized equations are evaluated, and the CELS is used until the appropriate convergence criteria is met. Otherwise there is no major difference from the FMV- or V-cycle MG described by McCormick [4].

Results

Preliminary results have been generated for a square driven cavity problem, written in the following nondimensional form:

$$(\vec{v}\cdot\vec{\nabla})\vec{v} = -\vec{\nabla}p + \frac{1}{Re}\nabla^2\vec{v} \qquad ; \qquad \vec{\nabla}\cdot\vec{v} = 0 \tag{13}$$

with u and v zero on the boundaries of the domain, except for the top wall, where $u = 1$, and p at the center node in the domain is set to zero.

All runs were performed using FORTRAN on a DEC R4000 Model 610 AXP, however, it should be noted that the code is not optimal, as the solver and coefficient routines are written in a general manner for solution on nonuniform grids. Thus far, a five to ten fold reduction in execution times has been obtained from the use of multigrid. Detailed results will be presented at the conference.

Conclusion

The successful implementation of a MG–CVFEM for the solution of viscous incompressible flows has clarified several issues prior to the application of FAC grid techniques in the context of CVFEM's. Areas of future research will concentrate on continued refinement of the MG–CVFEM, in particular the simplification and optimization of the implementation, and the inclusion of turbulence models in the context of both incompressible and compressible flows.

References

[1] Prakash, C., An Improved Control Volume Finite-Element Method for Heat and Mass Transfer, and for Fluid Flow Using Equal Order Velocity-Pressure Interpolation, *Numer. Heat Transfer*, vol. 9, pp. 253-276, 1986.

[2] Hookey, N.A. and Baliga, B.R., Evaluation and Enhancements of Some Control Volume Finite-Element Methods—Part 2. Incompressible Fluid Flow Problems, *Numer. Heat Transfer*, vol. 14, pp. 273-293, 1988.

[3] Hookey, N.A., A Control Volume Finite-Element Method for Steady Two-Dimensional Viscous Compressible Flows, Ph.D. thesis, McGill Univ., Montréal, 1989.

[4] McCormick, S.F., *Multilevel Adaptive Methods for Partial Differential Equations*, SIAM, Philadelphia, 1989.

[5] Patankar, S.V., *Numerical Heat Transfer and Fluid Flow*, Hemisphere Publishing Corp., Washington, 1980.

[6] Galpin, P.F., Van Doormaal, J.P. and Raithby, G.D., Solution of the Incompressible Mass and Momentum Equations by Application of a Coupled Equation Line Solver, *Int. J. Numer. Meths. Fluids*, vol. 5, pp. 615-625, 1985.

[7] Baliga, B.R. and Patankar, S.V., Elliptic Systems: Finite-Element Method II, in Minkowycz, W.J., Sparrow, E.M., Schneider, G.E. and Pletcher, R.H. (eds.), *Handbook of Numerical Heat Transfer*, Chap. 11, pp. 421-461, Wiley, New York, 1988.

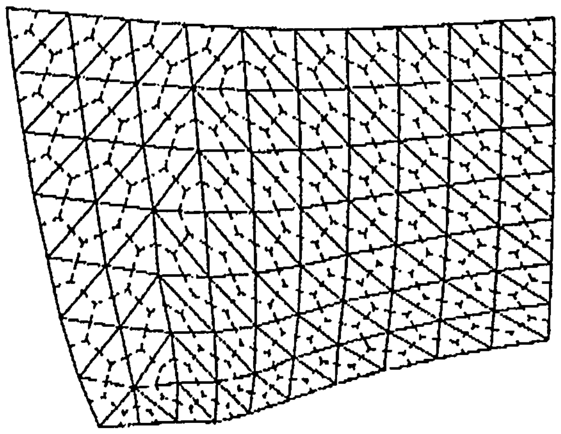


Figure 1: Discretization of a domain with triangular elements (solid lines) and their associated control volumes (dashed lines).

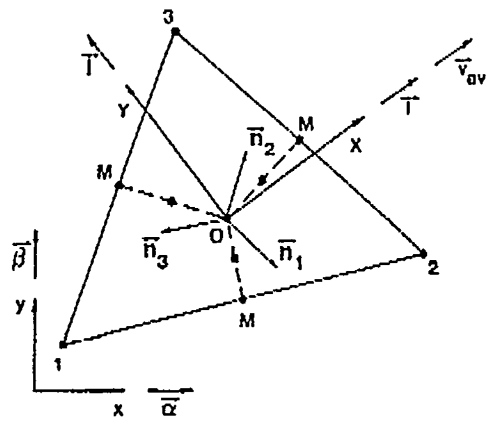


Figure 2: A typical three-node triangular element, with the global x, y and local, flow-oriented X, Y coordinate systems.

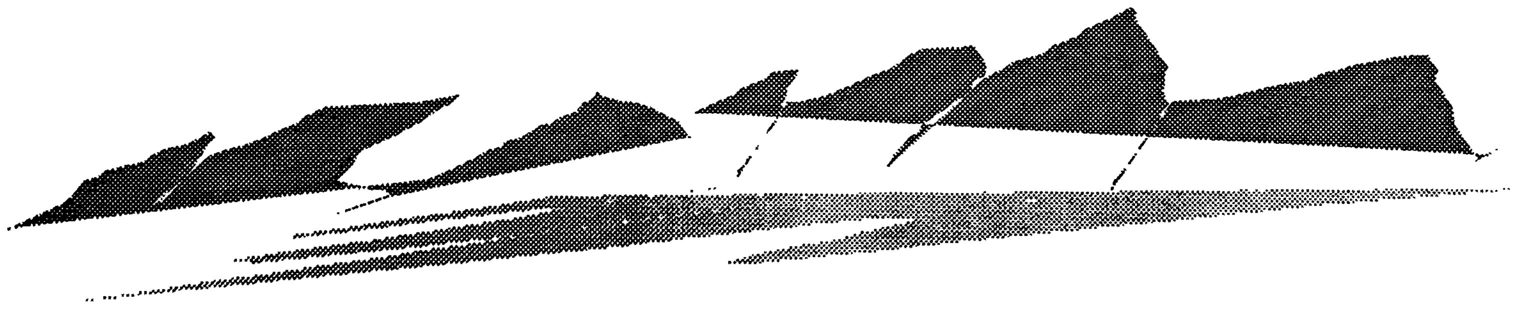
Saturday, April 9

Preconditioners II
Chair: Steve Ashby
Room A

10:15 - 10:40 P. Amodio
Parallel Preconditioning for the Solution of Nonsymmetric Banded Linear Systems

10:40 - 11:05 J.E. Pasciak
Preconditioning the Pressure Operator for the Time Dependent Stokes Problem

11:05 - 11:30 S. Holmgren
A Framework for the construction of preconditioners for systems of PDE



Parallel preconditioning for the solution of nonsymmetric banded linear systems

P. Amodio[*] F. Mazzia[*]

5ᵏ

Many computational techniques require the solution of banded linear systems. Common examples derive from the solution of partial differential equations and of boundary value problems. In particular we are interested in the parallel solution of block Hessemberg linear systems

$$Gx = \mathbf{f}, \tag{1}$$

where

$$G = \begin{pmatrix} D_1 & E_{12} & \dots & E_{1k} & & \\ C_2 & D_2 & E_{23} & & \ddots & \\ & \ddots & \ddots & \ddots & & E_{N-k+1,N} \\ & & \ddots & \ddots & \ddots & \vdots \\ & & & \ddots & D_{N-1} & E_{N-1,N} \\ & & & & C_N & D_N \end{pmatrix}, \tag{2}$$

arising from the solution of ordinary differential equations by means of boundary value methods (BVMs), even if the considered preconditioning may be applied to any block banded linear system.

BVMs have been extensively investigated in the last few years and their stability properties give promising results [6, 7]. In [2] a new class of BVMs, called Reverse Adams, which are BV-A-stable for orders up to 6, and BV-A_0-stable for orders up to 9, have been studied. To analyze the structure of the obtained coefficient matrix, consider the solution of a linear initial value problem

$$\begin{cases} \mathbf{y}' = A(t)\mathbf{y} + \mathbf{b}(t) & t \in [t_0,\ t_f] \\ \mathbf{y}(t_0) = \mathbf{y}_0 \end{cases} \tag{3}$$

by means of a k-step Reverse Adams BVM. This method leads to the discrete problem

[*]Dipartimento di Matematica, Università di Bari, Via E. Orabona 4, I-70125 Bari, Italy, e-mail: 00110570@vm.csata.it

$$\begin{cases} \mathbf{y}_n - \mathbf{y}_{n-1} = h_n \sum_{i=0}^{k} \beta_{in}^k (A(t_{n-1+i}) + \mathbf{b}(t_{n-1+i})), & n = 1, \ldots, N-k+1 \\ \mathbf{y}_n - \mathbf{y}_{n-1} = h_n \sum_{i=0}^{j} \beta_{in}^j (A(t_{n-1+i}) + \mathbf{b}(t_{n-1+i})), & n = N-k+2, \ldots, N, \\ & j = N-n+1 \\ \mathbf{y}_0 \quad \text{given} & \end{cases} \tag{4}$$

where h_i is the stepsize, $t_0 < t_1 < \ldots < t_N = t_f$ are the mesh points, $t_i = t_{i-1} + h_i$, and β_{in}^k are the coefficients of the k-step Reverse Adams method with variable stepsize.

In matrix form, (4) is equivalent to the linear system (1) whose coefficient matrix G is like in (2). For $k = 2$ the block tridiagonal linear system has alreday been solved in [4] by means of a parallel version of the Gauss-Seidel iteration and in [5, 3] by means of parallel preconditionings based on the implicit Euler method and the lower block bidiagonal part of the coefficient matrix. All these methods result to be competitive with respect to the existing ones, in particular with respect to direct methods when $A(t)$ is of medium-large size.

The generalization of all these methods to the solution of (1) - (2) may be not efficient. Direct methods (for example domain decomposition methods) require too much fill-in vectors, while the Gauss-Seidel method may loose its converging properties, since it is not always true that the iteration matrix has spectral radius less than one. Therefore, our attention is turned to conjugate gradient-type iterative methods that are suitable, provided that a good preconditioning improves its rate of convergence.

First of all, consider the following splitting for the matrix G:

$$G = L + U,$$

where

$$L = \begin{pmatrix} D_1 & & & \\ C_2 & D_2 & & \\ & \ddots & \ddots & \\ & & C_N & D_N \end{pmatrix}$$

is non singular, since we choose the stepsizes in the stability domains of the BVM considered.

For small-medium size blocks, the preconditioning based on L seems to be quite efficient. The solution of the preconditioned system may be performed in parallel by means of one of the algorithms derived from the parallel factorizations of tridiagonal matrices described in [1].

When the blocks are large and sparse, the parallel direct method which solves the linear system with the preconditioning as coefficient matrix results to be too expensive because any parallel factorization perform matrix-matrix multiplications which destroys the sparsity pattern of its blocks. Therefore we need different preconditionings for this class of matrices.

Consider as an approximation of G^{-1} the matrix $P^{-1} = (I - L^{-1}U)L^{-1}$. This is equivalent of performing two steps of the Gauss-Seidel iteration with the null vector as initial approximation. We have:

$$P^{-1}G = (I - L^{-1}U)L^{-1}(L + U) = I - (L^{-1}U)^2,$$

that is, the eigenvalues of the preconditioned matrix are inside the ball with center in 1 and radius $\rho((L^{-1}U)^2)$.

The parallel implementation of this algorithm is obtained by using two steps of the parallel Gauss-Seidel iteration [4] as preconditioning. Suppose to have p processors, we consider the splitting $G = L_p + U_p$, where L_p is upper block bidiagonal with lower bidiagonal matrices as main diagonal (see Figure 1).

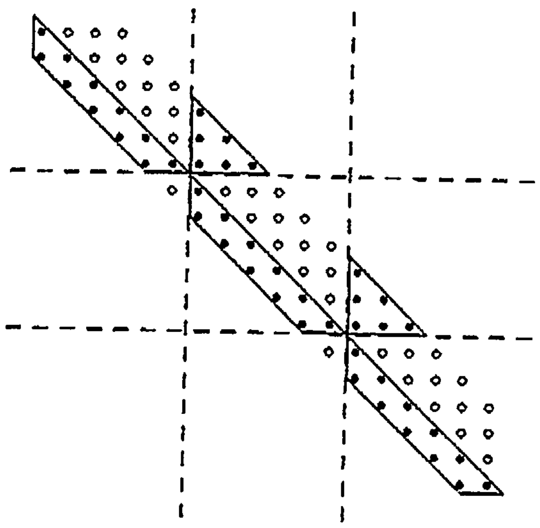


Figure 1: Parallel splitting on $p = 3$ processors for the matrix G. The elements of L_3 are represented by black points.

The most important properties of this splitting are that, for $p \leq n/2$, it is highly parallelizable and $\rho(L_p^{-1}U_p) = \rho(L^{-1}U)$. Therefore, we expect the same rate of convergence and an efficiency of almost 1.

Moreover, since in our problem the elements of the last upper off-diagonals of U_p and L_p are small in modulus, it is possible to neglect them and consider the preconditioning matrix $P_p'^{-1} = (I - L_p'^{-1}U_p')L_p'^{-1}$, where U_p' and L_p' have only few (one or two) upper off-diagonals. This choice minimizes data communications and reduces the number of operations of the whole algorithm.

References

[1] P. Amodio, L. Brugnano, T. Politi, *Parallel Factorizations for Tridiagonal Matrices*, SIAM J. Numer. Anal., 30 (1993) 813–823.

[2] P. Amodio, F. Iavernaro, F. Mazzia. Boundary Value Methods based on Adams-type methods. Report n. 23/93 of the Dipartimento di Matematica, Università di Bari, Italy, 1993.

[3] P. Amodio, F. Mazzia. *Parallel Block Preconditioning for the Solution of Boundary Value Methods*, Report n. 12/93 of the Dipartimento di Matematica, Università di Bari, Italy, 1993.

[4] P. Amodio, F. Mazzia. *A parallel Gauss-Seidel method for block tridiagonal linear systems*, Report n. 13/93 of the Dipartimento di Matematica, Università di Bari, Italy, 1993.

[5] L. Brugnano, D. Trigiante. *A Parallel Preconditioning Technique for BVM methods*, Appl. Num. Math. (in press).

[6] L. Lopez, D. Trigiante. *Boundary Value Methods and BV-stability in the Solution of Initial Value Problems*, Appl. Num. Math. **11**, 225–239, 1993.

[7] P. Marzulli, D. Trigiante. *On some Numerical Methods for Solving ODE*, Quaderno n. 4/1993 of the Istituto di Matematiche Applicate "U. Dini", Pisa, Italy, 1993.

Preconditioning the pressure operator for the time dependent Stokes problem

Authors: James H. Bramble and Joseph E. Pasciak

57

Abstract

In implicit time stepping procedures for the linearized Navier Stokes equations, a linear perturbed Stokes problem must be solved at each time step. Many methods for doing this require a good preconditioner for the resulting pressure operator (Schur complement). In contrast to the time independent Stokes equations where the pressure operator is well conditioned, the pressure operator for the perturbed system becomes more illconditioned as the time step is reduced (and/or the Reynolds number is increased). We escribe methods for solving the coupled velocity/pressure systems and, in particular, show how to construct good preconditioners for the poorly conditioned pressure operator.

A Framework for the Construction of Preconditioners for Systems of PDE

S. Holmgren[‡] and K. Otto[‡]

Department of Scientific Computing, Uppsala University
Box 120, S-751 04 Uppsala
Sweden

December, 1993

Abstract

We consider the solution of systems of partial differential equations (PDE) in 2D or 3D using preconditioned CG-like iterative methods. The PDE is discretized using a finite difference scheme with arbitrary order of accuracy. The arising sparse and highly structured system of equations is preconditioned using a discretization of a modified PDE, possibly exploiting a different discretization stencil. The preconditioner corresponds to a separable problem, and 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 preconditioner solve is of the same complexity. Also, since the preconditioner 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 it is shown how to compute the parameters describing them subject to different optimality constraints. Some of these results recover results derived by e.g. R. Chan, T. Chan, and E. Tyrtyshnikov, but here they are stated in a "PDE context".

Numerical experiments where different preconditioners are exploited are presented. Primarily, high-order accurate discretizations for first-order PDE problems are studied, but also second-order derivatives are considered. The results indicate that utilizing preconditioners based on fast solvers for modified PDE problems yields good solution algorithms. These results extend previously derived theoretical and numerical results for second-order approximations for first-order PDE, exploiting preconditioners based on fast Fourier transforms.

[‡] This work was supported by the Swedish National Board for Industrial and Technical Development (NUTEK)

Saturday, April 9

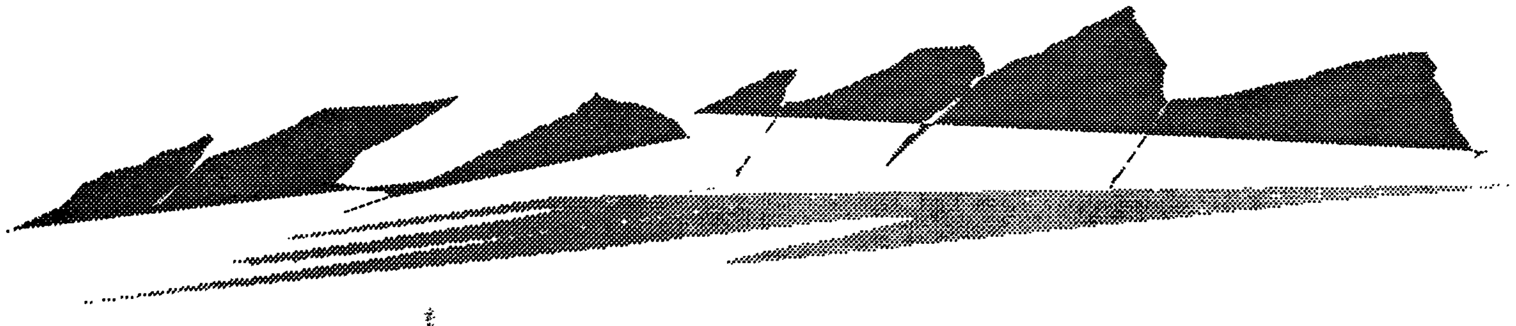
Applications V
Chair:
Room B

10:15 - 10:40 Maryse Page
Iterative Solvers for Navier-Stokes Equations - Experiments with Turbulence Model

10:40 - 11:05 Eli Tziperman
Multilevel Turbulence Simulations

11:05 - 11:30 M. Kamon
Preconditioning Techniques for Constrained Vector Potential Integral Equations, with Application to 3-D Magnetoquasistatic Analysis of Electron Packages

12:00 - 4:30 Informal Discussion



Iterative Solvers for Navier-Stokes Equations - Experiments with turbulence model

Maryse Page*
IREQ - Institut de Recherche d'Hydro-Québec
1800, montée Ste-Julie, Varennes, Canada, J3X 1S1
and
André Garon
Mechanical Engineering Dept, Ecole Polytechnique de Montréal,
C.P. 6079, Succ. "A", Montréal, Canada, H3C 3A7

February 15, 1994

Extended abstract submitted to the Colorado Conference on Iterative Methods,
Breckenridge, Co, April 5-9, 1994

Abstract

In the framework of developing software for the prediction of flows in hydraulic turbine components, Reynolds averaged Navier-Stokes equations coupled with k-ω two-equation turbulence model are discretized by finite element method. Since the resulting matrices are large, sparse and nonsymmetric, strategies based on CG-type iterative methods must be devised. A segregated solution strategy decouples the momentum equation, the k transport equation and the ω transport equation. These sets of equations must be solved while satisfying constraint equations. Experiments with orthogonal projection method are presented for the imposition of essential boundary conditions in a weak sense.

Introduction

To carry out computations of incompressible turbulent viscous flow in 3D domains, Reynolds averaged Navier-Stokes equations coupled with k-ω two-equation turbulence model are discretized by finite element. After linearizations by Newton-Raphson, it results in the solution of linear systems of equations. These systems are characterized by large, sparse and nonsymmetric matrices. To solve these linear systems of equations, we are routinely using a direct method based upon LU factorization. The consideration of larger problems leads us to consider CG-type iterative solvers for these systems. It implies modification of solution strategies, which must take into account the constraints to which the set of equations are subjected.

This paper starts with the description of the governing equations and their discretization. The solution strategies and the solution algorithms are presented, followed by an application.

*Also Ph.D. Student from Mechanical Engineering Dept, Ecole Polytechnique de Montréal

Governing Equations and Discretization

The Reynolds averaged Navier-Stokes equations for incompressible, turbulent viscous flow are formulated as

$$\begin{aligned} \rho\,(\boldsymbol{u}\cdot\nabla)\,\boldsymbol{u} - \nabla\cdot\boldsymbol{\sigma} &= 0 && (1)\\ \nabla\cdot\boldsymbol{u} &= 0 && (2) \end{aligned}$$

with

$$\boldsymbol{\sigma} = -p\;\boldsymbol{g} + \mu_{eff}\;\boldsymbol{\tau}(\boldsymbol{u}), \quad \boldsymbol{\tau}(\boldsymbol{u}) = [\nabla\boldsymbol{u} + \nabla^T\boldsymbol{u}], \quad \mu_{eff} = \mu + \mu_T$$

where $\boldsymbol{u}$ is the fluid velocity vector, ρ is the density, μ is the dynamic viscosity, μ_T is the turbulent viscosity, p is the pressure, $\boldsymbol{g}$ is the metric tensor and $\boldsymbol{\sigma}$ is the stress tensor. The momentum and the continuity equations are completed by the kinetic energy k transport equation and the turbulent frequency ω transport equation (k-ω two-equation model [1]):

$$\begin{aligned} \rho\,(\boldsymbol{u}\cdot\nabla)\,k &= \nabla\cdot(\mu_k\nabla k) - \rho C_\mu k\omega + \mu_T\boldsymbol{D} && (3)\\ \rho\,(\boldsymbol{u}\cdot\nabla)\,\omega &= \nabla\cdot(\mu_\omega\nabla\omega) - \rho C_{\omega 2}\omega^2 + \rho C_{\omega 1}\boldsymbol{D} && (4) \end{aligned}$$

with

$$\mu_k = \mu + \frac{\mu_T}{\sigma_k}, \quad \mu_\omega = \mu + \frac{\mu_T}{\sigma_\omega}, \quad \boldsymbol{D} = \frac{\boldsymbol{\tau}:\boldsymbol{\tau}}{2}, \quad \mu_T = \frac{\rho k}{\omega}$$

where $C_\mu = 0.09$, $C_{\omega 1} = 5/9$, $C_{\omega 2} = 3/40$, $\sigma_k = 2$ and $\sigma_\omega = 2$.

On a computational domain Ω with a boundary Γ and an external normal $\boldsymbol{n}$, the weak Galerkin weighted residual formulation of Reynold averaged Navier-Stokes and k-ω turbulence model are the following:

$$\begin{aligned} \int_\Omega [\rho(\boldsymbol{u}\cdot\nabla)\boldsymbol{u}]\cdot\widehat{\boldsymbol{u}}\;d\Omega + \int_\Omega \mu_{eff}\boldsymbol{\tau}(\boldsymbol{u}):\nabla\widehat{\boldsymbol{u}}\;d\Omega - \int_\Omega p\nabla\cdot\widehat{\boldsymbol{u}}\;d\Omega &= \int_\Gamma (\boldsymbol{\sigma}\cdot\boldsymbol{n})\cdot\widehat{\boldsymbol{u}}\;d\Gamma && (5)\\ -\int_\Omega q\nabla\cdot\boldsymbol{u}\;d\Omega &= 0 && (6)\\ \int_\Omega [\rho(\boldsymbol{u}\cdot\nabla)k]\;\widehat{k}\;d\Omega + \int_\Omega \mu_k\nabla k\cdot\nabla\widehat{k}\;d\Omega + \int_\Omega \rho C_\mu k\omega\widehat{k}\;d\Omega &= \int_\Omega \mu_T\boldsymbol{D}\widehat{k}\;d\Omega + \int_\Gamma \mu_k\frac{\partial k}{\partial n}\widehat{k}\;d\Gamma && (7)\\ \int_\Omega [\rho(\boldsymbol{u}\cdot\nabla)\omega]\;\widehat{\omega}\;d\Omega + \int_\Omega \mu_\omega\nabla\omega\cdot\nabla\widehat{\omega}\;d\Omega + \int_\Omega \rho C_{\omega 2}\omega^2\widehat{\omega}\;d\Omega &= \int_\Omega \rho C_{\omega 1}\boldsymbol{D}\widehat{\omega}\;d\Omega + \int_\Gamma \mu_\omega\frac{\partial \omega}{\partial n}\widehat{\omega}\;d\Gamma && (8) \end{aligned}$$

In practice, a streamline-upwind Petrov-Galerkin technique (SUPG) [2] is used to modify the test functions of the convection and source terms of the transport equations. Continuous quadratic approximations are used for velocity, kinetic energy and turbulent frequency, while linear discontinuous approximation is used for the pressure. The turbulent viscosity μ_T is computed by projection to the nodes of the elements on a L^2 basis.

Solution Strategies

The coupling of the two transport equations to the Navier-Stokes equations leads to a highly nonlinear set of equations. Also, it results in a large matrix system that requires high storage especially if a direct method is used. Gauss-Seidel approach is used, hence solving momentum, then k transport, then ω transport. From one equation to the other, the updated values of the former equations are injected in the current one, resulting in a block Gauss-Seidel decomposition of the global system.

Launder-Spalding law of the wall [1] is used to model the flow between the physical wall and boundary of the computational domain. On these boundaries, a zero-normal velocity constraint is introduced.

With the above segregated solution strategy, the values of k and ω must be updated at each global step. It was found convenient to impose the essential boundary conditions in a weak sense. For example, to solve $-\nabla^2 u = 0$ with $u = u_p$ on some part of the boundary, the following variational formulation can be written:

$$\int_\Omega \nabla\widehat{u}\cdot\nabla u \, d\Omega - \int_{\Gamma_p} \widehat{u}\, p \, d\Gamma = 0 \tag{9}$$

$$-\int_{\Gamma_p} \widehat{p}\,(u - u_p)\, d\Gamma = 0 \tag{10}$$

It can be shown that p is in fact the multiplier $\frac{\partial u}{\partial n}$. This formulation of the problem results in the following system of equations:

$$\begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} U \\ P \end{bmatrix} = \begin{bmatrix} F \\ D \end{bmatrix} \tag{11}$$

This system of equations is very similar to the Stokes problem, where U stands for the velocity vector and P stands for the pressure. In addition to the incompressibility constraint, other constraints can be applied: nodal periodicity, flow rate, zero-normal velocity and so forth.

Solution Algorithms

We are interested in solving system of equations similar to (11), where A is a $n \times n$ nonsymmetric matrix and B is a $m \times n$ matrix. There are mainly three approaches to solve this type of systems: mixed, penalized and projected. For the description of these approaches, homogeneous constraint is considered.

With the mixed approach, all the unknowns are solved simultaneously. In addition to the fact that the matrix is large ($(n+m)\times(n+m)$), the matrix can be indefinite.

A penalty method reduces the number of unknowns by eliminating the variable P and replacing it by an approximation ($r_p BU$). The constraint is only respected up to a certain level depending on the value r_p. Moreover, the penalty parameter r_p must be sufficiently large (e.g., $r_p = 10^8$), thus leading to ill-conditioning and to possible rounding-off errors. It inhibits the use of iterative methods. To reduce the round-off errors, an improved approach is the augmented Lagrangian algorithm based on Uzawa method [3]. It stems from a descent method. Starting with an initial guess for P,

```
Repeat
    $(A + rB^T B)\, U^n = F - B^T P^n$
    $P^{n+1} = P^n + rBU^n$
Until $||F - B^T P - AU|| > \epsilon$ or $||BU|| > \epsilon_{CTR}$
```

While the penalty parameter r can be smaller than r_p (e.g., $r = 10^5$), the system is still ill-conditioned. The term $rB^T B$ can be constructed element by element if a discontinuous approximation is used to interpolate P. For a continuous approximation, the construction of $rB^T B$ requires the explicit construction of the matrices B^T and B. In such a case, it is preferable to consider the next approach.

The last approach is based on an orthogonal projection method which reformulates the system as $\mathcal{P}A\mathcal{P} = \mathcal{P}f$, with $\mathcal{P} = I - B^T(B^T)^\dagger$ and $C^\dagger$ is the Moore-Penrose pseudoinverse of $n \times m$ matrix C, as described by Bramley [4]. The orthogonal projection method based on conjugate gradient method was analyzed by Bramley [4] for the generalized Stokes problem. This approach is particularly suited to CG-type iterative methods which require the results of matrix vector products. The orthogonal projector $\mathcal{P}$ on the null space of B doesn't have to be explicitly constructed. The typical operation $y = \mathcal{P}v$ is decomposed in the following way:

$$\begin{aligned} y &= (I - B^T(B^T)^\dagger)\, v \\ y &= (I - B^T(BB^T)^{-1}B)\, v \\ y &= v - B^T R^{-1} R^{-T} B\, v \end{aligned}$$

where R is the Cholesky factor of the symmetric and positive definite $(m \times m)$ BB^T matrix. The iterative methods proceed in the space where the constraint is satisfied. The variable P is computed by the following relation $P = (B^T)^\dagger(F - BU)$.

Applications

To test the imposition of essential boundary conditions in a weak sense, a two-dimensional advection-diffusion problem is considered

$$\boldsymbol{u}\cdot\nabla\varphi = \nabla\cdot(\alpha\nabla\varphi)$$

on a unit square, with $\alpha = 0.02$, $\boldsymbol{u} = (\cos 67.5^o, \sin 67.5^o)$ and $\varphi|_\Gamma = 0.0$ except on a portion of a side where $\varphi|_{\Gamma_b} = 1.0$. The computational domain is discretized by 800 triangular elements with linear interpolation.

This new constrained problem represented by equation (11) is solved with the orthogonal projection approach. Since the matrix A is nonsymmetric, the projection method must be based on CG-type iterative methods. For this numerical experiment, these iterative methods are BICG [5] CGS [6], BICGSTAB [7], QMR-two-term [8], TFQMR [9], GMRES(k) [10] and CGNE [11].

To have a point of comparison, the usual way of treating essential boundary conditions has also been considered. The same CG-type iterative methods have been applied.

The numerical experiments have been performed within MATLAB 4.0. The convergence histories of the relative residuals $(||r||_2/||r_o||_2)$ for both approaches are presented in figure 1. For the system of equations resulting from the usual treatment of boundary conditions, all the iterative methods are converging to the specified convergence criteria. For the new constrained problem solved by the projection approach, GMRES(10), BICGSTAB and CGNE (CGNR, not shown) methods succeeded in finding the solution, while CGS, BiCG, QMR and TFQMR failed. Further tests will be performed and presented at the conference.

Acknowledgement

This work was partially supported by the National Science and Engineering Research Council of Canada (NSERC). We would like to thank the CEntre de Recherche en Calcul Appliqué (CERCA) for computing support and facilities.

References

[1] D.C. Wilcox. *Turbulence Modeling for CFD.* DCW Industries, Inc., La Cañada, 1993.

[2] A.N. Brooks and T.J.R. Hughes. Streamline Upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations. *Comput. Meths. Appl. Mech. Engrg*, 32:199–259, 1982.

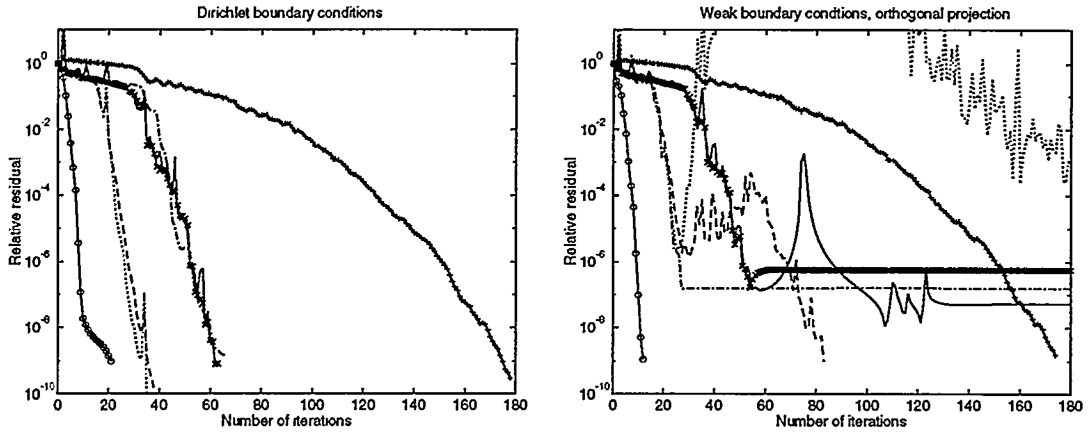


Figure 1: Convergence histories: BICG (solid), CGS (dotted), BICGSTAB (dashed), QMR (two-term) (x-mark), TFQMR (dashdotted), GMRES(10) (circle) at each restart, CGNE (plus)

[3] M. Fortin and R. Glowinski. *Augmented Lagrangian Methods.* North-Holland, Amsterdam, 1983.

[4] R. Bramley. An orthogonal projection algorithm for generalized Stokes problems. Report 1190, CRSD, University of Illinois at Urbana-Champaign, January 1992.

[5] R. Fletcher. Conjugate gradient methods for indefinite systems. In *Lecture Notes in Mathematics, 506*, pages 73–89. 1976. Proc. Dundee Conf. on Numer. Anal., 1975.

[6] P. Sonneveld. CGS, a fast Lanczos-type solver for nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 10(1):36–52, 1989.

[7] H.A. van der Vorst. Bi-CGSTAB: A fast and smoothly converging variant for Bi-CG for the solution on nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 13(2):631–644, 1992.

[8] R.W. Freund and N.M. Nachtigal. An implementation of the QMR method based on coupled two-term recurrences. Technical Report 92.15, RIACS, NASA Ames Research Center, Moffett Field, June 1992. To appear in SIAM Journal on Scientific and Statistical Computing, vol. 15, 1994.

[9] R.W. Freund. A Transpose-Free Quasi-Minimal Residual algorithm for non-Hermitian linear systems. *SIAM J. Sci. Statist. Comput.*, 14:470–482, 1993.

[10] Y. Saad and M.H. Schultz. GMRES: A generalized mininal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 3(7):856–889, 1986.

[11] R.W. Freund, G.H. Golub, and N.M. Nachtigal. Iterative solution of linear systems. *Acta Numerica*, (1):57–100, 1992.

[illegible]

Eli Tziperman
Geophysical Fluid Dynamics Lab
Princeton University
Princeton, NJ 08542
USA
ett@gfdl.gov
Tel: (609) 452-6531
FAX: 609-987-5063

Multilevel Turbulence Simulations

Eli Tziperman, Irad Yavneh and Shlomo Ta'asan
The Weizmann Institute of Science, Rehovot 76100, Israel.

We propose a novel method for the simulation of turbulent flows, that is motivated by and based on the Multigrid (MG) formalism. The method, called Multilevel Turbulence Simulations (MTS), is potentially more efficient and more accurate than LES.

In many physical problems one is interested in the effects of the small scales on the larger ones, or in a typical realization of the flow, and not in the detailed time history of each small scale feature. MTS take advantage of the fact that the detailed simulation of small scales is not needed at all times, in order to make the calculation significantly more efficient, while accurately accounting for the effects of the small scales on the larger scale of interest.

In MTS, models of several resolutions are used to represent the turbulent flow. The model equations in each coarse level incorporate a closure term (roughly corresponding to the tau correction in the MG formalism that accounts for the effects of the unresolvable scales on that grid. The finer resolution grids are used only a small portion of the simulation time in order to evaluate the closure terms for the coarser grids, while the coarse resolution grids are then used to accurately and efficiently calculate the evolution of the larger scales.

The methods efficiency relative to direct simulations is of the order of the ratio of required integration time to the smallest eddies turnover time, potentially resulting in orders of magnitude improvement for a large class of turbulence problems.

PRECONDITIONING TECHNIQUES FOR CONSTRAINED VECTOR POTENTIAL INTEGRAL EQUATIONS, WITH APPLICATION TO 3-D MAGNETOQUASISTATIC ANALYSIS OF ELECTRONIC PACKAGES*

M. KAMON† AND J. R. PHILLIPS‡

Abstract.
In this paper techniques are presented for preconditioning equations generated by discretizing constrained vector integral equations associated with magnetoquasistatic analysis. Standard preconditioning approaches often fail on these problems. We present a specialized preconditioning technique and prove convergence bounds independent of the constraint equations and electromagnetic excitation frequency. Computational results from analyzing several electronic packaging examples are given to demonstrate that the new preconditioning approach can sometimes reduce the number of GMRES iterations by more than an order of magnitude.

1. Introduction. The recently developed multipole-accelerated iterative methods for solving potential integral equations have renewed interest in using discretized integral formulations for the numerical solution of geometrically complicated three-dimensional problems [1, 2]. As multipole-based approaches use implicit matrix representations which can not be easily directly factored, the success of such approaches hinges on reliable convergence of the underlying iterative method. To aid in insuring rapid iteration convergence, multilevel and local inversion preconditioners have been developed for discretized integral equations, and these techniques substantially accelerate convergence rates for fine discretizations [3, 4]. However, multipole-accelerated iterative methods have provided the greatest benefit for engineering applications, which, because of the low accuracy requirements, rarely use fine discretizations.

In this paper, we address the problem of preconditioning systems of equations generated from coarse discretizations of integral equations associated with magnetoquasistatic analysis of complicated three-dimensional geometries. Such analyses are used to aid in the design of a wide variety of electronic packages and electromechanical systems [5]. In the next section, we briefly describe a low-order vortex-like scheme for numerically solving the constrained vector integral equation associated with magnetoquasistatic analysis. In section 3, we examine an electronic packaging example to demonstrate that several standard preconditioning techniques fail to significantly accelerate GMRES [6] convergence. In section 4, we develop a specialized sparsification approach to preconditioning, and prove that the approach eliminates the deleterious effects of the constraint equations. Also, we show that in the uniform discretization, the preconditioner results in a frequency-independent convergence bound. In section 5, we present computational results from combining these new preconditioning techniques with our multipole-accelerated iterative approach. The results are used to show that on difficult electronic packaging examples, the new preconditioning approach can sometimes reduce the number of GMRES iterations by more than an order of magnitude.

2. Background. An integral formulation for magnetoquasistatic analysis can be derived as a special case of the Fourier-transformed Maxwell's equations [7, 8]. The derivation leads to a volume-integral equation of the form

$$J(r) + i\omega \int_{V'} \frac{J(r')}{\|r - r'\|} dv' = -\nabla\Phi(r), \tag{1}$$

with the constraint

$$\nabla \cdot J = I_s(r), \tag{2}$$

where $r, r' \in \mathbf{R}^3$ are locations in 3-space, V is the volume containing electrical conductors, $\Phi(r) \in \mathbf{C}$ is a scalar potential, $J(r) \in \mathbf{C}^3$ is an electric current density, $I_s(r)$ is a given applied current source whose

* This work was supported by the Defense Advanced Research Projects Agency contract N00014-91-J-1698, the National Science Foundation contract (MIP-8858764 A02), a National Science Foundation fellowship, and grants from I.B.M. and Digital Equipment Corporation.

† Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139. (matt@rle-vlsi.mit.edu)

‡ Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology Cambridge, MA 02139. (jphill@rle-vlsi.mit.edu)

volume integral is zero, and ω is the Fourier transform frequency. Of primary engineering interest is the relationship between an applied source and the resulting scalar potential as a function of ω.

2.1. Galerkin Discretization. Typically, (1) is solved by first representing the unknown current density, $\boldsymbol{J}$, as a weighted sum of orthogonal basis functions,

$$J(\boldsymbol{r}) = \sum_{i=1}^{b} I_i \boldsymbol{w}_i(\boldsymbol{r}). \tag{3}$$

Here, $\boldsymbol{w}_1(\boldsymbol{r}), \ldots, \boldsymbol{w}_b(\boldsymbol{r}) : \mathbf{R}^3 \rightarrow \mathbf{R}^3$ are the vector basis functions, and $I_1, \ldots, I_b \in \mathbf{C}$ are the unknown weights. A system of equations for the weights is then generated by applying a Galerkin procedure to the volume integral in (1). That is, the weights are determined by insuring that for each $\boldsymbol{w}_j$,

$$< \boldsymbol{w}_j, (\sum_{i=1}^{b} I_i \boldsymbol{w}_i(\boldsymbol{r}) + i\omega \int_V \frac{\sum_{i=1}^{b} I_i \boldsymbol{w}_i(\boldsymbol{r}) \cdot \boldsymbol{w}_j(\boldsymbol{r}')}{\|\boldsymbol{r} - \boldsymbol{r}'\|} dv' - (-\nabla\Phi(r))) >= 0 \tag{4}$$

where $< a(r), b(r) >$ denotes the inner product given by $\int a(r') \cdot b(r') dV$. Exploiting the basis function orthogonality leads to a simpler system of the form

$$I_j \int_V \int_{V'} \boldsymbol{w}_i(\boldsymbol{r}') \cdot \boldsymbol{w}_i(\boldsymbol{r}) dv' dv + i\omega \int_V \int_{V'} \frac{\sum_{i=1}^{b} I_i \boldsymbol{w}_j(\boldsymbol{r}) \cdot \boldsymbol{w}_i(\boldsymbol{r}')}{\|\boldsymbol{r} - \boldsymbol{r}'\|} dv' dv + \int_{V'} \boldsymbol{w}_j(\boldsymbol{r}') \nabla\Phi(r') dv' = 0 \tag{5}$$

For most engineering calculations, only moderate accuracy is required, and therefore methods based on easily implemented piecewise-constant expansion functions are commonly used. In these methods, a conductor volume is divided into b brick-shaped filaments, as shown in Figure 1. The current density in the i^{th} filament is then approximated as $J = \frac{I_{b_i}}{a_i} l_i$, where $l_i \in \mathbf{R}^3$ is the unit vector along the filament length, I_{b_i} is the net current flow, and a_i is the filament cross-sectional area. Substituting this low order representation in (5) results in a system of the form

$$(R + i\omega L) I_b - V_b = Z I_b - V_b = 0, \tag{6}$$

where $I_b \in \mathbf{C}^b$ is the vector of filament currents, $R \in \mathbf{R}^{b \times b}$ is a diagonal matrix whose diagonals are given by $R_{i,i} = (\textit{filament length})/a_i$, $L \in \mathbf{R}^{b \times b}$ is a dense, symmetric positive definite matrix whose elements are given by

$$L_{i,j} = \frac{1}{a_i a_j} \int_V \int_{V'} \frac{l_i \cdot l_j}{\|\boldsymbol{r} - \boldsymbol{r}'\|} dv' dv, \tag{7}$$

and $V_b \in \mathbf{C}^b$ is the vector of filament voltages whose i^{th}entry is the scalar potential difference at the two ends of the i^{th} filament, averaged over the filament's cross-section.

2.2. Equation System. Current conservation, (2), implies that at the n points where filaments join, the associated filament and source currents must sum to zero. This is directly a statement of Kirchoff's current law law [9], and can be represented as

$$A I_b = I_s, \tag{8}$$

where each column of the incidence matrix $A \in \mathbf{R}^{n \times b}$ contains two nonzero entries, -1 and 1 and I_s is a mostly zero vector with nonzeros corresponding to the source currents in (2).

This nodal analysis approach to formulating current conservation can be combined with (6), yielding a system of equations for the branch currents and scalar potentials given by

$$\begin{bmatrix} Z & -A^t \\ A & 0 \end{bmatrix} \begin{bmatrix} I_b \\ \tilde{\Phi}_n \end{bmatrix} = \begin{bmatrix} 0 \\ I_s \end{bmatrix}, \tag{9}$$

where $\tilde{\Phi}_n$ is the average of Φ over the filament cross-sections that meet at a given point.

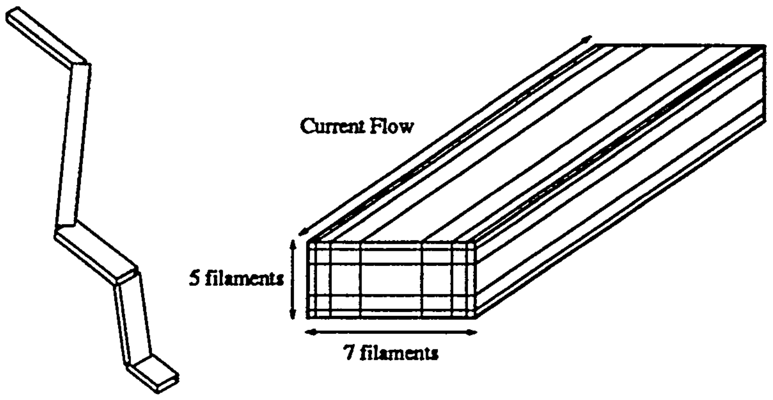


FIG. 1. *Single pin of a pin-connect divided into 5 sections, each of which is a bundle of 35 filaments.*

For complicated problems, the number of branches can easily exceed 20,000. Therefore, since $Z \in \mathbf{C}^{b \times b}$ is dense, (9) will be too expensive to factor directly for complicated problems. Also, iterative methods converge very slowly when applied to solving (9), as the system has two different scales of equations, and so it is extremely poorly conditioned. Instead, it is possible to reformulate the system using mesh analysis [9] to eliminate the explicit current conservation equations. Mesh analysis avoids explicitly representing current conservation in much the same way as a vortex method avoids explicitly enforcing a zero divergence condition.

Mesh analysis can most easily be described by referring to the graph of nodes and branches which represents the connected network of filaments. Specifically, the n points where filaments join are associated with n nodes in the graph, and the filaments themselves are associated with b branches in the graph. The meshes in this graph are all the closed loops of branches which do not enclose any other branch. The unknowns are mesh currents, denoted I_m, and each mesh loop has an associated mesh current flowing around it. When necessary, filament currents can be determined from mesh currents by computing differences. Note also that the since Φ is a scalar potential, the integral of its gradient around any closed loop is zero. This implies that the sum of the branch voltages (defined in (6)) around any mesh must be zero unless there is an outside source, and these constraints can be represented by

$$MV_b = V_s, \tag{10}$$

where V_b is the vector of voltages across each branch except for the source branches, V_s is a mostly zero source vector, and $M \in \mathbf{R}^{b \times m}$ is the mesh matrix, where $m = b' - n + 1$ is the number of meshes and b' is the number of current filaments plus the number of source branches. The relationship between branch currents and branch voltages given in (6) is unchanged, but the branch currents must now be computed from the mesh currents using

$$M^t I_m = I_b, \tag{11}$$

where $I_m \in \mathbf{C}^m$ is the vector of mesh currents, and M^t is the transpose of the mesh matrix above. Combining (11) with (10) and (6) yields

$$MZM^t I_m = V_s. \tag{12}$$

Although MZM^t is generally much better conditioned than the system in (9), for many examples the matrix is still quite ill-conditioned. Consider the two microprocessor packaging examples shown in Figures 2 and 3. Figure 2 shows thirty-five metal pins of a structure used to connect an integrated circuit chip to a printed circuit board. A coarse discretization of this structure yields a 689×689 MZM^t matrix which has a condition number, $\kappa \approx 10^4$. Figure 3 is an example of a printed circuit board to which the connector in Figure 2 might be attached. The printed circuit board is two thin metal sheets sandwiching 255 small copper lines. Coarsely discretizing this structure yields a 751×751 matrix with a condition number, $\kappa \approx 10^6$. From the spectrum shown in Fig. 4 of MLM^t, one can see that, while most of the

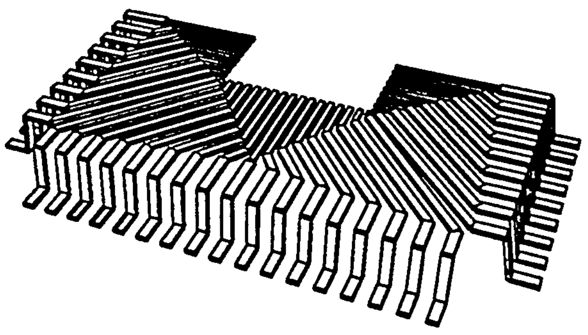


FIG. 2. *Half of a pin-connect package. Thirty-five pins shown.*

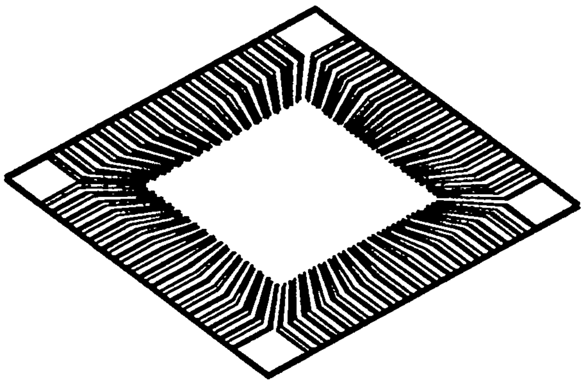


FIG. 3. *A portion of a printed circuit board. Two thin resistive planes sandwich 255 copper lines. Only the outline of the planes is drawn.*

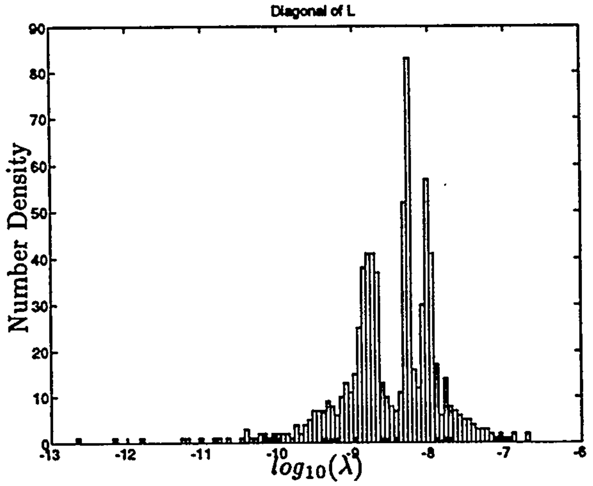


FIG. 4. *Eigenvalue spectrum of MLM^t for a coarse discretization of the printed circuit board example*

eigenvalues are in the interval $[10^{-10}, 10^{-7}]$, the remaining isolated eigenvalues are located toward the origin in the interval $[10^{-13}, 10^{-10}]$. Such small eigenvalues are not easily cancelled with the polynomial produced by the Krylov-subspace methods, so they will significantly slow convergence unless eliminated through preconditioning.

3. Standard preconditioning methods. In this section we discuss the results of applying various standard preconditioners to MZM^t. In general, the GMRES iterative method applied to solving (12) can be significantly accelerated by preconditioning if there is an easily computed good approximation to the inverse of MZM^t. We denote the approximation to $(MZM^t)^{-1}$ by P^{-1}, in which case preconditioning the GMRES algorithm is equivalent to using GMRES to solve

$$(MZM^t)P^{-1}x = V_s \tag{13}$$

for the unknown vector x. The mesh currents are then computed with $I_m = P^{-1}x$.

Local Inversion. An easily computed good approximation to $(MZM^t)^{-1}$ can be constructed by noting that the most tightly coupled meshes are ones which are physically close. To exploit this observation, for each mesh i, the submatrix of MZM^t corresponding to all meshes near mesh i is inverted directly. Then, the row of the inverted submatrix associated with mesh i becomes the i^{th} row of P^{-1}. This idea was originally suggested in [3] and [10]. We refer to this preconditioner as a "local-inversion" preconditioner, because it is formed by inverting physically localized problems.

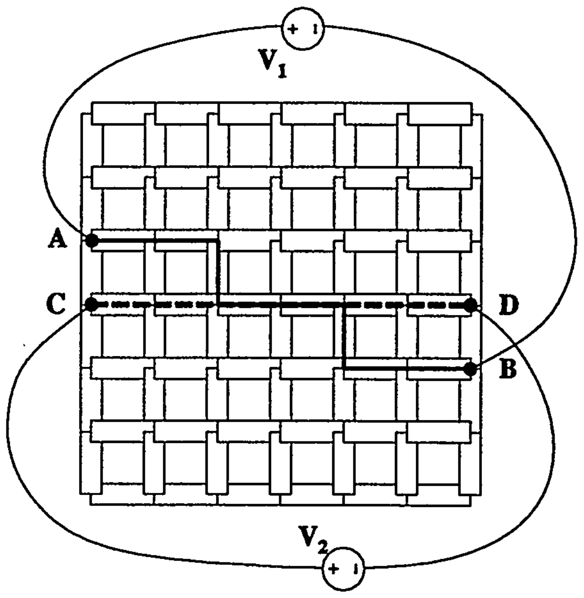


FIG. 5. *Two ground plane meshes due to external sources. One mesh includes the filaments along the path from point A to B and the other from C to D. The filaments that make up the plane are drawn one-third their actual width for illustration.*

A problem arises in forming this preconditioner when a mesh contains many branches. In this case, the branches may span much of the problem domain and what is 'local' is no longer obvious. Consider the example in Fig. 5 of two sources attached to a thin square sheet discretized into an $n \times n$ grid of meshes. Regardless of the level of discretization, most of the meshes include only four branches but the meshes resulting from the two sources include roughly n branches and span much of the problem domain. Therefore, much of the problem can be physically close to these large meshes. For this reason, the large meshes associated with sources cannot be included in the preconditioner, otherwise excessively large subproblems would be inverted directly.

Sparsity Pattern Based Preconditioners. Another idea is to perform incomplete LU factorization of MZM^t based on the sparsity pattern of MRM^t. Incomplete LU is generally ineffective, however, because MZM^t is not necessarily diagonally dominant and therefore ignored terms can become more significant. Another approach along the same lines is to "sparsify" MZM^t to have the sparsity pattern of MRM^t and then compute the exact LU factorization of the sparsified matrix, P.

Fig. 6 shows the results of applying the local-inversion preconditioner, the sparsity-based preconditioner, plus an example of the sparsified-L class of preconditioners to be discussed below. The MZM^t matrix is 751×751 and corresponds to the printed-circuit board example of Fig. 3 in the high frequency limit, that is, as $\omega \to \infty$. The high frequency case is chosen because it has been found to demonstrate the worst case convergence for the sparsified preconditioners, as discussed in the next section. A point worth noting here is that for these sparsified preconditioners in the low frequency limit, $\omega \to 0$, both P and $MZM^t \to MRM^t$ and therefore $(MZM^t)P^{-1} \to I$. The local-inversion preconditioner shows approximately the same convergence behavior at low and high frequency.

From Fig. 6 it is apparent that local-inversion and sparsity-based preconditioning only slightly accelerated convergence. Note that this example includes approximately 300 large meshes.

4. Positive definite sparsifications of L. .

To develop a better preconditioner, instead of sparsifying MZM^t as described above, consider sparsifying the partial inductance matrix, L, and then generating the preconditioner by directly factoring the sparse result, $P = M(R + j\omega L_s)M^t$, where L_s is the sparsified partial inductance matrix. We call this class of preconditioners "sparsified-L."

4.1. The high frequency limit.

As $\omega \to \infty$, the preconditioned matrix reduces to $(MLM^t)(ML_sM^t)^{-1}$. In what follows, it will be shown that L_s should be chosen to be symmetric positive

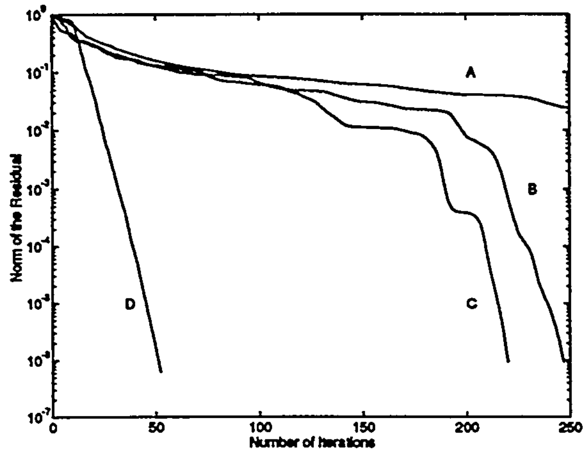


FIG. 6. *Convergence of GMRES applied to PCB example with no preconditioning (A), sparsity-based preconditioning (B), local-inversion preconditioning (C), and sparsified-L preconditioning using the diagonal of L (D)*

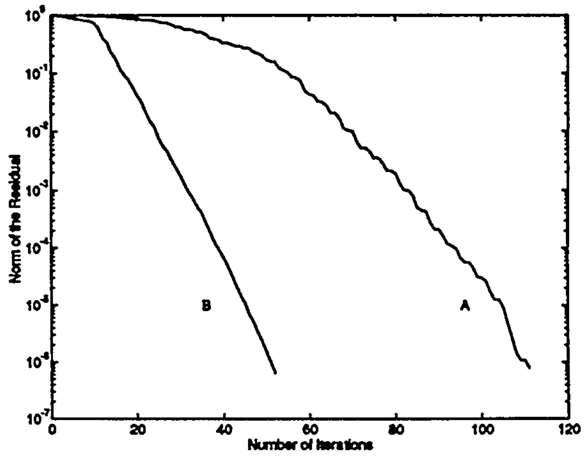


FIG. 7. *Convergence of GMRES applied to PCB example with threshold preconditioning for $\epsilon = 10^{-1}$ (A), and diagonal-of-L preconditioning (B)*

definite for this sparsified-L class of preconditioners to be effective.

Lemma 4.1. The product of real symmetric positive definite matrices has positive eigenvalues.

Proof. Let A and B be real symmetric positive definite matrices, then $A^{-1/2}$ and $B^{-1/2}$ exist and are also symmetric positive definite. AB has the same eigenvalues as

$$D = A^{-1/2}ABA^{1/2} = A^{1/2}BA^{1/2} = (B^{1/2}A^{1/2})^t(B^{1/2}A^{1/2}) = C^tC. \tag{14}$$

Then, for any vector x, $x^tDx = (Cx)^t(Cx) = y^ty > 0$, where $y = Cx$. Therefore, D has positive eigenvalues. □

Theorem 4.2. If L_s is symmetric positive definite, then the preconditioned system, $(MLM^t)(ML_sM^t)^{-1}$ has positive eigenvalues.

Proof. For any x, let $y = M^tx$. Then $x^t(MLM^t)x = y^tLy > 0$ since L is positive definite. Following a similar argument for ML_sM^t and using Lemma 4.1, the theorem is proved. □

As an example, consider choosing L_s to be a threshold sparsification of L, that is, form L_s by zeroing all terms except those that satisfy $L_{ij}^2 > \epsilon\,|L_{ii}L_{jj}|$, for some ϵ. In this case, L_s is not necessarily positive definite. Fig. 7 compares this preconditioner for $\epsilon = 0.1$ to the preconditioner formed by taking L_s to be only the diagonal of L, which is obviously positive definite. Fig. 7 clearly indicates that using the threshold sparsification preconditioner results in slower convergence than using the diagonal-of-L preconditioner. This can be explained by examining the spectra of the preconditioned matrices in Fig. 8. For both cases, the eigenvalues seem similarly clustered, except the threshold preconditioned matrix has a distribution of negative eigenvalues while the diagonal-of-L preconditioned matrix does not.

Theorem 4.2 leads to the result that the condition number of preconditioned system in the high frequency limit is bounded independent of the mesh matrix, M.

Theorem 4.3. If L_s is positive definite, then

$$\kappa[(MLM^t)(ML_sM^t)^{-1}] \le \kappa(LL_s^{-1})$$

where, for a matrix with positive eigenvalues A, the condition number $\kappa(A)$ is defined as $\kappa(A) = \lambda_{\max}(A)/\lambda_{\min}(A)$ and $\lambda_{\max}(A)$, $\lambda_{\min}(A)$ are the maximum and minimum eigenvalues, respectively.

Proof. For the matrix $A = MLM^t$ with preconditioner $P = ML_sM^t$, and with the maximum eigenvalue $\lambda_{\max}(AP^{-1})$, there is some y such that $AP^{-1}y = \lambda_{\max}y$. Then for $x = P^{-1}y$, the generalized eigenvalue problem $MLM^tx = \lambda_{\max}ML_sM^tx$ is satisfied. Therefore, $\lambda_{\max}(AP^{-1}) =$

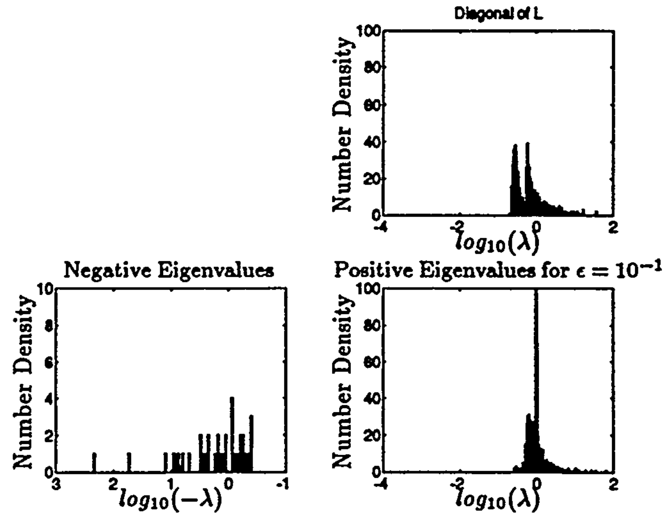


FIG. 8. *Eigenvalues for the diagonal-of-L preconditioned system and the threshold preconditioned system.*

$x^tMLM^tx/x^tML_sM^tx$ and so there is also a vector $u = M^tx$ such that $\lambda_{\max}(AP^{-1}) = u^tLu/u^tL_su$ and so

$$\lambda_{\max}(AP^{-1}) \leq \max_u \frac{u^tLu}{u^tL_su}$$

By a similar argument,

$$\lambda_{\min}(AP^{-1}) \geq \min_v \frac{v^tLv}{v^tL_sv}$$

and thus

$$\kappa[MLM^t(ML_sM^t)^{-1}] \leq \kappa(LL_s^{-1})$$

□

The above two theorems lead to the conclusion that one should focus on choosing positive definite L_s matrices. As described above, the sparsest approach would be to take the diagonal of L. Another approach is to divide physical space into cubes and then to include in L_s principal submatrices of L corresponding to the groups of filaments contained inside each cube where a filament may be included in only one cube. Thus, by appropriately numbering the branches, L_s can be written as a block diagonal matrix, and therefore we refer to it as a "cube-block" preconditioner.

Theorem 4.4. L_s for the cube-block preconditioner is positive definite.

Proof. The set of eigenvalues of a block diagonal matrix is the union of the sets of eigenvalues from each block. Since L is symmetric positive definite, so are all of its principal submatrices (See, for instance [11], p. 397). Given the block diagonals of L_s are principal submatrices of L, the theorem is proved. □

4.2. The general case. Under certain conditions the bound on GMRES convergence in the limit as $\omega \to \infty$ holds for all ω.

Theorem 4.5. Given a problem discretized with filaments of uniform size, and assuming that the GMRES algorithm uses the diagonal-L preconditioner, the residual at iteration k, $r^k = b - \tilde{Z}(\omega)x^k$, where $\tilde{Z}(w)$ is the preconditioned Z, satisfies

$$\frac{||r^k||}{||r^0||} \leq 2\left[\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right]^k \quad , \tag{15}$$

where $\kappa = \kappa(LL_s^{-1}) = \kappa(L)$, independent of frequency. The following observation and short lemma will be used to prove Theorem 4.5.

Observation 4.6. If all filaments are of the same size, the matrices R and L are constant along the main diagonal. The preconditioner P constructed from the main diagonal of L is $P = (r + i\omega l)MM^t$, where r and l are the diagonal elements of R and L respectively.

Lemma 4.7. Given $Z = M(rI + i\omega L)M^t$, and $P = M(r + i\omega l)M^t$, the preconditioned matrix

$$\tilde{Z} = P^{-\frac{1}{2}} Z P^{-\frac{1}{2}} \tag{16}$$

is of the form

$$\tilde{Z} = Ce^{j\theta}[T + j\sigma I] \tag{17}$$

where $T = T^t$ is real-symmetric and $C, \sigma, \theta \in \mathbf{R}$.

Proof. As MM^t is symmetric positive-definite, P^{-1} may be factored as $P^{-1} = P^{-\frac{1}{2}}P^{-\frac{1}{2}}$, with

$$P^{-\frac{1}{2}} = \frac{1}{\sqrt{r + i\omega l}}(MM^t)^{-\frac{1}{2}} \tag{18}$$

Combining with Eq. 16,

$$P^{-\frac{1}{2}}ZP^{-\frac{1}{2}} = \left[\frac{r}{i\omega}I + (MM^t)^{-\frac{1}{2}}MLM^t(MM^t)^{-\frac{1}{2}}\right]\frac{i\omega}{r + i\omega l} \tag{19}$$

or

$$P^{-\frac{1}{2}}ZP^{-\frac{1}{2}} = \frac{\omega^2}{r^2 + \omega^2 l^2}e^{i\theta}\left[i\sigma I + T\right] \tag{20}$$

with $\theta = \tan^{-1}\frac{r}{\omega l}$, $\sigma = -r/\omega$,

$$T = (MM^t)^{-\frac{1}{2}}MLM^t(MM^t)^{-\frac{1}{2}} \tag{21}$$

MLM^t is positive-definite since L is, and therefore T is symmetric positive-definite. □

We are now ready to prove Theorem 4.5.

Proof. From Lemma 4.7, $\tilde{Z}$ is of the form $T + i\sigma I$, T symmetric positive definite. Therefore, using Theorem 4 in [12], the computed iterates x^k satisfy

$$\frac{||b - Ax^k||}{||b - Ax^0||} \le \frac{2}{R^k + 1/R^k} \le \frac{2}{R^k} \tag{22}$$

with $R = c(\omega) + \sqrt{c(\omega)^2 - 1}$, and where

$$c(\omega) = \frac{\sqrt{\lambda_{\max}(T)^2 + \sigma^2} + \sqrt{\lambda_{\min}(T)^2 + \sigma^2}}{\lambda_{\max}(T) - \lambda_{\min}(T)} \tag{23}$$

Since

$$c(\omega) < c_\infty \equiv \frac{\lambda_{\max}(T) + \lambda_{\min}(T)}{\lambda_{\max}(T) - \lambda_{\min}(T)} = \frac{\kappa(T) + 1}{\kappa(T) - 1} \tag{24}$$

Then

$$R \le c_\infty + \sqrt{c_\infty^2 - 1} = \frac{\sqrt{\kappa(T)} + 1}{\sqrt{\kappa(T)} - 1} \le \frac{\sqrt{\kappa(L)} + 1}{\sqrt{\kappa(L)} - 1} \tag{25}$$

which combined with Eq. 22 proves the theorem. □

Remark. The preconditioned matrix $\tilde{Z}$ is normal and its eigenvalues lie on a line in the complex plane.

Proof. Defining $C = \frac{\omega^2}{r^2 + \omega^2 l^2}$, $\tilde{Z}$ is normal since $\tilde{Z}\tilde{Z}^H = C^2(T^2 + \sigma^2 I) = \tilde{Z}^H\tilde{Z}$. The eigenvalues $\lambda(\tilde{Z}) = Ce^{i\theta}(i\sigma + \lambda(T))$ clearly lie on a line, as the $\lambda(T)$ are real. □

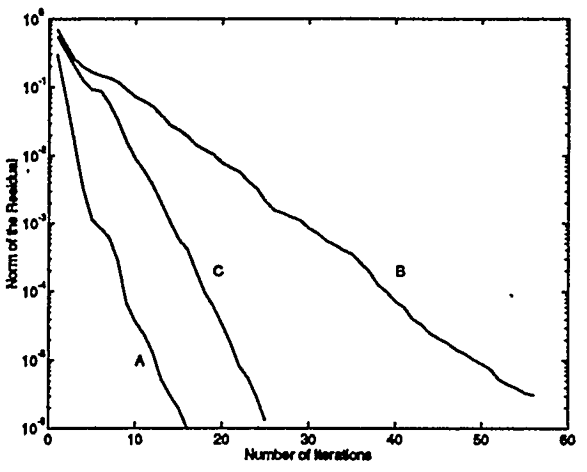


FIG. 9. *Convergence of GMRES applied to the 35-pin package example with cube-block preconditioning (A), diagonal-of-L preconditioning (B), and local-inversion preconditioning (C).*

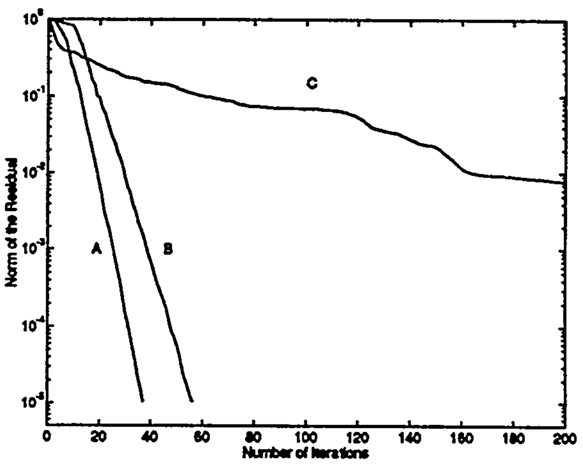


FIG. 10. *Convergence of GMRES applied to the PCB example with cube-block preconditioning (A), and diagonal-of-L preconditioning (B), and local-inversion preconditioning (C).*

Preconditioner type	Size of MZM^t	Preconditioner factor time	Total execution time	Total number of iterations	Average # of iters. per solve
diagonal-L	751x751	0.26	450.46	729	41
cube-block	751x751	6.07	254.35	374	21
diagonal-L	1099x1099	0.81	1042.57	760	42
cube-block	1099x1099	11.86	755.12	518	29
diagonal-L	2101x2101	3.43	1901.58	760	42
cube-block	2101x2101	44.91	1381.15	502	28
diagonal-L	4351x4351	15.87	5522.79	842	47
cube-block	4351x4351	174.13	4609.96	641	36
diagonal-L	7501x7501	46.24	8894.92	883	49
cube-block	7501x7501	452.11	7309.18	635	35

TABLE 1

Execution times and iteration counts for diagonal-of-L and cube-block preconditioning of the printed circuit board example. Times are in CPU seconds for the DEC AXP3000/500.

5. Computational results. To compare the relative merits of the cube-block, diagonal-of-L, and local-inversion preconditioners, consider the pin-connect example of Fig. 2 and the printed circuit board (PCB) example of Fig. 3. For this experiment, the pin-connect was discretized into 3488 filaments which corresponds to 3305 meshes and each of the thin planes in the PCB was discretized into a 60×60 grid of meshes giving a total 7501 meshes including the copper lines. The GMRES error in the solution at high frequency as a function of iteration is plotted in Fig. 9 for the pin-connect example, and in Fig. 10 for the PCB example. As the figures clearly show, the block diagonal preconditioners are an improvement over the diagonal-of-L and local-inversion preconditioners. It is worth noting that, unlike the PCB example, for the pin-connect example, local-inversion preconditioning did better than diagonal-of-L. This behavior can be expected since there are only 35 large meshes which must be excluded from the local-inversion preconditioner.

From Table 1 it can be observed that the time to compute the preconditioners is negligible compared to the total execution time, although for larger problems, the time required to compute the cube-block preconditioner may become significant. Also, the required number of iterations for either of the preconditioners does not grow rapidly with problem size.

6. Conclusions and Acknowledgments. In this paper we developed the sparsified-L class of preconditioners for the linear systems generated by integral equations associated with magnetoquasistatic analysis. This class of preconditioner was shown to eliminate the apparent ill-conditioning caused by the constraint equations. For the two industrial examples presented in this paper, the preconditioners can significantly reduce the number of GMRES iterations.

The authors would like to thank Jacob White for his enthusiasm and guidance.

REFERENCES

[1] K. Nabors and J. White, "Fast capacitance extraction of general three-dimensional structures," *IEEE Trans. on Microwave Theory and Techniques*, To Appear 1992.

[2] L. Greengard and V. Rokhlin, "A fast algorithm for particle simulations," *Journal of Computational Physics*, vol. 73, pp. 325–348, December 1987.

[3] S. A. Vavasis, "Preconditioning for boundary integral equations," *SIAM J. Matrix Anal. Appl.*, vol. 13, pp. 905–925, 1992.

[4] K. E. Atkinson, "A survey of boundary integral equation methods for the numerical solution of laplace's equation in three dimensions," in *Numerical Solution of Integral Equations* (M. A. Goldberg, ed.), pp. 1–34, New York: Plenum Press, 1990.

[5] A. E. Ruehli and P. A. Brennan, "Efficient capacitance calculations for three-dimensional multiconductor systems," *IEEE Transactions on Microwave Theory and Techniques*, vol. 21, pp. 76–82, February 1973.

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

[7] A. E. Ruehli, "Survey of computer-aided electrical analysis of integrated circuit interconnections," *IBM Journal of Research and Development*, vol. 23, pp. 626–639, November 1979.

[8] A. C. Cangellaris, J. L. Prince, and L. P. Vakanas, "Frequency-dependent inductance and resistance calculation for three-dimensional structures in high-speed interconnect systems," *IEEE Transactions on Components, Hybrids, and Manufacturing Technology*, vol. 13, pp. 154–159, March 1990.

[9] C. A. Desoer and E. S. Kuh, *Basic Circuit Theory*. New York: McGraw-Hill, 1969.

[10] K. Nabors, S. Kim, J. White, and S. D. Senturia, "An adaptive multipole algorithm for 3-d capacitance calculation," in *Proceedings of the International Conference on Computer Design*, October 1991.

[11] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge: Cambridge University Press, 1985.

[12] R. Freund, "On conjugate-gradient type methos and polynomial preconditioners for a class of complex non-hermitian matrices," *Numer. Math.*, vol. 57, pp. 285–312, 1990.

Saturday, April 9

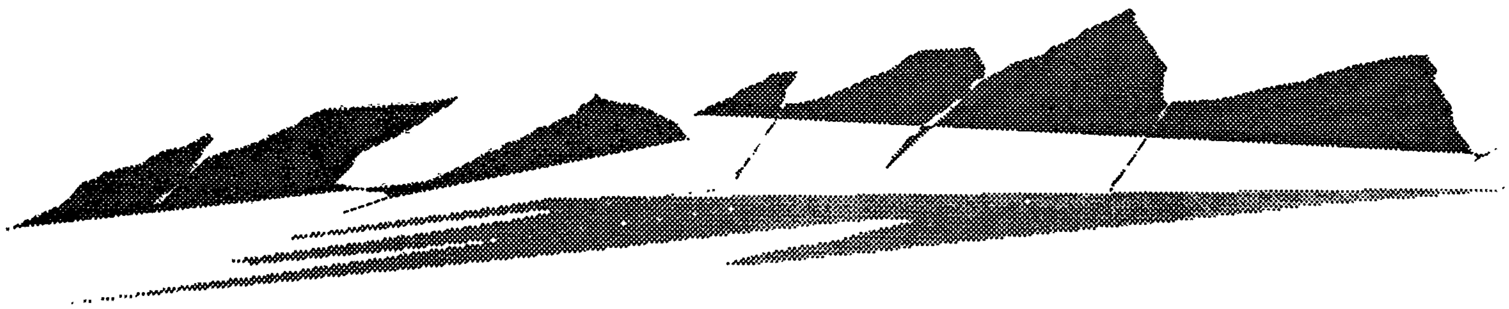
Toeplitz and Circulant Matrix Solvers
Chair:
Room A

4:45 - 5:10 Thomas Huckle
Iterative Methods for Toeplitz-Like Matrices

5:10 - 5:35 Paul Saylor
A Modified Direct Preconditioner for Indefinite Symmetric Toeplitz Systems

5:35 - 6:00 Eugene E. Tyrtyshnikov
Circulant Preconditioners with Unbounded Inverses: Why Non-Optimal Preconditioners may Possess a Better Quality than Optimal Ones

6:00 - 6:25 Seymour Parter
A Remark on Band-Toeplitz Preconditions for Hermitian Toeplitz Systems



Iterative Methods for Toeplitz-like Matrices

Thomas Huckle
Institut für Angewandte Mathematik und Statistik
Universität Würzburg
D-97074 Würzburg, F.R.G.

Abstract. In this paper we will give a survey on iterative methods for solving linear equations with Toeplitz matrices, Block Toeplitz matrices, Toeplitz plus Hankel matrices, and matrices with low displacement rank. We will treat the following subjects:

- optimal (w)-circulant preconditioners as a generalization of circulant preconditioners;
- Optimal implementation of circulant-like preconditioners in the complex and real case;
- preconditioning of near-singular matrices; what kind of preconditioners can be used in this case;
- circulant preconditioning for more general classes of Toeplitz matrices; what can be said about matrices with coefficients that are not l_1-sequences;
- preconditioners for Toeplitz least squares problems, for block Toeplitz matrices, and for Toeplitz plus Hankel matrices;

Key Words. Fourier Transform, Sine Transform, Toeplitz matrices, preconditioned conjugate gradient method

AMS(MOS) Subject Classifications. 65F10,65N06

TITLE:

A Modified Direct Preconditioner for Indefinite Symmetric Toeplitz Systems

AUTHORS:
Paul Concus
Lawrence Berkeley Lab
Berkeley, CA

and

Paul Saylor
Dept of Computer Science
The University of Illinois
Urbana, IL.

ABSTRACT:

A modification is presented of the classical $O(n^2)$ algorithm of Trench for the direct solution of Toeplitz systems of equations. The Trench algorithm can be guaranteed to be stable only for matrices that are (symmetric) positive definite; it is generally unstable otherwise. The modification permits extension of the algorithm to compute an approximate inverse in the indefinite symmetric case, for which the unmodified algorithm breaks down when principal submatrices are singular. As a preconditioner, this approximate inverse has an advantage that only matrix-vector multiplications are required for the solution of a linear system, without forward and backward solves. The approximate inverse so obtained can be sufficiently accurate, moreover, that, when it is used as a preconditioner for the applications investigated, subsequent iteration may not even be necessary. Numerical results are given for several test matrices. The perturbation to the original matrix that defines the modification is related to a perturbation in a quantity generated in the Trench algorithm; the associated stability of the Trench algorithm is discussed.

Name	Eugene E. Tyrtyshnikov
Affiliation	Institute of Numerical Mathematics of the Russian Academy of Sciences
Street Address	Leninski Prosp. 32-A Moscow 117334
Country	Russia
E-mail Address	tee@adonis.iasnet.com
Fax	(095) 938-1821

ABSTRACT:

Circulant Preconditioners with Unbounded Inverses: Why Non-Optimal Preconditioners May Possess a Better Quality Than Optimal Ones

Eugene E. Tyrtyshnikov

Institute of Numerical Mathematics
Russian Academy of Sciences
Leninski Prosp. 32-A
Moscow 117334, Russia
E-mail: tee@adonis.iasnet.com

There exist several preconditioning strategies for systems of linear equations with Toeplitz coefficient matrices. The most popular of them are based on the Strang circulants and the Chan optimal circulants. Let A_n be an n-by-n Toeplitz matrix. Then the Strang preconditioner S_n copies the central n/2 diagonals of A_n while other diagonals are determined by the circulant properties of S_n. The Chan circulant C_n coincides with the minimizer of the deviation A_n - C_n in the sense of the matrix Frobenius norm. At the first glance the Chan circulant should provide a faster convergence rate since it exploits more information on the coefficient matrix.

The preconditioning quality is heavily dependent on clusterization of the preconditioned eigenvalues. According to recent results by R.Chan we know that both considered circulants possess the clustering property if the coefficient Toeplitz matrices A_n are generated by a function which first belongs to the Wiener class and second is separated from zero. Both circulants provide approximately the same clustering rate, and therefore both should possess the same preconditioning quality.

However, the most interesting case is the one when the generating function may take the zero value, and hence the circulants have unbounded in n inverses. In these cases the Strang preconditioners may appear to be singular and we recommend to use the so called improved Strang preconditioners (in which a zero eigenvalue of the Strang circulant is replaced by some positive value).

For example, if the generating function is of the form

$$f(x) = 2 - 2\cos(x)$$

the Chan optimal circulants provide the preconditioned matrices of sizes

$$n = 256, 512, 1024$$

which have respectively

$$26, 40, 77$$

eigenvalues located at least at the distance 0.1 from the point 1. For the Strang circulants the corresponding numbers of eigenvalues are equal to

$$3, 2, 3.$$

Therefore, the (improved) Strang circulants may lead to a better clustering rate than the Chan optimal circulants.

We show that the 'Strang vs Chan' comparision problem is indeed reduced to the 'Dirichlet vs Fejer' (or 'Fourier vs Cesaro') comparison problem. One could expect that the smoother the generating function is the faster the clustering rate should be. We prove that this statement holds true only for the Strang circulants. At the same time, the smoothness of the generating functions does not affect the clustering rate of the Chan optimal circulants. Various numerical examples provide excellent illustrations of our theory of the clustering rate estimates.

We also prove that the very restrictive requirement - that the generating function should belong to the Wiener class - can be substantially relaxed. We show that for nonsymmetric Toeplitz coefficient matrices the clustering point exists for the singular values of the preconditined matrix.

We emphasize that our clusterization theorems can be extended to the case of multilevel Toeplitz matrices preconditioned by multilevel circulant preconditioners. The multilevel case is very important for industrial applications since in this case fast direct solution methods can not compete with the iterative solution methods.

A Remark on Band-Toeplitz Preconditions for Hermitian Toeplitz Systems

by

Seymour V. Parter**

Jaechil You*

Abstract: This note presents a modification of an idea of R.H. Chan and P.T.P. Tang. Let $f(0) \geq 0$ be a real valued, bounded, continuous function defined on $(-\pi, \pi)$. Let $T_n[f]$ be the Toeplitz matrix of order $n+1$ generated by $f(0)$. Chan and Tang suggest that for given $\ell \geq 1$ one chose $g_\ell(0) \geq 0$ as a real valued function of fixed degree ℓ which minimizes $\left\|\frac{(f-g_\ell)}{g_\ell}\right\|_\infty$. They construct $g(0)$ via the Remez algorithm. Then $T_n[g_\ell]^{-1}$ is used as the precondition for $T_n[f]$. We suggest that $g(0)$ be chosen as the even trigonometric polynomial of minimal degree which "matches" $f(0)$ at all points θ_j at which $f(\theta)$ assumes its minimum. Clearly this $g(\theta)$ is much easier to determine taht the $g_\ell(\theta)$. This choice is based on earlier work on the extreme eigenvalues of Hermitian Toeplitz matrices and the more recent work of Manteuffel and Parter on Preconditioning finite element discretizations of elliptic operators. It is shown that the condition number of $T_n[g]^{-1}\ T_N[F]$ is uniformly bounded for all n. Experimental results demonstrate the efficacy of these preconditioners.

** Department of Computer Science and Department of Mathematics, University of Wisconsin-Madison, Madison, WI 53706

* Department of Mathematics, University of Wisconsin-Madison, Madison, WI 53706

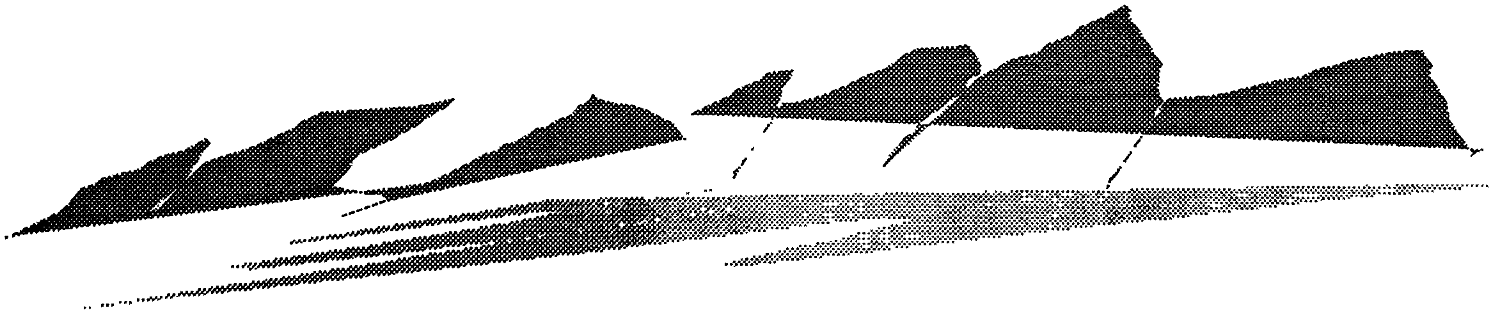
Saturday, April 9

Saddle Point Problems
Chair:
Room B

4:45 - 5:10 Andy Wathen
An Optimal Iterative Solver for the Stokes Problem

5:10 - 5:35 Bruno Welfert
On the Convergence of Inexact Uzawa Algorithms

5:35 - 6:00 Xiezhang Li
The Asymptotic Convergence Factor for a Polygon Under a Perturbation



An optimal iterative solver for the Stokes Problem

Andy Wathen, University of Bristol, UK
David Silvester, UMIST, UK

Discretisations of the classical Stokes Problem for slow viscous incompressible flow give rise to systems of equations of the form

$$\begin{pmatrix} A & B^t \\ B & -C \end{pmatrix} \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} f \\ O \end{pmatrix} \tag{1}$$

for the velocity u and the pressure p, where the coefficient matrix is symmetric but necessarily indefinite. The square submatrix A is symmetric and positive definite and represents a discrete (vector) Laplacian and C may be the zero matrix or more generally will be symmetric positive semi-definite.

For 'stabilised' discretisations ($C \neq 0$) and discretisations which are inherently 'stable' ($C = 0$) and so do not admit spurious pressure components even as the mesh size, h, approaches zero, the Schur compliment $-C - BA^{-1}B^T$ has spectral condition number independent of h (given also that B is bounded).

In this paper we will show how this property together with a multigrid preconditioner only for the Laplacian block A yields an optimal solver for the Stokes problem through use of the Minimum Residual iteration. That is, combining Minimum Residual iteration for (1) with a block preconditioner which comprises a small number of multigrid V-cycles for the Laplacian block A together with a simple diagonal scaling block provides an iterative solution procedure for which the computational work grows only linearly with the problem size.

We will give numerical results and some comparison with the more traditional approach of solving the a Schur compliment system by (Hestenes-Stiefel) Conjugate Gradients.

References.

1. Wathen, A.J. & Silvester, D.J., 1993, 'Fast iterative solution of stabilised Stokes systems Part I: Using simple diagonal preconditioners', SIAM J. Numer. Anal. **30(3)**, 630-649.

2. Silvester, D.J. & Wathen, A.J., 'Fast iterative solution of stabilised Stokes systems Part II: Using general block preconditioners', to appear in SIAM J. Numer Anal., September 1994.

On the convergence of inexact Uzawa algorithms

Bruno D. Welfert
Department of Mathematics
Arizona State University
Tempe, AZ 85287-1804

We consider the solution of symmetric indefinite systems of the type

$$\begin{pmatrix} A & B^t \\ B & -C \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix}, \tag{1}$$

where A and C are symmetric positive definite and semi-definite, respectively. Systems of the type (1) arise frequently in quadratic minimization problems, as well as mixed finite element discretizations of fluid flow equations.

The Uzawa algorithm is based on a (block) preconditioning of the system (1) by the matrix

$$M = \begin{pmatrix} A & \\ B & -\frac{1}{\alpha}I \end{pmatrix}$$

for some $\alpha > 0$, and is appropriate even when $C = 0$. The corresponding iteration converges if and only if

$$\alpha < \frac{2}{\rho(BA^{-1}B^t)}. \tag{2}$$

. An inexact version of this algorithm was presented in [1], where M is now of the type

$$M = \begin{pmatrix} M_{11} & \\ B & -\frac{1}{\alpha}I \end{pmatrix}.$$

It has been observed that the choice $M_{11} = A$ is not always optimal, i.e., does not always yield the smallest asymptotic convergence rate for the resulting algorithm.

In this presentation, we investigate the effect of M_{11} on the global convergence and give a few heuristics, confirmed by numerical experiments, for choosing M_{11}.

Alternate (symmetric) preconditionings will also be analyzed and compared numerically.

[1] H. C. Elman and G. H. Golub, *Inexact and Preconditioned Uzawa Algorithms for Saddle Point Problems*, Tech. Rep. CS-TR-3075, University of Maryland, College Park, May 1993.

The Asymptotic Convergence Factor for a Polygon under a Perturbation

XIEZHANG LI [1]

Abstract

Let

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

be a large system of linear equations, where $A \in \mathbb{C}^{N\times N}$, nonsingular and $\mathbf{b} \in \mathbb{C}^{N}$. A few iterative methods for solving (1) have recently been presented in the case where A is nonsymmetric. Many of their algorithms consist of two phases:

Phase I: estimate the extreme eigenvalues of A;

Phase II: construct and apply an iterative method based on the estimates.

For convenience, (1) is rewritten as an equivalent fixed-point form,

$$\mathbf{x} = T\mathbf{x} + \mathbf{c}. \tag{2}$$

Let Ω be a compact set excluding 1 in the complex plane, and let its complement in the extended complex plane be simply connected. The *asymptotic convergence factor* (ACF) for Ω, denoted by $\kappa(\Omega)$, measures the rate of convergence for the *asymptotically optimal semiiterative methods* for solving (2), where $\sigma(T) \subset \Omega$.

Suppose that Ω is a polygon whose vertices are the extreme eigenvalues of T. We consider how $\kappa(\Omega)$ changes if a vertex of Ω is perturbated. In this paper, a general variation formula for $\kappa(\Omega)$ is presented by the application of Hadamard's variation formula.

Let $\tilde{\Omega}$ be a polygon obtained as a result of a perturbation ϵe^{it} of a vertex of Ω, where ϵ is a small positive parameter and $0 \le t < 2\pi$. The ACF for $\tilde{\Omega}$ can be represented up to the first order of the perturbation,

$$\kappa(\tilde{\Omega}) = \kappa(\Omega)(1 + q\epsilon + o(\epsilon)), \tag{3}$$

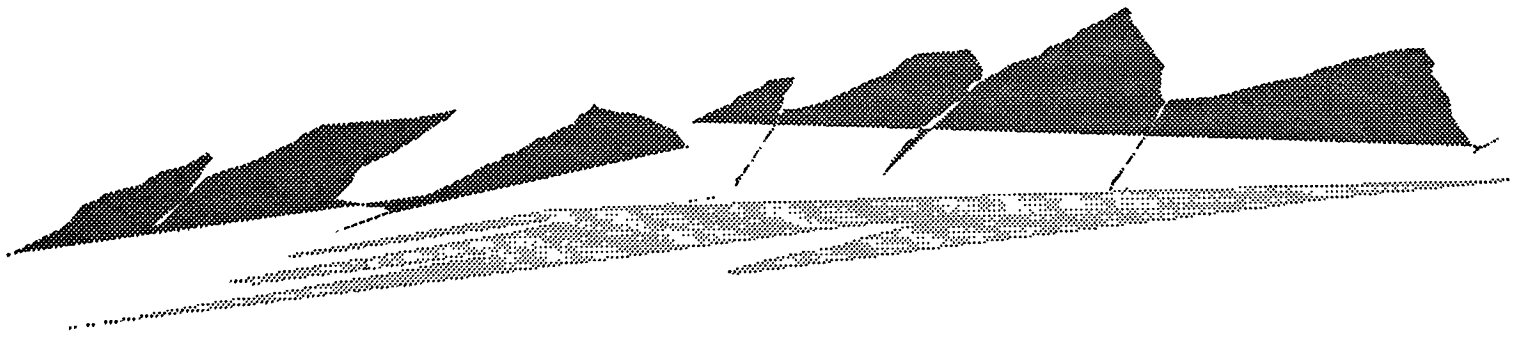
where the sensitivity q of the ACF to a perturbation is given by an integral independent of ϵ,

Some numerical experiments are given in this paper.

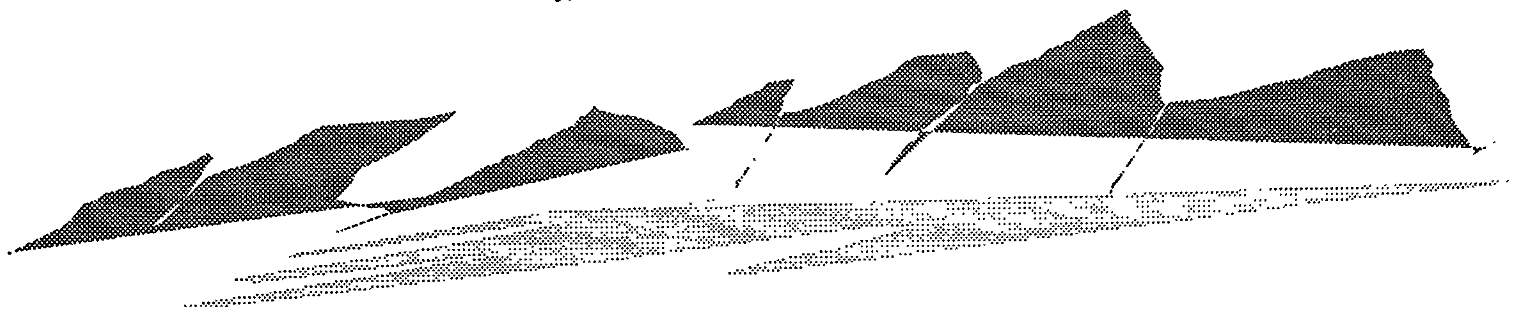
[1]Department of Mathematics & Computer Science, Georgia Southern University, Statesboro, GA 30460, U. S. A. (e-mail: xli@gsu.cs.gasou.edu)

INDEX

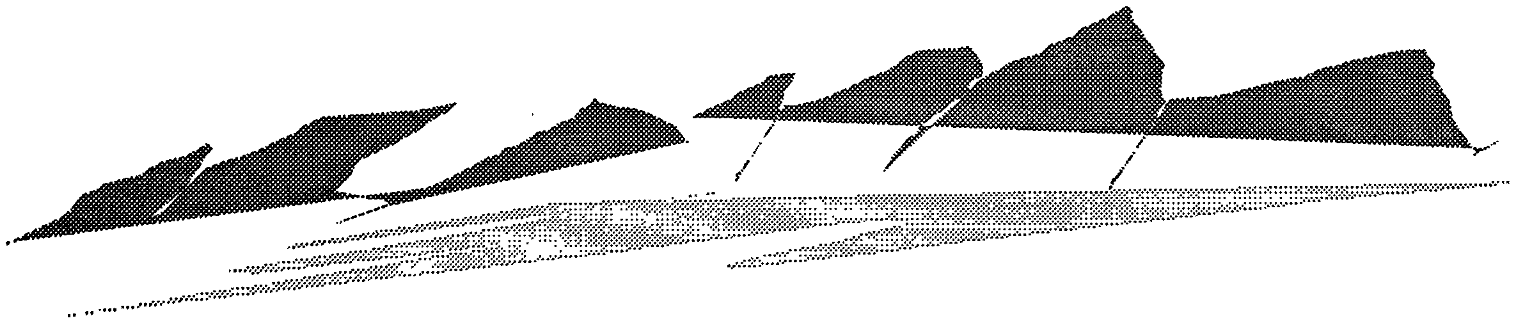
*Fernando Alvarado**	Block-Bordered Diagonalization and Parallel Iterative Solvers Saturday, 9:15 - 9:40, Room A
*P. Amodio**	Parallel Preconditioning for the Solution of Nonsymmetric Banded Linear Systems Saturday, 10:15 - 10:40, Room A
*Colin Aro**	Preconditioned Time-Difference Methods for Advection- Diffusion-Reaction Equations Friday, 5:10 - 5:35, Room B
*S.F. Ashby**	Modeling Groundwater Flow on Massively Parallel Computers Friday, 8:00 - 8:25, Room B
*Karen R. Baker**	Modeling the Diffusion of Phosphorus in Silicon in 3-D Friday, 11:05 - 11:30, Room B
*Teri Barth**	Variable Metric Conjugate Gradient Methods Thursday, 10:40 - 11:05, Room A
*A. Basermann**	Parallelizing Iterative Solvers for Sparse Systems of Equations and Eigenproblems on Distributed-Memory Machines Thursday, 10:40 - 11:05, Room B
J.R. Bates	A Nonlinear Multigrid Solver for a Semi-Lagrangian Potential Vorticity-Based Barotropic Model on the Sphere Friday, 5:35 - 6:00, Room A
*R. Bauer**	Preconditioned CG-Solvers and Finite Element Grids Friday, 10:40 - 11:05, Room B
*S.W. Bova**	Iterative Methods for Stationary Convection-Dominated Transport Problems Saturday, 8:00 - 8:25, Room B
*W.E. Boyse**	A Sparse Matrix Iterative Method for Efficiently Computing Multiple Simultaneous Solutions Wednesday, 5:10 - 5:35, Room A



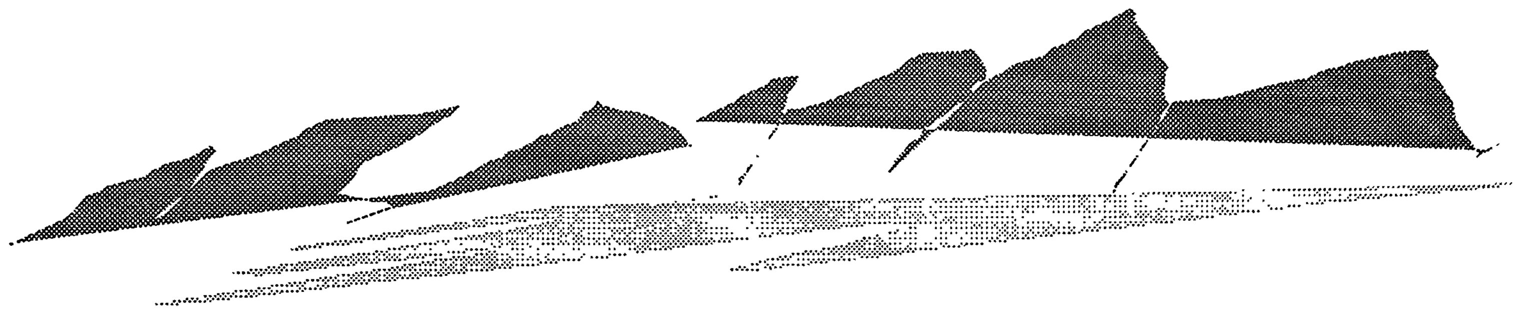
James H. Bramble	Preconditioning the Pressure Operator for the Time Dependent Stokes Problem Friday, 5:35 - 6:00, Room A
Randall Bramley	CIMGS: An Incomplete Orthogonal Factorization Preconditioner Saturday, 8:25 - 8:50. Room A
*Randall Bramley**	Solving Linear Inequalities in a Least Squares Sense Tuesday, 11:05 - 11:30, Room B
*Clemens W. Brand**	Preconditioned Iterations to Calculate Extreme Eigenvalues Tuesday, 4:45 - 5:10, Room B
A. Brandt	A Nonlinear Multigrid Solver for a Semi-Lagrangian Potential Vorticity-Based Barotropic Model on the Sphere Friday, 5:35 - 6:00, Room A
*Are Magnus Bruaset**	Object-Oriented Design of Preconditioned Iterative Methods Wednesday, 4:45 - 5:10, Room B
Monika ten Bruggencate	Block-Bordered Diagonalization and Parallel Iterative Solvers Saturday, 9:15 - 9:40, Room A
*Martin Bucker**	An Implementation of the TFQMR-Algorithm on a Distributed Memory Machine Thursday, 11:30 - 11:55, Room B
*Xiao-Chuan Cai**	Domain Decomposition Based Iterative Methods for Nonlinear Elliptic Finite Element Problems Tuesday, 10:15 - 10:40, Room A
G.F. Carey	PCG: A Software Package for the Iterative Solution of Linear Systems on Scalar, Vector & Parallel Computers Wednesday, 8:00 - 8:25, Room B
G.F. Carey	Maximizing Sparse Matrix Vector Product Performance in MIMD Computers Thursday, 8:50 - 9:15, Room B
G.F. Carey	On the Relationship Between ODE Solvers and Iterative Solvers for Linear Equations Thursday, 5:10 - 5:35, Room B
G.F. Carey	Iterative Methods for Stationary Convection-Dominated Transport Problems Saturday, 8:00 - 8:25, Room B
M.L.B. Carvalho	A Parallel Implementation of an EBE Solver for the Finite Element Method Thursday, 11:05 - 11:30, Room B



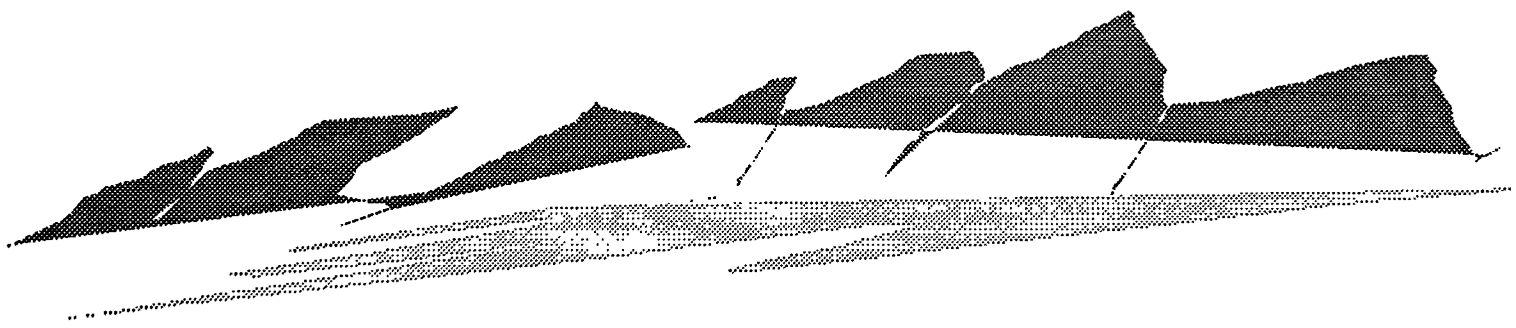
Tony F. Chan	Composite-Step Product Methods for Solving Nonsymmetric Linear Systems Thursday, 9:15 - 9:40, Room A
*Tony Chan**	Multigrid and Multilevel Domain Decomposition for Unstructured Grids Tuesday, 11:05 - 11:30, Room A
Jen Yuan Chen	MGMRES: A Generalization of GMRES for Solving Large Sparse Nonsymmetric Linear Systems Thursday, 11:30 - 11:55, Room A
*Edmond Chow**	Approximate Inverse Preconditioners for General Sparse Matrices Saturday, 8:00 - 8:25, Room A
*Christina Christara**	Schwarz and Multilevel Methods for Quadratic Spline Collocation Tuesday, 9:15 - 9:40, Room A
K.A. Cliffe	Two Grid Iteration With a Conjugate Gradient Fine Grid Smoother Applied to a Groundwater Flow Model Friday, 8:25 - 8:50, Room B
Paul Concus	A Modified Direct Preconditioner for Indefinite Symmetric Toeplitz Systems Saturday, 5:10 - 5:35, Room A
*Jane Cullum**	Peaks, Plateaus, Numerical Instabilities, and Achievable Accuracy in Galerkin and Norm Minimizing Procedures for Solving Ax=b Wednesday, 10:40 - 11:05, Room A
*Tom Cwik**	An Iterative Parallel Sparse Matrix Equation Solver with Application to Finite Element Modeling of Electromagnetic Scattering Friday, 9:15 - 9:40, Room B
Hasan Dag	Block-Bordered Diagonalization and Parallel Iterative Solvers Saturday, 9:15 - 9:40, Room A
*Tugral Dayar**	On the Effects of Using the GTH method in the Iterative Aggregation/Disaggregation Technique Wednesday, 5:35 - 6:00, Room A
*J.E. Dendy, Jr.**	Grandchild of the Frequency Decomposition Multigrid Methods Friday, 4:45 - 5:10, Room A
*Craig C. Douglas**	Constructive Interference II: Semi-Chaotic Multigrid Methods Friday, 8:25 - 8:50, Room A
*Tobin Driscoll**	Conformal Mapping and Convergence of Krylov Iterations Wednesday, 8:00 - 8:25, Room A



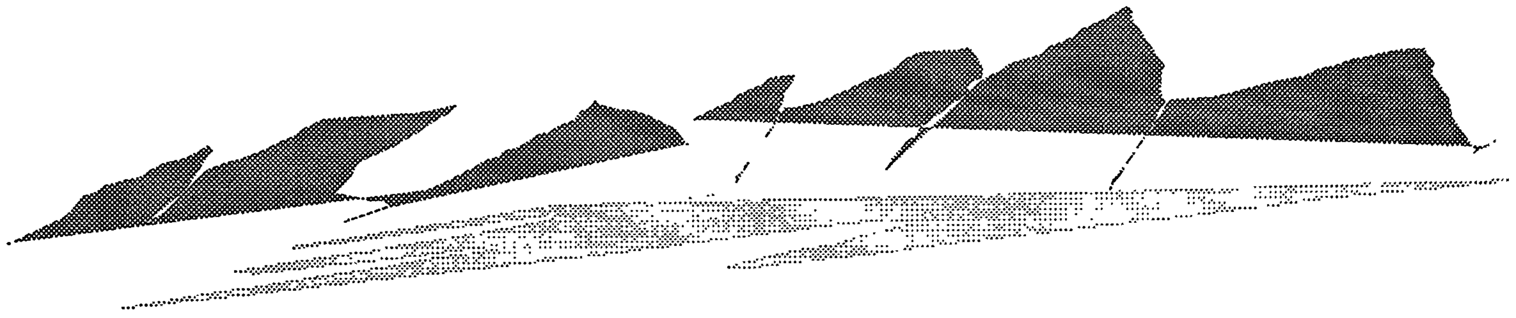
*Vladimir Druskin**	Explicit and Implicit ODE Solvers Using Krylov Subspace Optimization: Application to the Diffusion Equation andParabolic Maxwell's System Thursday, 4:45 - 5:10, Room B
H. Carter Edwards	VOILA-A Visual Object-Oriented Iterative Linear Algebra Problem Solving Environment Wednesday, 5:10 - 5:35, Room B
Stanley C. Eisenstat	Choosing the Forcing Terms in an Inexact Newton Method Tuesday, 8:50 - 9:15, Room B
Howard Elman	Fast Non-Symmetric Iterations and Efficient Preconditioning for Navier-Stokes Equations Friday, 6:00 - 6:25, Room B
Richard E. Ewing	Implementations of the Optimal Multigrid Algorithm for the Cell-Centered Finite Difference on Equilateral Triangular Grids Friday, 8:00 - 8:25, Room A
Vance Faber	Minimal Residual Method Stronger than Polynomial Preconditioning Wednesday, 10:15 - 10:40, Room A
R.D. Falgout	Modeling Groundwater Flow on Massively Parallel Computers Friday, 8:00 - 8:25, Room B
Charlotte F. Fischer	Overlapping Domain Decomposition Preconditioners for the Generalized Davidson Method for the Eigenvalue Problem Tuesday, 5:10 - 5:35, Room B
T.W. Fogwell	Modeling Groundwater Flow on Massively Parallel Computers Friday, 8:00 - 8:25, Room B
*Diederik Fokkema**	Generalized Conjugate Gradient Squared Thursday, 8:00 - 8:25, Room A
Roland Freund	A Look-Ahead Variant of TFQMR Thursday, 8:50 - 9:15, Room A
*Roland Freund**	Block Quasi-Minimal Residual Iterations for Non-Hermitian Linear Systems Thursday, 8:25 - 8:50, Room A
Kyle Gallivan	CIMGS: An Incomplete Orthogonal Factorization Preconditioner Saturday, 8:25 - 8:50. Room A
*E. Gallopoulos**	Matrix-Valued Polynomials in Lanczos Type Methods Thursday, 10:15 - 10:40, Room A



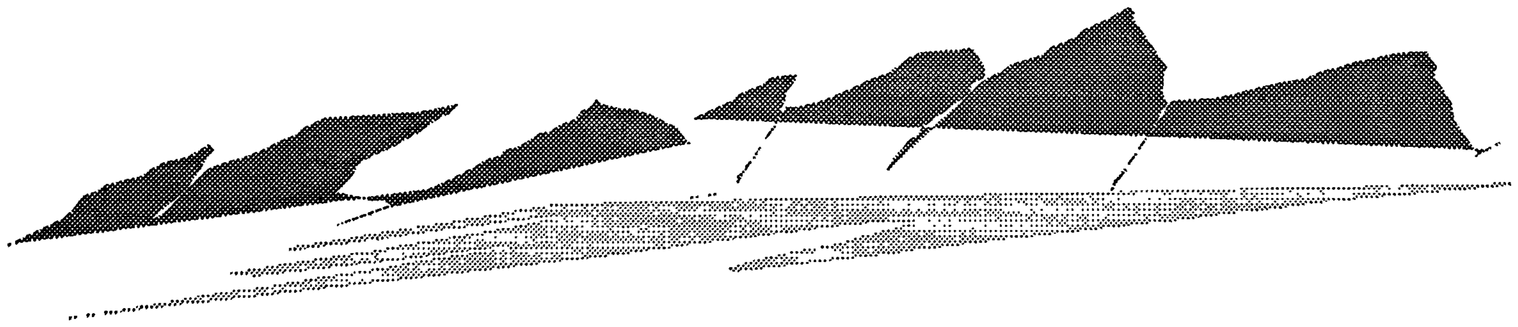
*Marc Garbey**	A Schwarz Alternating Procedure for Singular Perturbation Problems Tuesday, 8:50 - 9:15, Room A
Andre Garon	Iterative Solvers for Navier-Stokes Equations - Experiments with Turbulence Model Saturday, 10:15 - 10:40, Room B
R. Ghanem	Multilevel Adaptive Solution Procedure for Material Nonlinear Problems in Visual Programming Environment Wednesday, 5:35 - 6:00, Room B
*Eldar Giladi**	On the Interplay Between Inner and Outer Iterations for a Class of Iterative Methods Wednesday, 6:00 - 6:25, Room A
*Michael Griebel**	On the Relation Between Traditional Iterative Methods and Modern Multilevel/Domain Decomposition Methods Friday, 10:15 - 10:40, Room A
William Gropp	Portable, Parallel, Reusable Krylov Space Codes Wednesday, 9:15 - 9:40, Room B
*Karl Gustafson**	Computational Trigonometry Wednesday, 11:05 - 11:30, Room A
*D.Rh. Gwynllyw**	Preconditioned Iterative Methods for Unsteady Non- Newtonian Flow Between Eccentrically Rotating Cylinders Friday, 5:35 - 6:00, Room B
*M.J. Hagger**	Two Grid Iteration With a Conjugate Gradient Fine Grid Smoother Applied to a Groundwater Flow Model Friday, 8:25 - 8:50, Room B
*Thomas Hagstrom**	Experimental and Theoretical Studies of Iterative Methods for Nonlinear, Nonsymmetric Systems Arising in Combustion Friday, 4:45 - 5:10, Room B
*Linda Hayes**	VOILA-A Visual Object-Oriented Iterative Linear Algebra Problem Solving Environment Wednesday, 5:10 - 5:35, Room B
Linda Hayes	ITPACK Project: Past, Present, and Future Wednesday, 6:00 - 6:25, Room B



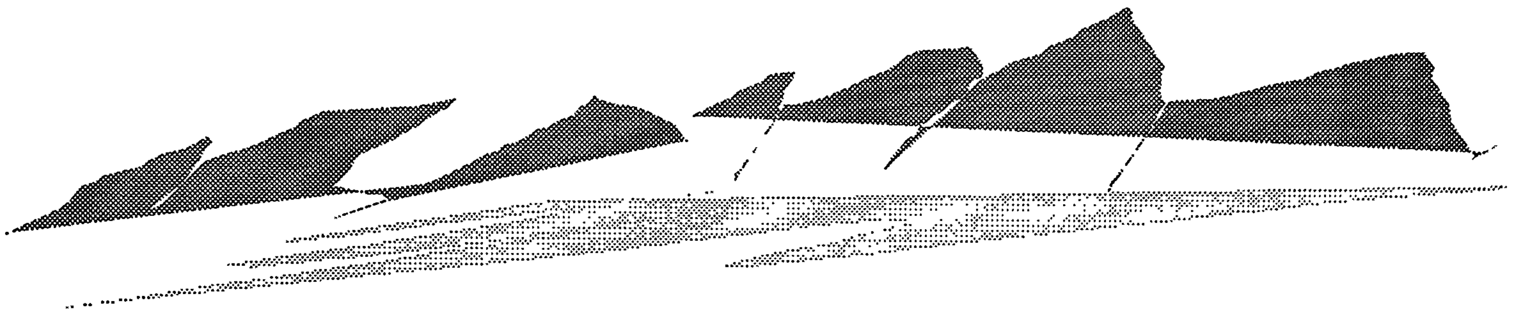
Qing He	Parallel Algorithms for Unconstrained Optimization by Multisplitting with Inexact Subspace Search - The Abstract Tuesday, 10:40 - 11:05, Room B
*Qing He**	Parallel Algorithms for Unconstrained Optimizations by Multisplitting Thursday, 4:45 - 5:10, Room A
*Matthias Heinkenschloss**	Numerical Solution of Control Problems Governed by Nonlinear Differential Equations Tuesday, 11:30 - 11:55, Room B
*Lina Hemmingsson**	Analysis of Semi-Toeplitz Preconditioners for First-Order PDEs Thursday, 5:10 - 5:35, Room A
*Van Henson**	On Multigrid Methods for Image Reconstruction from Projections Friday, 5:10 - 5:35, Room A
Mike Heroux	Advancements and Performance of Iterative Methods In Industrial Applications Codes on Cray Parallel/Vector Supercomputers Tuesday, 10:40 - 11:05, Room B
*Michael Heroux**	Performance Analysis of High Quality Parallel Preconditioners Applied to 3d Finite Element Structural Analysis Wednesday, 11:30 - 11:55, Room B
*S. Holmgren**	A Framework for the construction of preconditioners for systems of PDE Saturday, 11:05 - 11:30, Room A
*N.A. Hookey**	Simulation of Viscous Flows Using a Multigrid-Control Volume Finite Element Method Saturday, 9:15 - 9:40, Room B
Louis Howell*	A Multilevel Approximate Projection for Incompressible Flow Calculations Saturday, 8:50 - 9:15. Room B
Jie Hu	A Divide-and-Inner Product Parallel Algorithm for Polynomial Evaluation Thursday, 8:00 - 8:25, Room B
*Dan Hu**	Parallelizing Sylvester-Like Operations on a Distributed Memory Computer Thursday, 8:25 - 8:50, Room B
*Thomas Huckle**	Iterative Methods for Toeplitz-Like Matrices Saturday, 4:45 - 5:10, Room A



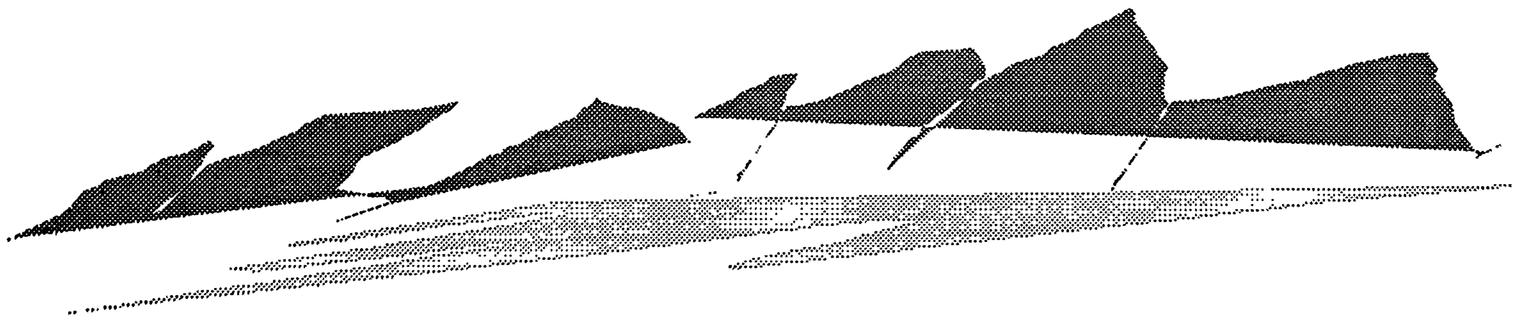
Scott A. Hutchinson	Parallel Performance of a Preconditioned CG Solver for Unstructured Finite Element Applications Thursday, 8:00 - 8:25, Room B
*Scott Hutchinson**	A Two-Level Parallel Direct Search Implementation for Arbitrarily Sized Objective Functions Tuesday, 10:15 - 10:40, Room B
*Masao Igarasi**	On the Convergence Processes of Newton-Raphson Iteration Methods Tuesday, 8:25 - 8:50, Room B
Vahraz Jamnejad	An Iterative Parallel Sparse Matrix Equation Solver with Application to Finite Element Modeling of Electromagnetic Scattering Friday, 9:15 - 9:40, Room B
*Jan Janssen**	Multigrid Waveform Relaxation on Spatial Finite Element Meshes Friday, 8:50 - 9:15, Room A
Wayne Joubert	Minimal Residual Method Stronger than Polynomial Preconditioning Wednesday, 10:15 - 10:40, Room A
Wayne Joubert	Some Comparison of Restarted GMRES and QMR for Linear and Nonlinear Problems Thursday, 11:05 - 11:30, Room A
Wayne Joubert	On the Relationship Between ODE Solvers and Iterative Solvers for Linear Equations Thursday, 5:10 - 5:35, Room B
*Wayne Joubert**	PCG: A Software Package for the Iterative Solution of Linear Systems on Scalar, Vector & Parallel Computers Wednesday, 8:00 - 8:25, Room B
*Michael Jung**	Implicit Extrapolation Methods for Multilevel Finite Element Computations Friday, 11:05 - 11:30, Room A
*M. Kamon**	Preconditioning Techniques for Constrained Vector Potential Integral Equations, with Application to 3-D Magnetoquasistatic Analysis of Electron Packages Saturday, 11:05 - 11:30, Room B
*Yimin Kang**	Convergence Analysis of Combinations of Different Methods Thursday, 6:00 - 6:25, Room B
Hans G.Kaper	A Schwarz Alternating Procedure for Singular Perturbation Problems Tuesday, 8:50 - 9:15, Room A



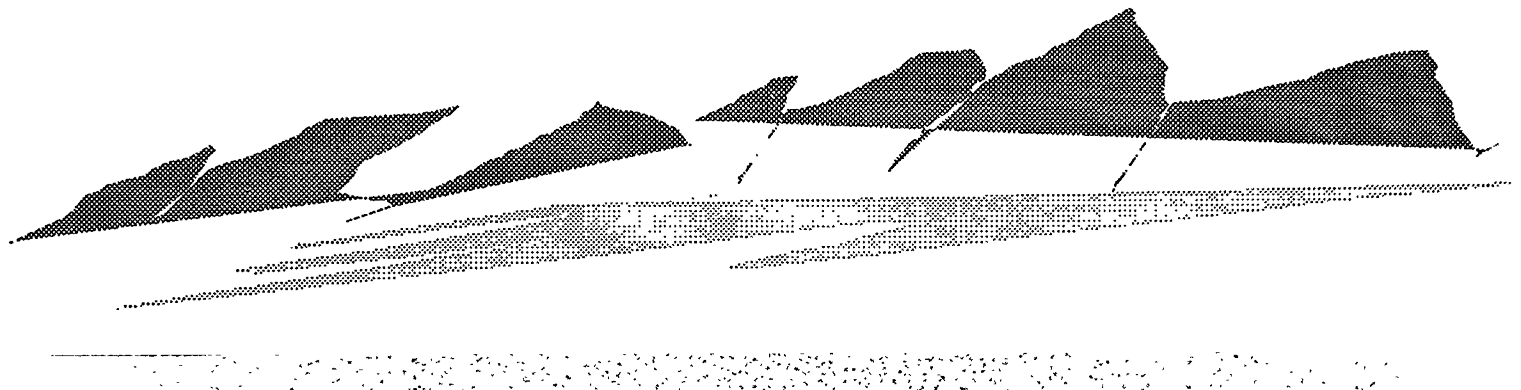
*C.T. Kelley**	GMRES and Integral Operators Tuesday, 5:35 - 6:00, Room A
*D. Kim**	Multilevel Adaptive Solution Procedure for Material Nonlinear Problems in Visual Programming Environment Wednesday, 5:35 - 6:00, Room B
*Seongjai Kim**	Parallel Iterative Procedures for Approximate Solutions of Wave Propagation by Finite Element and Finite Difference Methods Tuesday, 10:40 - 11:05, Room A
*David R. Kincaid**	ITPACK Project: Past, Present, and Future Wednesday, 6:00 - 6:25, Room B
*Emanuel Knill**	Minimal Residual Method Stronger than Polynomial Preconditioning Wednesday, 10:15 - 10:40, Room A
Leonid Knizhnerman	Explicit and Implicit ODE Solvers Using Krylov Subspace Optimization: Application to the Diffusion Equation andParabolic Maxwell's System Thursday, 4:45 - 5:10, Room B
*H.S. Kohli**	Maximizing Sparse Matrix Vector Product Performance in MIMD Computers Thursday, 8:50 - 9:15, Room B
Lilia Kolotilina	Performance Analysis of High Quality Parallel Preconditioners Applied to 3d Finite Element Structural Analysis Wednesday, 11:30 - 11:55, Room B
*L. Kolotilina**	Incomplete Block SSOR Preconditionings for High Order Discretizations Saturday, 8:50 - 9:15. Room A
T. Korsmeyer	Comparing Precorrected-FFT and Fast Multipole Algorithms for Solving Three-Dimensional Potential Integral Equations Tuesday, 4:45 - 5:10, Room A
E.B. Las Casas	A Parallel Implementation of an EBE Solver for the Finite Element Method Thursday, 11:05 - 11:30, Room B
Y. Li	A Nonlinear Multigrid Solver for a Semi-Lagrangian Potential Vorticity-Based Barotropic Model on the Sphere Friday, 5:35 - 6:00, Room A
*Lei Li**	A Divide-and-Inner Product Parallel Algorithm for Polynomial Evaluation Thursday, 8:00 - 8:25, Room B
*Xiezhang Li**	The Asymptotic Convergence Factor for a Polygon Under a Perturbation Saturday, 5:35 - 6:00, Room B



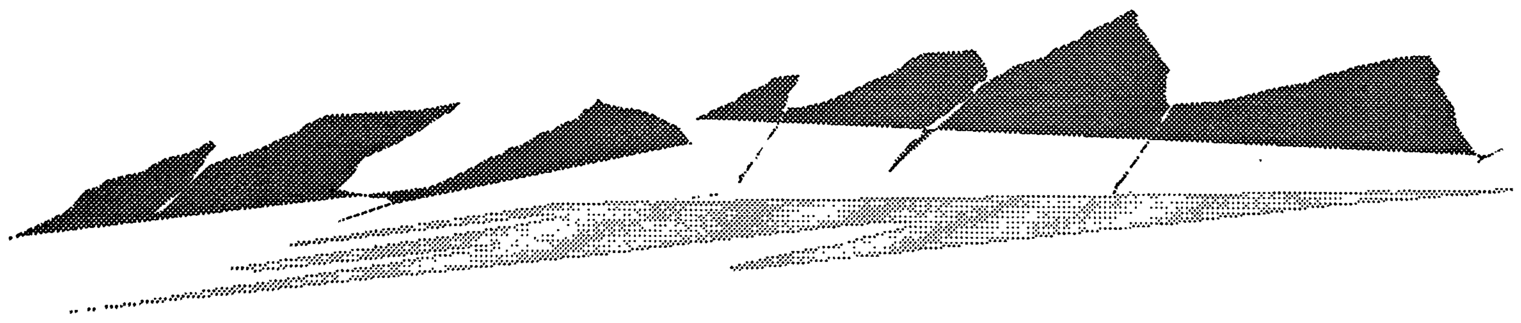
Mark Limber	On Multigrid Methods for Image Reconstruction from Projections	Friday, 5:10 - 5:35, Room A
Zhining Liu	Multigrid Mapping and Box Relaxtion for Simulation of the Whole Process of Flow Transition in 3-D Boundary Layers	Friday, 6:00 - 6:25, Room A
*Chaoqun Liu**	Multigrid Mapping and Box Relaxtion for Simulation of the Whole Process of Flow Transition in 3-D Boundary Layers	Friday, 6:00 - 6:25, Room A
*A. Lorber**	On the Relationship Between ODE Solvers and Iterative Solvers for Linear Equations	Thursday, 5:10 - 5:35, Room B
*Andrew Lumsdaine**	Krylov-Subspace Acceleration of Time Periodic Waveform Relaxation	Thursday, 5:35 - 6:00, Room B
Thomas Manteuffel	Minimal Residual Method Stronger than Polynomial Preconditioning	Wednesday, 10:15 - 10:40, Room A
F. Mazzia	Parallel Preconditioning for the Solution of Nonsymmetric Banded Linear Systems	Saturday, 10:15 - 10:40, Room A
Steve McCormick	A Nonlinear Multigrid Solver for a Semi-Lagrangian Potential Vorticity-Based Barotropic Model on the Sphere	Friday, 5:35 - 6:00, Room A
*Steve McCormick**	Multilevel First-Order System Least Squares for PDE'S	Friday, 11:30 - 11:55, Room A
*Steven M. McKay**	The Use of the Spectral Method within the Fast Adaptive Composite Gid Method	Tuesday, 11:30 - 11:55, Room A
R.T. McLay	Maximizing Sparse Matrix Vector Product Performance in MIMD Computers	Thursday, 8:50 - 9:15, Room B
Bruce McMillin	Adapting Implicit Methods to Parallel Processors	Thursday, 10:15 - 10:40, Room B
*A.J. Meir**	Velocity-Vorticity Formulation of Three-Dimensional, Steady, Viscous, Incompressible Flows	Saturday, 8:25 - 8:50. Room B



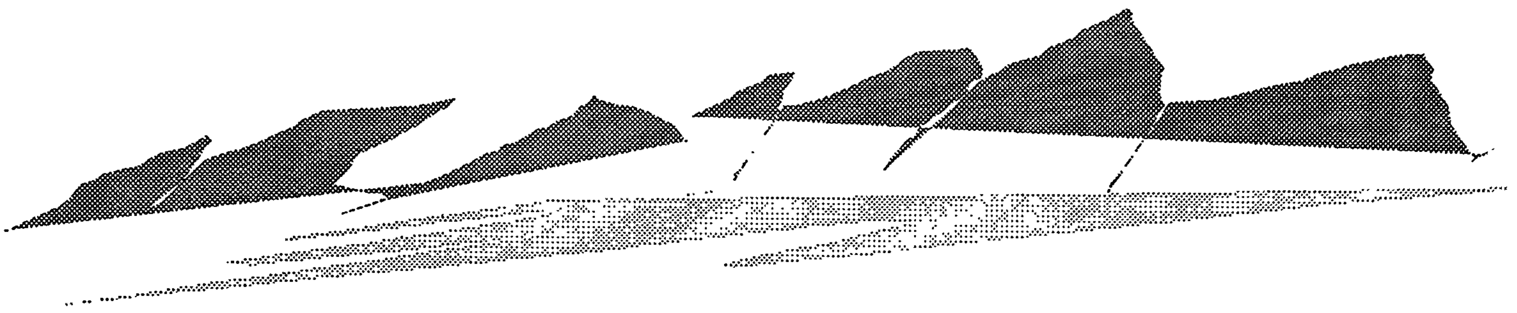
*N. J. Meyers**	An Iterative Method for the Solution of Linear Systems Using the Faber Polynomials for Annular Sectors Wednesday, 8:50 - 9:15, Room A
Juan C. Meza	A Multigrid Preconditioner for the Semiconductor Equations Friday, 10:15 - 10:40, Room B
Harry K. Moffat	A Two-Level Parallel Direct Search Implementation for Arbitrarily Sized Objective Functions Tuesday, 10:15 - 10:40, Room B
Harry K. Moffat	Parallel Performance of a Preconditioned CG Solver for Unstructured Finite Element Applications Thursday, 8:00 - 8:25, Room B
*Ron Morgan**	Some Comparison of Restarted GMRES and QMR for Linear and Nonlinear Problems Thursday, 11:05 - 11:30, Room A
*Noel M. Nachtigal**	A Look-Ahead Variant of TFQMR Thursday, 8:50 - 9:15, Room A
Tadao Nakamura	A Divide-and-Inner Product Parallel Algorithm for Polynomial Evaluation Thursday, 8:00 - 8:25, Room B
*Olavi Nevanlinna**	Convergence of Arnoldi Method Wednesday, 11:30 - 11:55, Room A
Kwong T. Ng	A Two-Level Parallel Direct Search Implementation for Arbitrarily Sized Objective Functions Tuesday, 10:15 - 10:40, Room B
Andy Nikishin	Performance Analysis of High Quality Parallel Preconditioners Applied to 3d Finite Element Structural Analysis Wednesday, 11:30 - 11:55, Room B
M.E. Oman	The Numerical Solution of Total Variation Minimization Problems in Image Processing Tuesday, 5:10 - 5:35, Room A
K. Otto	A Framework for the construction of preconditioners for systems of PDE Saturday, 11:05 - 11:30, Room A
*Maryse Page**	Iterative Solvers for Navier-Stokes Equations - Experiments with Turbulence Model Saturday, 10:15 - 10:40, Room B



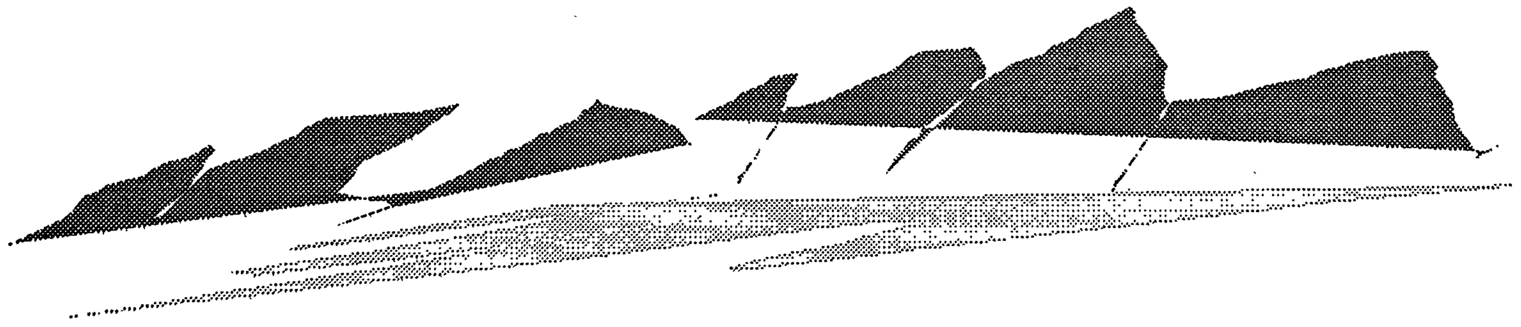
*Victor Pan**	New Algorithms for the Symmetric Tridiagonal Eigenvalue Computation Tuesday, 5:35 - 6:00, Room B
*Seymour Parter**	A Remark on Band-Toeplitz Preconditions for HermitianToeplitz Systems Saturday, 6:00 - 6:25, Room A
*J.E. Pasciak**	Preconditioning the Pressure Operator for the Time Dependent Stokes Problem Saturday, 10:40 - 11:05, Room A
*Michael Pernice**	Domain Decomposed Preconditioners with Krylov Subspace Methods as Subdomain Solvers Tuesday, 8:00 - 8:25, Room A
Svetozara Petrova	Preconditioned Iterations to Calculate Extreme Eigenvalues
J.R. Phillips	Comparing Precorrected-FFT and Fast Multipole Algorithms for Solving Three-Dimensional Potential Integral Equations Tuesday, 4:45 - 5:10, Room A
*J.R. Phillips**	Preconditioning Techniques for Constrained Vector Potential Integral Equations, with Application to 3-D Magnetoquasistatic Analysis of Electron Packages Saturday, 11:05 - 11:30, Room B
Tim Phillips	Preconditioned Iterative Methods for Unsteady Non- Newtonian Flow Between Eccentrically Rotating Cylinders Friday, 5:35 - 6:00, Room B
*Claude Pommerell**	Migration of Vectorized Iterative Solvers to Distributed Memory Architectures Wednesday, 8:25 - 8:50, Room B
*Gene Poole**	Advancements and Performance of Iterative Methods In Industrial Applications Codes on Cray Parallel/Vector Supercomputers Wednesday, 10:40 - 11:05, Room B
Krishnan Radhakrishnan	Experimental and Theoretical Studies of Iterative Methods for Nonlinear, Nonsymmetric Systems Arising in Combustion
*Jussi Rahola**	Solution of Dense Systems of Linear Equations in Electromagnetic Scattering Calculations Friday, 8:50 - 9:15, Room B
*Larry Reeves**	Adapting Implicit Methods to Parallel Processors Thursday, 10:15 - 10:40, Room B



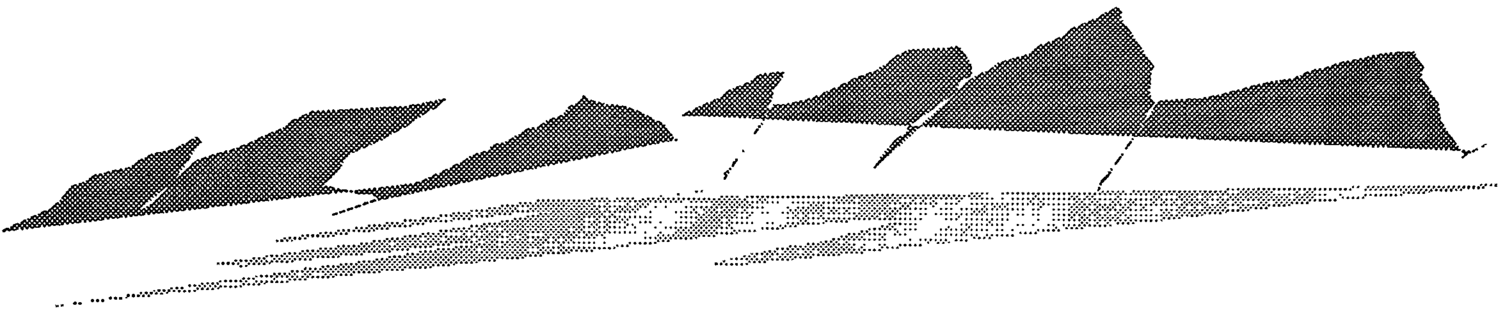
*Rosemary Renaut**	Parallel Algorithms for Unconstrained Optimization by Multisplitting with Inexact Subspace Search - The Abstract Tuesday, 10:40 - 11:05, Room B
*Shu-Mei C. Richman**	A Component Analysis Based On Serial Results for Analyzing Performance of Parallel Iterative Programs Wednesday, 11:05 - 11:30, Room B
Bruce T. Robinson	On Multigrid Methods for Image Reconstruction from Projections Friday, 5:10 - 5:35, Room A
Garry Rodrigue	Preconditioned Time-Difference Methods for Advection- Diffusion-Reaction Equations Friday, 5:10 - 5:35, Room B
Ulrich Rude	Implicit Extrapolation Methods for Multilevel Finite Element Computations Friday, 11:05 - 11:30, Room A
*John Ruge**	A Nonlinear Multigrid Solver for a Semi-Lagrangian Potential Vorticity-Based Barotropic Model on the Sphere Friday, 5:35 - 6:00, Room A
Roland Ruhl	Migration of Vectorized Iterative Solvers to Distributed Memory Architectures Wednesday, 8:25 - 8:50, Room B
Youcef Saad	Overlapping Domain Decomposition Preconditioners for the Generalized Davidson Method for the Eigenvalue Problem Tuesday, 5:10 - 5:35, Room B
Youcef Saad	Approximate Inverse Preconditioners for General Sparse Matrices Saturday, 8:00 - 8:25, Room A
*Youcef Saad**	P_SPARSLIB: A Parallel Sparse Iterative Solution Package Wednesday, 8:50 - 9:15, Room B
Ove Saevareid	Implementations of the Optimal Multigrid Algorithm for the Cell-Centered Finite Difference on Equilateral Triangular Grids Friday, 8:00 - 8:25, Room A
*Paul Saylor**	A Modified Direct Preconditioner for Indefinite Symmetric Toeplitz Systems Saturday, 5:10 - 5:35, Room A
A.A. Seidl	A Sparse Matrix Iterative Method for Efficiently Computing Multiple Simultaneous Solutions Wednesday, 5:10 - 5:35, Room A



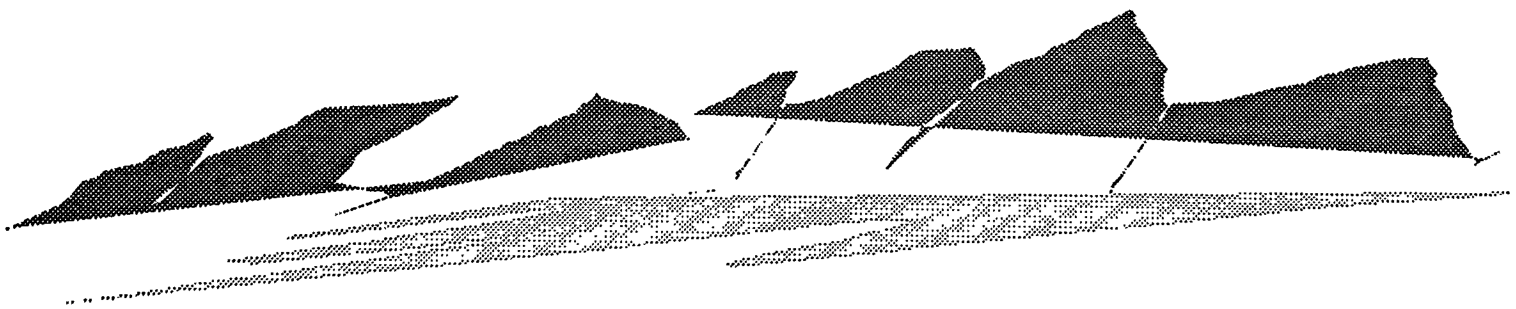
S. Selberherr	Preconditioned CG-Solvers and Finite Element Grids Friday, 10:40 - 11:05, Room B
John N. Shadid	A Two-Level Parallel Direct Search Implementation for Arbitrarily Sized Objective Functions Tuesday, 10:15 - 10:40, Room B
*John Shadid**	Parallel Performance of a Preconditioned CG Solver for Unstructured Finite Element Applications Thursday, 9:15 - 9:40, Room B
Qasim Sheikh	Performance Analysis of High Quality Parallel Preconditioners Applied to 3d Finite Element Structural Analysis Wednesday, 11:30 - 11:55, Room B
*Jian Shen**	Implementations of the Optimal Multigrid Algorithm for the Cell-Centered Finite Difference on Equilateral Triangular Grids Friday, 8:00 - 8:25, Room A
*R.P. Silva**	A Parallel Implementation of an EBE Solver for the Finite Element Method Thursday, 11:05 - 11:30, Room B
David Silvester	An Optimal Iterative Solver for the Stokes Problem Saturday, 4:45 - 5:10, Room B
*David Silvester**	Fast Non-Symmetric Iterations and Efficient Preconditioning for Navier-Stokes Equations Friday, 6:00 - 6:25, Room B
V. Simoncini	Matrix-Valued Polynomials in Lanczos Type Methods Thursday, 10:15 - 10:40, Room A
Gerard L.G. Sleijpen	Generalized Conjugate Gradient Squared Thursday, 8:00 - 8:25, Room A
Barry Smith	Schwarz and Multilevel Methods for Quadratic Spline Collocation Tuesday, 9:15 - 9:40, Room A
Barry Smith	Multigrid and Multilevel Domain Decomposition for Unstructured Grids Tuesday, 11:05 - 11:30, Room A
*Barry Smith**	Portable, Parallel, Reusable Krylov Space Codes Wednesday, 9:15 - 9:40, Room B
Danny C. Sorensen	Parallelizing Sylvester-Like Operations on a Distributed Memory Computer Thursday, 8:25 - 8:50, Room B



A. Spence	Two Grid Iteration With a Conjugate Gradient Fine Grid Smoother Applied to a Groundwater Flow Model Friday, 8:25 - 8:50, Room B
*Gerhard Starke**	Subspace Orthogonalization for Substructuring Preconditioners for Nonsymmetric Systems of Linear Equations Wednesday, 9:15 - 9:40, Room A
*Andreas Stathopoulos**	Overlapping Domain Decomposition Preconditioners for the Generalized Davidson Method for the Eigenvalue Problem Tuesday, 5:10 - 5:35, Room B
William J. Stewart	On the Effects of Using the GTH method in the Iterative Aggregation/Disaggregation Technique Thursday, 8:25 - 8:50, Room B
S.L. Swift	Maximizing Sparse Matrix Vector Product Performance in MIMD Computers Thursday, 8:50 - 9:15, Room B
*Tedd Szeto**	Composite-Step Product Methods for Solving Nonsymmetric Linear Systems Thursday, 9:15 - 9:40, Room A
Shlomo Ta'asan	Multilevel Turbulence Simulations Saturday, 10:40 - 11:05, Room B
*Johannes Tausch**	Equivalent Preconditioners for Boundary Element Methods Thursday, 5:35 - 6:00, Room A
C.C. Tazartes	Grandchild of the Frequency Decomposition Multigrid Methods Friday, 4:45 - 5:10, Room A
*Kim-Chuan Toh**	Convergence Estimates for Iterative Methods Via the Kreiss Matrix Theorem on a General Complex Domain Wednesday, 8:25 - 8:50, Room A
A.F.B. Tompson	Modeling Groundwater Flow on Massively Parallel Computers Friday, 8:00 - 8:25, Room B
C.S. Tong	Domain Decomposition Methods for Solving an Image Problem Tuesday, 8:25 - 8:50, Room A
Lloyd N. Trefethen	Conformal Mapping and Convergence of Krylov Iterations Wednesday, 8:00 - 8:25, Room A
Lloyd N. Trefethen	Convergence Estimates for Iterative Methods Via the Kreiss Matrix Theorem on a General Complex Domain Wednesday, 8:25 - 8:50, Room A



*Anne E. Trefethen**	The Conjugate Gradient NAS Parallel Benchmark on the IBM SP1 Wednesday, 10:15 - 10:40, Room B
*W.K. Tsui **	Domain Decomposition Methods for Solving an Image Problem Tuesday, 8:25 - 8:50, Room A
*Ray S. Tuminaro**	A Multigrid Preconditioner for the Semiconductor Equations Friday, 10:15 - 10:40, Room B
*Eugene E. Tyrtyshnikov**	Circulant Preconditioners with Unbounded Inverses: Why Non-Optimal Preconditioners may Possess a Better Quality than Optimal Ones Saturday, 5:35 - 6:00, Room A
*Eli Tziperman**	Multilevel Turbulence Simulations Saturday, 10:40 - 11:05, Room B
Stefan Vandewalle	Multigrid Waveform Relaxation on Spatial Finite Element Meshes Friday, 8:50 - 9:15, Room A
*Stefan Vandewalle**	Time-Parallel~Iterative Methods for Parabolic PDEs: Multigrid Waveform Relaxation and Time-Parallel Multigrid Friday, 9:15 - 9:40, Room A
Curt Vogel	Iterative Methods for Distributed Parameter Estimation in Parabolic PDE Tuesday, 6:00 - 6:25, Room A
*Curt Vogel**	The Numerical Solution of Total Variation Minimization Problems in Image Processing Tuesday, 5:10 - 5:35, Room A
*Eugene L. Wachspress**	Recent ADI Iteration Analysis and Results Wednesday, 4:45 - 5:10, Room A
*J.G. Wade**	Iterative Methods for Distributed Parameter Estimation in Parabolic PDE Tuesday, 6:00 - 6:25, Room A
*Homer F. Walker**	Choosing the Forcing Terms in an Inexact Newton Method Tuesday, 8:50 - 9:15, Room B
*Xiaoge Wang**	CIMGS: An Incomplete Orthogonal Factorization Preconditioner Saturday, 8:25 - 8:50. Room A
*Andy Wathen**	An Optimal Iterative Solver for the Stokes Problem Saturday, 4:45 - 5:10, Room B



*Bruno Welfert**	On the Convergence of Inexact Uzawa Algorithms Saturday, 5:10 - 5:35, Room B
*J. White**	Comparing Precorrected-FFT and Fast Multipole Algorithms for Solving Three-Dimensional Potential Integral Equations Tuesday, 4:45 - 5:10, Room A
Beata Winnicka	Solving Linear Inequalities in a Least Squares Sense Tuesday, 11:05 - 11:30, Room B
Donald Wolitzer	Preconditioned Time-Difference Methods for Advection- Diffusion-Reaction Equations Friday, 5:10 - 5:35, Room B
Z.Q. Xue	GMRES and Integral Operators Tuesday, 5:35 - 6:00, Room A
Irad Yavneh	Multilevel Turbulence Simulations Saturday, 10:40 - 11:05, Room B
*Irad Yavneh**	Multigrid with Red Black SOR Revisited Friday, 10:40 - 11:05, Room A
Alex Yeremin	Performance Analysis of High Quality Parallel Preconditioners Applied to 3d Finite Element Structural Analysis Wednesday, 11:30 - 11:55, Room B
David M. Young	ITPACK Project: Past, Present, and Future Wednesday, 6:00 - 6:25, Room B
*David Young**	MGMRES: A Generalization of GMRES for Solving Large Sparse Nonsymmetric Linear Systems Thursday, 11:30 - 11:55, Room A
Tong Zhang	The Conjugate Gradient NAS Parallel Benchmark on the IBM SP1 Wednesday, 10:15 - 10:40, Room B
Cinzia Zuffada	An Iterative Parallel Sparse Matrix Equation Solver with Application to Finite Element Modeling of Electromagnetic Scattering Friday, 9:15 - 9:40, Room B