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

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

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

MASTER

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

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!

Tuesday, April 5

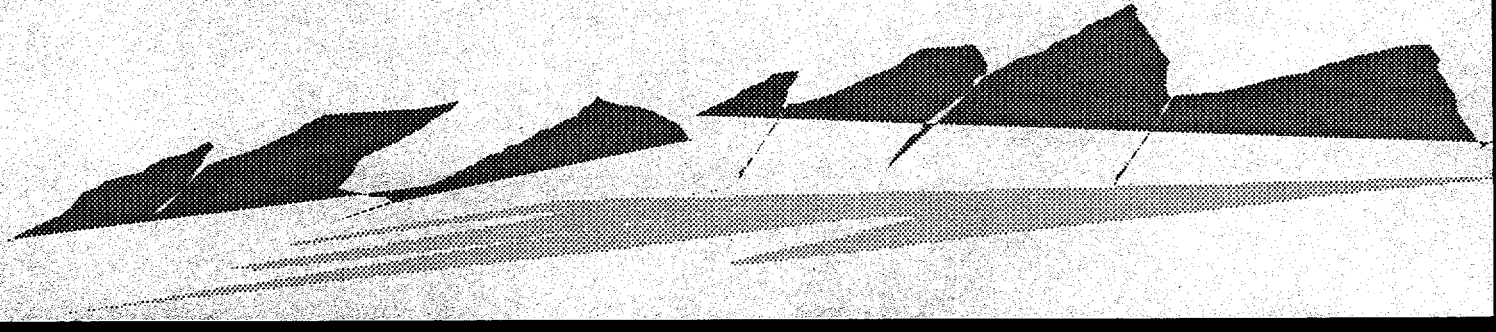
Domain Decomposition Methods I
Chair: Seymour Parter
Room A

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



Domain Decomposed Preconditioners with Krylov Subspace Methods as Subdomain Solvers

Michael Pernice
Utah Supercomputing Institute
University of Utah
Salt Lake City, Utah 84112
usimap@sneffels.usi.utah.edu

Extended Abstract

Domain decomposed preconditioners for nonsymmetric partial differential equations typically require the solution of problems on the subdomains. Most implementations employ exact solvers to obtain these solutions. Consequently work and storage requirements for the subdomain problems grow rapidly with the size of the subdomain problems. Subdomain solves constitute the single largest computational cost of a domain decomposed preconditioner, and improving the efficiency of this phase of the computation will have a significant impact on the performance of the overall method.

The small local memory available on the nodes of most message-passing multicomputers motivates consideration of the use of an iterative method for solving subdomain problems. For large-scale systems of equations that are derived from three-dimensional problems, memory considerations alone may dictate the need for using iterative methods for the subdomain problems [7]. In addition to reduced storage requirements, use of an iterative solver on the subdomains allows flexibility in specifying the accuracy of the subdomain solutions. Substantial savings in solution time is possible if the quality of the domain decomposed preconditioner is not degraded too much by relaxing the accuracy of the subdomain solutions.

While some work in this direction has been conducted for symmetric problems [1, 3, 7], similar studies for nonsymmetric problems appear not to have been pursued. This work represents a first step in this direction, and explores the effectiveness of performing subdomain solves using several transpose-free Krylov subspace methods, in particular GMRES [6], transpose-free QMR [2], CGS [8], and a smoothed version of CGS [9]. Depending on the difficulty of the subdomain problem and the convergence tolerance used, a reduction in solution time is possible in addition to the reduced memory requirements. The domain decomposed preconditioner is a Schur complement method in which the interface operators are approximated using interface probing. However, the results apply to overlapping Schwarz methods as well.

Subdomain solves are carried out until a prespecified accuracy is obtained. As a result the outer iterative method must accomodate variable preconditioning. For this reason FGMRES [5] is used as the base iterative method. While a strategy which

uses a fixed number of iterations on the subdomains may be employed, this would rule out methods that initially increase or fail to reduce the residual, such as CGS and quasi-minimal smoothed methods. It turns out that the low cost per iteration of these methods makes them desirable in many circumstances.

Numerical experiments conducted with model convection-diffusion problems show that the effectiveness of the domain-decomposed preconditioner (as measured by the number of iterations to satisfy a fixed convergence criterion) is not reduced as the accuracy of the subdomain solves decreases to around 10^{-4}. However as this accuracy is further reduced, some deterioration in the overall convergence rate is seen; in many instances this reduced effectiveness is offset by reduced work on the subdomains. However, very low accuracy of the subdomain solves, around 10^{-1}, has a pronounced negative effect on the domain decomposed preconditioners (except in the cases where GMRES is used), and has even been observed to lead to convergence failure in some cases. These observations have been seen to hold in both constant- and variable-convection problems.

This work has been submitted for inclusion in the Proceedings of the Seventh International Conference on Domain Decomposition Methods in Scientific Computing. A complete report [4] is available on request.

REFERENCES

[1] C. Börgers, *The Neumann-Dirichlet domain decomposition method with inexact solvers on the subdomains*, Numer. Math., 55 (1989), pp. 123–136.

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

[3] A. Meyer, *A parallel preconditioned conjugate gradient method using domain decomposition and inexact solvers on each domain*, Computing, 45 (1990), pp. 217–234.

[4] M. Pernice, *Domain decomposed preconditioners wuth Krylov subspace methods as subdomain solvers*, Tech. Report 45, Utah Supercomputing Institute, 1993.

[5] Y. Saad, *A flexible inner-outer preconditioned GMRES algorithm*, SIAM J. Sci. Comput., 14 (1993), pp. 461–469.

[6] Y. Saad and M. H. Schultz, *GMRES: A generalized minimum residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856–869.

[7] B. F. Smith, *A parallel implementation of an iterative substructuring algorithm for problems in three dimensions*, SIAM J. Sci. Comput., 14 (1993), pp. 406–423.

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

[9] L. Zhou and H. F. Walker, *Residual smoothing techniques for iterative methods*, Tech. Report 58, Department of Mathematics and Statistics, Utah State University, May 1992.

Domain decomposition methods for solving an image problem

W.K. Tsui and C.S. Tong
Department of Mathematics
Hong Kong Baptist College

14 Dec 1993

Abstract

The domain decomposition method is a technique to break up a problem so that ensuing sub-problems can be solved on a parallel computer. In order to improve the convergence rate of the capacitance systems, preconditioned conjugate gradient methods are commonly used. In the last decade, most of the efficient preconditioners are based on elliptic partial differential equations which are particularly useful for solving elliptic partial differential equations. In this paper, we apply the so called covering preconditioner, which is based on the information of the operator under investigation. Therefore, it is good for varies kinds of applications, specifically, we shall apply the preconditioned domain decomposition method for solving an image restoration problem. The image restoration problem is to extract an original image which has been degraded by a known convolution process and additive Gaussian noise.

A Schwarz Alternating Procedure for Singular Perturbation Problems

Authors: Marc Garbey,
LAN, Universit\'(e) Claude Bernard Lyon I, 69622
Villeurbanne Cedex, France
e-mail: mgarbey@lanl.univ-lyon1.fr

Hans G. Kaper
MCS Division, Argonne National Laboratory
Argonne, IL 60439
e-mail: kaper@mcs.anl.gov

We show that the Schwarz alternating procedure offers a good algorithm for the numerical solution of singular perturbation problems, provided the domain decomposition is properly designed to resolve the boundary and transition layers. We give sharp estimates for the optimal position of the domain boundaries and present convergence rates of the algorithm for various second-order singular perturbation problems. The splitting of the operator is domain-dependent, and the iterative solution of each subproblem is based on a modified asymptotic expansion of the operator. We show that this asymptotic-induced method leads to a family of efficient massively parallel algorithms and report on implementation results for a turning-point problem and a combustion problem,

Schwarz and Multilevel Methods for Quadratic Spline Collocation

Christina C. Christara
Department of Computer Science
University of Toronto
Toronto, Ontario M5S 1A4
CANADA
ccc@cs.toronto.edu

Barry Smith
Department of Mathematics
University of California, Los Angeles
Los Angeles, CA 90024-1555
U.S.A.
bsmith@math.ucla.edu

Smooth spline collocation methods offer an alternative to Galerkin finite element methods, as well as to Hermite spline collocation methods, for the solution of linear elliptic Partial Differential Equations (PDEs).

Recently, optimal order of convergence spline collocation methods have been developed for certain degree splines. Convergence proofs for smooth spline collocation methods are generally more difficult than for Galerkin finite elements or Hermite spline collocation, and they require stronger assumptions and more restrictions. However, numerical tests indicate that spline collocation methods are applicable to a wider class of problems, than the analysis requires, and are very competitive to finite element methods, with respect to efficiency.

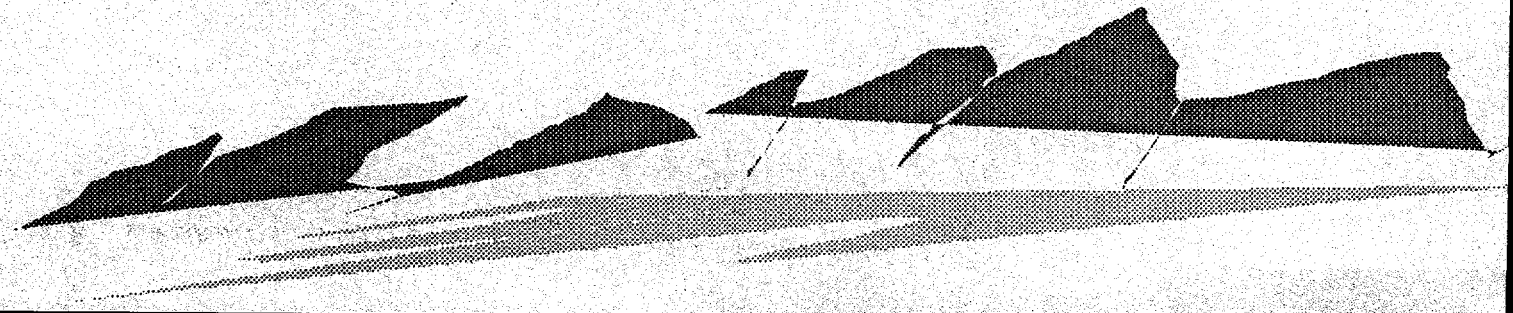
We will discuss Schwarz and multilevel methods for the solution of elliptic PDEs using quadratic spline collocation, and compare these with domain decomposition methods using substructuring. Numerical tests on a variety of parallel machines will also be presented. In addition, preliminary convergence analysis using Schwarz and/or maximum principle techniques will be presented.

Tuesday, April 5

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



On the Convergence Processes of Newton-Raphson Iteration Methods

Masao Igarasi
Nihon University
1866 Kameino Fujisawa Kanagawa
Japan

Abstract not available

Title: Choosing the forcing terms in an inexact Newton method

Authors:

Stanley C. Eisenstat
Department of Computer Science
Yale University
New Haven, CT 06520

Homer F. Walker (speaker)
Mathematics and Statistics Department
Utah State University
Logan, UT 84322-3900

Abstract: An *inexact Newton method* is a generalization of Newton's method for solving $F(x) = 0$, $F : \mathbb{R}^n \to \mathbb{R}^n$, in which each step reduces the norm of the local linear model of F. At the kth iteration, the norm reduction is usefully expressed by the *inexact Newton condition*

$$\|F(x_k) + F'(x_k)\, s_k\| \le \eta_k \|F(x_k)\|, \qquad \eta_k \in [0, 1),$$

where x_k is the current approximate solution and s_k is the step. In many applications, an η_k is first specified, and then an s_k is found for which the inexact Newton condition holds. Thus η_k is often called a "forcing term". In practice, the choice of the forcing terms is usually critical to the efficiency of the method and can affect robustness as well. Here, we outline several promising choices, discuss theoretical support for them, and compare their performance in a Newton iterative (truncated Newton) method applied to several large-scale problems.

Tuesday, April 5

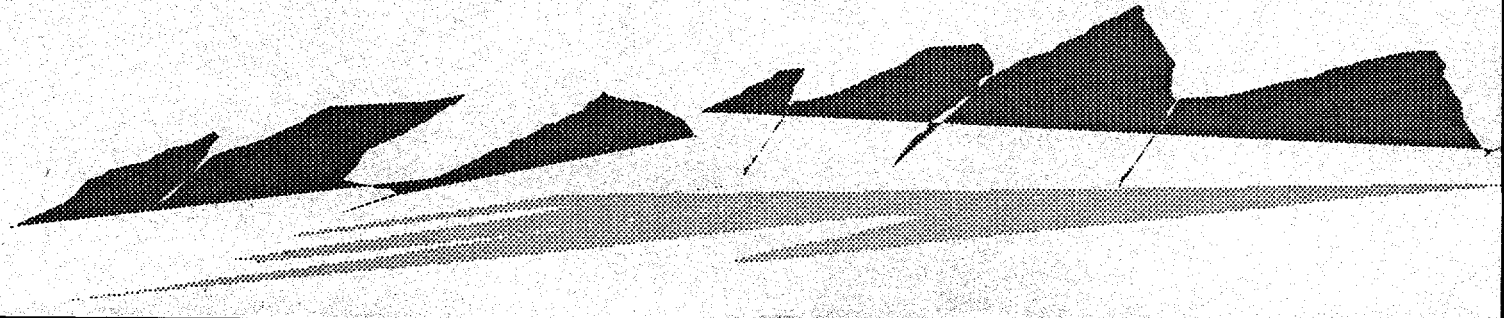
Domain Decomposition Methods II
Chair: Seymour Parter
Room A

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



Domain Decomposition Based Iterative Methods for Nonlinear Elliptic Finite Element Problems

Xiao-Chuan Cai*

The class of overlapping Schwarz algorithms has been extensively studied for linear elliptic finite element problems. In this presentation, we consider the solution of systems of nonlinear algebraic equations arise from the finite element discretization of some nonlinear elliptic equations. Several overlapping Schwarz algorithms, including the additive and multiplicative versions, with inexact Newton acceleration will be discussed. We show that the convergence rate of the Newton's method is independent of the mesh size used in the finite element discretization, and also independent of the number of subdomains into which the original domain in decomposed. Numerical examples will be presented.

*Department of Computer Science, University of Colorado, Boulder, CO 80309. *email:* cai@cs.colorado.edu.

Parallel Iterative Procedures for Approximate Solutions of Wave Propagation by Finite Element and Finite Difference Methods

Seongjai Kim*

Abstract. Parallelel iterative procedures based on domain decomposition techniques are defined and analyzed for the numerical solution of wave propagation by finite element and finite difference methods. For finite element methods, in a Lagrangian framework, an efficient way for choosing the algorithm parameter as well as the algorithm convergence are indicated. Some heuristic arguments for finding the algorithm parameter for finite difference schemes are addressed. Numerical results are presented to indicate the effectiveness of the methods.

Key words. Domain decomposition method, Parallel iterative algorithm, Wave propagation, Robin interface boundary condition.

1. Introduction.

Wave propagation in real media shows the effects of attenuation and dispersion. Therefore a realistic simulation of wave propagation should be able to reproduce these two effects. Wave equations are often served by a suitable radiation condition at infinity. Such problems can be solved numerically by first truncating the given unbounded domain, imposing a suitable outgoing radiation condition on the (artificial) boundary of the truncated bounded domain and then solving the resulting problem using discretization methods.

Let $\Omega \subset \mathbb{R}^d$, $d \leq 3$, be a bounded domain with a Lipschitz boundary $\Gamma = \partial\Omega$. Consider the following wave problem

$$
\begin{aligned}
-\Delta u - a(x)^2 u + iq(x)^2 u &= f(x), \quad x \in \Omega, \\
\frac{\partial u}{\partial \nu} + i\alpha(x) u &= 0, \qquad x \in \Gamma,
\end{aligned}
\tag{1.1}
$$

where i is the imaginary unit and the coefficients $a(x)$, $q(x)$ and $\alpha(x)$ satisfy

$$
\begin{aligned}
&0 < a_0 \leq a(x) \leq a_1 < \infty, \\
&0 \leq q_0 \leq q(x) \leq q_1 < \infty, \\
&\alpha = \alpha_r - i\alpha_i, \quad \alpha_r > 0, \ \alpha_i \geq 0,
\end{aligned}
$$

and are sufficiently regular that the existence and unuqueness of a solution of (1.1) lying in $H^s(\Omega)$ for some $s \geq 1$ for reasonable f are assured. The coefficient α is properly chosen such that the second equation of (1.1) represents a first–order outgoing radiation condition

*Department of Mathematics, Purdue University, W. Lafayette, IN 47907

that allows normally incident waves to pass out of Ω transparently. The problem (1.1) describes the propagation of time–harmonic waves such as, e.g., electromagnetic waves in (conducting) media, discretizations of the time–dependent Schrödinger equations by implicit difference schemes, inverse scattering problems, seismic waves, and underwater acoustics.

The wave problem (1.1) seems to be difficult to solve. In addition to having a complex–valued solution, the problem (1.1) is neither Hermitian symmetric nor coersive; as a consequence, most standard iterative methods either fail to converge or converge so slowly as to be impractical. The purposes here are to define massively parallelizable domain decomposition iterative procedures and indicate efficient automatic strategies for choosing iteration parameters.

Concerning the iterative numerical solvers, we refer to Bayliss, Goldstein and Turkel and Freund [1, 7] for the conjugate gradient–type algorithms, and Douglas, Hensley and Roberts [4] for an ADI algorithm. A strip–type domain decomposition method by finite differences for the problem (1.1) in a rectangular domain is constructed by the author [8].

An outline of the paper is as follows. In §2 a domain decomposition is considered using the Robin interface condition. In §3 a finite element procedure is illustrated and the corresponding parallel iterative procedure is defined. A convergence result and an efficient strategy for finding the algorithm parameter are presented in the section. In §4, an automatic (but heuristic) strategy for finding efficient algorithm parameters for finite difference solutions of the problem is presented. The section 5 reports some numerical results to show the effectiveness of the algorithms. The last section indicates the conclusions and possible applications.

2. Domain decomposition method.

Let $\{\Omega_j,\ j = 1, \ldots, M\}$ be a partition of Ω:

$$\overline{\Omega} = \cup_{j=1}^{M} \overline{\Omega}_j; \quad \Omega_j \cap \Omega_k = \emptyset, \quad j \neq k.$$

Assume that $\partial\Omega_j$, $j = 1, \ldots, M$, is also Lipschitz and that Ω_j is star–shaped. In practice, with the exception of perhaps a few Ω_j's along Γ, each Ω_j would be convex with a piecewise–smooth boundary. Let

$$\Gamma_j = \Gamma \cap \partial\Omega_j, \quad \Gamma_{jk} = \Gamma_{kj} = \partial\Omega_j \cap \partial\Omega_k, \quad \Sigma = \cup_{j,k=1}^{M} \Gamma_{jk}.$$

The weak formulation of the problem (1.1) is given by seeking $u \in V = H^1(\Omega)$ such that

$$(\nabla u, \nabla v)_\Omega - ((a^2 - iq^2)u, v)_\Omega + \langle i\alpha u, v\rangle_\Gamma = (f, v)_\Omega, \ v \in V, \tag{2.1}$$

Let us consider the decomposition of the problem (1.1) over $\{\Omega_j\}$. The problem (1.1) is equivalent to the following: Find u_j, $j = 1, \ldots, M$, such that

$$-\Delta u_j - a(x)^2 u_j + iq(x)^2 u_j = f(x), \quad x \in \Omega_j, \tag{2.2}$$

$$\frac{\partial u_j}{\partial \nu_j} + i\alpha(x) u_j = 0, \qquad x \in \Gamma_j, \tag{2.3}$$

$$\frac{\partial u_j}{\partial \nu_j} + i\beta\, u_j = -\frac{\partial u_k}{\partial \nu_k} + i\beta\, u_k, \qquad x \in \Gamma_{jk}, \tag{2.4}$$

where ν_j is the unit outward normal from Ω_j and β is a complex (normally chosen to be a constant) function on Σ with $\mathrm{Re}\beta > 0$. The consistency conditions are replaced by the Robin interface boundary condition (2.4). This replacement is often more convenient, see [11, 8].

Now, let $V_j = H^1(\Omega_j)$. Testing (2.2) against $v \in V_j$ and using (2.3)–(2.4), we obtain the weak formulation for (2.2)–(2.4) over the partition $\{\Omega_j\}$: find $u_j \in V_j$, $j = 1, \ldots, M$, such that

$$
\begin{aligned}
&(\nabla u_j, \nabla v)_{\Omega_j} - ((a^2 - iq^2)u_j, v)_{\Omega_j} + \sum_k \langle i\beta\, u_j, v\rangle_{\Gamma_{jk}} + \langle i\alpha u_j, v\rangle_{\Gamma_j} \\
&\qquad = \sum_k \langle -\frac{\partial u_k}{\partial \nu_k} + i\beta u_k, v\rangle_{\Gamma_{kj}} + (f, v)_{\Omega_j}, \quad v \in V_j.
\end{aligned} \tag{2.5}
$$

Basic idea of a domain decomposition iterative method is to localize the computations to smaller subdomain problems. It is feasible to localize to each Ω_j by evaluating the quantities in (2.2)–(2.4) [resp. (2.5)] related to Ω_j at the new iterate level and those in (2.2)–(2.4) [resp. (2.5)] related to neighboring subdomains Ω_k such that $\Gamma_{jk} \neq \emptyset$ at the old level.

3. The finite element method.

Let V^h be a finite element space of V with the regular triangulations [3] not crossing the interfaces Σ and of the maximum diameter h, and the local finite element spaces are defined by $V_j^h = \{v|_{\Omega_j} : v \in V^h\}$. Our analysis below shall include the case in which $\{\Omega_j\}$ is a partition of Ω into individual elements. The finite element approximation to (1.1) is given by restricting (2.1) to the space V^h; for the existence and uniqueness of the approximate solution and convergence properties of the method, we refer to [3, 6, 5].

The finite element domain decomposition iterative algorithm can be constructed as follows: given arbitrary initial guess $\{u_j^{h,0} \in V_j^h : j = 1, \ldots, M\}$, we build inductively the sequences $\{u_j^{h,n} \in V_j^h : j = 1, \ldots, M\}$, $n \geq 1$, by solving

$$
\begin{aligned}
&(\nabla u_j^{h,n}, \nabla v)_{\Omega_j} - ((a^2 - iq^2)u_j^{h,n}, v)_{\Omega_j} + \sum_k \langle i\beta\, u_j^{h,n}, v\rangle_{\Gamma_{jk}} + \langle i\alpha u_j^{h,n}, v\rangle_{\Gamma_j} \\
&\qquad = \sum_k \langle -\frac{\partial u_k^{h,n-1}}{\partial \nu_k} + i\beta u_k^{h,n-1}, v\rangle_{\Gamma_{kj}} + (f, v)_{\Omega_j}, \ v \in V_j^h.
\end{aligned} \tag{3.1}
$$

Here, when we consider the decomposition of the domain into the individual elements, the following observation is very critical; Since the method is trying to impose the continuity of both the displacement and the flux on each interface, there will be a flux conservation error, i.e., $\dfrac{\partial u_j^h}{\partial \nu_j} \neq -\dfrac{\partial u_k^h}{\partial \nu_k}$ unless the approximate solution $u^h \in V^h$ is a linear function over the domain Ω, a totally uninteresting case. So, let us introduce Lagrangian multipliers on the interfaces $\{\Gamma_{jk}\}$. Assume that, when $v_j^h \in V_j^h$, the normal component of its flux $\dfrac{\partial v_j^h}{\partial \nu_j}$ on Γ_{jk} is a polynomial of some fixed degree τ and that τ is independent of Γ_{jk}. Set

$$
\Lambda^h = \{\lambda : \lambda|_{\Gamma_{jk}} \in P_\tau(\Gamma_{jk}) = \Lambda_{jk}, \Gamma_{jk} \neq \emptyset\};
$$

note that there are two copies of P_τ assigned to the set Γ_{jk}: Λ_{jk} and Λ_{kj}. Consider the Lagrangian multiplier to be λ_{jk} as seen from Ω_j and λ_{kj} as seen from Ω_k. Modify (2.4)

to read

$$-\lambda_{jk} + i\beta u_j = \lambda_{kj} + i\beta u_k, \quad x \in \Gamma_{jk}. \tag{3.2}$$

Then, the parallel iterative process by the hybridized finite element method can be defined, by dropping the superscript h, as follows:

$$\text{Select } u_j^0 \in V_j, \ j = 1, \ldots, M, \text{ arbitrarily,} \tag{3.3}$$

then recursively compute the sequences $u_j^n \in V_j, \ n \geq 1$, by solving

$$\begin{aligned} &(\nabla u_j^n, \nabla v)_{\Omega_j} - ((a^2 - iq^2)u_j^n, v)_{\Omega_j} + \sum_k \langle i\beta u_j^n, v\rangle_{\Gamma_{jk}} + \langle i\alpha u_j^n, v\rangle_{\Gamma_j} \\ &\qquad = \sum_k \langle \lambda_{kj}^{n-1} + i\beta u_k^{n-1}, v\rangle_{\Gamma_{jk}} + (f, v)_{\Omega_j}, \quad v \in V_j, \end{aligned} \tag{3.4}$$

$$\lambda_{jk}^n = i\beta u_j^n - (\lambda_{kj}^{n-1} + i\beta u_k^{n-1}), \quad x \in \Gamma_{jk}. \tag{3.5}$$

Note that (3.4) is independent of λ_{jk}^n and determines u_j^n; then λ_{jk}^n is computed by (3.5). It should be noticed that the algorithm (3.3)–(3.5) is not requiring extra costs on computation, but, in fact, it is cheaper and leads to an easier numerical implementation.

We have the following theorem.

Theorem 3.1 ([9]). *Assume $q \geq q_0 > 0$ and $\beta > 0$. The iterates $\{u_j^n, \lambda_{jk}^n\} \in V_j \times \Lambda_{jk}$ in the algorithm (3.3)–(3.5) converge to the solution $\{u_j, \lambda_{jk}\}$ of the global hybridized finite element procedure in the following senses:*

$$\begin{aligned} &u_j^n \to u_j = u^*|_{\Omega_j} \ \text{ in } L^2(\Omega_j), \\ &\lambda_{jk}^n \ \text{ and } \ -\lambda_{kj}^n \to \lambda_{jk} = -\lambda_{kj} \ \text{ in } L^2(\Gamma_{jk}), \end{aligned}$$

where $u^* \in V^h$ is the solution of global finite element method. If $\rho(A)$ is the spectral radius of the iteration matrix A, then $\rho(A) < 1$. Thus, the iterative procedure (3.3)–(3.5) is convergent for every $\beta > 0$.

In order to present the rate of convergence in terms of the problem coefficients, the number of subdomains and the mesh size, we assume the partition $\{\Omega_j\}$ is quasiregular with the size H. Let $H = \mathcal{O}(h)$. This includes the decomposition of the domain into the individual elements. We quote again the following theorem without the proof.

Theorem 3.2 ([9]). *Assume that the positive constant parameter β in the iterative procedure (3.3)–(3.5) satisfies*

$$\beta = \sqrt{h^{-2} + h^2(a_1^4 + q_1^4)}\,. \tag{3.6}$$

Then, the spectral radius $\rho(A)$ is minimized and bounded as follows:

$$\rho(A) \leq 1 - \frac{q_0^2}{K(h^{-4} + a_1^4 + q_1^4)^{1/2}} \equiv \gamma_0, \tag{3.7}$$

and the iteration (3.3)–(3.5) converges with an error at the n–th iteration bounded asymptotically by $\mathcal{O}(\gamma_0^n)$.

Now, we consider two particular examples for (3.7). First, assume $q = \mathcal{O}(1)$. Then, by choosing the parameter $\beta = \mathcal{O}(h^{-1})$, it follows that the iteration (3.3)–(3.5) converges with rate bounded by

$$\gamma_0 = 1 - ch^2. \tag{3.8}$$

For our second example, we consider discretizations of time–dependent problems, e.g., Schrödinger equation, by implicit difference schemes, then $q^2 \equiv \frac{1}{\Delta t}$. Assume $\Delta t = \mathcal{O}(h^2)$. Then the choice $\beta = \mathcal{O}(h^{-1})$ leads to the estimate

$$\gamma_0 = 1 - c, \tag{3.9}$$

for some positive $c < 1$. This bound implies that only some fixed number of iterations, independent on both H and h, are required for each time step.

4. The finite difference method.

In this section we consider two–dimensional problems. When $q \equiv 0$, the standard iterative algorithms (relaxation and extrapolation schemes) fail to converge and conjugate gradient–type algorithms converge so slowly to be impractical.

Let Ω be a two–dimensional bounded domain which can be divided by retangular subdomains with the interfaces parallel to either coordinate axis. Let δ_x^2 denote the centered second order difference with respect to x, and ∂_ν, $\partial_{\mathfrak{f}}$ and $\partial_{\mathfrak{b}}$ be the centered, forward and backward differences,respectively, for the first order derivatives, in the direction of the outer normal (here, an exterior bordering of the domain is assumed). Let $\Delta_h = \delta_x^2 + \delta_y^2$. Then, one proper finite difference approximation to (2.2)–(2.4) for two–dimensional problems can be defined by

$$-\Delta_h u_j^n - a^2 u_j^n + iq^2 u_j^n = f, \qquad x \in \Omega_j, \tag{4.1}$$

$$\partial_\nu u_j^n + i\alpha u_j^n = 0, \qquad x \in \Gamma_j, \tag{4.2}$$

$$\partial_{\mathfrak{f}} u_j^n + i\beta u_j^n = -\partial_{\mathfrak{b}} u_k^{n-1} + i\beta u_k^{n-1}, \quad x \in \Gamma_{jk}. \tag{4.3}$$

Note that the Robin boundary condition is approximated by a combination of forward–backward differences. This combination is very necessary for both convergence and efficience, and the second–order approximation of the (centered) five point finite difference scheme would not be destroyed. For each subdomain Ω_j, only the subdomains sharing an edge as an iterior boundary are considered as the adjacent subdomains Ω_k. For some proper ordering of grid points, the matrix representation for the iterative algorithm (4.1)–(4.3) is of the form:

$$u^n = Q^{-1} R u^{n-1} + Q^{-1} f, \quad n = 1, 2, \cdots. \tag{4.4}$$

For the constant coefficient problems in rectangular domains (using a tensor product argument for the strip type domain decompositions), we can find the algorithm parameters β such that the algorithm converges [8]. In this section we present a heuristic, automatic method of finding efficient algorithm parameter β for general coefficient problems. Consider an L–shape domain and the domain decomposition depicted in Fig. 4.1 (i). There the bold lines denote the interfaces. We shall determine β, line by line, by using horizental or vertical mesh lines.

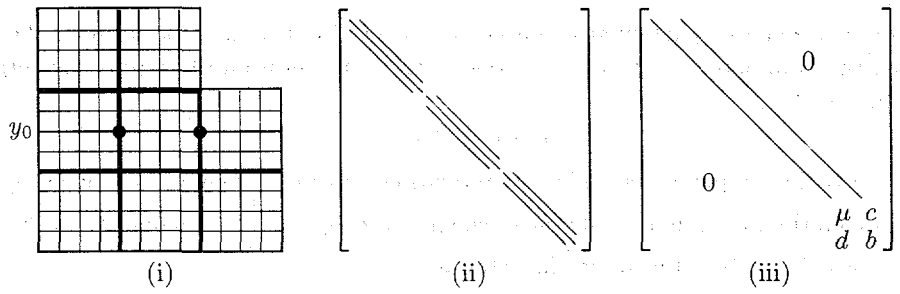


Fig. 4.1: (i). *A decomposition of the domain Ω with mesh lines.* (ii). *The tridiagonal system for the restricted one–dimensional problem on Ω^{y_0}.* (iii). *The matrix U of the LU–factorization performed up to the $(m-1)$–th row.*

Let us find the values of β on the dotted points along the line $y = y_0$. Ignoring the term u_{yy}, we restrict our problem to the one–dimensional subspace $\Omega^{y_0} := \{(x, y) \in \Omega : y = y_0\}$ decomposed into three subdomains with two dotted points being the interfaces. If the points in Ω^{y_0} are ordered from left to right, each subproblem can be solved by inverting a block diagonal matrix (three blocks), where each block is a tridiagonal matrix of dimension, say, m, see Fig. 4.1 (ii). First, consider the tridiagonal matrix corresponding to the left–end subdomain. When an LU–factorization is performed up to the $(m-1)$–th row, the matrix U is depicted in Fig. 4.1 (iii). This factorization is possible even the tridiagonal matrix has not been assembled on the interface point. We choose β satisfying

$$-\frac{c}{\mu} = 1 - i\beta h, \tag{4.5}$$

on the first interface point. By using this β, one can complete not only the last row of the matrix but also the first row of the next matrix corresponding to the mid–subdomain. Now, we consider the second tridiagonal matrix. After performing LU–factorization up to the $(m-1)$–th row, choose β for the second interface point as in the first case. For each mesh line having interface points, including both the horizontal and the vertical mesh lines, this searching can be continued. This procedure is readily extendable to the multiple decompositions of more general domains, clearly. We will refer this procedure to *ADOP* (alternating direction optimal procedure) in the remainder of this paper. It is not so difficult to check the following lemma.

Lemma 4.1. *Let G_1 be the iteration matrix of the reduced one–dimensional problem of* (4.1)–(4.3) *restricted on Ω^{y_0}, with the parameter β found by ADOP. Then the spectral radius of G_1 is zero, i.e., $\rho(G_1) = 0$.*

The above lemma implies the *ADOP* parameter β would be chosen in such a way that the spectral radii of the iteration matrices of the one–dimensional alternating direction problems are zero. In the next section, efficiency of *ADOP* will be numerically checked.

5. Numerical results.

This section reports on some experimental data to illustrate the effectiveness of our parallel algorithms. The program is implemented in FORTRAN77 with complex double precision, and is performed using an IBM RS/6000, serial machine. Let $\Omega = (0, 1)^2$. The

$M_x \times M_y$	2×2	4×4	8×8	64×64
n	9	21	22	81
ρ	0.714	0.866	0.873	0.963

Table 5.1: $1/h = 64$, $\omega = 50$, $c = c_1$ *and* $q = 60$.

$M_x \times M_y$	64×64	128×128	256×256
n	76	78	80
r_2^n	1.4e-1	3.0e-2	7.2e-3
ρ	0.975	0.956	0.940

Table 5.2: $1/h = M_x$, $\omega = 80$, $c = c_1$ *and* $q = h^{-1}$.

coefficients are chosen as follows:

$$a(x,y) = \frac{\omega}{c(x,y)}, \quad \alpha(x,y) = \omega, \quad q = \text{a nonnegative constant},$$

where $\omega > 0$ is the angular frequency and $c(x,y)$ is the wave speed. We choose typically four different functions for $c(x,y)$:

$$\begin{aligned} c_1(x,y) &\equiv 1, \\ c_2(x,y) &= e^{xy}(2 - \sin(3\pi x)), \\ c_3(x,y) &= (2 - \sin(2\pi x))(2 - \sin(4\pi y)), \\ c_4(x,y) &= 1 + 2x^3 + y. \end{aligned} \tag{5.1}$$

The source function f is selected such that the true solution $u(x,y) = \dfrac{\phi(x) \cdot \phi(y)}{\omega^2}$, where $\phi(x) = e^{i\omega(x-1)} + e^{-i\omega x} - 2$. Each subproblem is solved directly. The error is estimated on the relative L^p-norm r_p^n and the iteration is stopped when $s_\infty^n \leq 10^{-4}$, where

$$r_p^n = \frac{\|U^n - u\|_{L^p(\Omega)}}{\|u\|_{L^p(\Omega)}}, \quad s_\infty^n = \frac{\|U^n - U^{n-1}\|_{L^\infty(\Omega)}}{\|U^n\|_{L^\infty(\Omega)}},$$

with $p = 2$ for finite element solutions, $p = \infty$ for finite difference solutions and U^n is the approximate solution of the n–th iteration. The zero initial values, $U^0 \equiv 0$, are assumed.

Finite element parallel iterations. Here we report conputational results for finite element parallel iterations. The bilinear splines on quadrilateral elements are used as the basis functions. When we consider the decompositions of the domain into the individual elements ($H = h$), the cost of one domain decomposition iteration is approximately as four times as that of one classical Jcacobi iteration. We choose the wave speed $c(x,y) = 1$.

Table 5.1 contains the number of iterations n and the average rate of convergence ρ for various decompositions $M_x \times M_y$, when $1/h = 64$, $\omega = 50$ and $q = 60$. When $H = 8h \sim 16h$, the fastest solutions (in CPU–time) are obtained.

In Table 5.2, the iteration numbers, relative errors and the average convergence rates are presented, when $1/h = M_x$, $\omega = 80$ and $q = h^{-1}$: the domain is decomposed into individual elements. If we choose the time step $\Delta t = h^2$ for time–discretizations of the Schrödinger equation by the backward difference, as we expect in anlaysis, a fixed number of iterations are needed in each time step, see (3.9).

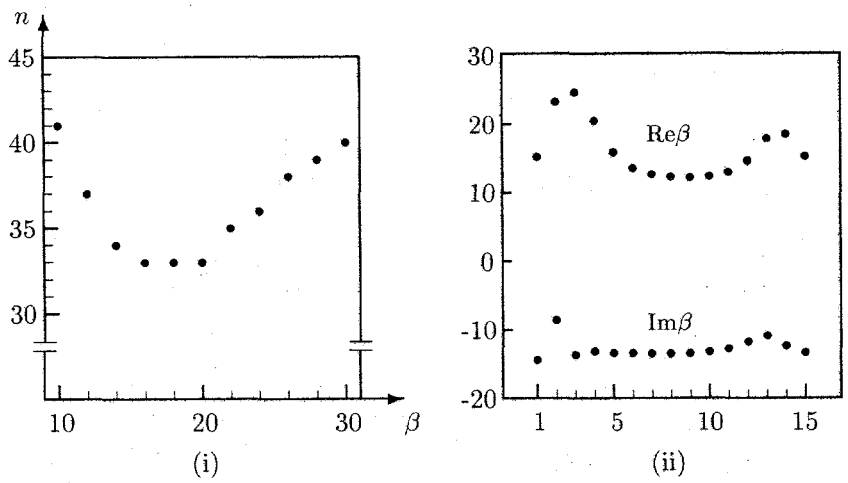


Fig. 5.1: *The box decomposition ($M_x \times M_y = 16 \times 16$); when $\omega = 30$, $c = c_2$, $q = 20$ and $1/h = 64$.* (i). *The constant parameters β vs. the iteration numbers n.* (ii). *The ADOP parameter β along the mesh line $y = 0.5$ (the values of β on the 15 interface points).*

When $q \equiv 0$, in general, we cannot get a numerical convergence unless the grid is sufficiently fine.

Finite difference iterative procedures. We report the numerical results for finite difference parallel iterations using the parameters found by *ADOP*. In the case of $q_0 > 0$, it is numerically checked that the parameter β obtained by *ADOP* introduces a faster convergence than any other constant parameters. When $q \equiv 0$, *ADOP* results in a good convergence in certain conditions, while constant parameters cannot be used unless c is a constant.

In Fig. 5.1 (i), the iteration numbers n with constant parameter β varing from 10 to 30 are reported, when $\omega = 30$, $c(x, y) = c_2(x, y)$, $q = 20$, $1/h = 64$ and the decomposition of the domain $M_x \times M_y = 16 \times 16$. When $16 \leq \beta \leq 20$, we have the fastest convergence, and it needs 33 iterations. For the *ADOP* parameter, the iteration converges after 28 iterations. The *ADOP* parameter along the mesh line $y = 0.5$ is depicted in Fig. 5.1 (ii).

Notice that the *ADOP* parameter, in general, cannot be a constant even for constant coefficient problems. When we consider the strip decomposition $M_x \times M_y = 16 \times 1$, the iteration converges after only 14 iterations, while 19 iterations is needed for the tolerence $s_\infty^n \leq 10^{-6}$.

Table 5.3 presents the numerical results, when $1/h = 128$, $\omega = 50$, $c(x, y) = c_4(x, y)$ and $q \equiv 0$. From a computational point of view, the case $q \equiv 0$ is very interesting: we do not know of any efficient iterative algorithm for such a case. The standard iterative methods (relaxations and extrapolations) and conjugate gradient–type algorithms either fail to converge or converge so slowly. When the domain is decomposed into a relatively large number of box subdomains, some error oscillations are observed. The following is numerically checked: If $\max(\frac{\omega}{c})h \leq \frac{1}{4}$ and $H \geq 4h$ for the case $q \equiv 0$, *ADOP* introduces a numerical convergence for strip type decompositions (in fact, for constant coefficient cases, theoretical convergence can be obtained).

Tables 5.4 and 5.5 show the results for the strip decompositions and box decomposi-

$M_x \times M_y$	8×1	16×1	32×1	64×1	128×1
n	45	109	192	359	609
ρ	0.930	0.971	0.983	0.991	0.995

Table 5.3: $1/h = 128$, $\omega = 50$, $c(x,y) = c_4(x,y)$ *and* $q \equiv 0$.

$M_x \times M_y$	8×1	16×1	32×1	64×1	128×1
n	6	6	9	15	26
ρ	0.495	0.495	0.626	0.755	0.850

Table 5.4: $1/h = 128$, $\omega = 60$, $c(x,y) = c_3(x,y)$ *and* $q = 75$.

$M_x \times M_y$	8×8	16×16	32×32	64×64	128×128
n	8	8	14	26	49
ρ	0.590	0.590	0.740	0.850	0.918

Table 5.5: $1/h = 128$, $\omega = 60$, $c(x,y) = c_3(x,y)$ *and* $q = 75$.

tions, respectively, when $1/h = 128$, $\omega = 60$, $c(x,y) = c_3(x,y)$ and $q = 75$. The error for the global one–domain finite difference solution is 1.48%. When M_x, $M_y \leq 32$, after first four iterations, the errors are reduced less than 2%. For all wave speeds in (5.1), we have experienced similar results.

6. Conclusions and applications.

Massively parallelizable iterative algorithms are defined and analyzed for finite element and finite difference approximates of wave propagation. We can find an efficient iteration parameter which minimizes the spectral radius of the iteration matrix for the finite element problems in conducting media. Such a well–selected parameter shall accelerate the convergence of the iteration. It is numerically checked that the Lagrangian multipliers are good approximations for the flux, which can be expected from (3.1) and (3.4). This information can be applicable, for instance, to adoptive procedures for mesh refinement.

For finite difference approximate solutions, using the parameters obtained by *ADOP*, the iteration converges fast for the problem in both conducting media and ideally non–conducting media. *ADOP* is tested using various coefficients and different spacings. It results in a faster convergence than any other constant parameters. In addition to being efficient, *ADOP* is an automatic procedure, and its cost is never expensive.

When an iterative (domain decomposition) algorithm is designed, iteration parameters are often introduced to accelerate the convergence of the iteration. For certain model problems, the parameters can be selected easily and effectively. However, when one considers the iterative algorithm for a realistic problem, the problem of choosing iteration parameters may not be so simple. Furthermore, for real–valued problems the iteration is more sensitive to the algorithm parameters than conplex–valued problems; For real–valued problems, the right–hand side of (4.5) would be $1 - \beta h$. So the rate of change of the value is larger than complex cases, when β is moving. It is hoped that *ADOP* could be an answer for the problem of choosing iteration parameters.

An variant of *ADOP* is being developed for the finite element approximations with

the bassis functions presented by a tensor product. The procedure *ADOP* is readily applicable to the other types of second order partial differential equations, in particular, to singularly perturbed problems such as time–dependent problems more efficiently [10].

Acknowledgement

The author thanks Professor Jim Douglas Jr. for his valuable comments and Dr. Neil N. Carlson for his encouragement.

References

[1] A. Bayliss, C. Goldstein and E. Turkel, *An iterative method for the Helmholtz equation*, J. Comput. Phys. 49 (1983) pp. 443–457.

[2] A. Bayliss, C. Goldstein and E. Turkel, *On accuracy conditions for the numerical computation of waves*, J. Comput. Phys. 59 (1985) pp. 396–404.

[3] P.G. Ciarlet, *Finite Element Methods for Elliptic Problems*, North–Holland, Amsterdam, 1978.

[4] J. Douglas, Jr., J. L. Hensley, and J. E. Roberts, *An alternating–direction iteration method for Helmholtz problems*, Technical Report #214, Mathematics Department, Purdue University, W. Lafayette, IN 47907, April 1993.

[5] J. Douglas, Jr., J. E. Santos and D. Sheen, *Approximation of scalar waves in the space–frequency domain*, Technical Report #217, Mathematics Department, Purdue University, W. Lafayette, IN 47907, May 1993.

[6] J. Douglas, Jr., J. E. Santos, D. Sheen and L. S. Bennethum, *Frequency domain treatment of one–dimensional scalar waves*, Mathematical Models and Methods in Applied Sciences 13 (1993), pp. 171–194.

[7] R. W. Freund, *Conjugate gradient–type methods for linear systems with complex symmetric coefficient matriices*, SIAM J. Sci. Stat. Comput. 13 (1992), pp. 425–448.

[8] S. Kim, *A parallelizable iterative procedure for the Helmholtz problem*, submitted for publication.

[9] S. Kim, *A domain decomposition method for finite element solutions of wave propagation*, preprint.

[10] S. Kim, *Parallel domain decomposition methods for second–order partial differential equations*, in preparation.

[11] P.L. Lions, *On the Schwarz alternating method III: a variant for nonoverlapping subdomains*, in Third International Symposium on Domain Decomposition Method for Partial Differential Equations, T.F. Chan, R. Glowinski, J. Periaux and O. B. Widlund (eds), SIAM, Philadelphia, 1990.

Department of Mathematics
Purdue University
W. Lafayette, IN 47907
E-mail: *skim@math.purdue.edu*

Multigrid and Multilevel Domain Decomposition for Unstructured Grids

Tony Chan
Department of Mathematics
405 Hilgard Ave.
Los Angeles, CA 90024-1555
chan@math.ucla.edu

Barry Smith
Department of Mathematics
405 Hilgard Ave.
Los Angeles, CA 90024-1555
bsmith@math.ucla.edu

Multigrid has proven itself to be a very versatile method for the iterative solution of linear and nonlinear systems of equations arising from the discretization of PDES. In some applications, however, no natural multilevel structure of grids is available, and these must be generated as part of the solution procedure.

In our presentation we will consider the problem of generating a multigrid algorithm when only a fine, unstructured grid is given. Our techniques generate a sequence of coarser grids by first forming an approximate maximal independent set of the vertices and then applying a Cavendish type algorithm to form the coarser triangulation. Numerical tests indicate that convergence using this approach can be as fast as standard multigrid on a structured mesh, at least in two dimensions.

The Use of the Spectral Method within the Fast Adaptive Composite Grid Method

by

Steven M. McKay
Brigham Young University

The use of efficient algorithms for the solution of partial differential equations has been sought for many years. The fast adaptive composite grid (FAC) method combines an efficient algorithm with high accuracy to obtain low cost solutions to partial differential equations. The FAC method achieves fast solution by combining solutions on different grids with varying discretizations and using multigrid like techniques to find fast solution. Recently, the continuous FAC (CFAC) method has been developed which utilizes an analytic solution within a subdomain to iterate to a solution of the problem. This has been shown to achieve excellent results when the analytic solution can be found. The CFAC method will be extended to allow solvers which construct a function for the solution, e.g., spectral and finite element methods. In this discussion, the spectral methods will be used to provide a fast, accurate solution to the partial differential equation. As spectral methods are more accurate than finite difference methods, the ensuing accuracy from this hybrid method outside of the subdomain will be investigated.

Tuesday, April 5

Nonlinear Problems II
Homer Walker
Room B

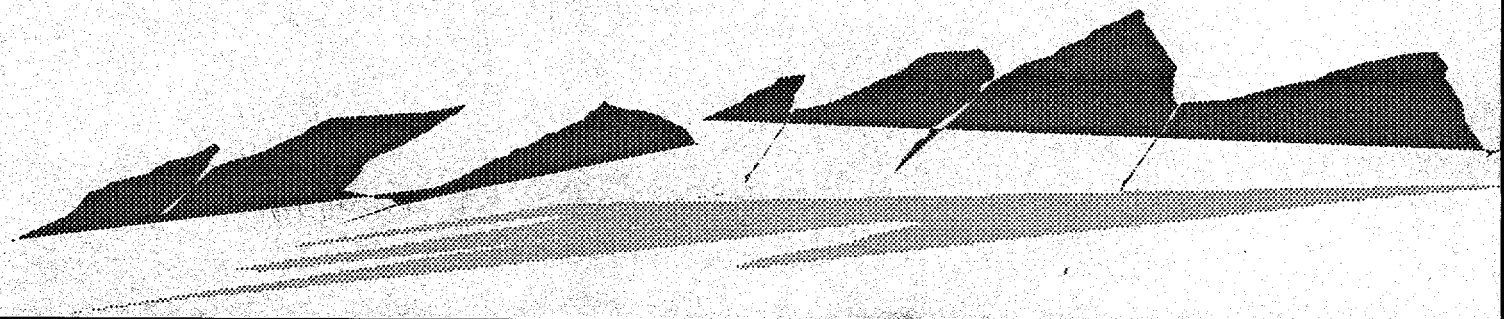
10:15 - 10:40 Scott Hutchinson
A Two-Level Parallel Direct Search Implementation for Arbitrarily Sized Objective Functions

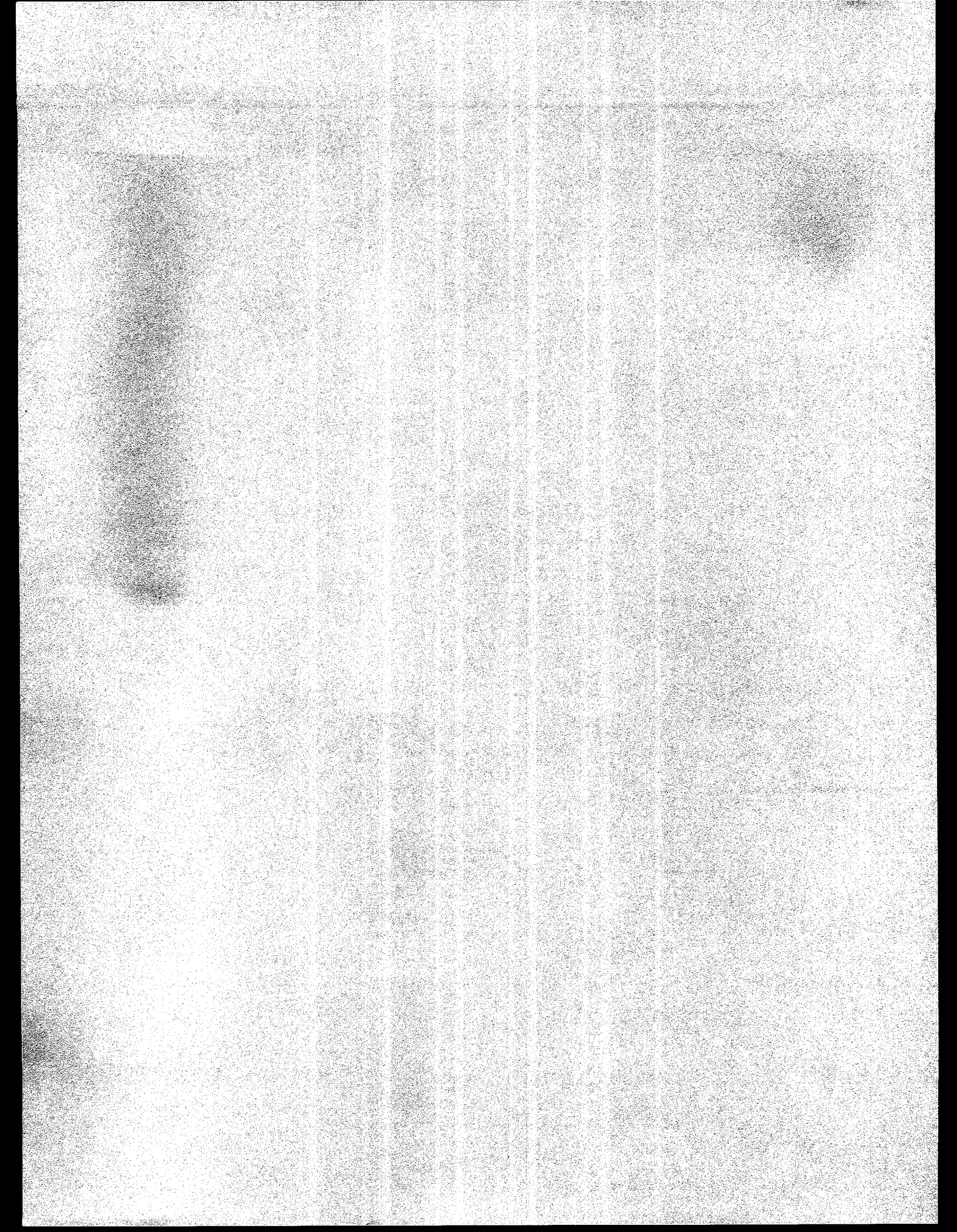
10:40 - 11:05 Rosemary Renaut
Parallel Algorithms for Unconstrained Optimization by Multisplitting with Inexact Subspace Search - The Abstract

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





A Two-Level Parallel Direct Search Implementation for Arbitrarily Sized Objective Functions*

Scott A. Hutchinson† John N. Shadid†

Harry K. Moffat‡ Kwong T. Ng§

February 21, 1994

Abstract

In the past, many optimization schemes for massively parallel computers have attempted to achieve parallel efficiency using one of two methods. In the case of large and expensive objective function calculations, the optimization itself may be run in serial and the objective function calculations parallelized. In contrast, if the objective function calculations are relatively inexpensive and can be performed on a single processor, then the actual optimization routine itself may be parallelized. In this paper, a scheme based upon the Parallel Direct Search (PDS) technique is presented which allows the objective function calculations to be done on an arbitrarily large number (p_2) of processors. If, p, the number of processors available, is greater than or equal to $2p_2$ then the optimization may be parallelized as well. This allows for efficient use of computational resources since the objective function calculations can be performed on the number of processors that allow for peak parallel efficiency and then further speedup may be achieved by parallelizing the optimization.

Results are presented for an optimization problem which involves the solution of a PDE using a finite-element algorithm as part of the objective function calculation. The optimum number of processors for the finite-element calculations is less than $p/2$. Thus, the PDS method is also parallelized. Performance comparisons are given for a nCUBE 2 implementation.

1 Introduction

In this paper, we describe a generalization of the parallel direct search (PDS) optimization algorithm [2, 7] to a two-level scheme wherein both the objective function calculation and the optimization may be parallelized. In the past, most optimization schemes for massively parallel computers have attempted to achieve parallel efficiency at one of two levels. In the case of large and expensive objective function calculations, the optimization itself is run in serial and the objective function calculations are parallelized [1]. On the other hand, if the objective function calculations are relatively inexpensive and can be performed on a single processor,

*This work was partially 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-AC04-76DP00789 and by NIH grant R01-HL-44747.

†Parallel Computational Sciences Department, Sandia National Laboratories, Albuquerque, NM

‡Chemical Processing Science Department, Sandia National Laboratories, Albuquerque, NM

§Department of Electrical and Computer Engineering, New Mexico State University, Las Cruces, NM

then the actual optimization routine itself may be parallelized [2, 3]. Through the use of PDS, we are able to achieve parallel speedup using both techniques, provided we have sufficient computational resources.

The idea is simply that for parallel applications (i.e., parallel objective function calculations) one can consider the implementation's efficiency for a given problem size and determine an optimal number of processors such that the parallel efficiency is maximized. Often, this number may be less than the total number of available processors. Thus, using more than this optimal number may be considered to be inefficient even though the additional processors may provide a speedup. In terms of the two-level PDS optimization, we can calculate objective functions on the optimal number of processors, and achieve further speedup by distributing the optimization work among the remaining processors.

In the following section, we give a brief overview of the PDS optimization technique and discuss how we have generalized the approach. Following this, some computational experiments are described and results given for a particular optimization problem which involves a finite-element solution of a boundary-value problem as part of the objective function calculation. Lastly, a summary is given.

2 Two-Level Parallel Direct Search

The multi-directional direct search methods have recently come into favor as derivative-free optimization techniques [2, 7] for solving the nonlinear unconstrained problem

$$\min_{x \in \mathbf{R}^n} f(x) \tag{1}$$

where $f : \mathbf{R}^n \rightarrow \mathbf{R}$. One popular example of the direct search method is the Nelder-Mead simplex algorithm [4]. Dennis and Torczon's PDS method uses a multi-directional direct search approach which can be shown to converge under some mild assumptions [7, 8]. Further, the algorithm has been developed in the context of a parallel optimization scheme, achieving parallel speedup by distributing the objective function calculations over some number of processors.

2.1 Parallel Direct Search

The PDS algorithm uses a set of simplex rotations, expansions and contractions to search in several directions simultaneously. Since these steps are well defined and simultaneous, it is possible to define a search scheme *a priori* and perform the simultaneous objective function evaluations in parallel. In a preprocessing step, a search scheme (grid) of a predetermined size (s) is generated using the possible simplex operations. This grid amounts to a set of points at which the objective function will be evaluated. At each iteration, the grid is moved to coincide with the current minimum location. Thus, the method also has the ability to navigate non-convex optimization problems.

To parallelize the optimization, the s points in the grid are subdivided and distributed among p processors. Thus, at each iteration, the individual processors each perform $\lfloor s/p \rfloor$ or $\lfloor s/p \rfloor + 1$ objective function evaluations. The assumption, however, is that each processor has the computational resources to evaluate $f(x)$. Further, given this, we can use at most s processors if $s < p$.

2.2 Generalized Two-Level Parallel Direct Search

In many optimization problems, the objective function is derived from complex physical processes which may themselves be difficult to model. Under these conditions, the evaluation of $f(x)$ may be computationally expensive and need to be carried out in parallel (e.g., parallel finite-element models). Thus, the assumption above is no longer valid and, if used, the PDS method simply degenerates to a serial technique where the objective function is parallelized. Thus, the possible objective functions range from those that can be performed efficiently on a single processor to those that may require all the available processors. The metric that should decide the proper number of processors to use for the objective function calculation is their *parallel efficiency*.

Parallel efficiency relative to the minimum number of processors on which the application can run, p_{min}, is defined in terms of the parallel speedup

$$E(p) = \frac{S(p)}{p_{min}} \tag{2}$$

where the speedup is

$$S(p) = \frac{T_{min}}{T_p} \tag{3}$$

Here, T_{min} is the solution time required on p_{min} processors and T_p is the solution time on p processors ($p_{min} \leq p$).

Beginning with speedup, we can consider that speedup due to both the parallelization of the objective function calculations (level-1) and that due to the parallelization of the PDS optimization algorithm (level-2). The level-1 speedup is defined as

$$S^1(p_1) = \frac{T^1_{min}}{T^1_{p_1}} \qquad p_1 \geq p_{min} \tag{4}$$

and the level-2 speedup as

$$S^2(p_2) = \frac{T^2_{min}}{T^2_{p_2}} \qquad p_2 \geq 1 \tag{5}$$

where $p_2 = \lfloor p/p_1 \rfloor$, the number parallel PDS optimization processes. Then, the effective level-two speedup relative to p_{min} processors, can be written

$$S^2_{eff}(p_1 p_2) = \frac{sT^1_{min}}{\frac{s}{p_2}T^1_{min} + \tau(p_2)} \tag{6}$$

$$= \frac{p_2 T^1_{min}}{T^1_{min} + \frac{p_2}{s}\tau(p_2)} \tag{7}$$

where we have assumed that the speedup scales with some function $\tau(p_2)$.

Thus, for the given problem size and machine size, one can determine the optimal number of processors (p^*) which should be used in calculating the objective function. In the present implementation, if $p^* \leq p/2$ then we may also parallelize the optimization using PDS. This defines the two-level approach. Each objective function calculation is performed on p^* processors while $\hat{p} = \lfloor p/p^* \rfloor$ of these processes are performed in parallel. Note that for $p^* = p$ the original PDS algorithm results but is simply run in serial while if $p^* = 1$, the original *parallel* PDS algorithm results.

Since there is very little overhead associated with parallelizing the optimization[1], the parallel speedup here is nearly linear. Thus, if the application speedup is less than the optimization speedup, the optimal number of processors on which to evaluate the objective function is $p^* = p_{min}$. However, if the application is perfectly parallel (linear speedup with slope 1), then assuming some overhead associated with the parallel optimization, $p^* = p$ (serial PDS).

3 Computational Experiments

To assess the performance of the two-level PDS method, we have chosen an optimization problem which involves a finite-element estimation in the evaluation of the objective function. The specific problem is to optimize the electrode position and voltage for an implantable defibrillator configuration [1]. The finite-element problem models the electric current density throughout the thoracic cavity and involves a mesh of 262,110 trilinear tetrahedral elements and 47,113 nodes. While this is a large finite-element problem, it is certainly not one which taxes the resources of a 1024 processor nCUBE 2. In fact, the problem can be solved on only 32 processors of such a machine ($p_{min} = 32$). The finite-element solver is one being developed for message-passing massively parallel computers at Sandia National Laboratories [5, 6].

For the optimization problem, the unknowns are the coordinates in $\mathbf{R}^3$ of each of the two electrode centers and the voltage at one of the electrodes (the other is held at ground). Thus, $n = 7$. Further, the objective function is nonlinear and unconstrained. Thus, it is a suitable real-world test problem for the two-level PDS method.

We first look at the efficiency for the solution of the objective function problem described above. Figure 1 illustrates the scaled efficiency for the test problem over a range of machine sizes. Note, that, as expected, the maximum efficiency occurs for $p = p_{min} = 32$.

We next use this number for the study of the two-level PDS speedup. Figure 2 illustrates the nearly linear speedup of the iterative portion of the two-level PDS algorithm[2].

4 Summary

We have implemented a generalized, two-level version of Dennis and Torczon's parallel direct search (PDS) nonlinear unconstrained optimization algorithm. In this implementation, parallelization may be achieved on two levels — both the objective function calculations and the optimization. The parallelization of the optimization portion yields a nearly linear speedup, thus, the optimal number of processors on which to evaluate the objective function is p^*, that number which yields the highest efficiency for the objective function calculations. For the example problem considered, $p^* = p_{min}$, the minimum number of processors which may be used to run the application.

[1] In the current implementation, this overhead is primarily due to file I/O associated with transferring data between the controlling front-end process and the parallel optimization processes.

[2] There is an initialization portion of the PDS code which, in the current implementation, runs on p_{min} processors and calculates the objective function values for each vertex in the initial simplex. Thus, the overall speedup of the algorithm would depend upon home many iterations are required for convergence

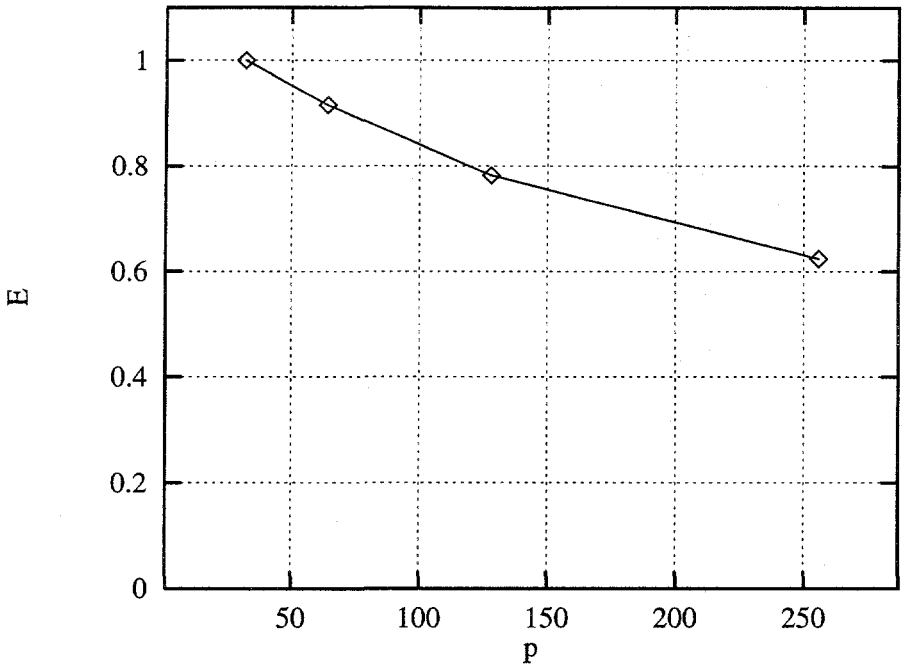


Figure 1: Relative parallel efficiency of the objective function calculations ($p_{min} = 32$)

References

[1] Scott A. Hutchinson. *Parallel Finite-Element Analysis and Optimization of Electrical Defibrillation.* PhD thesis, New Mexico State University, December 1993.

[2] Jr. J.E. Dennis and Virginia Torczon. Direct search methods on parallel machines. *SIAM J. Optimization*, 1(4):448–474, 1991.

[3] J.C. Meza and M.L. Martinez. A numerical study of hybrid optimization methods for the molecular conformation problems. Technical Report SAND93-8233, Sandia National Laboratories, May 1993.

[4] J.A. Nelder and R. Mead. A simplex method for function minimization. *Comput. J.*, 7:308–313, 1965.

[5] J.N. Shadid, S.A. Hutchinson, and H.K. Moffat. Parallel performance of a preconditioned CG solver for unstructured finite element applcaitions. In *Proceedings of the Colorado Conference on Iterative Methods*, Breckenridge, Colorado, April 1994.

[6] J.N. Shadid, H.K. Moffat, and S.A. Hutchinson. A parallel unstructured finite element implementation for MIMD machines. in preparation, 1994.

[7] Virginia Torczon. On the convergence of the multidirectional search algorithm. *SIAM J. Optimization*, 1:123–145, 1991.

[8] Virginia Torczon. On the convergence of pattern search methods. Technical Report TR93-10, Rice University, June 1993.

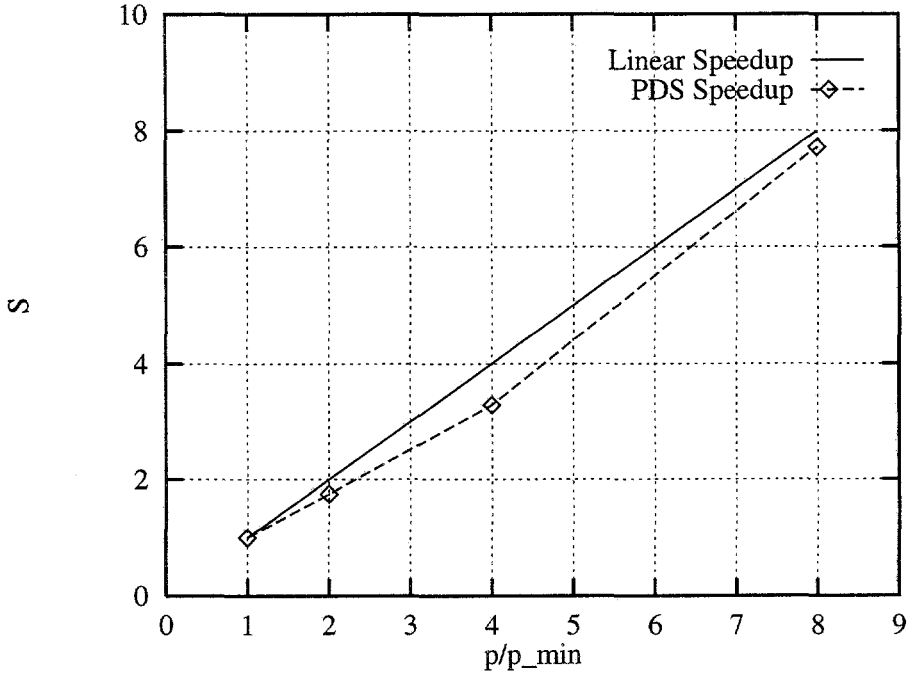


Figure 2: Relative speedup of two-level PDS optimization

PARALLEL ALGORITHMS FOR UNCONSTRAINED OPTIMIZATION BY MULTISPLITTING WITH INEXACT SUBSPACE SEARCH – THE ABSTRACT *

ROSEMARY RENAUT † AND QING HE ‡

Key words. parallel algorithms, multisplitting, unconstrained optimization, QR decompositon, inexact line search, Householder QR decomposition, hypercube computer

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

1. Introduction. In [1] a new parallel iterative algorithm for unconstrained optimization by multisplitting is proposed. In this algorithm the original problem is split into a set of small optimization subproblems which are solved using well known sequential algorithms. These algorithms are iterative in nature, e.g. DFP variable metric method. Here we use sequential algorithms based on an *inexact subspace search*, which is an extension to the usual idea of an inexact line search. Essentially the idea of the inexact line search for nonlinear minimization is that at each iteration we only find an approximate minimum in the line search direction. Hence by *inexact subspace search*, we mean that, instead of finding the minimum of the subproblem at each interation, we do an incomplete down hill search to give an approximate minimum.

Some convergence and numerical results for this algorithm will be presented. Further, the original theory will be generalized to the situation with a singular Hessian. Applications for nonlinear least squares problems will be presented. Experimental results will be presented for implementations on an Intel iPSC/860 Hypercube with 64 nodes as well as on the Intel Paragon. In the next section we present the original algorithms and indicate our modifications.

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

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

* 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 Mathematics, Arizona State University, Tempe, AZ 85287-1804 (renaut@math.la.asu.edu).

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

where $x = (X_1, X_2, ..., X_r)^T$ is partitioned consistently with A. For $1 \leq j \leq r$, define

$$b_j(x) = b - \sum_{i \neq j}^{r} A_i X_i \tag{2.2}$$

and take positive scalars α_i^k, $\alpha_i^k > \alpha_i > 0$, where k is the iteration number, such that

$$\sum_{i=1}^{r} \alpha_i^k = 1, \tag{2.3}$$

then the solution to 2.1 can be found in parallel by solving the subproblems

$$\textit{minimize } \|A_j X_j - b_j(x^k)\|, \ 1 \leq j \leq r. \tag{2.4}$$

The solution is then given from Algorithm 1 by $x^{k+1} = \sum_{j=1}^{r} \alpha_j^{k+1} Z^{j,k+1}$ where for $1 \leq h \leq r$,

$$Z_h^{j,k+1} = \begin{cases} Y_j^{k+1} & \text{if h = j} \\ X_h^k & \text{otherwise.} \end{cases}$$

Let B_i^k be an $m \times 1$ vector, for $1 \leq$ i $\leq$ r and k ≥ 0.

ALGORITHM 1. *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* $j \neq i$
   - *4.2 calculate* $b_i(x^k) = b - \sum_{j \neq i}^{r} B_j^k$
   - *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

End

In this paper we replace the exact minimisation at step 4.3 by the inexact subspace search.

2.2. Nonlinear Minimization. Let f: $R^n \rightarrow R^1$ have positive semidefinite Hessian and have only one stationary point x_1.
The problem is to minimize f(x) in the bounded neighbourhood D of x_1.
For $1 \leq j \leq r$, let

$$\overline{x_j} = (\overline{X_1}^j, \overline{X_2}^j, ..., \overline{X_r}^j) \tag{2.5}$$

where, for $1 \leq h \leq r$

$$\overline{X_h}^j = \begin{cases} Y_j & \text{if h = j} \\ X_h & \text{otherwise.} \end{cases} \tag{2.6}$$

Then the solution of the problem is given by $x^{k+1} = \sum_{j=1}^{r} \alpha_j^{k+1} Z^{j,k+1}$ where for $1 \leq h \leq r$,

$$Z_h^{j,k+1} = \begin{cases} Y_j^{k+1} & \text{if h = j} \\ X_h^k & \text{otherwise.} \end{cases}$$

In the algorithm let $f_j(x, Y_j) = f(\overline{x_j})$.

ALGORITHM 2. *For all processor 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

End

As in Algorithm 1 we replace an exact minimisation at step 3.2 by an inexact subspace search.

REFERENCES

[1] QING HE, *Parallel Algorithms for Unconstrained Optimizations by Multisplitting*, To appear.

SOLVING LINEAR INEQUALITIES IN A LEAST SQUARES SENSE

RANDALL BRAMLEY AND BEATA WINNICKA
COMPUTER SCIENCE DEPARTMENT
INDIANA UNIVERSITY
BLOOMINGTON IN 47405

*

Let $A \in \Re^{m \times n}$ be an arbitrary real matrix, and let $b \in \Re^m$ a given vector. A familiar problem in computational linear algebra is to solve the system $Ax = b$ in a least squares sense; that is, to find an x^* minimizing $||Ax - b||$, where $\|\cdot\|$ refers to the vector two-norm. Such an x^* solves the normal equations $A^T(Ax - b) = 0$, and the optimal residual $r^* = b - Ax^*$ is unique (although x^* need not be). The least squares problem is usually interpreted as corresponding to multiple observations, represented by the rows of A and b, on a vector of data x. The observations may be inconsistent, and in this case a solution is sought that minimizes the norm of the residuals.

A less familiar problem to numerical linear algebraists is the solution of systems of linear *inequalities* $Ax \leq b$ in a least squares sense, but the motivation is similar: if a set of observations places upper or lower bounds on linear combinations of variables, we want to find x^* minimizing $||(Ax - b)_+||$, where the i^{th} component of the vector v_+ is the maximum of zero and the i^{th} component of v.

When the system $Ax \leq b$ is consistent, that is, when a solution exists that satisfies all the inequalities, then phase I of any standard linear programming method can find it. Futhermore when the system is not consistent, linear programming can identify that case, but does not provide any kind of an "optimal" solution. Other methods developed for solving linear inequalities include an unusual algorithm by G.W. Stewart, which defines a function that diverges in a direction that converges to a solution of the inequalities; if no solution exists, the function converges to a unique minimum.

One way of solving the problem is to state it as the quadratic programming problem in (x, z)

$$(1) \qquad (\text{QP}) = \begin{cases} \min \frac{1}{2} z^T z \\ \text{subject to } Ax - b \leq z \end{cases}$$

However, there are serious numerical difficulties with solving a quadratic programming problem that has a singular objective function; furthermore, most methods require an active set strategy that can be difficult to implement, particularly when it is necessary

* Work supported by NSF grant CCR-9120105

to decide which entries to drop from the active set. The analogue of an active set for the algorithm described in this paper is automatically determined without difficult decisions of when to add or drop constraints.

The only algorithm specifically designed for solving linear inequalities in a least squares sense was developed by S.–P. Han. That algorithm requires finding the minimum norm least squares (equality) solution to systems $A_I x = b_I$, where A_I is a submatrix of A consisting of rows of A. This implies that a singular value decomposition or a complete orthogonal decomposition of A_I is required on every iteration. Both of these decompositions are relatively expensive to compute, and there currently are no effective update/downdate methods that allow the reuse of work performed on a previous iteration. This paper will show that a small change in Han's algorithm allows an implementation using a QR factorization with column pivoting instead, and both the robustness and finite termination are retained. We also show that the algorithm has potential applications beyond simple data analysis by applying it to linear separability problems. That application requires finding a hyperplane that best separates two point sets; when the two sets are not linearly separable, a hyperplane that correctly separates the largest number of points is desired.

Numerical Solution of Control Problems Governed by Nonlinear Differential Equations

Matthias Heinkenschloss

Interdisciplinary Center for Applied Mathematics
Department of Mathematics
Virginia Polytechnic Institute
and State University
e–mail: na.heinken@na–net.ornl.gov

In this presentation we investigate an iterative method for the solution of optimal control problems. These problems are formulated as constrained optimization problems with constraints arising from the state equation and in the form of bound constraints on the control. The method for the solution of these problems uses the special structure of the problem arising from the bound constraint and the state equation. Its it is derived from SQP methods and projected Newton methods and combines the advantages of both methods. The bound constraint is satisfied by all iterates using a projection, the nonlinear state equation is satisfied in the limit. Only a linearized state equation has to be solved in every iteration. The solution of the linearized problems are done using multilevel methods and GMRES.

In an abstract formulation the control problems under investigation are given in the form

$$\begin{array}{ll} \min & f(x), \\ s.t. & c(x) = 0 \\ & u \geq 0 \end{array} \tag{1}$$

where $x = (y, u)$ and $f : Y \times U \to \mathbb{R}$, $c : Y \times U \to Z$ are smooth functions. In this formulation U is the space of controls, Y is the state space and the equation $c(x) = 0$ represents the state equation. In the applications we have in mind the state equation is given by nonlinear partial differential equations. The bound constraints $u \geq 0$ can be replaced by $a \leq u \leq b$. However we use the form (1) to keep the presentation simple.

Problems like this frequently occur in practice. One example which will be used in our numerical testings is the control of so–called phase field models. These models describe the solid–liquid phase transition in a pure material [2], [4], [5]. They involve the temperature u of the material and the phase function φ which indicates the liquid or solid state of the material. The

boundary between the two phases can be described by $\Gamma(x,t) = \{(x,t) \in Q \mid \varphi(x,t) = 0\}$. The medium is purely solid if $\varphi(x,t) = -1$ and it is purely liquid if $\varphi(x,t) = +1$. The model consists of a coupled system of two nonlinear parabolic differential equations. The corresponding control problem is to follow a prescribed temperature and/or phase by controlling the heat input to the system through distributed or boundary controls. In the following formulation of this problem we follow the notation in [2], [3], [6] and denote the control by f and the states by (u,φ). The mathematical description is given by

$$\min_{f \in \mathcal{F}_{ad}} \frac{\alpha}{2}\|u - u_d\|^2_{L^2(Q)} + \frac{\beta}{2}\|\varphi - \varphi_d\|^2_{L^2(Q)} + \frac{\gamma}{2}\|f\|^2_{L^2(Q)}.$$

subject to the state equation

$$\begin{aligned} u_t + \tfrac{l}{2}\varphi_t &= \kappa \Delta u + f \\ \tau \varphi_t &= \xi^2 \Delta \varphi + g(\varphi) + 2u \end{aligned} \qquad \text{on } Q = \Omega \times (0,T).$$

with Neumann boundary conditions $\frac{\partial}{\partial n}u = 0$, $\frac{\partial}{\partial n}\varphi = 0$ on $\partial\Omega \times (0,T)$, and initial conditions given by $u(x,0) = u_0(x)$, $\varphi(x,0) = \varphi_0(x)$ $x \in \Omega$. The admissible set $\mathcal{F}_{ad}$ is defined by bound constraints on the control:

$$\mathcal{F}_{ad} = \{ f \mid f_{low} \le f \le f_{upp} \text{ on } Q\}.$$

The state equation contains the nonlinear function g which usually is of the type $g(z) = 0.5z - 0.5z^3$. The constants κ, ℓ, τ, and ξ denote the heat conductivity, the latent heat, the relaxation time, and the length scale of the interface, respectively.

Difficulties in the numerical solution of these types of problems arise from the nonlinearity of the state equation, the presence of of control constraints and the size of the problem. For example, the number of variables in the discretized optimality system for the optimal control of the phase field model is of the order 10^6.

Our algorithm is derived by combining SQP methods for the solution of equality constraint problems with the projected Newton method for problems with simple bound constraints. A more detailed description is given below. The algorithm only requires the solution of the linearized state equation and the (linear) adjoint equation in each step and generates iterates which satisfy the bound constraints. Moreover we introduce a merit function and a Amijo–like line search procedure to enforce global convergence.

For the control problem governed by the phase field model the algorithm allows a decoupling of the system of parabolic equations such that only linear parabolic equations have to be solved. The implementation of this algorithm uses multilevel methods and GMRES to compute the steps.

As mentioned before our method is derived combining SQP methods and projected Newton methods. Projected Newton methods for the solution of optimization problems with simple bound constraints have been introduced by Bertsekas [1]. For the solution of a problem like (1) these methods require the elimination of the constraint $c(x) = 0$. If we assume that the derivative of c with respect to y is continuously invertible, then, by the implicit function theorem there exists a function $y(u)$ implicitly defined through

$$c(y(u), u) = 0.$$

Using this function (1) can be formulated as

$$\begin{array}{ll} \min & \phi(u) = f(y(u), u). \\ s.t. & u \geq 0 \end{array} \tag{2}$$

Under the assumption on the derivative C_y the objective function ϕ is twice differentiable.

For finite dimensional problems projected Newton methods generate iterates of the form

$$u_{k+1} = P(u_k - \alpha_k D_k^{-1} \nabla \phi(u_k)), \tag{3}$$

where P denotes the projection onto $\{u | u \geq 0\}$. The matrix D_k is a projection of the Hessian $\nabla^2 \phi$ and is given by

$$D_k = I_{A_k^+} \operatorname{diag}(\nabla^2 \phi(u_k))\, I_{A_k^+} + I_{A_k^{+c}} \nabla^2 \phi(u_k) I_{A_k^{+c}},$$

where $A_k^+ = A^+(u_k) = \{i \mid (u_k)_i = 0,\ \partial\phi(u_k)/\partial u_i > 0\}$, $A_k^{+c} = \{1, \ldots, n\} \setminus A_k^+$, and $I_A : \mathbb{R}^n \to \mathbb{R}^n$ is the indicator function for the index set A defined by

$$I_A : \mathbb{R}^n \to \mathbb{R}^n, \quad (I_A x)_i = \begin{cases} x_i & i \in A, \\ 0 & i \notin A. \end{cases}$$

The matrix D_k is called diagonal with respect to A^+. It is shown by Bertsekas that, under appropriate assumptions the active set is identified after finitely many iterations: $A(u_k) = A(u_*) = \{i \mid u_i = 0\}$ for all $k \geq k_*$. For

iterations $k \geq k_*$ the projected Newton method is equivalent to Newton's method for the unconstrained problems on the set of inactive indices and, thus, is locally q–quadratic convergent. The analysis of projected Newton methods is generalized to infinite dimensional problems in [7]. The simple projection in (3) and the convergence properties make the projected Newton method very attractive for problems like (1) for which the state y can be efficiently computed as a function of the control u. However, for control problems governed by nonlinear partial differential equations the solution of the state equation in every iteration is extremely expensive and makes this method impractical. Instead, one would like to use a method that make use of the simple projection, but solves the nonlinear state equation only as the sequence of iterates converges. Such a generalization of projected Newton methods was introduced by Kelley and Sachs [8] and analyzed for control problems governed by ODEs. Their starting point is the system of necessary optimality conditions for (1). The result is an algorithm which in each step requires the solution of a projected linearized Kuhn–Tucker system and which is locally convergent with q–order $1 + p$ for some $p \in (0, 1)$.

We use a different approach for the generalization of the projected Newton method in finite dimensions. We compare SQP methods for the solution of

$$\begin{array}{ll} \min & f(x) \\ s.t. & c(x) = 0 \end{array} \tag{4}$$

with the projected Newton method. To ensure global convergence of the algorithm we introduce a constraint merit function and incorporate an Amijo–like line search procedure. Locally, our method is very similar to the algorithm proposed in [8].

The Lagrange function for (4) is given by

$$L(x, \lambda) = f(x) - \lambda^T c(x), \tag{5}$$

and the Kuhn–Tucker conditions for (4) have the form

$$\begin{array}{rcrcl} \nabla_x L(x, \lambda) & = & \nabla f(x) - C(x)\lambda & = & 0, \\ \nabla_\lambda L(x, \lambda) & = & c(x) & = & 0, \end{array} \tag{6}$$

where C^T denotes the Jacobian of c. If Newton's method is applied for the solution of the Kuhn–Tucker system, then at each iteration the following system has to be solved:

$$\begin{pmatrix} \nabla_{xx} L(x, \lambda) & -C(x) \\ C(x)^T & 0 \end{pmatrix} \begin{pmatrix} s_x \\ s_\lambda \end{pmatrix} = - \begin{pmatrix} \nabla_x f(x) - C(x)\lambda \\ c(x) \end{pmatrix}. \tag{7}$$

If $\nabla_{xx}L(x,\lambda)$ is positive definite on the null space of $C(x)^T$, then (7) is uniquely solvable. Using the fact that the Jacobian C_y of c with respect to y is invertible, one can establish the following equations for the step $s_x = (s_y, s_u)$:

$$\begin{aligned} s_u &= -B(x,\lambda)^{-1}T(x)^T(\nabla_x f(x) + \nabla^2_{xx}L(x,\lambda)r(x)), \\ s_y &= -C_y(x)^{-T}C_u(x)s_u - C_y(x)^{-T}c(x), \end{aligned}$$

where

$$r(x) = \begin{pmatrix} -C_y(x)^{-T}c(x) \\ 0 \end{pmatrix}, \quad T(x) = \begin{pmatrix} -C_y(x)^{-T}C_u(x)^T \\ I \end{pmatrix},$$

and $B(x,\lambda) = T(x)^T\nabla^2_{xx}L(x,\lambda)\,T(x)$.

We extend the projected Newton method in the following way: The Hessian $\nabla^2\phi$ is replaced by B_k, an approximation of the reduced Hessian $B(x_k,\lambda_k)$ and the subset of the active set is determined using a modification of the reduced gradient $T(x)^T\nabla_x f(x)$:

$$A^+(u) = \{i \mid u_i = 0, \quad (T(x)^T\nabla_x f(x) + a(x))_i > 0\}.$$

If the projected matrix

$$D_k = I_{A_k^+}\mathrm{diag}(B_k)I_{A_k^+} + I_{A_k^{+c}}B_k I_{A_k^{+c}}$$

is invertible, we define

$$\begin{aligned} s_u &= -D_k^{-1}(T(x_k)^T\nabla_x f(x_k) + b(x_k)), \\ u_k(\alpha) &= P(u_k + \alpha s_u), \\ y_k(\alpha) &= y_k - \alpha C_y(x_k)^{-T}c(x_k) - C_y(x_k)^{-T}C_u(x_k)^T(u_k - P(u_k + \alpha s_u)). \end{aligned} \tag{8}$$

The vectors $a(x_k), b(x_k)$ and the approximation B_k of the reduced Hessian are motivated by the desire to avoid second order derivatives and to allow for quasi–Newton methods. Our formula for the Lagrange multiplier estimate is given by

$$\lambda(x) = C_y(x)^{-1}\nabla_y f(x). \tag{9}$$

Our algorithm generates iterates of the form

$$(y_{k+1}, u_{k+1}) = (y_k(\alpha_k), u_k(\alpha_k)),$$

where $\alpha_k > 0$ is some step size. The new Lagrange multiplier estimate is given by $\lambda_{k+1} = \lambda(y_{k+1}, u_{k+1})$, cf. (9). For the choice of the step size we

need a merit function which monitors the progress in reducing the function value of f and in approaching the feasible set $\{x|c(x) = 0\}$. We use the merit function

$$\Phi(x, \rho) = f(x) - \lambda(x)^T c(x) + \frac{\rho}{2}||c(x)||_2^2.$$

Since our algorithm generates iterates which satisfy the inequality constraints we have the unusual situation of a constrained merit function:

$$\min_{u \geq 0} \Phi(x, \rho).$$

The step size α_k is chosen by an Amijo–like step size rule such that a sufficient decrease in the merit function is obtained.

If the constraints are linear, i.e. if $c(y, u) = Cy + Bu$, then our algorithm is equivalent to the projected Newton method by Bertsekas, provided the starting point (y_0, u_0) is feasible.

Form the presentation (8) one can see that quasi–Newton methods can be used to approximate second order information. However, if second order information is available, then the iteration can also be performed by solving the projected linearized Kuhn–Tucker system

$$\begin{pmatrix} \nabla_{yy} L & \nabla_{yu} L I_{A^+{}^c} & -C_y \\ I_{A^+{}^c} \nabla_{uy} L & I_{A^+{}^c} \nabla_{uu} L I_{A^+{}^c} + \Lambda_k & -I_{A^+{}^c} C_u \\ C_y^T & C_u^T I_{A^+{}^c} & 0 \end{pmatrix} \begin{pmatrix} s_y \\ s_u \\ s_\lambda \end{pmatrix} = - \begin{pmatrix} \nabla_y f - C_y \lambda \\ \nabla_u f - C_u \lambda \\ c \end{pmatrix}, \tag{10}$$

where Λ_k is a positive definite diagonal matrix satisfying $\Lambda_k = I_{A^+} \Lambda_k I_{A^+}$, i.e. Λ_k is zero in all rows and columns with indices $i \in A^{x^c}$. In (10) we omitted the arguments x_k, λ_k and used A^+ instead of A_k^+. For certain control problems this formulation may be favorable. In fact, for the control of the phase field model (10) is used. The structure of the control problem allows a reformulation of this system, which results in a decoupling of the equations. The new system is then solved using a multilevel method in combination with GMRES. In our numerical tests the method shows a fast linear convergence and the time needed to solve the nonlinear problem (1) roughly grows linearly with the discretization. It can also be observed that the method hardly requires more work than for the unconstrained case.

References

[1] D. B. Bertsekas. Projected Newton methods for optimization problems with simple constraints. *SIAM J. Control Optim.*, 20:221–246, 1982.

[2] G. Caginalp. An analysis of a phase field model of a free boundary. *Archive for Rational Mechanics and Analysis*, 92:205–245, 1986.

[3] Z. Chen. *Das Phasenfeldmodell für das Problem der Phasenübergänge: Analysis, Numerik und Steuerung.* PhD thesis, Institut für Mathematik, Universität Augsburg, Germany, 1991.

[4] J. B. Collins and H. Levine. Diffusive interface model of diffusion limited crystal growth. *Physica Review B*, 31:6119–6122, 1985.

[5] G. Fix. Numerical simulation of free boundary problems using phase field models. In J. R. Whiteman, editor, *The Mathematics of Finite Elements and Applications IV, MAFELAP 1981*, pages 265–279. Academic Press, London, New York, ..., 1982.

[6] K.-H. Hoffmann and L. Jiang. Optimal control problem of a phase field model for solidification. *Numer. Funct. Anal.*, 13:11–27, 1992.

[7] C. T. Kelley and E. W. Sachs. Multilevel algorithms for constrained compact fixed point problems. *SIAM J. Scientific and Stat. Computing*, to appear, 1993.

[8] C. T. Kelley and E. W. Sachs. Solution of optimal control problems by a pointwise projected Newton method. *Department of Mathematics, North Carolina State University, Raleigh, NC*, 1993.

Tuesday, April 5

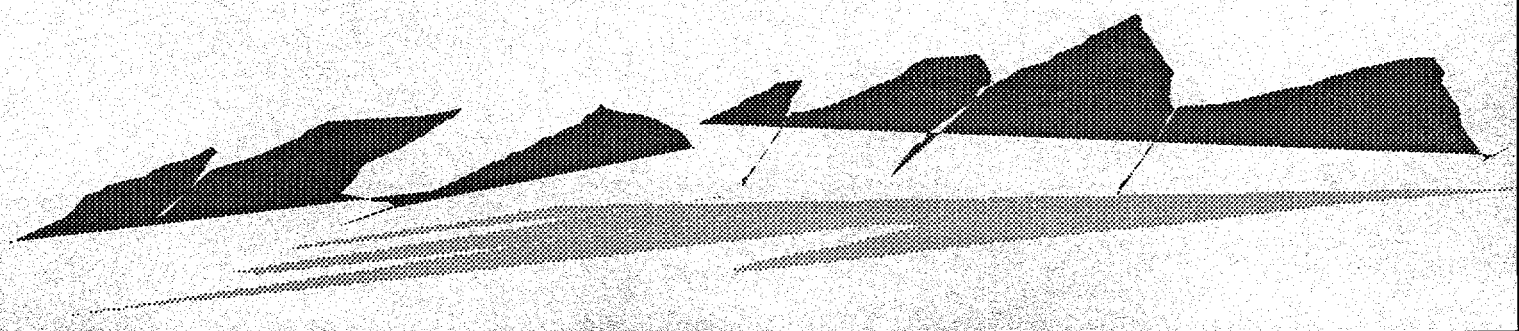
Integral Equations and Inverse Problems
Chair: Nick Trefethen
Room A

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



COMPARING PRECORRECTED-FFT AND FAST MULTIPOLE ALGORITHMS FOR SOLVING THREE-DIMENSIONAL POTENTIAL INTEGRAL EQUATIONS

J. WHITE*, J. R. PHILLIPS† AND T. KORSMEYER‡

1. Introduction. Mixed first- and second-kind surface integral equations with $\frac{1}{r}$ and $\frac{\partial}{\partial n}\frac{1}{r}$ kernels are generated by a variety of three-dimensional engineering problems. For such problems, Nyström type algorithms can not be used directly, but an expansion for the unknown, rather than for the entire integrand, can be assumed and the product of the singular kernal and the unknown integrated analytically. Combining such an approach with a Galerkin or collocation scheme for computing the expansion coefficients is a general approach, but generates dense matrix problems. Recently developed fast algorithms for solving these dense matrix problems have been based on multipole-accelerated iterative methods [1, 2, 3, 8], in which the fast multipole algorithm is used to rapidly compute the matrix-vector products in a Krylov-subspace based iterative method. Another approach to rapidly computing the dense matrix-vector products associated with discretized integral equations follows more along the lines of a multigrid algorithm [4], and involves projecting the surface unknowns onto a regular grid, then computing using the grid, and finally interpolating the results from the regular grid back to the surfaces.

In this paper, we describe a precorrected-FFT approach which can replace the fast multipole algorithm for accelerating the dense matrix-vector product associated with discretized potential integral equations. The precorrected-FFT method, described below, is an order $n \log n$ algorithm, and is asymptotically slower than the order n fast multipole algorithm. However, initial experimental results indicate the method may have a significant constant factor advantage for a variety of engineering problems.

2. Problem Formulation. In this paper we consider only a simplified discretization applied to a first kind formulation, though the techniques generalize to mixed formulations and more complicated discretizations. The approach used is to formulate the exterior Dirichlet problem using a single layer charge density denoted σ. It then follows that σ must satisfy the integral equation

$$\psi(x) = \int_{surfaces} \sigma(x') \frac{1}{\|x - x'\|} da', \qquad x \in surfaces. \tag{1}$$

where $\psi(x)$ is the known surface potential, da' is the incremental conductor surface area, x, $x' \in \mathbf{R}^3$, and $\|x\|$ is the usual Euclidean length of x.

* Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139. (white@rle-vlsi.mit.edu)

† Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology Cambridge, MA 02139. (jphill@rle-vlsi.mit.edu)

‡ Dept. of Ocean Engineering Massachusetts Institute of Technology Cambridge, MA 02139. (xmeyer@f-c.mit.edu)

A standard approach to numerically solving (1) for σ is to use a piece-wise constant Galerkin scheme. That is, the surfaces are discretized into n panels, and it is assumed that on each panel i, a charge, q_i, is uniformly distributed. Then for each panel, an equation is written which relates the known i-th panel potential, denoted $\overline{p}_i$, to the sum of the contributions to that potential from the n charge distributions on all n panels. The result is a dense linear system,

$$Pq = \overline{p} \tag{2}$$

where $P \in \mathbf{R}^{n \times n}$, q is the vector of panel charges, $\overline{p} \in \mathbf{R}^n$ is the vector of known panel potentials, and

$$P_{ij} = \frac{1}{a_i a_j} \int_{panel_i} \int_{panel_j} \frac{1}{\|x - x'\|} \, dada', \tag{3}$$

where a_i and a_j are the areas of the i-th and j-th panel.

The dense linear system of (2) can be solved to compute panel charges from a given set of panel potentials. If Gaussian elimination is used to solve (2), the number of operations is order n^3. Clearly, this approach becomes computationally intractable if the number of panels exceeds several hundred. Instead, consider solving the linear system (2) using a Krylov-subspace style iterative method. The dominant costs in such an algorithm will be calculating the n^2 entries of P using (3) before the iterations begin, and performing n^2 operations to compute Pq on each iteration. Described below is a precorrected-FFT based algorithm which, through the use of approximate grid projections, avoids forming most of P and reduces the cost of forming Pq to order $n \log n$ operations.

3. The precorrected-FFT method. After a three dimensional problem has been discretized into panels, consider then subdividing the cube containing the problem into an $m \times m \times m$ array of small cubes so that each small cube contains only a few panels. Several sparsification techniques for P are based on the idea of directly computing only those portions of Pq associated with interactions between panels in neighboring cubes. The rest of Pq is then somehow approximated to accelerate the computation [7].

One approach to computing distant interactions is to exploit the fact that evaluation points distant from a cube can be accurately computed by representing the given cube's charge distribution using a small number of weighted point charges. In particular, if the point charges all lie on a uniform grid, then FFT can be used to compute the potential at these grid points due to the grid charges. More specifically, one method for approximating Pq in $n \log n$ operations has four steps:

- directly compute nearby interactions,
- project the panel charges onto a uniform grid of point charges,
- compute the grid potentials due to grid charges using an FFT, and
- interpolate the grid potentials onto the panels.

The difficulty, as will be made clearer below, is that the calculations using the FFT on the grid do not accurately approximate the nearby interactions. And in addition, this poor approximation must be subtracted from the result before accurate direct calculation

of nearby interactions can be substituted. This subtraction can be performed at almost no cost by modifying the way the nearby interactions are computed, a step we refer to as precorrection, and is described below.

3.1. Projecting onto a grid. For panel charges contained within a given cube, the potentials at evaluation points distant from the given cube can be accurately computed by representing the given cube's charge distribution with a small number of appropriately weighted point charges on a uniform grid thoughout the given cube's volume. For example, consider the cube embedded in the center of a $3 \times 3 \times 3$ array of cubes, and assume that the potential will be evaluated at points exterior to the 27 cube array. Then, since the potential satisfies Laplace's equation, the error in the point charge approximation over the entire exterior can be minimized by minimizing the potential error on the surface of the cube array.

The above observation suggests a scheme for computing the grid charges used to represent charge in a given cube a. First, test points are selected on the surface of the $3 \times 3 \times 3$ cube array which has cube a as its center. Then, potentials due to the grid charges are forced to match the potential due to the cube's actual charge distribution at the test points. Since such collocation equations are linear in the charge distribution, this projection operation which generates a subset of the grid charges, denoted q_a^g, can be represented as a matrix, W_a, operating on a vector representing the panel charges in cube a, q_a. In particular, if there are G grid charges and A panels, then

$$q_a^g = W_a q_a = \begin{bmatrix} P^{gt} \\ 1 \ldots 1 \end{bmatrix}^{-1} \begin{bmatrix} P^{qt} \\ 1 \ldots 1 \end{bmatrix} q_a, \tag{4}$$

where $P^{gt} \in \Re^{(G-1)\times G}$ is the mapping between grid charges and test point potentials and is given by

$$P^{gt}_{i,j} = \frac{1}{\|x_i^t - x_j^g\|}; \tag{5}$$

$P^{qt} \in \Re^{(G-1)\times A}$ is the mapping between panel charges and test point potentials and is given by

$$P^{qt}_{i,j} = \int_{panel\ j} \sigma(x') \frac{1}{\|x_i^t - x'\|} da'. \tag{6}$$

Here x_i^t and x_j^g are the position of the i-th test point and the j-th grid point. The rows of ones in (4) insure that the sum of grid charges is equal to the net charge in the cube. For an alternative approach, based more generally on matching multipole expansion coefficients, see [6].

3.2. Using the FFT. For a general three dimensional problem, consider subdividing a cube containing the entire problem domain into a $m \times m \times m$ array of small cubes. Then, the collocation approach above can be used to generate point charge approximations for

charge distributions in every cube, effectively projecting the charge density onto a three-dimensional grid. For example, if the representative point charges are placed at the cube vertices, then the resulting charge distribution will be projected to a $(m+1) \times (m+1) \times (m+1)$ uniform grid. Fast multipole algorithms also effectively create a uniform grid by constructing multipole expansions at the center of each cube, but due to sharing, the point charge approach can be more efficient. For example, a point charge at a cube vertex is used to represent charge in the eight cubes which share that vertex.

Once the charge has been projected to a grid, computing the potentials at the grid points due to the grid charges is a three-dimensional convolution. We denote this as

$$\psi_g(i,j,k) = \sum_{i',j',k'} h(i-i', j-j', k-k') q_g(i',j',k'). \tag{7}$$

where i,j,k and i',j',k' are triplets specifying the grid points, ψ_g is the vector of grid point potentials, q_g is the vector of grid point charges, and $h(i-i', j-j', k-k')$ is the inverse distance between grid points i,j,k and i',j',k'. As will be made clear below, $h(0,0,0)$ can be arbitrarily defined, and is set to zero. The above convolution can be computed in $n \log n$ time using the Fast Fourier Transform.

Once the grid potentials have been computed, they can be interpolated to the panels in each cube using the transpose of W_a [4]. Therefore, projection, followed by convolution, followed by interpolation, can be represented as

$$\psi_{fft} = W^t H W q, \tag{8}$$

where q is the vector of panel charges, ψ_{fft} is an approximation to the panel potentials, W is the concatenation of the W_a's for each cube, and H is the matrix representing the convolution in (7).

3.3. Precorrecting. In ψ_{fft} of (8), the portions of Pq associated with neighboring cube interactions have already been computed, though this close interaction has been poorly approximated in the projection/interpolation. Before computing a better approximation, it is necessary to remove the contribution of the inaccurate approximation. In particular, denote as $P_{a,b}$ the portion of P associated with the interaction between neighboring cubes a and b, denote the potential at grid points in cube a due to grid charges in cube b as $H_{a,b}$, and denote ψ_a and q_b as the panel potentials and charges in cubes a and b respectively. Then

$$\psi_a = \psi_{a_{fft}} + \left(P_{a,b} - W_a^t H_{a,b} W_b^t)\right) q_b \tag{9}$$

will be a much better approximation to ψ_a.

Assuming that the Pq product will be computed many times in the inner loop of an iterative algorithm,

$$P_{a,b}^{cor} = \left(P_{a,b} - W_a^t H_{a,b} W_b^t)\right) \tag{10}$$

will be expensive to initially compute, but will cost no more to subsequently apply than $P_{a,b}$.

4. Conclusions. Using the above notation, the precorrected-FFT algorithm for computing Pq can be briefly described as two steps. First compute

$$\psi_{fft} = W^t H W q \tag{11}$$

using the FFT to sparsify H. Then, for each cube a, compute

$$\psi_a = \sum_{b \in neighbors} P^{cor}_{a,b} q_b + \psi_{a_{fft}}; \tag{12}$$

Numerical experiments indicate that using a uniform grid with an average of eight grid points associated with each cube (making $G = 27$ in (4)) results in potentials outside nearest neighbors which have errors similar to those produced by evaluating second-order multipole expansions (which have nine coefficients) outside second nearest neighbors. Assuming a homogenous distribution of eight panels per cube, this implies that for commensurate accuracy, the precorrected-FFT method has a complexity of $(243 + K_{fft} \log 2n) * n$, and the fast multipole algorithm has a complexity of at least $2700n$. So, although the fast multipole algorithm is asymptotically faster, for an efficient FFT program, this will only be of practical significance for extremely large n.

It should be noted that the above result is *not* general. The fast multipole algorithm retains its linear-time behavior even in the arbitrarily inhomogenous case [5]. The precorrected-FFT method is preferable when the distribution can be made to look homogenous. More detailed experiments are required to better understand these pragmatic issues.

REFERENCES

[1] V. Rohklin, "Rapid solution of integral equation of classical potential theory," *J. Comput. Phys.*, vol. 60, pp. 187–207, 1985.

[2] A. Greenbaum, L. Greengard, and G. B. McFadden, "Laplace's equation and the Dirichlet-Neumann map in multiply connected domains," tech. rep., Courant Institute, New York, March 1991.

[3] K. Nabors and J. White, "Fastcap: A multipole accelerated 3-D capacitance extraction program," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 10, pp. 1447–1459, November 1991.

[4] A. Brandt, "Multilevel computations of integral transforms and particle interactions with oscillatory Kernals," *Computer Physics Communications*, no. 65, pp. 24-38, 1991.

[5] K. Nabors, F. T. Korsmeyer, F. T. Leighton, and J. White, "Multipole accelerated preconditioned iterative methods for three-dimensional potential integral equations of the first kind," *To Appear, SIAM J. on Sci. and Stat. Comp.*

[6] C. L. Berman, "Grid-multipole calculations," *IBM Research Report*, no. RC 19068(83210), 1993.

[7] L. Greengard, *The Rapid evaluation of potential fields in particle systems.* Cambridge, Massachusetts: M.I.T. Press, 1988.

[8] T. Korsmeyer, D. K. P. Yue, K. Nabors, J. White, "Multipole-accelerated preconditioned iterative methods for three-dimensional potential problems," *Boundary Element Methods 15 (BEM15)*, Worcester, MA, August 1993.

THE NUMERICAL SOLUTION OF TOTAL VARIATION MINIMIZATION PROBLEMS IN IMAGE PROCESSING

C. R. VOGEL * AND M. E. OMAN †

Consider the minimization of penalized least squares functionals of the form

$$f(u) = \frac{1}{2}\|Au - z\|^2 + \alpha \int_\Omega |\nabla u|\, dx. \tag{1}$$

Here A is a bounded linear operator, z represents data, $\|\cdot\|$ is a Hilbert space norm, α is a positive parameter, $\int_\Omega |\nabla u|\, dx$ represents the total variation (TV) of a function $u \in BV(\Omega)$, the class of functions of bounded variation on a bounded region Ω, and $|\cdot|$ denotes Euclidean norm. In image processing, u represents an image which is to be recovered from noisy data z. Certain "blurring processes" may be represented by the action of an operator A on the image u. Two cases of special interest are

(i) **Denoising.** Here $A = I$, the identity operator. One wishes to extract the exact image from a noisy recorded image $z = u_{exact} + \epsilon$.

(ii) **Deblurring.** Here A is a convolution operator

$$Au(x) = \int_\Omega k(x-y)\, u(y)\, dy, \quad x \in \Omega,$$

and one wishes to deconvolve noisy data $z = Au_{exact} + \epsilon$.

In the deblurring case, the operator equation $Au = z$ is ill-posed, and discretizations of it are highly ill-conditioned. The purpose of penalty terms like the TV functional $\int_\Omega |\nabla u|\, dx$ in (1) is to impose stability in a manner which incorporates certain a priori information about the unknown solution. The TV functional penalizes highly oscillatory solutions while allowing discontinuities or very steep gradients in the solution. The qualitative differences between solutions obtained with TV and the standard quadratic penalty functional $\int_\Omega |\nabla u|^2\, dx$ are illustrated in Figures 1 and 2 below.

For further discussion of TV methods in image processing, see the seminal papers by Rudin and Osher, et al, [7], [8]. The numerical methods presented in these two papers are analyzed in [6]. For a detailed discussion of TV and functions of bounded variation, see [5]. TV penalty methods are considered in [9]. For an abstract analysis of TV penalty methods, see [1]. TV methods have recently been applied to the electrical impedance tomography (EIT) problem, a parameter identification problem in elliptic PDE [4].

While the minimizer of the TV penalized functional (1) has very desirable features, the functional itself has some characteristics which pose very serious challenges for numerical analysts. The Euler-Lagrange equations (first order necessary conditions for a minimizer) are formally

$$g(u) \stackrel{\text{def}}{=} A^*(Au - z) - \alpha \nabla \cdot \left(\frac{\nabla u}{|\nabla u|} \right) = 0, \quad x \in \Omega, \tag{2}$$

* Department of Mathematical Sciences, Montana State University, Bozeman MT 59717, e-mail: vogel@math.montana.edu. Research was supported in part by the NSF under Grant DMS-9303222.

† Department of Mathematical Sciences, Montana State University, Bozeman MT 59717, e-mail: imsgmoma@mathfs.math.montana.edu.

$$\frac{\partial u}{\partial n} = 0, \quad x \in \partial\Omega. \tag{3}$$

When $A = I$ (the denoising case), this is a second order elliptic PDE. When A is an integral operator (the deconvolution case), it is an integro-differential equation. Certain difficulties arise from the fact that the differential operator, which can be expressed in "diffusion operator" form

$$\mathcal{D}(u) \stackrel{\text{def}}{=} -\nabla \cdot (\kappa(u)\nabla u), \quad \kappa(u) \stackrel{\text{def}}{=} \frac{1}{|\nabla u|}, \tag{4}$$

is highly nonlinear and degenerate in the sense that the diffusion coefficient κ is neither bounded above nor away from zero. Moreover, as a consequence of the nondifferentiability of the Euclidean norm at the origin, κ is not differentiable. Nondifferentiability and the lack of an upper bound can be overcome by replacing κ with a smooth approximation

$$\kappa_\beta(u) = \frac{1}{\sqrt{|\nabla u|^2 + \beta}}, \quad \beta > 0. \tag{5}$$

Even with this replacement, the operator $\mathcal{D}$ is still not coercive, or strongly elliptic. Additional difficulties arise when Ω is a region in two- or three-dimensional space. To obtain high resolution images, one must apply a high level of discretization, which results in very large (highly nonlinear) systems. When $A = I$, these systems are sparse. However, when A is an integral operator as is the case with deconvolution, these systems are no longer sparse.

The solution approach taken by Rudin and Osher in [7], [8] was essentially to replace (2) with a time–dependent PDE

$$\frac{\partial u}{\partial t} = -g(u), \quad x \in \Omega,\ t > 0, \tag{6}$$

and then integrate from some initial guess to steady state using an explicit time–marching scheme. This approach is essentially a gradient descent method with a fixed step size. It is very slow to converge and step size selection is problematic.

An alternative approach considered in [9] uses Newton's method to minimize a smooth version f_β of the functional in (1). With a line search added for robustness, the iteration takes the form

1. Obtain s by solving the linear system

$$H_\beta(u^{(k)})s = -g_\beta(u^{(k)}). \tag{7}$$

2. Find an approximate solution λ^* to the 1-D "line search" subproblem

$$\min_{0<\lambda\leq 1} f_\beta(u^{(k)} + \lambda s). \tag{8}$$

3. Update the approximate solution $u^{(k+1)} = u^{(k)} + \lambda^* s$.

Here f_β, g_β, and H_β are the objective function, gradient, and Hessian (second derivative), respectively, obtained by replacing $|\nabla u|$ with $\sqrt{|\nabla u|^2 + \beta}$ for some $\beta > 0$. For any positive β and any fixed discretization, this algorithm is locally quadratically convergent. However, the size of the neighborhood about the solution where quadratic convergence is obtained depends strongly on β. This neighborhood is large when

β is relatively large. As might be expected, the size of the neighborhood decreases dramatically as β becomes small.

An obvious shortcoming of Newton's method lies in the computational complexity of solving the linear system (7). Recently obtained results obtained using a "truncated Newton method" (see [3]) with a preconditioned conjugate gradient iteration to solve the linear system (7) will be presented. The performance of FFT-based preconditioners (see [2]) based on the block-Toeplitz matrix

$$C = A^*A + \alpha L_\beta \tag{9}$$

will be discussed. Here L_β is the matrix representing a discretization of the linear constant coefficient diffusion operator

$$\mathcal{L}_\beta u = -\frac{1}{\beta^{3/2}} \nabla \cdot (\nabla u).$$

This is an approximation to the Hessian of the functional $\int_\Omega \sqrt{|\nabla u|^2 + \beta}\, dx$ obtained by setting u equal to a constant.

Also to be presented are results obtained using a fixed point iteration:

$$u^{(k+1)} = \left(A^*A + \alpha D_\beta(u^{(k)}) \right)^{-1} A^* z, \quad k = 0, 1, 2, ... \tag{10}$$

Here $D_\beta(u)$ is a discretization of the linear diffusion operator

$$D_\beta(u)v = -\nabla \cdot (\kappa_\beta(u) \nabla v), \tag{11}$$

i.e., the linear operator in (10) depends on the previous iterate $u^{(k)}$ via the diffusion coefficient (5). In numerical experiments conducted up to this date, this iteration seems to display monotonically convergence in the sense that the objective functionals $f_\beta(u^k)$ converge to zero monotonically. The rate of convergence appears to be linear, but seems not to depend strongly on β. In fact, convergence has been obtained using extremely small values of β, for which Newton's method is not feasible. This fact, together with apparent global convergence properties, makes this method a strong competitor to variants of Newton's method.

Further work is proceeding in two directions: (i) acceleration of the convergence rate; and (ii) the efficient solution of the (large) linear systems in (10).

REFERENCES

[1] R. Acar and C. R. Vogel, *Analysis of Total Variation penalty methods for linear operator equations*, preprint, to be submitted to SIAM J. Numer. Analysis.

[2] R. H. Chan, J. G. Nagy, and R. J. Plemmons, *FFT-based preconditioners for Toeplitz-block least squares problems*, to appear in SIAM J. Numer. Analysis.

[3] R. S. Dembo and T. Steihaug, *Truncated-Newton algorithms for large-scale unconstrained optimization*, Math. Programming, vol. 26 (1983), pp. 190-212.

[4] D. C. Dobson and F. Santosa, *An image enhancement technique for electrical impedance tomography*, to appear in Inverse Problems.

[5] E. Giusti, *Minimal Surfaces and Functions of Bounded Variation*, Birkhäuser, 1984.

[6] P.L. Lions, S. Osher, and L. Rudin, *Denoising and Deblurring Algorithms with Constrained Nonlinear PDE's*, submitted to SIAM J. Numer. Analysis.

[7] L.I. Rudin, S. Osher, and E. Fatemi, *Nonlinear Total Variation Based Noise Removal Algorithms*, Physica D, vol 60 (1992), pp. 259-268.

[8] L.I. Rudin, S. Osher, and C. Fu, *Total Variation Based Restoration of Noisy, Blurred Images*, submitted to SIAM J. Numer. Analysis.

[9] C. R. Vogel, *Total Variation regularization for ill-posed problems*, submitted to Inverse Problems.

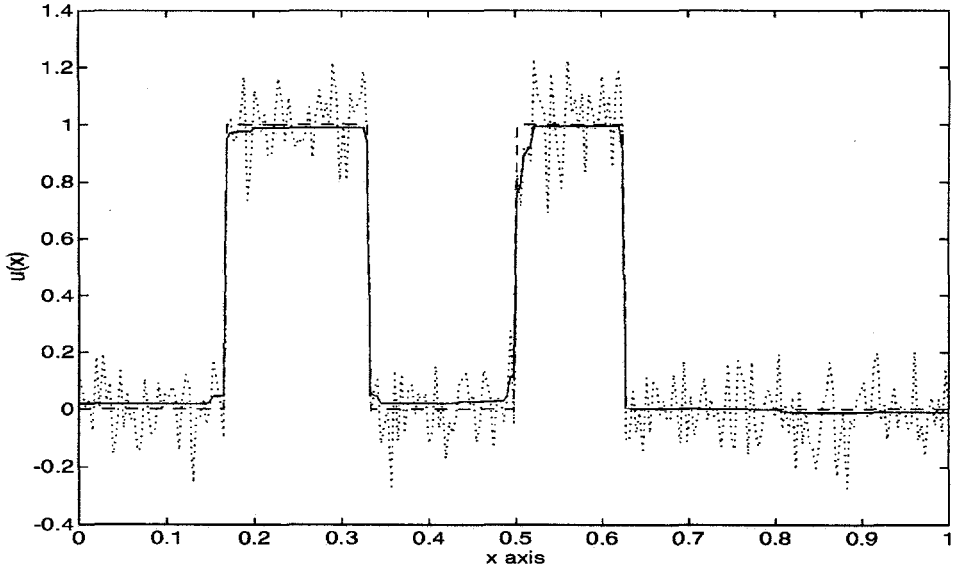


FIG. 1. *Denoised solution obtained using a smoothed TV penalty functional* $J_\beta(u) = \int_\Omega (|\nabla u|^2 + \beta)^{1/2}\, dx$. *Dashed line is exact solution; dotted line is noisy data; and solid line is denoised solution obtained using TV.*

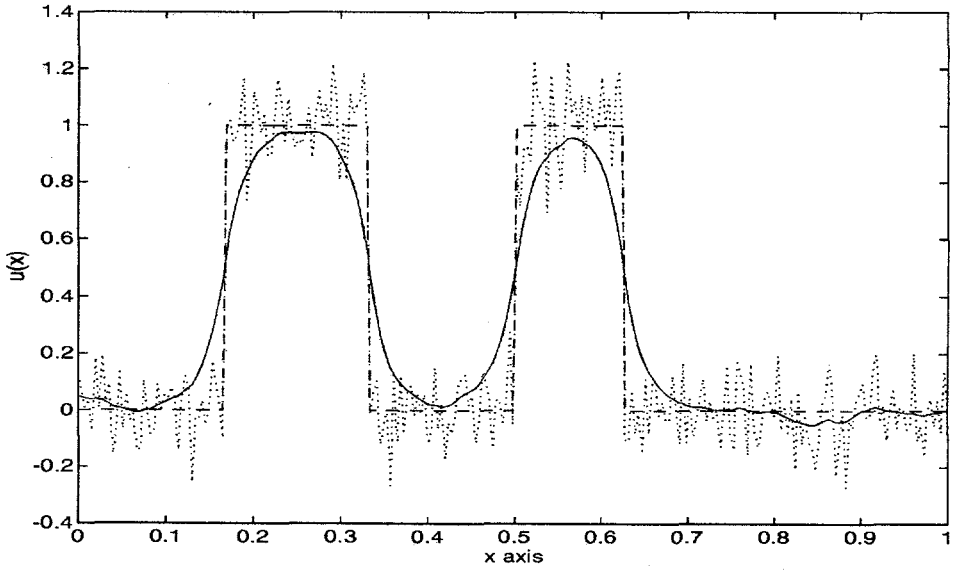


FIG. 2. *Solid line is denoised solution obtained using a quadratic derivative penalty functional* $J_\beta(u) = \int_\Omega |\nabla u|^2\, dx$.

GMRES AND INTEGRAL OPERATORS *

C. T. KELLEY† AND Z. Q. XUE†

Abstract. Many discretizations of integral equations and compact fixed point problems are collectively compact and strongly convergent in spaces of continuous functions. These properties not only lead to stable and convergent approximations but also can be used in the construction of fast multilevel algorithms. Recently the GMRES algorithm has become a standard coarse mesh solver. The purpose of this paper is to show how the special properties of integral operators and their approximations are reflected in the performance of the GMRES iteration and how these properties can be used to strengthen the norm in which convergence takes place.

We illustrate these ideas with composite Gauss rules for integral equations on the unit interval.

Key words. Multilevel methods, GMRES iteration, Pseudospectrum, Collective Compactness

AMS(MOS) subject classifications. 65F10, 65J10, 65R20,

1. Introduction. In this paper we consider the performance of the GMRES [9] iteration for linear equations of the form

$$Au = u - Ku = f \tag{1.1}$$

on a separable Hilbert space H. In (1.1) $K \in \mathcal{COM}(H)$ the space of compact operators on H. Throughout this paper we assume that A is a nonsingular linear operator on H. We consider convergence rate estimates of the form

$$\|r_k\|_H \leq \tau_k \|r_0\|_H \tag{1.2}$$

where the sequence of real numbers $\{\tau_k\}$ converges to zero and is independent of the right hand side f of (1.1). We also consider right hand side dependent estimates of the form

$$\|r_k\|_H \leq \tau(k, r_0). \tag{1.3}$$

The compactness of K implies that the GMRES iteration will converge superlinearly [6], [8]. These estimates are presented in both forms (1.2) and (1.3). One of our goals in this paper is to investigate how such convergence estimates are changed if the operator K is approximated.

Rates of convergence of the form (1.2) or (1.3) can be derived from resolvent integration [7], [8] for any K such that $I - K$ has bounded inverse and 1 is in the unbounded component of the spectrum of K. If K is compact more precise information can be obtained, in fact the GMRES iterates converge r-superlinearly to the solution in a way that is independent of the right hand side. This means that the sequence $\{\tau_k\}$ converges q-superlinearly to zero. In this case the sequence $\{\tau_k\}$ can be directly related to spectral properties of K.

* Extended abstract for the Colorado Conference on Iterative Methods. Submitted February 18, 1991.

† North Carolina State University, Center for Research in Scientific Computation and Department of Mathematics, Box 8205, Raleigh, N. C. 27695-8205, USA (Tim_Kelley@ncsu.edu, xue@math.ncsu.edu). This research was supported by National Science Foundation grant #DMS-9024622. Computing activity was also partially supported by an allocation of time from the North Carolina Supercomputing Center.

Our motivation is the case where K is an integral operator on $L^2(\Omega)$

$$Ku(x) = \int_\Omega k(x,y)u(y)\,dy \tag{1.4}$$

for a compact $\Omega \subset R^N$ and $k \in C(\Omega \times \Omega)$. Here $H = L^2(\Omega)$ and $K : H \to C(\Omega)$. In this case $K \in \mathcal{COM}(H)$ and $\mathcal{COM}(H, C(\Omega))$. Hence, if the right hand side f of (1.1) is continuous, then the solution u will also be continuous. It is natural, therefore, to try to solve (1.1) in the space of continuous functions. The GMRES iteration, however, is based on Hilbert space orthogonality and converges in the topology of L^2.

The typical discretizations based on quadrature [2] rules also raise questions. We will approximate integrals by quadrature rules

$$\int_\Omega f(x)\,dx \approx I_N(f) = \sum_{j=1}^N f(x_j^N)w_j^N$$

and assume that

$$\lim_{N\to\infty} I_N(f) = \int_\Omega f(x)\,dx$$

for all $f \in C(\Omega)$. We then approximate K by

$$K(u)(x) \approx K_N(u)(x) = \sum_{j=1}^N k(x, x_j^N)u(x_j^N)w_j^N.$$

It is known [1] that the sequence $\{K_N\}$ is strongly convergent (*i. e.* $K_N u \to Ku$) in $C(\Omega)$, but not norm convergent, and that the sequence is collectively compact. These two properties imply that $I - K_N$ has a bounded inverse in $C(\Omega)$ and that the solutions of $u - K_N u = f$ converge to the solution of (1.1).

The approximate equation

$$A_N u = u - K_N u = f \tag{1.5}$$

has a solution u^N for N sufficiently large. One can compute u^N by solving the finite dimensional system for $\bar{u} \in R^N$,

$$(\bar{A}_N \bar{u})_i = \bar{u}_i - \sum_{j=1}^N k(x_i^N, x_j^N)\bar{u}_j w_j^N = \bar{f}_i = f(x_i^N) \tag{1.6}$$

to recover $u^N(x_i^N) = \bar{u}_i$ for the values of u^N at the nodal points of the quadrature rule. Then

$$u^N(x) = f(x) + \sum_{j=1}^N k(x, x_j^N)\bar{u}_j w_j^N.$$

These considerations raise two questions. How is the sequence $\{\tau_k\}$ affected when (1.1) is replaced by (1.5)? We solve (1.1) in L^2 and (1.5) in R^N. Can we quantify the manner in which the convergence of the GMRES iteration for (1.1) governs that for (1.5)? The compactness properties of K and K_N are determined by the smoothness properties of the function k. Can the L^2 convergence of the GMRES iteration be replaced by convergence in the topology of $C(\Omega)$ (or an even stronger topology if the kernel k is sufficiently smooth) in an efficient way? The purpose of this paper is to provide a partial answer to the first question and to resolve the second.

2. Convergence Rate Estimates. If we let $\mathbf{P}_k$ denote the set of polynomials of degree k and denote the set of residual polynomials by

$$\mathcal{P}_k = \{p \in \mathbf{P}_k, p(0) = 1\}$$

it is well known [9] that

$$\|r_k\|_H \leq \|p(A)r_0\|_H$$

for all $p \in \mathcal{P}_k$ with equality holding for at least one $p_k \in \mathcal{P}_k$. This is an estimate of the form (1.3).

If we let $r_k^N = \bar{f} - \bar{A}_N \bar{u}_k$ be the residuals of the GMRES iteration for $\bar{A}_N \bar{u} = \bar{f}$ we have

$$\|r_k^N\|_{R^N} \leq \|p(\bar{A}_N)r_0^N\|_{R^N}$$

for all $p \in \mathcal{P}_k$. In particular, if $(\bar{u}_0^N)_i = u_0(x_i^N)$ we have by strong convergence that

$$\begin{aligned} \|r_k^N\|_{R^N} &\leq \|p_k(\bar{A}_N)r_0^N\|_{R^N} \\ &= \|p_k(A_N)r_0\|_{L^2} + o(1) \\ &= \|p_k(A)r_0\|_{L^2} + o(1). \end{aligned} \tag{2.1}$$

as $N \to \infty$. Hence the convergence of the GMRES iteration for the finite dimensional problem is governed by that of the infinite dimensional problem even though A_N and $\bar{A}_N$ are not defined on L^2. However, this is less than completely satisfactory in that the $o(1)$ term is dependent on L^∞ norms of u_0, f, and k, and moduli of continuity. It is also not clear how large N must be to capture the convergence rate at iteration k. While (2.1) may be all that one can expect in a general setting, one can obtain more precise results for special cases.

If the quadrature rule is a composite Gauss rule, it was shown in [5] that a norm convergent sequence of operators may be used in place of A_N. This fact was used to construct very efficient multilevel methods in that paper and can be used here to make more precise estimates than (2.1). We now turn to estimates of the form (1.2) and norm convergent sequences.

We will approximate K by operators that are near to K not only in the operator norm on H, but also in the Hilbert-Schmidt norm. This, in conjunction with results from [8], will immediately imply a rate estimate of the form (1.2) but with a sequence $\{\tau_k\}$ that is valid for all operators near K.

The central point here is that a Hilbert-Schmidt norm convergent sequence of finite-rank operator approximations to K can be constructed that give rise to the same finite dimensional systems as the traditional quadrature rule based strongly convergent sequence. Moreover, the GMRES iterates in the function space (with the L^2 inner product) and for the finite dimensional system (with a weighted R^N inner product) can be related by a simple unitary map. Hence the behavior of the GMRES iteration for the finite dimensional system is determined by that for the infinite dimensional problem.

As an example consider the composite midpoint rule. Here we take m subintervals $I_i^m = ((i-1)/m, i/m)$ for $1 \leq i \leq m$. The quadrature nodes are $x_i^m = (i-.5)/m$ and the weights are $h = 1/m$. If we approximate K by

$$\mathcal{K}_m u(x) = \int_0^1 k_m(x,y)u(y)\,dy \tag{2.2}$$

where k_m is the piecewise constant function on $[0,1] \times [0,1]$ defined for $x \in \text{Int}(I_i^m)$ and $y \in \text{Int}(I_j^m)$ by

$$k_m(x,y) = k(x_i^m, x_j^m) l_i^m(x) l_j^m(y). \tag{2.3}$$

then $\mathcal{K}_m \to K$ in the Hilbert-Schmidt norm. It is easy to see that the system

$$u - \mathcal{K}_m u = f \tag{2.4}$$

is nonsingular for m sufficiently large.

Let $\|\cdot\|_{HS}$ denote the Hilbert-Schmidt norm. If k is Lipschitz continuous, say, then $\|K - \mathcal{K}_m\|_{HS} = O(h)$ and we can use the methods of [8] to show that if $r_k^m = f - u_k + \mathcal{K}_m u_k$ are the residuals for the GMRES iteration for $u - \mathcal{K}_m u = f$ and (1.2) holds then

$$\|r_k^m\|_{L^2} \leq (\tau_k + O(h))\|r_0\|_{L^2}. \tag{2.5}$$

Let V^m be the space of piecewise constant functions on the intervals $\{I_i^m\}$. When restricted to V^m the system (2.4) is the same as the finite dimensional system (1.6) and therefore the iterations for the finite dimensional system also satisfy (2.5).

3. Convergence in a Stronger Norm. The results in this section require a more complex setting, which we now describe abstractly. Our setting is that of [4] where issues similar to those raised in this paper were considered in the context of Broyden's method [3] for linear and nonlinear equations. Let H be a real Hilbert space and let $X \subset H$ be a Banach space such that the inner product $(\cdot,\cdot)$ in H is continuous from $X \times X \to R$. This implies that there is C_X such that

$$\|u\|_H \leq C_X \|u\|_X \tag{3.1}$$

for all $u \in X$. Let $K \in \mathcal{COM}(H,X)$ the space of compact operators from H to X. Of course we may also regard K as an element of $\mathcal{COM}(H)$ the space of compact operators on H. In the context of the integral operator (1.4) discussed in § 1 $H = L^2(\Omega)$, $X = C(\Omega)$, and $C_X = \sqrt{m(\Omega)}$ where m is Lebesgue measure.

Having established a rate estimate (1.2) for the sequence of residuals, we show in this section that the GMRES iteration may be modified to produce a sequence that converges with the same rate in the norm of X.

PROPOSITION 3.1. *Let $\{u_k\}$ be the sequence of GMRES iterates. Assume that* (1.2) *holds for some sequence $\{\tau_k\}$. Let $\bar{u}_k = u_k + r_k$. Then*

$$\|\bar{u}_k - u^*\|_X \leq \|K\|_{\mathcal{L}(H,X)} \kappa_H(A) \tau_k \|u_0 - u^*\|_X. \tag{3.2}$$

Proof. First note that (1.2) implies that

$$\|u_k - u^*\|_H \leq \tau_k \kappa_H(A) \|u_0 - u^*\|_H. \tag{3.3}$$

Since

$$\bar{u}_k = u_k + r_k = f + K u_k,$$

continuity of K as a map from H to X implies that

$$\|\bar{u}_k - u^*\|_X = \|K(u_k - u^*)\|_X \leq \|K\|_{\mathcal{L}(H,X)} \|u_k - u^*\|_H.$$

This completes the proof. □

Note that $\bar{u}_k$ is as easy to compute as u_k upon exit from the main loop in GMRES. An algorithmic description of GMRES is:

ALGORITHM 3.1. *Algorithm* **gmres**(u, f, A, ϵ)

1. $r = f - Au$, $v_1 = r/\|r\|_2$, $\rho = \|r\|_2$, $\beta = \rho$, $k = 1$
2. *While* $\rho > \epsilon\|b\|_2$ *do*
   (a) $v_{k+1} = Av_k$
   for $j = 1, \ldots k$
   i. $h_{jk} = v_{k+1}^T v_j$
   ii. $v_{k+1} = v_{k+1} - h_{jk} v_j$
   (b) $h_{k+1,k} = \|v_{k+1}\|_2$
   (c) $v_{k+1} = v_{k+1}/\|v_{k+1}\|_2$
   (d) $e_1 = (1, 0, \ldots, 0)^T \in R^{k+1}$
   Minimize $\|\beta e_1 - H_k y\|_{R^k}$ *to obtain* y.
   (e) $\rho = \|\beta e_1 - H_k y\|_{R^k}$.
3. $u_k = u_0 + V_k y$.

We can use the fact that

$$r_k = f - Au_k = V_{k+1}(\beta e_1 + H_k y) = r_0 + V_{k+1} H_k y$$

to recover $\bar{u}_k$ with no additional operator-vector products involving A. In fact if $z = H_k y$ and we define a vector

$$w = (w_1, w_2, \ldots, w_{k+1})^T \in R^{k+1}$$

by $w_i = z_i + y_i$ for $1 \le i \le k$ and $w_{k+1} = z_{k+1}$, we have

$$\bar{u}_k = \bar{u}_0 + V_{k+1} w$$

which can simply replace the computation of u_k in step 3 of **gmres**.

REFERENCES

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

[2] K. E. ATKINSON, *A survey of numerical methods for Fredholm integral equations of the second kind*, SIAM, Philadelphia, 1976.

[3] C. G. BROYDEN, *A class of methods for solving simultaneous equations*, Math. Comp., 19 (1965), pp. 577–593.

[4] D. M. HWANG AND C. T. KELLEY, *Convergence of Broyden's method in Banach spaces*, SIAM J. on Optimization, 2 (1992), pp. 505–532.

[5] C. T. KELLEY, *A fast multilevel algorithm for integral equations.* SIAM Journal on Numerical Analysis, to appear.

[6] T. KERKHOVEN AND Y. SAAD, *On acceleration methods for coupled nonlinear elliptic systems*, Numerische Mathematik, 60 (1992), pp. 525–548.

[7] N. M. NACHTIGAL, S. C. REDDY, AND L. N. TREFETHEN, *How fast are nonsymmetric matrix iterations*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 778–795.

[8] O. NEVANLINNA, *Convergence of Iterations for Linear Equations*, Birkhäuser, Basel, 1993.

[9] Y. SAAD AND M. SCHULTZ, *GMRES a generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comp., 7 (1986), pp. 856–869.

ITERATIVE METHODS FOR DISTRIBUTED PARAMETER ESTIMATION IN PARABOLIC PDE

C. R. VOGEL* AND J. G. WADE†

The goal of the work presented is the development of effective iterative techniques for large-scale inverse or parameter estimation problems. In this extended abstract, a detailed description of the mathematical framework in which the authors view these problem is presented, followed by an outline of the ideas and algorithms developed.

1. Conceptual framework. Distributed parameter estimation problems often arise in mathematical modeling with partial differential equations. They can be viewed as inverse problems; the "forward problem" is that of using the fully specified model to predict the behavior of the system. The inverse or parameter estimation problem is: given the form of the model and some observed data from the system being modeled, determine the unknown parameter(s) in the model. These problems are of great practical and mathematical interest, and the development of efficient computational algorithms is an active area of study.

The estimation problem may be viewed mathematically as that of inverting the "forward" or "parameter to output" map $\mathcal{F} : Q \mapsto \mathcal{Z}$, where Q and $\mathcal{Z}$ are, respectively, the "parameter" and "observation" spaces. For a given $q \in Q$, the "output" of the model $\mathcal{F}(q)$ is compared to the data $\bar{\mathbf{z}} \in \mathcal{Z}$, and the goal is to find a suitable $q \in Q$ for which

$$\mathcal{F}(q) \approx \bar{\mathbf{z}}.$$

Generally $\mathcal{F}$ is composed of two maps, $\mathcal{F}(q) = \mathcal{C}\mathcal{S}(q)$. Given a $q \in Q$, the solution of the PDE comprising the model is given in terms of the "solution" operator $\mathcal{S}(q)$. Then, some partial information of the solution (to be compared with $\bar{\mathbf{z}}$) is extracted by application of the "observation operator" $\mathcal{C}$. Typically, $\mathcal{C}$ involves evaluation of traces or moments.

Since $\bar{\mathbf{z}}$ contains only partial knowledge of the system, the inversion is usually "ill-posed" the Hadamard sense. I.e., $\mathcal{F}^{-1}$ may not exist, may be one-to-many, or may be unbounded. Thus these problems share many features of other important problems such as those arising in image processing and first kind integral equations. A proper understanding of the ill-posedness is crucial for successful numerical treatment.

* Department of Mathematical Sciences, Montana State University, Bozeman MT 59717. Research was supported in part by the NSF under Grant DMS-9106609 and by the Center for Interfacial Microbial Process Engineering at Montana State University, an NSF-sponsored Engineering Research Center, and the Center's Industrial Associates.

† Department of Mathematics and Statistics, Bowling Green State University, Bowling Green, OH 43403-0221. Part of this work was carried out while the second author was a visitor at the Institute for Scientific Computation, Texas A&M University, College Station, TX 77843, and was supported in part by Department of Energy under contract #SK966-19.

A widly used approach, discussed below, is that of "output least-squares", in which the goal is to minimize

$$\Phi(q) \stackrel{\text{def}}{=} \frac{1}{2}\|\mathcal{F}(q) - \bar{\mathbf{z}}\|_Z^2. \tag{1.1}$$

To simplify the functional analysis, it is also assumed Q is a Hilbert space. One of the advantages of the least-squares approach is that it allows for incorporation of *regularization* techniques, such as that of Tikhonov, for attenuating the ill-posedness. Then (1.1) becomes

$$\Phi^{(\alpha)}(q) \stackrel{\text{def}}{=} \frac{1}{2}\|\mathcal{F}(q) - \bar{\mathbf{z}}\|_Z^2 + \frac{\alpha}{2}|q - q_0|_R^2 \tag{1.2}$$

for some $q_0 \in Q$, some $\alpha > 0$, and some seminorm $|\cdot|_R$. The parameter q_0 can embody any available prior knowledge of the system; this approach also has an interpretation in terms of Bayesian statistics [3]. The choices of α and the norm $|\cdot|_R$ affect the extent and nature of the regularization. A typical choice for $|\cdot|_R$ is the H^1 (Sobolev) seminorm.

This conceptual arrangement is fairly standard. It is substantially the same as that found in, for example, [1, 4, 6].

2. Example Problem. For focus, consider the following example. Let Ω be the unit square in $\mathbf{R}^2$, and consider the parameterized diffusion equation

$$\frac{\partial u}{\partial t}(t,x,y;q) = \nabla \cdot (q(x,y)\nabla(u(t,x,y;q))) \tag{2.3}$$

The initial conditions are $u(0,x,y;q) = 0$. The boundary conditions are specified in terms of the outward normal component of the flux $\Psi(t,x,y;q) \stackrel{\text{def}}{=} q(x,y)\nabla(u(t,x,y;q)$ and are as follows: along $y = 0$ and $y = 1$ no flux is allowed, along $x = 0$ the flux is specified by a function $b(t,y)$, and along $x = 1$ the flux is equal to $-ru(t,1,y;q)$ for some $r > 0$.

Let the observation operator $\mathcal{C}$ be the trace of u along Γ_{out} for $t \in [0,t_f]$, so that $[\mathcal{C}u](t,y) = u(t,1,y)$. The inverse problem in this example, then, is to determine $q(x,y)$ from knowledge of the response on Γ_{out} to a "signal" $b(t,\cdot)$ applied at Γ_{in}.

3. Minimization schemes. Many practical iterative optimization methods for least-squares problems fall broadly within the "quasi-Newton" class. Here, at each q_k in the iterations, a local quadratic model of $\Phi^{(\alpha)}$ is formed based the second order Taylor expansion. The Hessian is appoximated by, say, $H_k^{(\alpha)}$, which is constructed by various means. The methods which have this general structure include the Gauss-Newton (GN) method and its more robust variations such as the Levenberg-Marquardt method, as well as gradient-based secant methods such as BFGS.

For example, in the GN-based methods the Hessian is approximated by

$$H_k^{(\alpha)} \stackrel{\text{def}}{=} J^*(q_k)J(q_k) + \beta I, \tag{3.4}$$

where $\beta \geq \alpha$ and $J(q)$ is the Fréchet derivative operator, or Jacobian, of $\mathcal{F}$ at q. Thus these methods require that $J(q_k)$ or some approximation of it be computed on each iteration.

The "action" of this operator on a given $\delta q \in Q$ is

$$J(q_k)\delta q = \lim_{\tau \to \infty} \frac{1}{\tau} [\mathcal{F}(q_k + \tau \delta q) - \mathcal{F}(q_k)]. \tag{3.5}$$

It is usually possible to carry out this limit explicitly for a given problem. For the example described above, $J(q_k)\delta q = \mathcal{C}\delta u$ where $\delta u = \delta u(t, x, y)$ satisfies

$$\frac{\partial \delta u}{\partial t} = \nabla \cdot (q_k \nabla \delta u) + \nabla \cdot (\delta q \nabla u). \tag{3.6}$$

with homogeneous initial and boundary conditions.

4. Key properties of distributed parameter inverse problems. Generally, distributed parameter estimation problems are complicated by *(i)* the number of degrees of freedom, which is infinite in principle, and *(ii)* ill-conditioning, which can be severe.

Suppose that in the example above, q, u, y, b and $\bar{\mathbf{z}}$ are all appoximated on an $n \times n$ grid in space and with m levels in time. Then the approximation of $\mathcal{F}$ is a map from $\mathbf{R}^N$ to $\mathbf{R}^M$, where $N = n^2$ and $M = n \times m$, J is an $M \times N$ matrix, and the Hessian H is $N \times N$. The cost computing *one column* of J is the numerical solution of a PDE such as (3.6) (or (2.3) if finite differencing based on (3.5) is used). Moreover, it must be emphasized that J and H, though large, are generally *not sparse*. Hence, even the storage of the J and H can become a difficulty. For example, with $n = 32$, H is a 1024×1024 *full matrix*. For these reasons, inversion of $\mathcal{F}$ is an enormous computational problem even for values of n and m considered modest by numerical PDE standards.

Because of the ill-posedness of these problems, different criteria than are used for well-posed problems must be adopted by which to judge the success of methods. Rather than ask, "how accurately can we get the solution" (which, in general, is nonunique anyway), one should ask "how much information can we hope to obtain about q from the data, and how can we obtain it?" An extremely useful tool in this regard is the *singular value decomposition* (SVD) of J,

$$J = U\Sigma V^*.$$

Assume that the singular values $\{\sigma_j\}$ are arranged in decreasing order. The ill-posedness typical of distributed parameter inverse problems is reflected in that $\mathcal{F}$ and J are compact operators, so that $\sigma_j \to 0$ as $j \to \infty$. In fact this decay to zero often occurs at an *exponential* rate (see, e.g., [2, 4, 6]), in which case the problem may be called "severely ill-posed".

A key observation is this: although after discretization $J(q)$ is a large, full matrix, ill-posedness implies that it has a very low effective rank. Hence it can be approximated well

in terms of the dominant "few" singular values and vectors, by its "truncated singular value decompostion" (TSVD). For example, if after discretization, J is an $M \times N$ matrix, and if all but the largest L singular values are to be truncated, then

$$J \approx J_T = U_T \Sigma_T V_T^* \tag{4.7}$$

where $\Sigma_T = diag(\sigma_1, \ldots, \sigma_L)$ and U_T and V_T are matrices of size $M \times L$ and $N \times L$, respectively, defined in the obvious way. For severely ill-posed problems, $\sigma_{L+1}/\sigma_1 \ll 1$, and hence the approximation is accurate, for relatively small L, e.g., $L \ll N$.

Iterative methods for computing the TSVD in linear inverse problems were presented in [5]. They require the means by which to *apply* the operator to a given $\delta q \in Q$ as well as its adjoint J^* to a given $\delta \bar{z} \in \mathcal{Z}$. When $\mathcal{F}$ involve J involves solution of parameter-dependent PDE followed by pointwise or trace evaluation, this adjoint computation is non-trivial. These and other practical and theoretical issues will be discussed for these methods.

5. The contributions of this work. In [6], preliminary exploration of the the use of the TSVD for nonlinear problems was presented. A more complete development of these ideas is the goal of the present work. Specifically, modifications of quasi-Newton methods by the incorporation of the TSVD will be discussed. Practical and theoretical issues such as the choice of truncation level and its relationship to regularization and the computation and convergence of the adjoint J^* will be treated, with numerical results based on the example discussed above.

REFERENCES

[1] Banks, H. T. Kunisch, K., *Estimation Techniques for Distributed Parameter Systems*, Birkhäuser, 1989.

[2] Dobson, D.C., "Estimates on resolution and stabilization for the linearized inverse conductivity problem", Inverse Problems **8** (1992), p. 71–81.

[3] Fitzpatrick, B.G., "Bayesian analysis in inverse problems", Inverse Problems **7** (1991), p. 675–702.

[4] McCormick, S.F. and Wade, J.G., "Multigrid solution of linearized, regularized least-squares problems in electrical impedance tomography", Inverse Problems, to appear.

[5] Vogel, C.R. Wade, J.G., "Iterative SVD-based methods for ill-posed problems", special issue of SIAM Journal on Scientific and Statistical Computing for the 1992 Copper Mountain Conference on Iterative Methods, to appear.

[6] Vogel, C.R. Wade, J.G., "A modified Levenberg-Marquardt algorithm for large-scale inverse problems", submitted to the proceedings of the Conference on Computation and Control III, Bozeman, Montana, August, 1992.

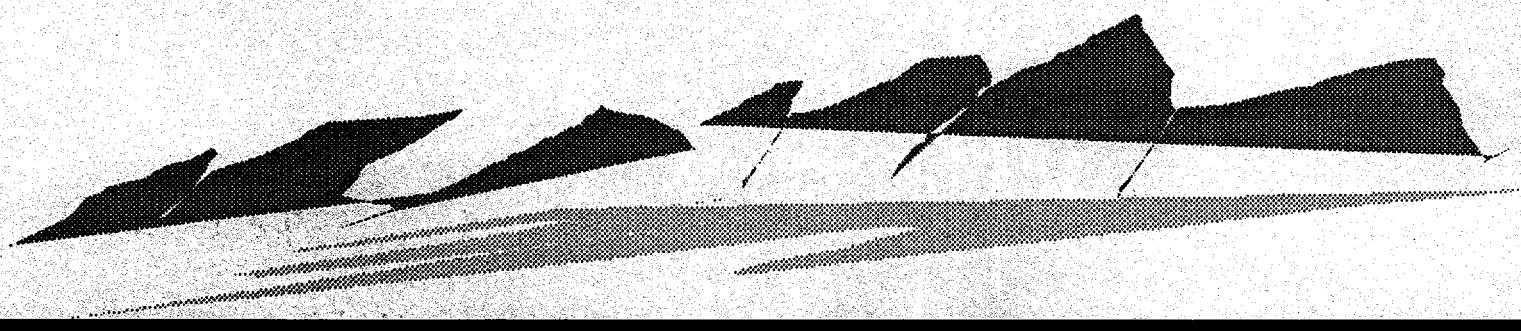
Tuesday, April 5

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



Tuesday, April 5

Eigenvalue Problems
Chair: Roland Freund
Room B

4:45 - 5:10 Clinton W. Jessup
Preconditioned Iterations to Calculate Extreme Eigenvalues

5:10 - 5:35 Andreas Stathopoulos
Overlapping Domain Decomposition Preconditioners for the Generalized Davidson Method for Eigenvalue Problems

5:35 - 6:00 Victor Pan
New Algorithms for the Symmetric Tridiagonal Eigenvalue Computation

Preconditioned Iterations to Calculate Extreme Eigenvalues

Clemens W. Brand Svetozara Petrova

Institut für Angewandte Mathematik
Montanuniversität Leoben
Franz-Josef-Str. 18
A-8700 Leoben, Austria

Common iterative algorithms to calculate a few extreme eigenvalues of a large, sparse matrix are Lanczos methods or power iterations. They converge at a rate proportional to the separation of the extreme eigenvalues from the rest of the spectrum. Appropriate preconditioning improves the separation of the eigenvalues. Davidson's method and its generalizations exploit this fact.

We examine a preconditioned iteration that resembles a truncated version of Davidson's method with a different preconditioning strategy.

Keywords. Davidson's method, eigenvalues, preconditioning, sparse matrices.

Overview. The matrices we are considering are discretizations of elliptic differential operators on large grids. They are sparse, symmetric and positive definite. In a typical application, a few of the eigenpairs at the lower end of the spectrum are sought. For matrices of this kind, the Lanczos algorithm constructs an orthonormal basis of a Krylov space. At each step, it adds another vector to the basis and uses essentially the Rayleigh-Ritz procedure to extract approximate eigenvectors from the Krylov space.

Davidson's method [3] also uses the Rayleigh-Ritz procedure, but enhances the subspace at each step by a vector $(D-\theta I)^{-1}(A-\theta I)y$, where D is the diagonal part of A; θ and y are the most recent approximations to the desired eigenvalue λ and eigenvector x. The subspace generated in this way is not a Krylov space, but as θ converges to λ, it asymptotically resembles a Krylov space generated by $(D-\lambda I)^{-1}(A-\lambda I)$. Frequently, the extreme eigenvalues of $(D-\lambda I)^{-1}(A-\lambda I)$ have a distribution more favorable than the eigenvalues of A, and the method then converges rapidly. $(D-\theta I)$ can be viewed as a preconditioner to $(A-\theta I)$. Generalizations of Davidson's method use more elaborate preconditioners to $(A-\theta I)$ [4, 5, 2].

For an $n \times n$-matrix A of the type we are considering, the Kaniel-Paige theory estimates $O(\sqrt{n})$ iterations for the Lanczos method to find an approximate eigenpair. Thus, a linear combination of $O(\sqrt{n})$ basis vectors will finally form the approximate eigenvector. Implementations of the Lanczos algorithm usually write these basis vectors to disc storage. However, for $n \approx 10^6$, even a basis of modest size (as compared to $\sqrt{n}$), say two hundred vectors, will occupy disc space in the gigabyte range. Davidson's method, too, has to store all basis vectors; but in addition, it has to orthogonalize explicitly each new vector with respect to all previously computed basis vectors. Therefore, both methods become inefficient as the dimension of the subspaces increases.

We investigate a preconditioned iteration that does not rely on high-dimensional subspaces. It uses the operator $\tilde{A}^{-1}(A - \theta I)$ to build up the subspace, where $\tilde{A}$ is a preconditioner to A (and not to $A - \theta I$, as in the usual version of Davidson's method; we note that some preconditioners we use would lead to unstable methods if they were applied to $A - \theta I$). For $\tilde{A} = A$ our method reduces essentially to inverse iteration, for $\tilde{A} = I$ it becomes a shifted power iteration (plus steepest descent optimization).

In its basic version, the method just builds up two-dimensional subspaces and restarts, i.e., it generates a new approximation for the eigenvector via a one-dimensional minimization of the Rayleigh quotient. Of course, higher-dimensional subspaces accelerate the convergence, but our numerical comparisons show that the increasing computational work and data traffic quickly balance against the higher rate of convergence.

We analyze the convergence of this method and prove that—with some assumptions on the matrix and the preconditioner—the method converges globally, for any initial approximate eigenpair (θ, y). For preconditionings based on modified incomplete factorization, the number of iterations required to reduce the initial error $\|Ay - \theta y\|$ by some fixed constant is of order $O(\sqrt{n})$ (the same as for the Lanczos method, but no high-dimensional subspaces are involved). For a large class of preconditioners we show that the speed of convergence depends on the condition number $\kappa(\tilde{A}^{-1}A)$. The general ideas in these proofs are: show that the only fixed points in the mapping associated with the iterative procedure are the exact eigenvectors of the original matrix; show that the eigenvector corresponding to the desired eigenvalue λ is the only point of attraction, while all other eigenvectors are repelling fixed points; estimate the spectral radius of the Jacobian to obtain bounds on the rate of convergence.

A block version of this method finds more than one eigenpair at either end of the spectrum. It is possible to insert as an inner iteration a low-dimensonal Lanczos process. In this variant, the outer iterations converge quadratically.

We present numerical experiments with various model problems and domains. For preconditioning we use DKR, MIC(p, q) (modified incomplete factorizations), INV(p) (block-incomplete factorizations) with small p and q (for the abbreviations see [1]), and hierarchical-basis multilevel preconditioners. We compare the number of iterations and the total computing time for these meth-

ods with a standard Lanczos Algorithm. We study the influence of subspace dimension, i.e., the restart frequency.

Our results generally indicate (especially for large problems, $n \approx 10^6$) that an efficient (multilevel) preconditioning and frequent restarts outperform methods that construct a large orthonormal basis (like the Lanczos method). For multilevel preconditioners, using polynomial acceleration with certain scaled and shifted Chebyshev polynomials at each level, the number of iterations (as is to be expected) does not depend on the size of the problem.

References

[1] P. Concus, G. H. Golub, and G. Meurant. Block preconditioning for the conjugate gradient method. *SIAM J. Sci. Stat. Comp.*, 6:220–252, 1985.

[2] M. Crouzeix, B. Philippe, and M. Sadkane. The Davidson Method. To appear in *SIAM J. Sci. Comp.*

[3] E. R. Davidson. The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real-symmetric matrices. *J. Comp. Phys.*,17:87–94, 1975.

[4] R. B. Morgan and D. S. Scott. Generalizations of Davidson's method for computing eigenvalues of sparse symmetric matrices. *SIAM J. Sci. Stat. Comp.*, 7(3):817–825, 1986.

[5] R. B. Morgan and D. S. Scott. Preconditioning the Lanczos algorithm for sparse symmetric eigenvalue problems. *SIAM J. Sci. Comp.*, 14(3):585–593, 1993.

Overlapping Domain Decomposition Preconditioners for the Generalized Davidson Method for the Eigenvalue Problem *

Andreas Stathopoulos ‡ Youcef Saad ‡ Charlotte. F. Fischer†

February 15, 1994

1 Introduction

The solution of the large, sparse, symmetric eigenvalue problem, $Ax = \lambda x$, is central to many scientific applications. Among many iterative methods that attempt to solve this problem, the Lanczos and the Generalized Davidson (GD) are the most widely used methods. The Lanczos method builds an orthogonal basis for the Krylov subspace, from which the required eigenvectors are approximated through a Rayleigh-Ritz procedure. Each Lanczos iteration is economical to compute but the number of iterations may grow significantly for difficult problems. The GD method can be considered a preconditioned version of Lanczos [4, 2]. In each step the Rayleigh-Ritz procedure is solved and explicit orthogonalization of the preconditioned residual $((M-\lambda I)^{-1}(A-\lambda I)x)$ is performed. Therefore, the GD method attempts to improve convergence and robustness at the expense of a more complicated step.

Variations of the Schwarz domain decomposition algorithms are extensively used for solving linear systems arising from Partial Differential Equations [3]. They solve restrictions of the problem on different subdomains independently, and then integrate the partial solutions. For this reason they have become popular on parallel computers. Solving the subdomain problems is expensive in large, sparse matrices and therefore Schwarz algorithms have been alternatively used as powerful preconditioners [3, 1]. Their power is further enchanced by the fact that they allow for domain overlapping. Increased domain overlapping as well as the use of a coarse grid solver compensates for the convergence deterioration with the number of subdomains. When no overlapping is used, the popular additive and multiplicative Schwarz algorithms are the usual block Jacobi and Gauss-Seidel ones.

Schwarz domain decomposition algorithms can be used to precondition the algebraic eigenvalue problem as well. The additive and multiplicative Schwarz methods have been applied on the GD method for solving the equation $(M - \lambda I)\delta = (A - \lambda I)x$ in a multicomputer environment. However, in this environment the subdomain overlapping causes some eigenvalue-specific problems. This paper presents the changes that are necessary to solve these problems. Some preliminary experimental results are also given.

*This work was supported by National Science Foundation under grant numbers ASC-9005687 and DMR-9217287, and by AHPCRC (University of Minnesota) under Army Research Office grant number DAAL03-89-C-0038.

†Computer Science Department, Vanderbilt University, Nashville, TN.

‡Computer Science Department, University of Minnesota

2 Problems from Subdomain Overlapping on GD

Assuming that the smallest eigenpair of $Ax = \lambda x$ is sought and B is an initial basis of an approximating subspace, a conceptual description of the GD algorithm follows[7, 6]:

GD Algorithm

Step 1. Compute projection $S = B^T AB$.
Step 2. Solve $Sc = \lambda c$
Step 3. Compute the residual $R = (ABc - \lambda Bc)$.
Step 4. Precondition: $R' = (M - \lambda I)^{-1} R$.
Step 5. Orthogonalize: $b' = (I - BB^T)R'$.
Step 6. Add normalized vector $b'/\|b'\|$ to B and repeat from 1.

The distribution of the algebraic problem onto the multicomputer is similar to the one in Sobolev spaces. Each processor holds a number of rows (nodes) of the matrix and the corresponding components of the eigenvectors and work arrays. Usually, a subdomain is associated with the rows on each processor and overlapping subdomains hold a number of the same rows.

On multicomputers, subdomain overlapping alters the dot product. The reason for this is that the overlapped regions of vectors contribute to the dot-product more than once. More specifically, if D is the overlap diagonal matrix, where $D_{i,i}$ is the number of processors on which row i appears, for any distributed vectors x, y the altered product $(.,.)_{ovl}$ is given by:

$$(x, y)_{ovl} = (x, y)_D = (Dx, y),$$

In linear systems of equations this is not a problem by itself, since the Galerkin condition does not depend on the dot-product. However, the Rayleigh-Ritz procedure is sensitive to the dot-product:

$$x = Bc,\ (Ax - \lambda x, B)_D = 0 \ \Rightarrow\ B^T DABc - \lambda B^T DBc = 0.$$

A second problem is that the $(.,.)_{ovl}$ dot-product describes the correct change only when the overlapping regions are the same in different processors. However, preconditioning operates on different subdomains with different sections of A, and the resulting overlapped regions in different processors do not coincide. Thus, the dot-product does not have a succinct formula as the above one and simple scaling cannot be used.

3 Proposed Solutions

• The problem of non coinciding overlapping regions in different processors, needs to be dealt with first, because otherwise the effect of the dot-product is not easily recordable. The problem can be faced by introducing a communication step after preconditioning. The overlapped regions of the preconditioned vector are communicated and weighed (averaged by D^{-1}) over all the processors. Since the size of overlaps is usually much smaller than the dimension of A this should not present a bottleneck in the algorithm. After this step, all overlapped vector components have identical values on the processors and the dot-product can be safely assumed to be $(.,.)_{ovl}$.

• There are two ways to deal with the change in the dot-product. The simplest (and most intuitive) one is to correct the effect of the dot-product by scaling one of the vectors back with D^{-1}. The dot-product defined by $(x, D^{-1}y)_{ovl} = (x, DD^{-1}y) = (x, y)$ is equal to the correct (non-overlapping) one. This obvious solution has two disadvantages: the GD library routine must be modified accordingly for the new dot-product, and requires $k+2$ scalings in each iteration, where k is the size of the basis.

• A different approach to the same problem is to consider the scaled matrix $A_s = D^{-\frac{1}{2}} A D^{\frac{1}{2}}$ and the scaled initial basis $B_s = D^{-\frac{1}{2}} B$. If the GD algorithm is applied to these scaled matrices with $(.,.)_{ovl}$ dot-product and with the above averaging after preconditioning, it gives the correct eigenvalue and the scaled eigenvector. To show this claim the following are necessary.

Let the rows of the matrix A span $\Re^N$. Let also the rows of the matrix local to subdomain i span the subspace L_i. If I_i is the orthocanonical basis of L_i, $P = I_i I_i^T$ is an orthogonal projector onto L_i. The *section* of A on L_i is defined as: $A_i = PAP = I_i I_i^T A I_i I_i^T$. Although A_i is not invertible, its restriction to L_i can be inverted. Let A_i^{-1} be defined as:

$$A_i^{-1} \equiv I_i (I_i^T A I_i)^{-1} I_i^T.$$

With this definition, the algebraic formulation of the additive and multiplicative Schwarz preconditioners is:

$$M^{-1} = A_1^{-1} + \ldots + A_q^{-1} \qquad \text{(additive)},$$
$$M^{-1} = (I - (I - A_1^{-1}A) \cdots (I - A_q^{-1}A))A^{-1} \qquad \text{(multiplicative)}.$$

Lemma 1 *The inverted sections of A_s are the scaled inverted sections of A, i.e.,*

$$(A_s)_i^{-1} = D^{-\frac{1}{2}} A_i^{-1} D^{\frac{1}{2}}.$$

Proof.

$$\begin{aligned}
(A_s)_i^{-1} &= I_i (I_i^T A_s I_i)^{-1} I_i^T \\
&= I_i (I_i^T D^{-\frac{1}{2}} A D^{\frac{1}{2}} I_i)^{-1} I_i^T \\
&= I_i (I_i^T I_i I_i^T D^{-\frac{1}{2}} A D^{\frac{1}{2}} I_i I_i^T I_i)^{-1} I_i^T \\
&= I_i ((I_i^T D^{-\frac{1}{2}} I_i)(I_i^T A I_i)(I_i^T D^{\frac{1}{2}} I_i))^{-1} I_i \\
&= I_i (I_i^T D^{\frac{1}{2}} I_i)^{-1} (I_i^T A I_i)^{-1} (I_i^T D^{-\frac{1}{2}} I_i)^{-1} I_i^T \qquad (1) \\
&= I_i I_i^T D^{-\frac{1}{2}} I_i (I_i^T A I_i)^{-1} I_i^T D^{\frac{1}{2}} I_i I_i^T \qquad (2) \\
&= D^{-\frac{1}{2}} I_i (I_i^T A I_i)^{-1} I_i^T D^{\frac{1}{2}} \\
&= D^{-\frac{1}{2}} A_i^{-1} D^{\frac{1}{2}}.
\end{aligned}$$

The transition from eq. (1) to eq. (2) is justified because D is a diagonal matrix and I_i part of the identity matrix. □

As a consequence of the above Lemma, the additive and multiplicative Schwarz preconditioners derived from the scaled A_s are simply the scaled preconditioners of A.

Proposition 1 *If M_s is the additive or multiplicative Schwarz preconditioner of A_s, and M is the corresponding preconditioner of A, then*

$$M_s^{-1} = D^{-\frac{1}{2}} M^{-1} D^{\frac{1}{2}}.$$

Proof. For the additive case and from Lemma 1:
$M_s^{-1} = \sum_{i=1}^q (A_s)_i^{-1} = \sum_{i=1}^q D^{-\frac{1}{2}} A_i^{-1} D^{\frac{1}{2}} = D^{-\frac{1}{2}} (\sum_{i=1}^q A_i^{-1}) D^{\frac{1}{2}} = D^{-\frac{1}{2}} M^{-1} D^{\frac{1}{2}}$.
For the multiplicative case:
$M_s^{-1} = (I - \prod_{i=1}^q (I - (A_s)_i^{-1} A_s)) A_s^{-1} = (I - \prod_{i=1}^q (I - D^{-\frac{1}{2}} A_i^{-1} A D^{\frac{1}{2}})) D^{-\frac{1}{2}} A^{-1} D^{\frac{1}{2}} =$
$= D^{-\frac{1}{2}} (I - \prod_{i=1}^q (I - A_i^{-1} A)) A^{-1} D^{\frac{1}{2}} = D^{-\frac{1}{2}} M^{-1} D^{\frac{1}{2}}$. □

With the above results it is easy to show that each step of the GD procedure, applied to A_s with starting basis B_s, computes the correct eigenvalues of A and their corresponding eigenvectors scaled by $D^{-\frac{1}{2}}$. Assume $\|B\| = 1$ for the original (not the overlapped) norm.

Step 1 The computed projection matrix is $S = (B_s, A_s B_s)_{ovl} = B^T D^{-\frac{1}{2}} D D^{-\frac{1}{2}} A D^{\frac{1}{2}} D^{-\frac{1}{2}} B = = B^T A B$, i.e., the same with the correct projection of the unscaled matrix.

Step 2 This depends only on S and therefore is not affected.

Step 3 The residual $R_s = (A_s - \lambda I) B_s c = D^{-\frac{1}{2}} R$, where R is the residual of the unscaled method.

Step 4 According to Proposition 1, $R'_s = M_s^{-1} R_s = D^{-\frac{1}{2}} M^{-1} D^{\frac{1}{2}} D^{-\frac{1}{2}} R = D^{-\frac{1}{2}} R'$, where R' is the preconditioned residual of the unscaled method.

Step 4' Both the scaled and unscaled versions, introduce different values on the overlapping regions of R' and R'_s. Thus, they both require a global summation and scaling: $R'_s \leftarrow D^{-1}(\sum_{j=1}^{q} R_s'^{(j)})$, $R' \leftarrow D^{-1}(\sum_{j=1}^{q} R'^{(j)})$, and the relation $R'_s = D^{-\frac{1}{2}} R'$ still holds.

Step 5 The orthogonalized vector $b'_s = R'_s - B_s (B_s, R'_s)_{ovl} = D^{-\frac{1}{2}} (R' - B B^T R') = D^{-\frac{1}{2}} b'$, where b' is the correctly orthogonalized vector of the unscaled version.

Step 6 The computed norm of b'_s is $\|b'_s\|^2 = (b'_s, b'_s)_{ovl} = b'^t D^{-\frac{1}{2}} D D^{\frac{1}{2}} b' = b'^T b'$. Therefore the norm computed is the correct norm of the new basis vector b' of the unscaled case. The algorithm can now repeat from Step 1.

Several applications scale the matrix A before the GD is applied. In these cases the scaling $D^{-\frac{1}{2}} A D^{\frac{1}{2}}$ can be done at no additional cost. M need not be explicitly scaled since it is extracted from A. Scaling the matrix A may also be beneficial in cases of sparse matrices with large overlap where the GD requires many iterations.

Another way of exploiting the above alternative algorithm is not to scale the matrix A but to record the effects of the scaled one in the matrix-vector multiplication and preconditioning routines. Since $A_s = D^{-\frac{1}{2}} A D^{\frac{1}{2}}$ and $M_s^{-1} = D^{-\frac{1}{2}} M^{-1} D^{\frac{1}{2}}$, an operation of any of these two matrices (say F) to a vector from the above algorithm can be formed as $D^{-\frac{1}{2}}(F(D^{\frac{1}{2}} b)))$. This is equivalent to scaling the vector by $D^{\frac{1}{2}}$ before the operation with F and by $D^{-\frac{1}{2}}$ after the operation, i.e., two scalings per operation. Moreover, the scaling after the preconditioning (by $D^{-\frac{1}{2}}$) can be combined with the scaling from averaging (by D^{-1}) into one scaling by $D^{-\frac{3}{2}}$. Thus, this approach costs 3 scalings per GD iteration and is much cheaper than the initial approach.

4 Preliminary Results and Conclusions

The use of overlapping domains in additive and multiplicative Schwarz preconditioners for the GD is demonstrated with some preliminary results on Tables 1 and 2. The codes used are described in [1, 5, 7], and the experiments are carried out on a PVM 4-node multiprocessor. No underlying grid is considered and the partitioning of the matrix is performed through automatic domain decomposition tools. The subdomain problems are solved with ILU(0).

On both tables the overlap is denoted by two numbers; the number of breadth first search levels that a domain is expanded, and the maximum number of additional nodes per level allowed. The number of GD iterations is reported for the additive (Add) and multiplicative (Mult) preconditioners. On Table 1, the matrix used is BCSSTK07 from the Harwell-Boeing collection. Its dimension is 420 and it has poor eigenvalue separation for the lowest part of the spectrum. The lowest eigenpair is sought for. Two steps of the corresponding Schwarz method are applied in each iteration. For the Add method the overlap (10,2) improves the convergence significantly, while (8,6) takes more than three times the steps. Improvements for the Mult method are more consistent with larger overlaps.

On Table 2, the matrix used comes from atomic structure calculations [7]. Its dimension is 748 and it has fairly good eigenvalue separation for the lower spectrum. The third lowest eigenpair is sought for. Two steps of the additive preconditioner (five for the multiplicative) are

Overlap Levels	0	2	10	4	8
Ovl / level	0	2	2	2	6
Iterations	454	402	360	>700	>1000

Overlap Levels	0	4	10	4	8
Ovl / level	0	2	2	6	6
Iterations	233	258	189	153	154

Table 1: BCSSTK07: Iterations for Additive (left) and Multiplicative (right) preconditioned GD

applied in each iteration. Similarly to the previous case, better and more consistent convergence improvements are observed for the multiplicative algorithm.

Overlap Levels	0	4	10	8	10
Ovl / level	0	2	2	6	6
Iterations	31	31	32	28	28

Overlap Levels	0	4	10	8	10
Ovl / level	0	2	2	6	6
Iterations	25	24	21	21	24

Table 2: Matrix 748: Iterations for Additive (left) and Multiplicative (right) preconditioned GD

These preliminary experiments show that similarly to algebraic linear systems an efficient partition of the algebraic eigenvalue problem requires knowledge of the underlying structure or the use of automatic domain decomposition tools.

Acknowledgement: The authors would like to thank Todd Goehring for the partitioning routines.

References

[1] Xiao-Chuan Cai and Y. Saad, *Overlapping Domain Decomposition Algorithms for General Sparse Matrices,* Preprint 93-027, Army High Performance Computing Research Center, University of Minnesota, 1993.

[2] E.R. Davidson, *The Iterative Calculation of a Few of the Lowest Eigenvalues and Corresponding Eigenvectors of Large Real-Symmetric Matrices*, J. Comput. Phys. 17 (1975) 87.

[3] M. Dryja, and O.B. Widlund, *Towards a unified theory of domain decomposition algorithms for elliptic problems,* in Third International Symposium on Domain Decomposition Methods for PDEs, eds. T.F.Chan et al. (SIAM, Philadelphia 1990).

[4] R.B. Morgan and D.S. Scott, *Generalizations of Davidson's Method for Computing Eigenvalues of Sparse Symmetric Matrices*, SIAM J. Sci. Stat. Comput. 7 (1986) 817.

[5] Y. Saad, *Krylov Subspace Methods in Distributed Computing Environments*, Preprint 92-126, Army High Performance Computing Research Center, University of Minnesota, 1992.

[6] A. Stathopoulos and C. F. Fischer, *Reducing Synchronization on the Parallel Davidson Method for the Large, Sparse, Eigenvalue Problem*, in: Proceed. Supercomputing '93, pp. 172 (ACM Press, Portland 1993).

[7] A. Stathopoulos and C.F. Fischer, *A Davidson Program for Finding a Few Selected Extreme Eigenpairs of a Large, Sparse, Real, Symmetric Matrix*, Comput. Phys. Commun., 79 (1994) 1.

New Algorithms for the Symmetric Tridiagonal Eigenvalue Computation

Victor Pan

Mathematics and Computer Science Department

Lehman College, City University of New York

Bronx, NY 10468

and International Computer Science Institute

Suite 600, 1947 Street

Berkeley, CA 94704

(Supported by NSF Grant CCR 9020690 and

by CUNY University Award #662478)

Summary. We present new algorithms that accelerate the bisection method for the symmetric eigenvalue problem. The algorithms rely on some new techniques, which include acceleration of Newton's iteration and can also be further applied to acceleration of some other iterative processes, in particular, of iterative algorithms for approximating polynomial zeros.

Key words: symmetric eigenvalue problem, bisection algorithm, Newton's iteration, convergence acceleration, polynomial zeros.

1991 Mathematics subject classification: 65F15, 65Y20, 65B99.

1. Introduction.

In this paper, we propose some new techniques in order to accelerate the convergence of the bisection method for the eigenvalues of a real symmetric tridiagonal (hereafter rst) matrix A (compare [Par], [GL], [B], [LPS], [PR]).

We recall (see our section 3) that the bisection algorithm approximates all the eigenvalues $\lambda_1, \ldots, \lambda_n$, $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$, of an $n \times n$ rst matrix A within an error bound t, $0 \leq t \leq \lambda_1 - \lambda_n$, by using

$$4n^2 \lceil H(t) \rceil + O(n) \tag{1.1}$$

arithmetic operations where

$$H(t) = \log_2((\lambda_1 - \lambda_n)/t) \, . \tag{1.2}$$

Our new algorithms involve

$$(4n^2 + O(n)) \left[(\log_2 H)^2 + \log_2 n + \nu \log_2 H \right] \tag{1.3}$$

arithmetic operations where ν varies from 4 to $5.5/\log_2 3 \approx 3.47$. (Note the decrease of the complexity bound from $O(H(t))$ in (1.1) to $O(\log H(t))$ in (1.3); see the derivation of the estimate (1.3) and some further comments in section 10 and, for a further improvement, see section 5 and table 10.6.) Furthermore, some features of our algorithms suggest that (unlike the bisection method) they tend to perform substantially better on the average input than the estimate (1.3) indicates.

The techniques used may be of some independent interest. In particular, by employing Newton's iteration for a k-fold zero of a polynomial, our algorithm 6.1 ensures a nearly quadratic convergence, *right from the start*, to single and multiple eigenvalues of A or to their clusters, as soon as each of these eigenvalues or, respectively, of their clusters is sufficiently well isolated from the other eigenvalues of A. The desired isolation is ensured by using the bisection procedure and the double exponential sieve algorithm of [BOT] with its new improvement (see some alternative techniques in [P87], [P89a], [PD]). And we show in

section 9 a simple but novel extension of Newton's iteration techniques, leading to a nearly cubic convergence rate, right from the start. This makes our techniques potentially useful for the study and design of other iterative algorithms. Actually, in this paper, we develop the earlier approach of [P87], originally applied to approximating complex polynomial zeros and based on Weyl's construction; our present techniques can in turn be easily extended in order to improve this approach of [P87] to the latter problem. This direction seems to be even more promising from the application point of view, because the polynomial zeros are usually sought with a much higher precision than the eigenvalues of a symmetric matrix. Another promising extension of this work is apparently to improving the known divide-and-conquer algorithms for the symmetric tridiagonal eigenvalue problem (see remark 4.1 in section 4).

Our paper is organized as follows: We recall fast methods for the evaluation of the characteristic polynomial of A and of its derivatives, in section 2, and the bisection algorithm, in section 3. In section 4, we present the structure of our main algorithm, give its informal description and also define the two basic concepts, of separation and isolation. In section 5, we recall the double exponential sieve process of [BOT] for the eigenvalue isolation. In section 6 we describe our algorithm 6.1 for approximating the well-isolated (clusters of) eigenvalues of A. Rapid convergence of this algorithm is proved in section 7, and the algorithm is further accelerated in section 9. In sections 8–10, we summarize our study by presenting (in sections 8 and 9) our two algorithms for approximating the eigenvalues of an rst matrix and (in section 10) their computational complexity estimates, shown both in formal expressions and in tables (for two samples of specific problem sizes).

2. Definitions and auxiliary results.

Hereafter, A denotes the $n \times n$ rst matrix, α_i denotes its entry (i,i), β_j denotes its entries $(j-1,j)$ and $(j,j-1)$; $p_i(x) = \det(xI_i - A_i)$ denotes the characteristic polynomial

of the $i \times i$ leading principal submatrix A_i of A,

$$A_n = A\ , \quad p_n(x) = p(x) = \det(xI_n - A)\ ,$$

$$p_0(x) = 1\ , \qquad p_1(x) = x - \alpha_1\ ,$$

$$p_j(x) = (x - \alpha_j)\, p_{j-1}(x) - \beta_{j-1}^2 p_{j-2}(x)\ , \tag{2.1}$$

where $i = 1, \ldots, n$; $j = 2, \ldots, n$. Due to (2.1), it suffices to use $2n - 3$ multiplications and $2n - 1$ additions to evaluate the sequence $p_1(x), \ldots, p_n(x)$ for any fixed x (assuming that the values β_h^2 have been precomputed for all h, in $n - 1$ multiplications). The number of sign agreements in this sequence equals the number $n_-(x)$ of the eigenvalues of A that are less than x ([GL, p. 438], [Par, p. 131]).

Furthermore, $4n - 8$ multiplications and $4n - 5$ additions suffice to evaluate both $p(x)$ and its derivative $p'(x)$ ([BP]). Indeed, introduce an auxiliary variable z, replace $p_h(x)$, for $h = j, j-1, j-2$, by $p_h(x+z) \bmod z^2 = p_h(x) + z\, p_h'(x)$ in (2.1), and recursively compute $p_h(x)$ and $p_h'(x)$ from the resulting expressions for $p_h(x + z) \bmod z^2$. Likewise, we may replace $p_h(x)$ for $h = j, j-1, j-2$ in (2.1) by $p_h(x + z) \bmod z^3$ and then recover $p(x)$, $p'(x)$ and $p''(x)$ from $p_h(x + z) \bmod z^3 = p_h(x) + zp_h'(x) + z^2 p_h''(x)/2$.

Remark 2.1. The above approach can be extended to the evaluation of higher order derivatives of the polynomial $p(x)$ and to computing its coefficients.

We will measure the computational complexity by the number of the evaluations of $n_-(x)$, $p(x)$ and $p'(x)$ and will deal with real semi-open intervals of the form

$$[a, b) = \{x,\ a \le x < b\}\ .$$

Hereafter, all logarithms are to the base 2, and $t > 0$ denotes the tolerance to the absolute errors of the output approximations to $\lambda_1, \ldots, \lambda_n$, the eigenvalues of A, where

$$\lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_n\ .$$

3. Bisection algorithm.

Next, we will recall the bisection algorithm for approximating the eigenvalues of A (compare [B], [PR], [LPS]).

Algorithm 3.1.

Input: an $n \times n$ rst matrix A, two real numbers λ_{n+1} and λ_0, such that $\lambda_{n+1} < \lambda_n \leq \lambda_1 < \lambda_0$, and an error tolerance $t > 0$.

Output: approximations (within the absolute error bound t) to the n eigenvalues of A in the interval $[\lambda_{n+1}, \lambda_0)$.

Initialize: call the interval $[\lambda_{n+1}, \lambda_0)$ *suspect.*

Recursive step: for every suspect interval $[q, r)$ such that $r - q > 2t$, compute $n_-(\frac{q+r}{2})$; call the subinterval $[\frac{q+r}{2}, r)$ *suspect* if $n_-(r) > n_-((q+r)/2)$; call the subinterval $[q, \frac{q+r}{2})$ *suspect* if $n_-(\frac{q+r}{2}) > n_-(q)$; remove the label "suspect" for the interval $[q, r)$.

Stopping criterion: end the computation when all the suspect intervals have length at most $2t$; for each of these intervals, output its midpoint and the number of the eigenvalues of A lying in it.

To estimate the computational cost of the bisection algorithm, note that in each its recursive step, there is at least one eigenvalue in each suspect interval, and thus, there are at most n suspect intervals at each recursive step.

On the other hand, in step s, each suspect interval has length $(\lambda_0 - \lambda_{n+1})/2^s$, so that $S = \lceil \log(\frac{\lambda_0 - \lambda_{n+1}}{t}) \rceil - 1$ steps suffice to arrive at the intervals of length at most $2t$, at which point we output the soluiton. Each s-th of these S steps requires $k(s) \leq n$ evaluations of $n_-(x)$, at the midpoints of each of the $k(s) \leq n$ suspect intervals. This leads to (1.1) since the extremal eigenvalues λ_1 and λ_n of an rst matrix can be easily approximated ([GL], [P90]), so that we may assume, say, that $\lambda_{n+1} - \lambda_0 \leq 2(\lambda_1 - \lambda_n)$.

4. Separation and isolation.

Our objective is to modify the bisection algorithm to accelerate its convergence. We

will still rely on the *recursive partitioning* of the input interval $[\lambda_{n+1}, \lambda_0)$ by its interior points, but not necessarily by the midpoints of suspect intervals. One of our two major tools will be Newton's iteration algorithm, which is well known to have local quadratic convergence. In fact, we will show (in sections 7 and 9) its superlinear (nearly quadratic or nearly cubic) convergence right from the start, provided that we start with an approximation to a single eigenvalue (or to a cluster of eigenvalues) sufficiently well isolated from all the other eigenvalues of A. We will quantitatively specify the term "sufficiently well" by using the concept of an ***isolation ratio*** ([P87], [P89]), abbreviated as ir. For an interval $J = [m - \ell, m + \ell)$, its isolation ratio, $\mathrm{ir}(J)$, equals H/ℓ, H being the distance from m to the nearest eigenvalue of A not lying in J itself, so that $H \geq \ell$, $\mathrm{ir}(J) \geq 1$.

Now, if we have a cluster of the eigenvalues of A in a fixed real interval J, containing no other eigenvalues of A and having its isolation ratio at least $8n$, then we will prove (near) quadratic convergence of Newton's iteration algorithm 6.1 of section 6 to this cluster, right from the start. Furthermore, our modified algorithm 9.1 of section 9 reaches nearly cubic convergence, right from the start, if $ir(J) \geq 2 + 4\sqrt{n}$. Moreover, the same algorithms remain effective under the above bound on $ir(J)$, no matter whether all the eigenvalues of A in J form a single cluster or not. In the latter case, the algorithms converge (with the same speed) not to a common approximation point for all the eigenvalues of A in J (such a point cannot exist in this case), but to a *splitting point* defined as follows:

A partition of an interval $[a, b)$ by its internal point x is called a ***separation step*** if

$$n_-(a) < n_-(x) < n_-(b) \ , \tag{4.1}$$

and then x is called a *splitting point.*

Clearly, there can be at most $n-1$ separation steps of the bisection or of any recursive partition algorithm. Due to this bound $n - 1$, we just need to devise a recursive partition algorithm whose every sufficiently long sequence of successive recursive steps, including no separation steps, rapidly converges to the eigenvalues of A. We will indeed show such an

algorithm in the next sections (where "sufficiently long" will be interpreted as the order of $\log n$), and in fact, the only remaining problem is to ensure high isolation ratios of the initial approximation intervals. Indeed, if the isolation ratios are high enough, already the Newton iteration algorithms rapidly converge either to an approximation point or to a splitting point. Since there can be at most $n-1$ separation steps, it follows that at most $2n-1$ (in the worst case!) applications of these algorithms will give us the desired approximations to the n eigenvalues of A.

Let us next turn to the problem of isolation. Looking for the increase of the isolation ratio of the original input interval, we will employ, in particular, the following simple result:

Fact 4.1. *Let $\mathrm{ir}(J) = 1+u$ for a suspect interval J (see algorithm 3.1). Then the bisection of J either outputs two suspect intervals of half length (and then the bisection is a separation step) or else defines a single suspect subinterval J^* of J (of half length) such that*

$$\mathrm{ir}(J^*) \geq 1+2u\ .$$

If h bisection steps have been successively applied to the interval J of fact 4.1 such that $\mathrm{ir}(J) \geq 1+u$ and if neither of them turned out to be a separation step, then their output suspect subinterval of J has length $|J|/2^h$ and has an isolation ratio of at least $1+2^h u$.

In the next section we will recall an algorithm from [BOT] that, for any interval J of fact 4.1, rapidly computes either a splitting point in J or a subinterval J_0 of J such that all the eigenvalues of A in J lie in J_0 and $\mathrm{ir}(J_0) \geq 3$, in which case $u \geq 2$, $1+2^h u \geq 1+2^{h+1}$. We will also show a further improvement of this algorithm.

Summarizing, our approach has the following structure (compare section 8).

1) apply improved algorithm of [BOT], to ensure (after sufficiently many separation steps) an isolation ratio of at least $1+2v$ for a fixed $v \geq 1$ ($v=1$ in the version of [BOT]), for every eigenvalue or every cluster of the eigenvalues of A;

2) apply the bisection algorithm to increase the isolation ratios to at least $8n$ or to at

least $2 + 4\sqrt{n}$;

3) apply Newton's iteration (algorithm 6.1) or its refinement (algorithm 9.1) to obtain approximations to the eigenvalues of A within the fixed tolerance to the errors.

The process may be interrupted (at most $n - 1$ times) by the separation steps, and then it is recursively repeated.

Remark 4.1. Steps 1) and 2) can be included in all the known divide-and-conquer (d.-c.) eigenvalue algorithms ([BP], [C], [DS]), to ensure faster convergence of the subsequent iterative processes (which vary for various d.-c. algorithms). Such an inclusion has been elaborated in [BP], but now we may also incorporate new improvements, of step 1 (shown in the next section) and of the subsequent Newton iteration (by using our algorithm of section 9). The power of the application of our approach is accentuated in the case of the d.-c. algorithms because they immediately furnish us with approximate separation of the eigenvalues of A.

5. Double exponential sieve.

For simplicity, we will study the case where a (single or multiple) eigenvalue λ lies in the interval $[0, 2)$ and is the only eigenvalue of A lying in this interval, and we will only comment on the extension to the general case at the end of this section.

By applying one bisection step, we will first specify in which interval, $[0, 1)$ or $[1, 2)$, λ lies. Without loss of generality, let $0 \leq \lambda < 1$. At this point, to see the main difficulty, assume that λ lies very near 0 and that so do some negative eigenvalues of A. (This situation so far has no perfect solution in the known implementations of the divide-and-conquer algorithms, [C], [DS].) In this case, the bisection process yields the isolation too slowly. Indeed, it is much more efficient in this case to compute $n_-(x_i)$ not for $x_i = 2^{-i}$ but for $x_i = 2^{-2^i}$, $i = 0, 1, \ldots$. This is precisely the strategy of our next algorithm, and this strategy (after its refinement) turned out to be the most effective means for rapidly ensuring higher isolation.

Let us next formally describe the algorithm.

For a fixed tolerance t, $0 < t < 1$, we will seek two real values a and b such that

$$0 \le a \le \lambda < b \le 1$$

and either

$$b - a \le 2t \tag{5.1}$$

(in which case $|\lambda - (a+b)/2| \le t$, so that the point $(a+b)/2$ approximates λ within the error bound t) or else

$$b \le 2a \tag{5.2}$$

(in which case the isolation ratio of the interval $[a, b)$ is at least 3).

The *double exponential sieve* algorithm of [BOT] computes the desired values a and b in $\lceil \log\log(1/(2t)) \rceil^2$ evaluations of $n_-(x)$. More precisely, the algorithm of [BOT] has been devised for approximating real polynomial zeros; we will restate it for the symmetric eigenvalue problem and will then improve it a little.

The algorithm first successively evaluates $n_-(x_i)$ for $x_i = 2^{-2^i}$, $i = 1, \dots, g_1$, where

$$g_1 = \min\{ \lceil \log\log(1/(2t)) \rceil \ , \quad \min_{i=1,2,\dots,} \{ i,\ n_-(x_i) < n_-(1) \} \} \ .$$

If $g_1 = \lceil \log\log(1/(2t)) \rceil$, then $0 \le \lambda \le 2t$, and we satisfy (5.1) by setting $a = 0$, $b = x_{g_1}$. Otherwise, λ lies in the interval $J = \{\lambda,\ a_1 = x_{g_1} \le \lambda < x_{g_1 - 1} = b_1\}$. If $g_i = 1$, then $\mathrm{ir}(J) \ge 5/3$, and we increase this ratio to at least $11/3$ in at most 2 applications of fact 4.1. Otherwise, we apply the same double exponential sieve procedure to the latter interval. Let g_2 denote the number of the evaluations of $n_-(x)$ in these applications. Then we either satisfy (5.1) by setting $a = a_1$, $b = a_2 = a_1 + (b_1 - a_1)2^{-2^{g_2}}$ or else obtain that

$$a_2 = a_1 + (b_1 - a_1)2^{-2^{g_2}} \le \lambda \le a_1 + (b_1 - a_1)2^{-2^{g_2 - 1}} = b_2 \ .$$

Since $b_1 - a_1 < 1$, the length of the latter interval, $b_2 - a_2$, is less than a_1 if $g_2 \ge g_1$, and in this case, we set $a = a_2$, $b = b_2$, and satisfy (5.2). Thus, it remains to consider the

case where $g_2 < g_1$. Recursively, we arrive at a decreasing sequence of positive integers $\{g_1, g_2, g_3, \ldots\}$ ending with a term g_u such that setting

$$a = a_u \ , \qquad b = b_u$$

satisfies (5.1) and/or (5.2). Since the sequence $\{g_1, g_2, \ldots\}$ strictly decreases, we have $u \le g_1$, so that the overall number of the evaluations of $n_-(x)$ in the entire process is at most

$$T_0 = \sum_{i=1}^{u} g_i \le 1 + 2 + \cdots + g_1 = (g_1 + 1)g_1/2 \ , \quad g_1 = \lceil \log\log(1/(2t)) \rceil \ . \tag{5.3}$$

Extension. If the input interval $[0, 2)$ has been replaced by any interval J of length $2L$, which may contain several eigenvalues of A, then the above process can be immediately extended so that, in at most $(g_1^*+1)g_1^*/2$ (for $g_1^* = \lceil \log\log(L/(2t)) \rceil^2$) evaluations of $n_-(x)$, we either compute a splitting point x in J [compare (4.1)], thus defining a separation of the eigenvalues of A from each other, or else output a subinterval $\widetilde{J}$ of J containing the same eigenvalues of A as J and such that $|\widetilde{J}| \le 2t$ (which ensures the desired approximation to all the $2L$ eigenvalues of A in J) and/or $\mathrm{ir}(\widetilde{J}) \ge 3$ (which ensures good initial isolation of these $2L$ eigenvalues from the other eigenvalues of A).

Improvement. The double exponential sieve process can be applied with any base $1 + v > 1$, rather than with the base $1 + v = 2$, as in [BOT], that is, we may initially set $x_i = (1+v)^{-2^i}$ and compute $n_-(x_i)$ as above, for $i = 1, \ldots, g_1(v)$; then we shall recursively apply a similar process to the intervals $\{\lambda, x_{g_k(v)} \le \lambda < x_{g_k(v)-1}\}$, $k = 1, 2, \ldots, u(v)$. In this case, the complexity estimate (5.3) (extended to any interval of length $2L$) changes into the estimate

$$T_0^*(v) = (g_1(v) + 1)\, g_1(v)/2, \ g_1(v) \le \lceil \log\left(\log(L/(2t))/\log(1 + v)\right) \rceil \ , \tag{5.4}$$

and the lower bound on the isolation ratio of the output interval of the double exponential sieve process remains equal to 3, except for the case where $g_1 = 1$, and then this ratio

changes from 3 to $1+2/v$. In the latter case, h applications of fact 4.1 increase the ratio to at least $1+2^{1+h}/v$, which is at least 3 if $h \geq \log v$. Thus, we shall extend the complexity bound (5.3) (adjusted to the interval L) to the following estimate:

$$T_0(v) = \max\{T_0^*(v), \lceil \log v \rceil\} \ , \tag{5.5}$$

which we may optimize by choosing an appropriate value for the parameter v. In particular, consider the two cases:

(a) $\log(L/(2t)) = 69$, $g_1(1) = 7$,

(b) $\log(L/(2t)) = 34$, $g_1(1) = 6$.

Then we may set $v = 512$ in case (a), $v = 64$ in case (b) and obtain from (5.3), (5.5) that

$$T_0 = T_0(1) = 28 \ , \qquad T_0(v) = 9 \ , \qquad \text{in case (a)},$$

$$T_0 = T_0(1) = 15 \ , \qquad T_0(v) = 6 \ , \qquad \text{in case (b)},$$

Due to the influence of the choice of v on the subsequent computations, it turns out to be more effective to set

$$v = 19 \ , \quad T_0(v) = 10 \ , \quad \text{in case a)}, \tag{5.6}$$

and

$$v = 3.4 \ , \quad T_0(v) = 6 \ , \quad \text{in case b)}. \tag{5.7}$$

Remark 5.1. The equations (5.3) and even (5.5) give us overly pessimistic estimates, because for a random λ, chosen under the uniform probability distribution on the interval from 0 to 1, we have that

$$\text{Probability}\{g_i(v) > i\} = (1+v)^{-2^i} \ ,$$

and similarly, Probability$\{g_k(v) > i\}$ also rapidly decreases to 0 as i grows, for all k. Moreover, the same effect can be achieved by means of the randomization of the choice of v, rather than λ.

6. Accelerated computation in the case of good initial isolation.

In this section we will propose an algorithm that, for a given interval J having a sufficiently high isolation ratio N, either separates the eigenvalues of A in J or approximates all of them (within a fixed tolerance t to the errors) at the cost of $O(\log\log(|J|/t))$ evaluations of $n_-(x)$. The choice of N will be specified in sections 7–10.

We will first formalize our task as the following computational problem:

Problem 6.1. Input: real a, b, N, t and integers j, k, $n_-(a)$, $n_-(b)$ such that

$$t > 0\,, \quad b > a\,, \quad j \geq 0\,, \quad k \geq 1\,, \quad j + k \leq n\,,$$

$$\lambda_{j+k+1} \leq a - C < a \leq \lambda_{j+k} \leq \cdots \leq \lambda_{j+1} < b < b + C \leq \lambda_j \qquad . \quad (6.1)$$

where $C = (b-a)(N-1)/2$.

(Due to the above relations, the interval $[a, b)$ contains exactly k eigenvalues of A, that is, $\lambda_{j+1}, \ldots, \lambda_{j+k}$, and has an isolation ratio of at least N.)

Output: a real μ such that

$$\lambda_{j+k} - t \leq \mu \leq \lambda_{j+1} + t \qquad (6.2)$$

and the integers $n_-(\mu - t)$, $n_-(\mu + t)$.

Once these output values are available, we have that either

$$n_-(\mu + t) = n_-(b)\,, \quad n_-(\mu - t) = n_-(a)$$

(in which case

$$\mu - t \leq \lambda_{j+k} \leq \cdots \leq \lambda_{j+1} < \mu + t\,,$$

so that μ approximates all the eigenvalues $\lambda_{j+1}, \ldots, \lambda_{j+k}$ within the error bound t) or $n_-(\mu + t) < n_-(b)$ [and then $\mu + t$ is a splitting point,

$$\lambda_{j+k} \leq \mu + t < \lambda_{j+1}\,]$$

or $n_-(\mu - t) > n_-(a)$ [and then $\mu - t$ is a splitting point,

$$\lambda_{j+k} < \mu - t \leq \lambda_{j+1}] .$$

To solve problem 6.1, we will apply Newton's iteration for computing the k-fold zero of a function. Since we deal with a cluster of the zeros of $p(x)$, well isolated from the other zeros of $p(x)$ (because we assume that N and C are large), this cluster should behave like a k-fold multiple zero of $p(x)$, and Newton's iteration is expected to converge rapidly. Our analysis in the next section confirms this expectation.

Let us next formally describe a recursive algorithm that realizes this idea in order to solve problem 6.1.

Algorithm 6.1. Initialization: set $a_0 = a$, $b_0 = b$, $N_0 = N$, $t_0 = (b_0 - a_0)/2$.

Recursive step i, $i = 0, 1, \ldots$. Choose a nonnegative h_i (according to remark 6.1 below) and compute

$$c_i = b_i + (b_i - a_i)h_i , \tag{6.3}$$

$$\mu_i = c_i - k\, p(c_i)/p'(c_i) , \tag{6.4}$$

$$t_{i+1} = 2(b_i - a_i)(h_i + 1)^2 M_i \,/\, (k - 2(h_i + 1)\, M_i) \tag{6.5}$$

where

$$M_i = \max\Big\{\frac{n - j - k}{N_i + 1 + 2h_i} \;,\; \frac{j}{N_i - 1 - 2h_i}\Big\} . \tag{6.6}$$

[Note that (6.4) represents Newton's iteration for approximating a k-fold zero of $p(x)$.] If $t_{i+1} \leq t$, then output $\mu = \mu_i$, compute and output $n_-(\mu - t)$, $n_-(\mu + t)$ and end. Otherwise compute $a_{i+1} = \mu_i - t_{i+1}$, $b_{i+1} = \mu_i + t_{i+1}$, $n_-(a_{i+1})$ and $n_-(b_{i+1})$. If

$$n_-(a) < n_-(a_{i+1}) ,$$

output $\mu = a_{i+1}$, compute and output $n_-(\mu - t)$, $n_-(\mu + t)$ and end. Otherwise, if

$$n_-(b_{i+1}) < n_-(b) ,$$

output $\mu = b_{i+1}$, compute and output $n_-(\mu - t)$, $n_-(\mu + t)$ and end. Otherwise, set

$$N_{i+1} = \frac{(N_i - 1)(k - 2(h_i + 1)M_i)}{4(h_i + 1)^2 M_i} - 1 \ , \tag{6.7}$$

where M_i is defined by (6.6), and go to step $i + 1$.

Remark 6.1. Estimating the complexity of our computations in sections 7, 8 and 9, we will focus on the cases where $h_i = 0$ and $h_i = 1$, providing also the analysis in the case of any positive $h_i < (N_i - 1)/2$ (in section 7) and $h_i < (N_i - 2)/2$ (in section 9). The two choices of $h_i = 0$ and $h_i = 1$ for all i lead to about the same complexity estimates for our algorithms. Due to the symmetry, this analysis can be extended to the choice of $h_i \le -1$ [after the respective adjustment of the expressions (6.5) and (6.6) for t_i and M_i]. The choice of $h_i \le -1$ in the cases where $a_i - \lambda_{j+k+1} > \lambda_j - b_i$ and of $h_i \ge 0$ otherwise should slightly improve the convergence of the algorithm.

7. Analysis of algorithm 6.1.

To analyze algorithm 6.1, we will assume that $2h_i < N_i - 1$ and will first recall that

$$-p'(x)/p(x) = \text{trace}((A - xI)^{-1}) = \sum_{r=1}^{n} 1/(\lambda_r - x) \ ,$$

$$a \le \lambda_r < b \ , \qquad r = j + 1, \ldots, j + k \ ,$$

$$\lambda_r \le a - C \ , \qquad r = j + k + 1, \ldots, n \ ,$$

$$\lambda_r \ge b + C \ , \qquad r = 1, \ldots, j \ ,$$

where $C = (b - a)(N - 1)/2$ [as in (6.1)].

For large C and N and for smaller positive h_0, the reciprocals of the eigenvalues of $A - I$ from the interval $[a, b)$ dominate in the entire sum $-p'(x)/p(x) = \sum_{r=1}^{n} 1/(\lambda_r - x)$ if $a \le x < b$. Therefore, $-p'(x)/(kp(x))$ well approximates the average value S_0/k of these k largest reciprocals. On the other hand, k/S_0 must lie between λ_{j+1} and λ_{j+k}, and consequently, $-kp(x)/p'(x)$ must lie in or near the interval $[\lambda_{j+1}, \lambda_{j+k})$. When we

compute $-kp(x)/p'(x)$, we may include the latter unknown interval in a small subinterval of $[a,b)$ lying about μ_0 of (6.4) and having a higher isolation ratio than $[a,b)$. We will next formalize this argument and will apply it recursively, to prove both rapid growth of the isolation ratio and rapid decrease of the interval length in the recursive process.

Examine step i, for $i = 0$; denote that $h = h_0$, $M_0 = M$ (to simplify the notation). Since $c_0 = b + (b-a)h$ [see (6.3)], the latter inequalities imply that

$$(a-b)(h+1) \le \lambda_r - c_0 < (a-b)h\ , \qquad r = j+1,\dots,j+k\ ,$$

$$\lambda_r - c_0 \le (N+1+2h)(a-b)/2\ , \qquad r = j+k+1,\dots,n\ ,$$

$$\lambda_r - c_0 \ge (N-1-2h)(b-a)/2\ , \qquad r = 1,\dots,j\ .$$

Therefore,

$$S_0 = \sum_{r=j+1}^{j+k} \frac{1}{\lambda_r - c_0} \le \frac{k}{(a-b)(h+1)}\ , \tag{7.1}$$

$$|S_0| \ge \frac{k}{(b-a)(h+1)}\ ; \tag{7.1a}$$

$$\frac{2(n-j-k)}{(N+1+2h)(a-b)} \le S_1 = \sum_{r=1+j+k}^{n} \frac{1}{\lambda_r - c_0} < 0\ ,$$

$$0 < S_2 = \sum_{r=1}^{j} \frac{1}{\lambda_r - c_0} \le \frac{2j}{(N-1-2h)(b-a)}\ ,$$

$$|S_1 + S_2| \le 2M/(b-a)\ ,$$

$$M = \max\Big\{\frac{n-j-k}{N+1+2h}\ ,\ \frac{j}{N-1-2h}\Big\} \le \frac{n-1}{N-1-2h}$$

[compare (6.6)]. It follows that

$$-\frac{p'(c_0)}{k\,p(c_0)} = \frac{1}{k}\sum_{r=1}^{n} \frac{1}{\lambda_r - c_0} = \frac{S_0}{k}\Big(1 + \frac{S_1+S_2}{S_0}\Big) = \frac{S_0}{k}(1+\rho)\ ,$$

$$|\rho| = |S_1+S_2|/|S_0| \le 2(h+1)\,M/k < \frac{2(h+1)(n-1)}{k(N-1-2h)}\ . \tag{7.2}$$

Consequently,

$$-\frac{k\,p(c_0)}{p'(c_0)} = \frac{k}{(1+\rho)S_0}\ . \tag{7.3}$$

We now deduce from (6.1) and (7.1) that

$$\frac{1}{\lambda_{j+1}-c_0} \le \frac{S_0}{k} = \frac{1}{k}\sum_{r=j+1}^{j+k}\frac{1}{\lambda_j - c_0} \le \frac{1}{\lambda_{j+k}-c_0}\ ,$$

and therefore,

$$(a-b)(h+1) \le \lambda_{j+k} - c_0 \le k/S_0 \le \lambda_{j+1} - c_0 < (a-b)h\ , \tag{7.4}$$

$$\lambda_{j+k} \le (k/S_0) + c_0 \le \lambda_{j+1}\ . \tag{7.5}$$

On the other hand, recall from (6.4) and (6.5), for $i = 0$, that

$$\mu_0 = c_0 - k\,p(c_0)/p'(c_0)\ , \tag{7.6}$$

$$t_1 = 2(b-a)(h+1)^2\,M/\big(k - 2(h+1)\,M\big)\ . \tag{7.7}$$

Our choice of h and N will guarantee that $|\rho| < 1$, and we deduce from (7.2) that

$$|\rho/(1+\rho)| \le \frac{2(h+1)M}{k(1-|\rho|)} \le \frac{2(h+1)M}{k(1-2(h+1)M/k)} = \frac{2(h+1)M}{k-2(h+1)M}\ .$$

Combining the latter inequality with the bound $k/|S_0| \le (b-a)(h+1)$, of (7.4), and with the equations

$$\frac{k|\rho|}{|(1+\rho)S_0|} = k\Big|\frac{p(c_0)}{p'(c_0)} + \frac{1}{S_0}\Big|$$

[implied by (7.3)] and (7.7), we obtain that

$$t_1 \ge \frac{k|\rho|}{|(1+\rho)S_0|} = k\Big|\frac{p(c_0)}{p'(c_0)} + \frac{1}{S_0}\Big|\ . \tag{7.8}$$

From (7.5) and (7.8), we now deduce that

$$\lambda_{j+1} + t_1 \ge (k/S_0) + c_0 + k\Big|\frac{p(c_0)}{p'(c_0)} + \frac{1}{S_0}\Big| \ge c_0 - kp(c_0)/p'(c_0)\ ,$$

and due to (7.6), we conclude that

$$\lambda_{j+1} + t_1 \geq \mu_0 \ .$$

Similarly, we deduce from (7.5), (7.6) and (7.8) that

$$\lambda_{j+k} - t_1 \leq \mu_0 \ .$$

If $t_1 \leq t$, we satisfy (6.2) by setting $\mu = \mu_0$. Otherwise, we have 3 cases:

Case a). $n_-(\mu_0 - t_1) > n_-(a)$. Then

$$\lambda_{j+k} < \mu_0 - t_1 \leq \lambda_{j+1} \ ,$$

and $\mu = \mu_0 - t_1$ satisfies (6.2).

Case b). $n_-(\mu_0 - t_1) = n_-(a)$, $n_-(\mu_0 + t_1) < n_-(b)$. Then

$$\lambda_{j+k} \leq \mu_0 + t_1 < \lambda_{j+1} \ ,$$

and $\mu = \mu_0 + t_1$ satisfies (6.2).

Case c). $n_-(\mu_0 - t_1) = n_-(a)$, $n_-(\mu_0 + t_1) = n_-(b)$. Then

$$\mu_0 - t_1 \leq \lambda_{j+k} \leq \lambda_{j+1} < \mu_0 + t_1 \ ,$$

and by setting

$$a_1 = \mu_0 - t_1 \ , \qquad b_1 = \mu_0 + t_1 \ , \tag{7.9}$$

we bracket the eigenvalues $\lambda_{j+k}, \ldots, \lambda_{j+1}$ in the interval $[a_1, b_1)$ of length $2t_1$. The distance from μ_0 to the nearest eigenvalue of A lying outside the interval $[a_1, b_1)$ is at least $C - t_1$ [for C of (6.1)]. Therefore, the isolation ratio of this interval is at least

$$N_1 = (C/t_1) - 1 = \frac{b-a}{2t_1}(N-1) - 1 = \frac{(N-1)(k-2(h+1)M)}{4(h+1)^2 M} - 1 \ , \tag{7.10}$$

which is (6.7) for $i = 0$, and we will choose $N = \gamma n$ so as to ensure that $N_1 + 1 > N - 1$ and, therefore, $2t_1 < b - a$.

Thus, the first step of algorithm 6.1 either solves problem 6.1 or reduces its input interval $[a, b)$ to a shorter interval $[a_1, b_1)$, containing exactly the same eigenvalues of A. In the latter case, we again arrive at problem 6.1, with a, b, N replaced by a_1, b_1, N_1, respectively [according to (7.9), (7.10)], and recursively repeat step i of algorithm 6.1, for $i = 1, 2, \ldots$.

From (6.6), we obtain that

$$M_i \leq \frac{n-1}{N_i - 1 - 2h_i} .$$

Substitute this bound into (6.7) and (6.5) and deduce that

$$N_{i+1} \geq (N_i - 1)(N_i - 1 - 2h_i)\frac{k - 2(h_i+1)M_i}{4(h_i+1)^2(n-1)} - 1 ,$$

$$b_{i+1} - a_{i+1} = 2t_{i+1} \leq 2t_i \frac{N_i - 1}{N_{i+1}} \leq \frac{4(b_i - a_i)(h_i+1)^2(n-1)}{(N_i - 1 - 2h_i)(k - 2(h_i+1)M_i)} ,$$

for $i = 0, 1, \ldots$. It follows that

$$\frac{b_{i+1} - a_{i+1}}{b_i - a_i} = O(\frac{n}{kN_i}) ,$$

$$N_i^2 / N_{i+1} = O(n/k) .$$

For a sufficiently large N, this implies a superlinear (and actually almost quadratic) growth of N_i (with the growth of i) and a respective decrease of t_i.

To give specific estimates, let us next set $h_i = 0$ for all i (compare remark 6.1). apply the above lower bound on N_{i+1}, and obtain that

$$\begin{aligned}(4n-4)(N_{i+1}+1) &\geq (N_i - 1)^2(k - 2M_i) \\ &\geq (N_i - 1)^2(k - (2n-2)/(N_i - 1)) = (N_i - 1)(N_i - 2n + 1) .\end{aligned}$$

Assume that $N = \gamma_0 n$, $\gamma_0 > 6$, and recursively deduce that

$$N_{i+1} > \gamma_{i+1} n , \quad \gamma_{i+1} = (\gamma_i - 2)\gamma_i / 4 , \quad i = 0, 1, \ldots$$

Set $\gamma_i = 4\gamma_i^* + 2$, $i = 0, 1, \ldots$, and obtain that $\gamma_{j+i}^* > (\gamma_j^*)^{2^i}$ for all integers $i > 0$ and $j \geq 0$. It follows that, for any fixed $j \geq 0$,

$$N_{i+j} > (4(\gamma_j^*)^{2^i} + 2)n \ ,$$

$$t_{i+j+1}/t_{i+j} \leq \frac{4n-4}{N_{i+j} - 2n + 1} < (\gamma_j^*)^{-2^i} \ ,$$

$$t_{i+j+1} < (\gamma_j^*)^{1-2^{i+1}} t_j \ , \qquad i = 0, 1, \ldots \tag{7.11}$$

In particular, for $j = 0$, we obtain that

$$t_{i+1} < (\gamma_0^*)^{1-2^{i+1}} (b-a)/2 \ .$$

Hereafter, we will denote that

$$H = \log((b-a)/(2t))$$

[which corresponds to (1.2) with $b - a = 2(\lambda_1 - \lambda_n)$], and we now obtain that

$$\widehat{T}(\gamma_0^*) = \lceil \log(1 + H/\log \gamma_0^*) \rceil - 1 \tag{7.12}$$

recursive steps of algorithm 6.1 suffice to solve problem 6.1 assuming that $N = \gamma n$, $\gamma > 6$, $\gamma_0^* > 1$, and $h_i = 0$ for all i.

In particlar, we have:

$$\widehat{T}(2) = \lceil \log(H+1) \rceil - 1 \ , \tag{7.12a}$$

$$\widehat{T}(4) = \lceil \log(H+2) \rceil - 2 \ , \tag{7.12b}$$

$$\widehat{T}(8) = \lceil \log(1 + H/3) \rceil - 1 \ , \tag{7.12c}$$

$$\widehat{T}(16) = \lceil \log(H+4) \rceil - 3 \ , \tag{7.12d}$$

and we need to set $N = 10n$, $N = 18n$, $N = 34n$ and $N = 66n$ in order to arrive at (7.12a), (7.12b), (7.12c) and (7.12d), respectively.

Let us now set $h_i = 1$ for all i. Then

$$M_i < (n-1)/(N_i - 3) ,$$

$$16(n-1)(N_{i+1}+1)/(N_i-1) \geq (N_i-3)(k-4M_i) \geq (N_i-3)(k-4(n-1)/(N_i-3))$$
$$= k(N_i-3) - 4(n-1) \geq N_i - 4n + 1$$

since $k \geq 1$.

It follows that, for $N = \gamma n$ and for a sufficiently large γ, we have:

$$N_{i+1} > \gamma_{i+1} n ,$$

$$\gamma_{i+1} = (\gamma_i - 4)\gamma_i/16 , \qquad i = 0, 1, \ldots$$

Set $\gamma_i = 16\gamma_i^* + 4$ and deduce that

$$\gamma_{i+1}^* > (\gamma_i^*)^2 , \qquad \gamma_{i+j}^* > (\gamma_j^*)^{2^i} ,$$

for all integers $i > 0$ and $j \geq 0$. Therefore, for any fixed integer $j \geq 0$, we have:

$$N_{i+j} > (16(\gamma_j^*)^{2^i} + 4)n ,$$

$$t_{i+j+1}/t_{i+j} \leq \frac{16n-16}{N_{i+j} - 4n + 1} < (\gamma_j^*)^{-2^i} ,$$

which again gives us (7.11), (7.12), (7.12a), (7.12b) and (7.12c), although this time for a slightly distinct expression of γ_0^* through $\gamma_0 = \gamma$, that is, for $\gamma_0^* = (\gamma_0 - 4)/16$. In particular (assuming $h_i = 1$ for all i), we now need to set $N = 36n$, $N = 68n$, $N = 132n$ and $N = 260n$ in order to arrive at (7.12a), (7.12b), (7.12c) and (7.12d), respectively.

Remark 7.1. The same techniques of the analysis would improve the bound (7.12) for $k > 1$. A small improvement (with a more difficult analysis job) could also be obtained based on choosing positive j in (7.11).

8. The eigenvalue algorithm.

Combining the algorithms of the previous sections, we will now accelerate the bisection algorithm 3.1. We will first restate the eigenvalue problem for any fixed interval $[a, b)$ as follows:

Problem 8.1. Input: an $n \times n$ rst matrix A, real a, b, t and the integers k, $n_-(a)$, $n_-(b)$ such that $t > 0$, $a < b$, $k = n_-(b) - n_-(a) > 0$, $0 \leq n_-(a) < n_-(b) \leq n$.

Output: a splitting point x, such that (4.1) holds, or an approximation, μ, within the tolerance t, to all the eigenvalues of A lying in $[a, b)$.

In the latter case, μ serves as a desired solution to the eigenvalue problem on $[a, b)$. In the former case, we reduce the eigenvalue problem on $[a, b)$ to two smaller eigenvalue problems on two subintervals. Proceeding recursively, we will arrive (in at most $k - 1$ steps) at the former case for problems 8.1 on all the subintervals; the solutions on these subintervals combined define a solution of the original eigenvalue problem on the input interval $[a, b)$.

An algorithm for the solution of problem 8.1 now follows, with the choice of the parameters v, N and h_i according to the recipes of sections 5–7.

Algorithm 8.1.

1°. Fix an appropriate positive v and apply the double exponential sieve process of section 5 to the interval $J = [a, b)$. The process either solves problem 8.1 or outputs a subinterval $\widehat{J}$ of $[a, b)$ containing the same eigenvalues as the input interval $[a, b)$, and in this case, the process also outputs ir_-, a lower bound 3 or $1 + 2/v$ on the isolation ratio of $\widehat{J}$.

2°. In the latter case, fix a sufficiently large N and apply $\lceil \log((N-1)(ir_- - 1)) \rceil$ bisection steps to the interval $\widehat{J}$. This either solves problem 8.1 or else outputs a subinterval J^* of $\widehat{J}$ containing the same eigenvalues as $\widehat{J}$ and having an isolation ratio of at least N (see fact 4.1).

3°. In the latter case, choose nonnegative h_i, $i = 0, 1, \ldots$, and apply algorithm 6.1 to the interval J^*.

Since there can be at most $k - 1$ separation steps, approximating all the k eigenvalues

of A in $[a,b)$ requires at most $2k-1$ calls for algorithm 8.1, that is, in a pessimistic estimate, at most $(2k-1)(T_0(v)+T_1(v,N))$ evaluations of $n_-(x)$, for $k \le n$, and $(2k-1)T_2(N)$ evaluations of $p(x)/p'(x)$ where

$$T_0(v) = \max\{T_0^*(v), \lceil \log v \rceil\}\ , \quad T_0^*(v) = (g_1(v)+1)g_1(v)/2\ ,\ g_1(v) = \lceil \log \frac{H-1}{\log(1+v)} \rceil\ , \tag{8.1}$$

[compare (5.4) and (5.5)],

$$T_1(v,N) = \lceil \log((N-1)v/2) \rceil\ , \tag{8.2}$$

$$T_2(N) = \lceil \log(cH+d) \rceil\ . \tag{8.3}$$

$H = \log((b-a)/(2t))$; $[a,b)$ is the input interval, $t > 0$ is the tolerance to the output errors, the constants c and d should be defined depending on the choice of the values N and h_i. In particular, by choosing $N = (4\gamma_0^* + 2)n$, $h_i = 0$, for all i, or $N = (16\gamma_0^* + 4)n$, $h_i = 1$, for all i, we arrive at $T_2(N) = \widehat{T}(\gamma_0^*)$, where $\widehat{T}(\gamma_0^*)$ satisfies (7.12).

Remark 8.1. The worst case factor $2k-1$ in the above estimates is overly pessimistic. Indeed, the number of recursive calls for each stage 1°, 2°, 3° of algorithm 8.1 actually varies from k to $2k-1$, and if it reaches the value $2k-1$, then the number of iteration steps in each call to stages 1° and 3° must be substantially less than the estimates (8.1), (8.3) shows (these estimates would have implied the output error of $t/(2\lceil \log k \rceil)$, and thus the algorithm should stop earlier).

9. Acceleration of algorithm 6.1

When we apply algorithm 6.1 at step 3° of algorithm 8.1, we may have available some approximations to all the eigenvalues of A [not only to $\lambda_{j+1}, \ldots, \lambda_{j+k}$, lying in the input interval $[a,b)$]. Next, we will use such approximations in order to accelerate the convergence of algorithm 6.1 and to decrease the value of N used in this agorithm. For this purpose, we will further weaken the already decayed influence of the reciprocals $1/(\lambda_r - c_i)$ of the

remote eigenvalues λ_r, for $r \le j$ and $r > j + k$, on the value $p'(c_i)/p(c_i)$. We yield this simply by subtracting (from this value) $1/(\lambda_r^* - c_i)$, the approximation to such reciprocals, readily available since λ_r^* are available. Our further analysis shows that, indeed, this well serves our purpose. To simplify the presentation, we will only show (in some detail) the first recursive step (accelerating step 0 of algorithm 6.1) and will assume that we are given approximations λ_r^* to all the eigenvalues λ_r of A, for $r = 1, \dots, n$, such that

$$|\lambda_r^* - \lambda_r| \le t^* = (b - a)/2 \ . \tag{9.1}$$

In fact, we may ensure this assumption by arranging the steps of algorithm 6.1 so as to always work with the *largest* of the available suspect intervals output at stage 2°.

We now modify algorithm 6.1 by replacing the values μ_i, involved in the expression (6.4) of the recursive step i, by the values μ_i^* defined as follows:

$$\mu_i^* = c_i + k/q_i \ ,$$

$$q_i = -\frac{p'(c_i)}{p(c_i)} - \sum_{r=1}^{j} \frac{1}{\lambda_r^* - c_i} - \sum_{r=j+k+1}^{n} \frac{1}{\lambda_r^* - c_i} \ ,$$

$i = 0, 1, \dots$. [The evaluation of the two latter sums and their subtraction from $-p'(c_i)/p(c_i)$ requires $3(n - k)$ extra arithmetic operations for each i.] Hereafter, we will refer to this modification of algorithm 6.1 as to **algorithm 9.1** and will refer to the respective modification of algorithm 8.1 as to **algorithm 8.1a**.

We need to specify the choice of the parameters h_i and N in this algorithm. We may set $h_i = 0$ for all i, which should lead to faster convergence of the algorithm, but leaves the value $r(c_i) = -p'(c_i)/p(c_i)$ unbounded. To avoid overflow, we should first compute the reciprocal $1/r(c_i)$ and end the computation detecting some eigenvalues of A in the interval $(b_i - t, b_i)$ if this reciprocal value is close to 0.

Another option is to assign small positive values to h_i, which should prevent the computer from overflow since

$$|p'(c_i)/p(c_i)| = \sum_{r=1}^{n} \frac{1}{\lambda_r - c_i} < \frac{k}{h_i} + \frac{n - k}{C - h_i} \ , \quad C = (b - a)(N - 1)/2 \ .$$

If V denotes the minimum value that causes overflow, then the bound $|p'(c_i)/p(c_i)| < V$ is guaranteed for any h_i exceeding $k/(V - \frac{n-k}{C-h_i})$, and thus it is sufficient to choose positive h_i of the above order.

To ensure rapid convergence of algorithm 9.1, we need a milder restriction on the parameter N than in the case of algorithm 6.1, namely, we will set

$$N = \Theta n^{1/2} + 2 \ ,$$

for a constant Θ to be specified later on.

Next, we will analyze algorithm 9.1 extending our analysis from section 7. Setting again $S_0 = \sum_{r=j+1}^{j+k} \frac{1}{\lambda_r - c_0}$, we obtain that

$$q_0 = S_0 + \sum_{r=1}^{j} d_r + \sum_{r=j+k+1}^{n} d_r \ , \tag{9.2}$$

$$d_r = \frac{1}{\lambda_r - c_0} - \frac{1}{\lambda_r^* - c_0} = \frac{\lambda_r^* - \lambda_r}{(\lambda_r - c_0)(\lambda_r^* - c_0)} \ , \quad \text{for all } r \ . \tag{9.3}$$

We recall that, unless $j < r \leq j + k$, we have the bound,

$$\frac{1}{|\lambda_r - c_0|} \leq \frac{2}{(N-1-2h)(b-a)} \ ,$$

which we extend to the bound

$$\frac{1}{|\lambda_r^* - c_0|} \leq \frac{2}{(N-2-2h)(b-a)} \ ,$$

due to (9.1).

Combine these bounds with (9.1)–(9.3) and obtain that

$$|q_0 - S_0| \leq (n-k)/((N-1-2h)(N-2-2h)t^*) \ .$$

Denote

$$\rho^* = (q_0 - S_0)/S_0 \ ,$$

so that $q_0 = (1+\rho^*)S_0$. Deduce from the latter relations and from (7.1a) and (9.1) that

$$k/|S_0| \le (b-a)(h+1) ,$$

$$|\rho^*| \le 2(h+1)(n-k)/(\,(N-1-2h)(N-2-2h)k) \,. \tag{9.4}$$

Our choice of (small) h and (large) N will guarantee that $|\rho^*| < 1$, and we will obtain the following bound:

$$|k/S_0 - k/q_0| = |\rho^*(k/S_0)/(1+\rho^*)| \le t_1^* ,$$

for

$$t_1^* = 2(h+1)^2(n-k)(b-a)/(\,(N-1-2h)(N-2-2h)k(1+\rho^*)) \,.$$

We have from (7.5) that

$$\frac{1}{\lambda_{j+1}-c_0} \le \frac{S_0}{k} \le \frac{1}{\lambda_{j+k}-c_0} , \quad \lambda_{j+k} \le \frac{k}{S_0} + c_0 \le \lambda_{j+1} ,$$

and it follows that

$$\lambda_{j+k} - t_1^* \le \mu_0^* = c_0 + k/q_0 \le \lambda_{j+1} + t_1^* \,.$$

Thus, step 0 of our modification of algorithm 6.1 [based on the replacement μ_0 of (6.4) by μ_0^*] either solves problem 8.1 or else brackets the eigenvalues $\lambda_{j+k}, \ldots, \lambda_{j+1}$ in the interval (a_1^*, b_1^*) of length $2t_1^*$ where

$$a_1^* = \mu_0^* - t_1^* , \qquad b_1^* = \mu_0^* + t_1^* \,.$$

The isolation ratio of this interval is at least

$$\begin{aligned} N_1^* &= (b-a)(N-1)/(2t_1^*) - 1 \\ &= |1+\rho^*|(N-1)(N-1-2h)(N-2-2h)k/(4(h+1)^2(n-k)) - 1 \,, \end{aligned} \tag{9.5}$$

which is substantially larger than N_{i+1} of (7.11) for $i = 0$ and for large $N = N_0$.

Replacing a, b, N by a_1^*, b_1^*, N_1^*, we may recursively repeat the computations. The above estimates for the growth of N and for the respective decrease of the interval length

$b-a$ can be recursively extended if we follow the (already pointed out) policy of always applying our algorithm to the currently largest approximation interval, thus improving the currently worst approximations to the eigenvalues of A. For $N = \Theta\, n^{0.5+v}$, $\Theta \geq 1$, we have that

$$\Theta\, n^{0.5}/N_1^* = O(1)\ ,$$

which shows a *nearly cubic growth* of the isolation ratio in the transition from $[a,b)$ to $[a_1^*, b_1^*)$. $b-a$, the length of the interval $[a,b)$, decreases at a similar rate in the transition to $b_1 - a_1^*$, the length of $[a_1^*, b_1^*)$.

Let us now set $h = 0$, denote $N = N_0^*$, and obtain that

$$(4n-4)(N_1^*+1) \geq (N_0^*-1)[(N_0^*-1)(N_0^*-2)-2(N_0^*-1)] = (N_0^*-1)((N_0^*)^2-3N_0^*-2n+4]\,.$$

Similarly, we may bound the isolation ratios N_{i+1}^* in terms of N_i^* at the next recursive steps $i = 1, 2, \ldots$ provided that $h_i = 0$ for all i.

Setting $N = N_0^* = \Theta_0 n^{1/2} + 2$, for $\Theta_0 > 4$, we may recursively deduce from these bounds that

$$N_{i+1}^* > \Theta_{i+1} n^{1/2} + 2\ , \quad \Theta_{i+1} = (\Theta_i^2 - 2)\Theta_i/4\ , \qquad i = 0, 1, \ldots\ .$$

Denote $\Theta_i^* = (\Theta_i/2) - 1$, and obtain that

$$\Theta_i = 2\Theta_i^* + 2\ ,$$

$$N_{i+1} > (2\Theta_{i+1}^* + 2)n^{1/2} + 2\ ,$$

$$\Theta_{i+1}^* > (\Theta_i^*)^3\ , \qquad i = 0, 1, \ldots\ .$$

Therefore, for all integers $i > 0$, $j \geq 0$, we have that

$$\Theta_{i+j}^* > (\Theta_j^*)^{3^i}\ ,$$

$$N_{i+j} > (2(\Theta_j^*)^{3^i} + 2)n^{1/2} + 2\ .$$

For any fixed $j \geq 0$, we have:

$$t_{i+j+1}/t_{i+j} = (N^*_{i+j} - 1)/(N^*_{i+j+1} + 1) \leq \frac{4n-4}{(N^*_{i+j})^2 - 3N^*_{i+j} - 2n + 4} < (\Theta^*_j)^{-3^i} ,$$

$i = 0, 1, \ldots$, and consequently,

$$t_{i+j+1} < (\Theta^*_j)^{(1-3^{i+1})/2} t_j , \quad i = 0, 1, \ldots \tag{9.6}$$

In particular, for $j = 0$, we obtain that

$$t_{i+1} < (\Theta^*_0)^{(1-3^{i+1})/2}(b-a)/2 ,$$

and therefore, for H denoting $\log((b-a)/(2t))$, we have that

$$\widetilde{T}(\Theta^*_0) = \lceil (\log(1 + 2H/\log\Theta^*_0))/\log 3 \rceil - 1 \tag{9.7}$$

recursive steps of algorithm 9.1 suffice to solve problem 6.1 provided that $h_i = 0$ for all i and $N = (2\Theta^*_0 + 2)n^{1/2} + 2$, $\Theta^*_0 > 1$, $\Theta_0 > 4$.

Remark 9.1. Setting $j > 1$, we may extend (9.7) to similar bounds for any $\Theta^*_0 > \sqrt{3/2} - 1$.

In particular, we obtain that

$$\widetilde{T}(2) = \lceil (\log(1 + 2H))/\log 3 \rceil - 1 , \tag{9.7a}$$

$$\widetilde{T}(4) = \lceil (\log(1 + H))/\log 3 \rceil - 1 , \tag{9.7b}$$

$$\widetilde{T}(8) = \lceil (\log(3 + 2H))/\log 3 \rceil - 2 , \tag{9.7c}$$

$$\widetilde{T}(16) = \lceil (\log(2 + H))/\log 3 \rceil - 1 , \tag{9.7d}$$

and we need to set $N = \Theta n^{1/2} + 2$, with Θ taking the values 4, 10, 18 and 34 in order to arrive at (9.7a), (9.7b), (9.7c) and (9.7d), respectively.

Let us also supply the estimates in the case where $h_i = 1$ for all i. In this case, we deduce from (9.4) and (9.5) that

$$16(n-1)N^*_1 + 1 \geq (N^*_0 - 1)[(N^*_0 - 3)(N^*_0 - 4) - 4n + 4] = (N^*_0 - 1)[(N^*_0)^2 - 7N^*_0 - 4n + 16]$$

and similarly bound the isolation ratios N_i^* of the intervals $[a_i, b_i)$ computed at the next recursive steps i of algorithm 9.1, for $i = 2, 3, \ldots$.

Setting $N_0 = \Theta_0 n^{1/2} + 2$, for $\Theta_0 > \sqrt{20}$, we deduce that

$$N_i > \Theta_i n^{1/2} + 2 \ , \quad \Theta_{i+1} = (\Theta_i^2 - 4)\Theta_i/16 \ , \quad i = 1, 2, \ldots$$

Now, we denote that

$$\Theta_i^* = (\Theta_i/4) - 2 \ , \quad \Theta_i = 4\Theta_i^* + 2$$

and deduce that

$$\Theta_{i+1}^* > (\Theta_i^*)^3 \ , \qquad i = 0, 1, \ldots \ .$$

Then, using the equations

$$\frac{t_{i+1}}{t_i} = \frac{N_i^* - 1}{N_{i+1}^* + 1} = \frac{16n - 16}{(N_i^*)^2 - 7N_i^* - 4n + 16} \ ,$$

we again deduce the bound (9.6), (9.7), (9.7a)–(9.7d), although this time we assume a distinct expression for Θ_0 through Θ_0^*, that is, $\Theta_0 = 4\Theta_0^* + 2$. In particular, we need to set that $N = N_0^* = \Theta_0 n^{1/2} + 2$ (for Θ_0 taking the values 10, 18, 34 and 66) in order to arrive at (9.7a), (9.7b), (9.7c) and (9.7d), respectively.

Remark 9.1. Seeking convergence acceleration at the expense of performing a little more work per iteration, we may generalize algorithms 6.1 and 9.1 by replacing (6.4) by more general expressions, such as

$$\tilde{\mu}_i = c_i - (k/\tilde{q}_i)^{1/d} \ ,$$

$$\tilde{q}_i = \sum_{r=j+1}^{n} \frac{1}{(\lambda_r - c_i)^d} - \sum_{r=1}^{j} \frac{1}{(\lambda_r^* - c_i)^d} - \sum_{r=j+k+1}^{n} \frac{1}{(\lambda_r^* - c_i)^d} \ ,$$

for some fixed natural $d > 1$, say, for $d = 3$. Note that the value

$$\sum_{r=1}^{n} \frac{1}{(\lambda_r - c_i)^d} = \mathrm{trace}\left((A_i - c_i I)^{-d}\right)$$

can be computed by extending the techniques of section 2. [In the extension of the same approach to approximating polynomial zeros, pointed out in the introduction, the latter value can be easily obtained from the d leading coefficients of the input polynomial $p(x)$, by using Newton's identities.]

10. Summary of the complexity estimates.

We are now ready to summarize our previous analysis into the estimates for the arithmetic complexity of the solution of problem 8.1, assuming that $k = n$, $a \leq \lambda_n \leq \lambda_1 \leq b$, $b - a \leq 2(\lambda_1 - \lambda_n)$. We recall (8.1)–(8.3), recall the need for $3n$ extra arithmetic operations in every iteration of algorithm 9.1 (versus algorithm 6.1), apply the operation count for the evaluation of $p(x)$ and $p'(x)$ from section 2, and obtain that

$$(4n^2 + O(n))(2T_0(v) + 2T_1(v, N) + \nu T_2(N)) \tag{10.1}$$

arithmetic operations suffice for the solution (for any choice of $v > 0$ and N, according to sections 5, 7 and 9) provided that either $\nu T_2(N) = 4\widehat{T}(\gamma_0^*)$ [compare (7.12)] or $\nu T_2(N) = 5.5\widetilde{T}(\Theta_0^*)$ [compare (9.7)], depending on which of the algorithms 8.1 or 8.1a we apply. The estimate (10.1) implies the estimate (1.3) of the introduction.

Next, we will calculate the value

$$T(N, 1) = 2T_0(1) + 2T_1(1, N) + \nu T_2(N) \tag{10.2}$$

for both our policies of choosing h_i in algorithms 6.1 and 9.1 (that is, for setting $h_i = 0$ or $h_i = 1$ for all i). We will consider the two cases:

a) $n = 1000$, $H = \log((b - a)/(2t)) = 70$,

b) $n = 500$, $H = \log((b - a)/(2t)) = 35$.

We first obtain from (8.1) that

$$g_1(1) = \lceil \log 69 \rceil = 7\ , \quad T_0(1) = 28\ , \text{ in the case a)}, \tag{10.3a}$$

$$g_1(1) = \lceil \log 34 \rceil = 6\ , \quad T_0(1) = 15\ , \text{ in the case b)}, \tag{10.3b}$$

The values of $T_1(1,N) = \lceil \log(N-1) \rceil - 1$ (as the functions in γ and Θ) and the values of $\nu T_2(N)$ (as the functions in γ_0^* and Θ_0^*), associated with the values of N of our interest, are displayed in Tables 10.1, 10.2 and 10.3. Tables 10.4 and 10.5 relate γ to γ_0^* and Θ to Θ_0^*. Table 10.6 collects the values $2T_0(1)$, $2T_1(1,N)$ and $\nu T_2(N)$ for all choices of γ_0^*, Θ_0^*, including also subdivision of the values $2T_1(1,N)$ into the two cases, where $h_i = 0$ and $h_i = 1$, respectively. In the two bottom lines of Table 10.6, we display the values $T(1,N)$, defined by (10.2), for each of the two cases (where $h_i = 0$ and $h_i = 1$).

The values shown as the numerators of fractions correspond to the case a), and ones shown as the denominators correspond to the case b).

The data for $T(1,N)$ can be compared to the values 128 (in case a)) and 64 (in case b)), which represent the complexity of the bisection algorithm. Table 10.6 shows that with a successful choice of the values γ_0^* and Θ_0^*, we may obtain superior worst case estimates. In parentheses in the third and in the two last lines of Table 10.6, we show these estimates decreased, due to optimizing the value v [compare (5.6) and (5.7)].

Furthermore, unlike the case of the bisection algorithm (whose worst and average case complexity is about the same, algorithms 8.1 and 8.1a, for a large class of input matrices, may actually perform substantially faster than the worst case bounds suggest (compare remarks 5.1, 7.1 and 8.1).

Table 10.1. $T_1(1,N)$ and $T_1(v,N)$ for $N=\gamma n$ and $N=\Theta n^{1/2}+2$, $v=19/3.4$.

γ	10	18	34	66	36	68	132	260					
Θ									4	10	18	34	66
$T_1(1,N)$	$\frac{13}{12}$	$\frac{14}{13}$	$\frac{15}{14}$	$\frac{16}{15}$	$\frac{15}{14}$	$\frac{16}{15}$	$\frac{17}{16}$	$\frac{17}{16}$	$\frac{6}{6}$	$\frac{8}{7}$	$\frac{9}{8}$	$\frac{10}{9}$	$\frac{11}{10}$
$(T_1(v,N))$	$(\frac{17}{15})$	$(\frac{18}{15})$	$(\frac{19}{16})$	$(\frac{20}{17})$	$(\frac{19}{16})$	$(\frac{20}{17})$	$(\frac{21}{18})$	$(\frac{22}{19})$	$(\frac{11}{8})$	$(\frac{12}{10})$	$(\frac{13}{10})$	$(\frac{14}{11})$	$(\frac{15}{12})$

Table 10.2. $\nu T_2(N)=4\widehat{T}(\gamma_0^*)$.

γ_0^*	2	4	8	16
$4\widehat{T}(\gamma_0^*)$	24/20	20/16	16/12	16/12

Table 10.3. $\nu T_2(N)=5.5\widetilde{T}(\Theta_0^*)$.

Θ_0^*	2	4	8	16
$5.5\widetilde{T}(\Theta_0^*)$	22/16.5	16.5/16.5	16.5/11	16.5/11

Table 10.4.

γ_0^*	2	4	8	16
γ (for $h_i = 0$)	10	18	34	66
γ (for $h_i = 1$)	36	68	132	260

Table 10.5.

Θ_0^*	2	4	8	16
Θ (for $h_i = 0$)	4	10	18	34
Θ (for $h_i = 1$)	10	18	34	66

Table 10.6.

(the case of $v = 19/3.4$ is shown in parentheses)

γ_0^*	2	4	8	16				
Θ_0^*					2	4	8	16
$2T_0(1)(2T_0(v))$ [(10.3a), (10.3b)]	56/30 (20/12)							
$2T_1(1,N)$ $(2T_1(v,N))$ $(h_i = 0)$ (Table 10.1)	26/24 (34/30)	28/26 (36/30)	30/28 (38/32)	32/30 (40/34)	12/12 (22/16)	16/14 (24/20)	18/16 (26/20)	20/18 (28/22)
$2T_1(1,N)$ $(2T_1(v,N))$ $(h_i = 1)$ (Table 10.1)	30/28 (38/32)	32/30 (40/34)	34/32 (42/36)	34/32 (44/38)	16/14 (24/20)	18/16 (26/20)	20/18 (28/22)	22/20 (30/24)
$\nu T_2(N)$ (Tables 10.2, and 10.3)	$\frac{24}{20}$	$\frac{20}{16}$	$\frac{16}{12}$	$\frac{16}{12}$	$\frac{22}{16.5}$	$\frac{16.5}{16.5}$	$\frac{16.5}{11}$	$\frac{16.5}{11}$
$T(1,N)$ $(h_i = 0)$ $(T(v,N))$	$\frac{106}{74}$ $(\frac{78}{62})$	$\frac{104}{72}$ $(\frac{76}{58})$	$\frac{102}{70}$ $(\frac{74}{56})$	$\frac{104}{72}$ $(\frac{76}{58})$	$\frac{90}{58.5}$ $(\frac{64}{44.5})$	$\frac{88.5}{60.5}$ $(\frac{60.5}{48.5})$	$\frac{90.5}{57}$ $(\frac{62.5}{43})$	$\frac{92.5}{59}$ $(\frac{64.5}{45})$
$T(1,N)$ $(h_i = 1)$ $(T(v,N))$	$\frac{110}{78}$ $(\frac{82}{64})$	$\frac{108}{76}$ $(\frac{80}{62})$	$\frac{106}{74}$ $(\frac{78}{60})$	$\frac{106}{74}$ $(\frac{80}{62})$	$\frac{94}{60.5}$ $(\frac{66}{48.5})$	$\frac{90.5}{62.5}$ $(\frac{62.5}{48.5})$	$\frac{92.5}{59}$ $(\frac{64.5}{45})$	$\frac{94.5}{61}$ $(\frac{66.5}{47})$

References

[B] H. J. Bernstein, An Accelerated Bisection Method for the Calculation of Eigenvalues of Symmetric Tridiagonal Matrices, *Numer. Math.*, 43, 153–160, 1984.

[BP] D. Bini, V. Pan, Practical Improvement of the Divide-and-Conquer Eigenvalue Algorithms, *Computing*, 48, 109–123, 1992.

[BOT] M. Ben-Or, P. Tiwari, Simple Algorithm for Approximating All Roots of a Polynomial with Real Roots, *J. of Complexity*, 6, 4, 417–442, 1990.

[C] J. J. M. Cuppen, A Divide and Conquer Algorithm for the Symmetric Tridiagonal Eigenproblem, *Numer. Math.*, 36, 177–195, 1981.

[DS] J. J. Dongarra, D. C. Sorensen, A Fully Parallel Algorithm for the Symmetric Eigenvalue Problem, *SIAM J. Sci. Stat. Computing*, 8, 2, 139–154, 1987.

[GL] G. M. Golub, C. F. van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, MD, 1989.

[H] P. Henrici, *Applied and Computational Complex Analysis,* Wiley, 1974.

[LPS] S.-S. Lo, B. Philippe, A. Sameh, A Multiprocessor Algorithm for the Symmetric Tridiagonal Eigenvalue Problem, *SIAM J. Sci. Stat. Computing,* 8, 155–165, 1987.

[P87] V. Pan, Sequential and Parallel Complexity of Approximate Evaluation of Polynomial Zeros, *Computers and Math. (with Applications),* 14, 8, 591–622, 1987.

[P89] V. Pan, Fast and Efficient Parallel Evaluation of the Zeros of a Polynomial having only Real Zeros, *Computer and Math. (with Applications),* 17, 11, 1475–1480, 1989.

[P89a] V. Pan, A New Algorithm for the Symmetric Eigenvalue Problem, Tech. Report TR 89-3, *Computer Science Dept., SUNYA,* Albany, NY, 1989.

[P90] V. Pan, Estimating Extremal Eigenvalues of a Symmetric Matrix, *Computers and Math. (with Applications),* 20, 2, 17–22, 1990.

[PR] B. N. Parlett, J. K. Reid, Tracking the Process of the Lanczos Algorithmn for Large Symmetric Eigenproblems, *IMA J. Num. Anal.,* 1, 135–155, 1981.

[PD] V. Pan, J. Demmel, A New Algorithm for the Symmetric Eigenvalue Problem, preprint, 1991.

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

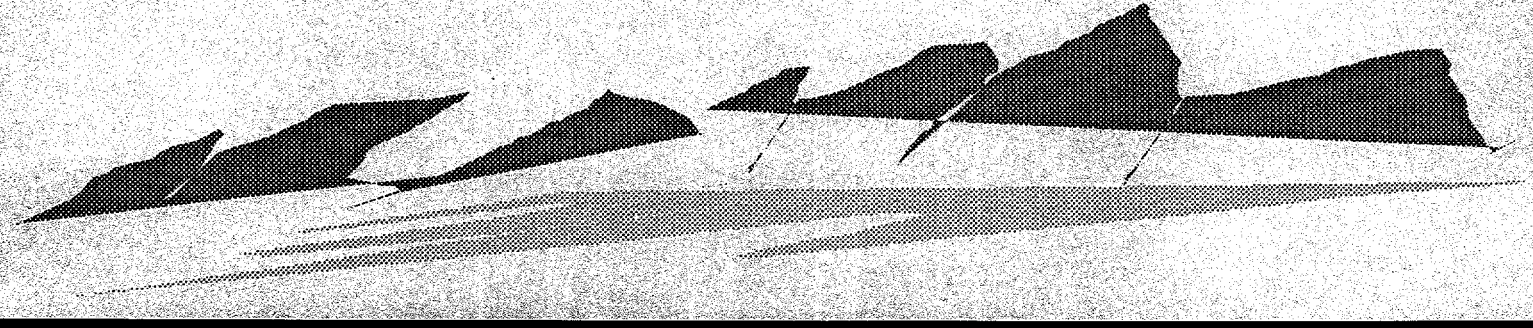
Tuesday, April 5

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

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



Tuesday Evening's Workshop

Iterative Software Kernels

Organizer: Iain Duff

Abstract

This workshop will focus on kernels for iterative software packages. Specifically, the three speakers will discuss various aspects of sparse BLAS kernels.

Speakers

- Iain Duff, Rutherford Appleton Laboratory
  "Current status of user level sparse BLAS"

  We discuss the current status of the User Level Sparse BLAS proposed by Duff, Marrone, and Radicati. In particular we indicate how it has evolved in response to comments from potential users. We indicate examples where the kernels have been used and discuss the status and availability of the software implementation.
  (Joint work with G. Radicati and M. Marrone.)

- Michael A. Heroux, Cray Research, Inc.
  "Current status of the Sparse BLAS Toolkit"

  We discuss the current status of the sparse BLAS toolkit proposed by the speaker and his collaborators. In particular, we will emphasize some of the latest modifications which make the toolkit more flexible and useful. We show examples of where optimal implementations of these kernels has had a dramatic impact on the performance of key application codes. Finally we discuss some of the software development issues, current status of the development project, and plans for future enhancements.
  (Joint work with Sandra Carney and Guangye Li.)

- Craig C. Douglas, Yale University
  "Adding Matrix-Matrix and Matrix-Matrix-Matrix Multiply to the Sparse BLAS Toolkit"

  We discuss a proposal to add a set of sparse matrix multiplication routines to the sparse BLAS toolkit. These routines would be of great benefit in multilevel and/or domain decomposition procedures in certain situations which are common in hard engineering and science problems. As examples, suppose only a fine grid matrix is available or the cost of the using a discretization module to generate smaller matrices is greater than the cost of solving the fine grid problem by a reasonable unigrid procedure. Memory issues will also be discussed.

Wednesday, April 6

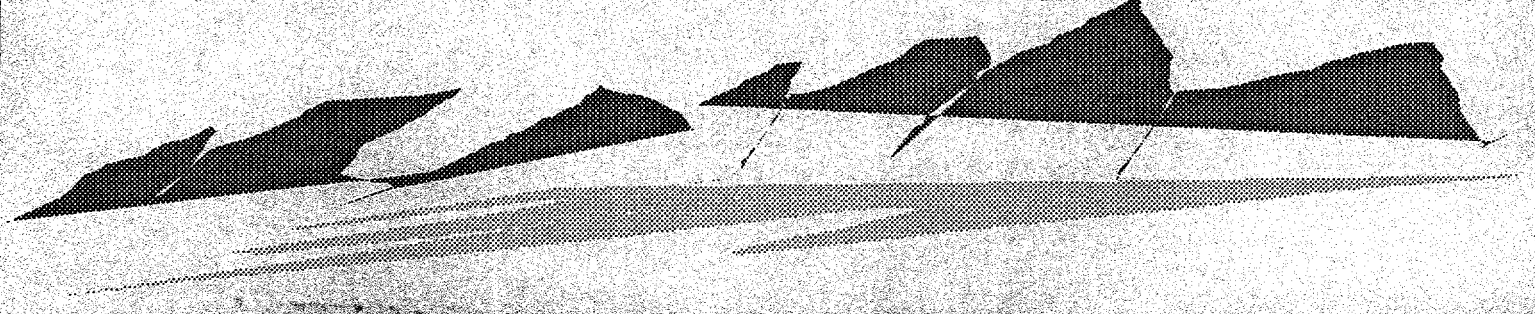
Nonsymmetric Solvers I
Chair: Tom Manteuffel
Room A

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 theFaber Polynomials for Annular Sectors

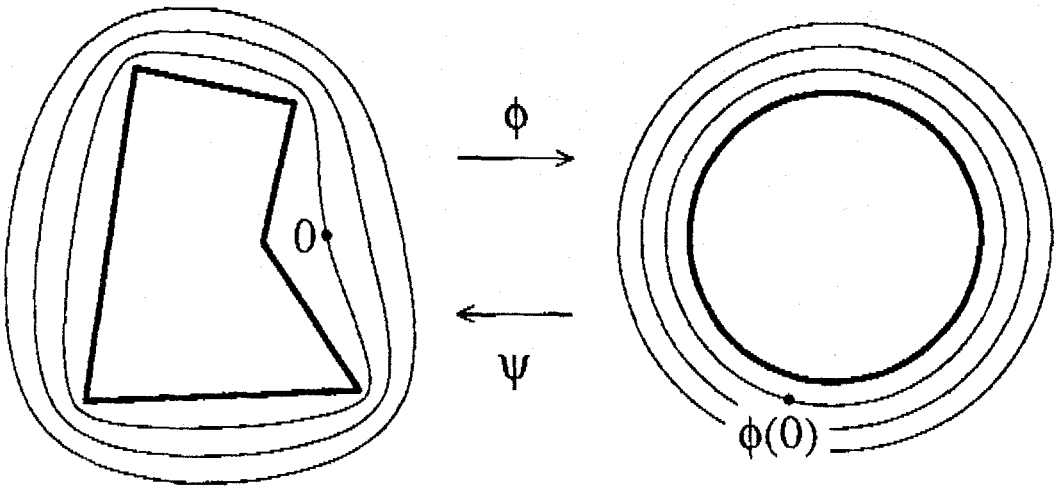
9:15 - 9:40 Gerhard Starke
Subspace Orthogonalization for Substructuring Preconditioners for Nonsymmetric Systems of Linear Equations



Conformal mapping and convergence of Krylov iterations

Tobin A. Driscoll* Lloyd N. Trefethen†

Connections between conformal mapping and matrix iterations have been known for many years. The idea underlying these connections is as follows. Suppose the spectrum of a matrix or operator A is contained in a Jordan region E in the complex plane, with $0 \notin E$. Let $\phi(z)$ denote a conformal map of the exterior of E onto the exterior of the unit disk D, with $\phi(\infty) = \infty$:



Then $1/|\phi(0)|$ is an upper bound for the optimal asymptotic convergence factor of any Krylov subspace iteration. This idea can be made precise in

*Speaker. Center for Applied Mathematics, Cornell University, Ithaca, NY 14853 (driscoll@cam.cornell.edu).

†Department of Computer Science, Cornell University, Ithaca, NY 14853 (LNT@cs.cornell.edu)

various ways, depending on the matrix iteration, on whether A is finite or infinite dimensional, and on what bounds are assumed on the non-normality of A.

This talk will explore these connections for a variety of matrix examples, making use of a new MATLAB Schwarz-Christoffel Mapping Toolbox developed by the first author. Unlike the earlier Fortran Schwarz-Christoffel package SCPACK, the new toolbox computes exterior as well as interior Schwarz-Christoffel maps, making it easy to experiment with spectra that are not necessarily symmetric about an axis.

With the aid of many graphs, we shall illustrate a variety of issues in matrix iterations, including:

- The behavior of GMRES vs. QMR, BCG, etc.,
- the gap between true and ideal iterations (i.e., between minimizing $\|p(A)b\|$ and $\|p(A)\|$),
- transient vs. asymptotic behavior.

In particular, we illustrate and comment upon the three-part convergence scenario described by Nevanlinna, in which the convergence of an iteration is first sublinear, then linear, then superlinear.

Convergence Estimates for Iterative Methods Via the Kreiss Matrix Theorem on a General Complex Domain

Kim-Chuan Toh* Lloyd N. Trefethen†

What properties of a nonsymmetric matrix A determine the convergence rate of iterations such as GMRES, QMR, and Arnoldi? If A is far from normal, should one replace the usual *Ritz values* $\rightarrow$ *eigenvalues* notion of convergence of Arnoldi by alternative notions such as *Arnoldi lemniscates* $\rightarrow$ *pseudospectra*?

Since Krylov subspace iterations can be interpreted as minimization processes involving polynomials of matrices, the answers to questions such as these depend upon mathematical problems of the following kind. Given a polynomial $p(z)$, how can we bound the norm $\|p(A)\|$ in terms of *(i)* the size of $p(z)$ on various sets in the complex plane, and *(ii)* the locations of the spectrum and pseudospectra of A?

This talk reports some progress towards solving these problems. In particular, we present theorems that generalize the Kreiss matrix theorem from the unit disk (for the monomial A^n) to a class of general complex domains (for polynomials $p(A)$).

Specifically, let A be an $N \times N$ matrix with spectrum Λ. The Kreiss Matrix Theorem gives bounds based on the resolvent norm $\|(zI - A)^{-1}\|$ for $\|A^n\|$ if Λ is in the unit disk, or for $\|e^{tA}\|$ if Λ is in the left half-plane. We generalize these results to the complex domain E, giving bounds for $\|F_n(A)\|$ if $\Lambda \subseteq E$, where F_n denotes the nth Faber polynomial associated with E.

*Speaker. Center for Applied Mathematics, Cornell University, Ithaca, NY 14853 (kc@cam.cornell.edu)

†Department of Computer Science, Cornell University, Ithaca, NY 14853 (LNT@cs.cornell.edu)

Let Φ be the conformal map from E^c onto the exterior of the unit disk. One of our bounds takes the form

$$\mathcal{K} \leq \sup_n \|F_n(A)\|, \qquad \|F_n(A)\| \leq C\mathcal{K}n,$$

where $\mathcal{K}$ is the "Kreiss constant" defined by

$$\mathcal{K} = \inf_c \left\{ \|(zI - A)^{-1}\| \leq c|\Phi'(z)| \,/\, (|\Phi(z)| - 1) \;\; \forall z \notin E \right\}$$

and C depends only on E. Analogous estimates are also established in which $\|F_n(A)\|$ is bounded in terms of N instead of n.

If $E = [-1, 1]$, then the Faber polynomials reduce to the Chebyshev polynomials. In this familiar special case our theorems establish connections between the norms $\|T_n(A)\|$ and the size of the Kreiss constant of A with respect to $[-1, 1]$. These results have implications for the convergence of Krylov subspace iterations applied to non-normal matrices with real eigenvalues.

Of course, matrix iterations cannot be expected to lead exactly to Faber polynomials for any particular set E. Fortunately, this is not a essential restriction. By means of an inequality due originally to Bernstein in 1912, our results can be extended to general polynomials $p(n)$.

An Iterative Method for the Solution of Linear Systems Using the Faber Polynomials for Annular Sectors

N. J. Myers,
Department of Mathematical Sciences,
University of Durham,
South Road,
Durham, DH1 3LE, England.
n.j.myers @ durham.ac.uk
December 1993

Abstract

We give a hybrid method for the iterative solution of linear systems of equations $A\mathbf{x} = \mathbf{b}$, where the matrix (A) is nonsingular, sparse and nonsymmetric. As in a method developed by Starke and Varga (1993) the method begins with a number of steps of the Arnoldi method to produce some information on the location of the spectrum of A. Our method then switches to an iterative method based on the Faber polynomials for an annular sector placed around these eigenvalue estimates. The Faber polynomials for an annular sector are used because, firstly an annular sector can easily be placed around any eigenvalue estimates bounded away from zero, and secondly the Faber polynomials are known analytically for an annular sector. Finally we give three numerical examples, two of which allow comparison with Starke and Varga's results. The third is an example of a matrix for which many iterative methods would fail, but our method converges.

1 Introduction

Hybrid algorithms have recently been used to solve

$$A\mathbf{x} = \mathbf{b} \tag{1}$$

where A is a large, sparse, real, nonsymmetric, nonsingular matrix. These methods usually involve two stages, the first is to use an iterative method that does not depend on any information about the coefficient matrix, and the second is to use a parameter dependent iterative method based on some information obtained in the first stage.

In the first stage Starke and Varga (1993) used Arnoldi's method to produce some estimates of the eigenvalues of the matrix A. They then placed a polygonal region, Ω, around these estimates. Using a numerical conformal mapping package they computed the parameters in the Schwarz–Christoffel representation of the conformal map, ψ, from the exterior of the unit disc onto the exterior of Ω. From these parameters they generated the Faber polynomials for the region Ω recursively. In the second stage they used these Faber polynomials as iteration polynomials. Similarly in our method we will use Arnoldi's method as the first stage. In the second stage, however, we will use the Faber polynomials for an annular sector, placed around the eigenvalue estimates, as the iteration polynomials. There are a few reasons for this choice: firstly an annular sector can easily be placed around any eigenvalue estimates bounded away from zero, secondly the Faber polynomials are known analytically for an annular sector, and finally using an annular sector allows us to consider cases of matrices that Starke and Varga's method could not deal with (for real matrices, specifically when we get two eigenvalue estimates λ_1, λ_2 such that $\lambda_1 < 0 < \lambda_2$). It is also expected that our method will perform particularly well for cases like example 2 here (example 6.3 in Starke and Varga), where Starke and Varga's region did not enclose the spectrum of the matrix and the annular sector will.

2 The Faber polynomials for an annular sector

According to the Riemann mapping theorem, given a region D in the extended complex plane, whose complement is simply connected, we can find a unique function ϕ, such that

$$\lim_{z\to\infty} \frac{\phi(z)}{z} = 1,$$

which maps the complement of D conformally onto $\{w : |w| > \rho\}$, the complement of a disc of radius ρ. The function ϕ has a Laurent expansion

$$\phi(z) = z + \alpha_0 + \frac{\alpha_1}{z} + \dots$$

about the point at infinity. The Faber polynomial $F_n(z)$, of degree n, is found by taking the polynomial part of the Laurent expansion of $[\phi(z)]^n$.

The mapping functions (or their inverses) are known analytically for only a few specific regions in the complex plane. In a recent paper (Coleman and Myers, 1993) we gave the inverse mapping and the Faber polynomials for the annular sector

$$Q = \{z : R \le |z| \le 1, \theta \le |\arg z| \le \pi\}, \qquad 0 < \theta \le \pi,$$

in terms of two parameters a and b which depend on θ and R, and using them we generated $\{\beta_i\}$ (the coefficients of w^{-i} in the Laurent expansion at infinity of the inverse mapping). From these coefficients we then generated polynomials $\phi_n(z)$ using the recurrence relation

$$\phi_n(z) = (z - \beta_0)\phi_{n-1}(z) - \sum_{k=1}^{n-1} \beta_k \phi_{n-k-1}(z) - \sum_{k=0}^{n-1} \beta_k z^{n-k} - (1+n)\beta_n,$$

and finally, the scaled Faber polynomials $\widetilde{F}_n(z) = F_n(z)/\rho^n$ were found by

$$\widetilde{F}_n(z) = \left(\frac{z}{\rho}\right)^n + \phi_{n-1}\left(\frac{z}{\rho}\right),$$

where ρ is the transfinite diameter of Q which also depends on a and b.

2.1 Scaling the sector and the Faber polynomials

We define the annular sector $Q(R_1, R_2, \theta)$ as

$$Q(R_1, R_2, \theta) = \{z : R_1 \le |z| \le R_2,\ \theta \le |\arg z| \le \pi\}.$$

Given the Faber polynomial of degree n,

$$F_n(z) = z^n + \rho^n \phi_{n-1}\left(\frac{z}{\rho}\right),$$

for the annular sector $Q(R, 1, \theta)$, with $R = R_1/R_2$, we will need to find the Faber polynomial of degree n for the annular sector

$$\bar{Q} = \{Z : R_1 \le |Z| \le R_2,\ \theta - \eta \le \arg Z \le 2\pi - \theta - \eta\}.$$

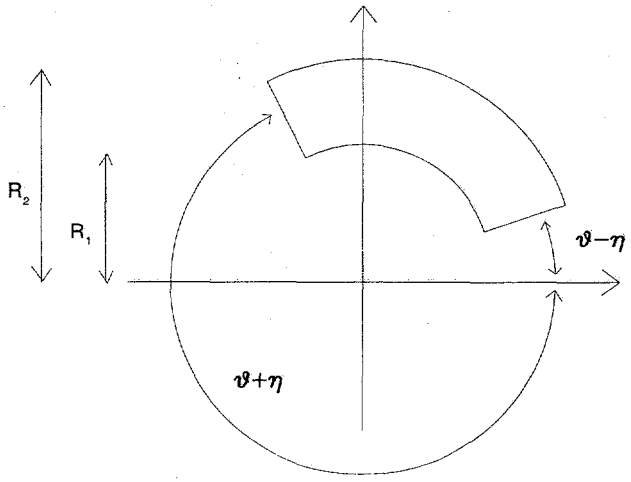


Figure 1. The annular sector $\bar{Q}$.

Figure 1 shows the annular sector $\bar{Q}$, which is simply the annular sector $Q(R_1, R_2, \theta)$ rotated through an angle $-\eta$ where $\eta \in [0, 2\pi)$ is the angle, measured in the clockwise direction, which the bisecting ray of the sector makes with the negative real axis.

Note that the exterior of $Q(R, 1, \theta)$ is mapped onto the exterior of $\bar{Q}$ by

$$Z = R_2 e^{-i\eta} z.$$

Then $F_n(z) = F_n(Ze^{i\eta}/R_2)$ and the corresponding monic Faber polynomial for $\bar{Q}$ is

$$\bar{F}_n(Z) = R_2^n e^{-in\eta} F_n\left(\frac{Ze^{i\eta}}{R_2}\right).$$

3 The hybrid method

3.1 Arnoldi's method

Arnoldi's method (Arnoldi, 1951) is used to compute m eigenvalue estimates of a nonsymmetric matrix $A \in \mathbb{R}^{n\times n}$. We start with an initial vector $\mathbf{v}_1 \in \mathbb{R}^n$ such that $\|\mathbf{v}_1\|_2 = 1$. An orthonormal basis for the Krylov subspace $K_m := \text{span}\left\{\mathbf{v}_1, \ldots, A^{m-1}\mathbf{v}_1\right\}$ is constructed via the Gram–Schmidt process. For $j = 1, 2, \ldots, m$

$$h_{i,j} := \mathbf{v}_i^T A \mathbf{v}_j \qquad i = 1, \ldots j$$

$$\widehat{\mathbf{v}}_{j+1} := A\mathbf{v}_j - \sum_{i=1}^{j} h_{i,j}\mathbf{v}_i$$

$$h_{j+1,j} := \|\widehat{\mathbf{v}}_{j+1}\|_2\,, \qquad \mathbf{v}_{j+1} := \frac{\widehat{\mathbf{v}}_{j+1}}{h_{j+1,j}}$$

provided $h_{j+1,j}$ is different from zero. Now with $V_m := [\mathbf{v}_1, \ldots, \mathbf{v}_m]$ (so $V_m \in \mathbb{R}^{n\times m}$) and $H_m := [h_{i,j}]_{\,1\le i\le j\le m}$ (so $H_m \in \mathbb{R}^{m\times m}$) we have

$$AV_m = V_m H_m + \widehat{\mathbf{v}}_{m+1}\mathbf{e}_m^T$$

and

$$V_m^T A V_m = H_m$$

with $\mathbf{e}_m = (0, \ldots, 0, 1)^T \in \mathbb{R}^m$.

The eigenvalues $\{\lambda_i\}_{i=1}^m$ of the Hessenberg matrix H_m were used as estimates for the eigenvalues of A.

3.2 Sector determination

Given some eigenvalue estimates $\{\lambda_i\}_{i=1}^m$, the next stage is to place a sector of an annulus around the eigenvalue estimates, and use the known, suitably scaled, Faber polynomials for this region as iteration polynomials.

The modulus and argument of each eigenvalue are found, with the arguments defined on $(-\pi, \pi]$. We set R_{min} to be the smallest eigenvalue modulus and R_{max}

to be the largest eigenvalue modulus. We then place the eigenvalue arguments in increasing order, $\{\alpha_i\}_{i=1}^m$, so that the smallest is first (α_1) and the largest is last (α_m), and look for the largest separation between adjacent arguments. That is we set x to be the largest of $2\pi - (\alpha_m - \alpha_1)$ and $\alpha_{i+1} - \alpha_i$ for $i = 1, \ldots, m-1$. The half–angle of the required sector is $\alpha = (2\pi - x)/2$. Finally we must determine η, which as mentioned previously is the angle the bisecting ray of the sector makes with the negative real axis. If $x = \alpha_{j+1} - \alpha_j$ for some $j \in \{1, \ldots, m-1\}$ then

$$\eta \equiv [\pi - \alpha_{j+1} - \alpha](\mathrm{mod} 2\pi) \in [0, 2\pi),$$

otherwise if $x = 2\pi - (\alpha_m - \alpha_1)$ then

$$\eta \equiv [\pi - \alpha_1 - \alpha](\mathrm{mod} 2\pi) \in [0, 2\pi).$$

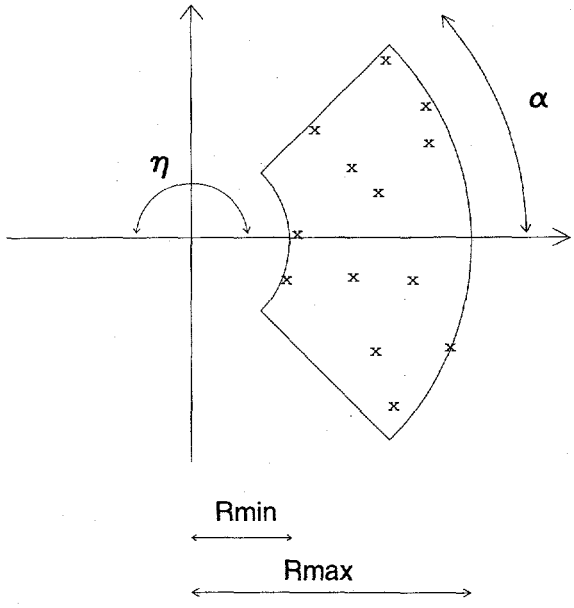


Figure 2. An annular sector placed around some eigenvalue estimates with the radii, the half–angle and the rotation from the negative real axis marked.

As stated above, the Faber polynomials for this region are used as iteration polynomials. Therefore a, b and ρ must be determined for this sector. The first step is to find a, b and ρ for the sector with inner radius R_{min}/R_{max}, outer radius 1 and angle $\theta = \pi - \alpha$, centred on the negative real axis. To do this we use modified Newton iteration with numerical integration to solve the two defining equations for a and b as described in Coleman and Myers (1993). The transfinite diameter is also found by numerical integration. Then we scale and rotate this sector onto the required sector, and change the Faber polynomials accordingly.

3.3 The iteration polynomial

Starke and Varga (1993), amongst others, have shown that Faber polynomials can be very useful for polynomial matrix iterations. Not only are Faber polynomials near–best with respect to the maximum norm, but their norms are also small on level sets for the region on which they are defined. This is a good property for dealing with non–normal matrices where eigenvalues can be very sensitive to perturbations.

Many iterative methods for solving (1) can be written as,

$$\mathbf{x}_m = \mathbf{x}_0 + q_{m-1}(A)\mathbf{r}_0,$$

where q_{m-1} a polynomial of degree $m-1$, is called the iteration polynomial. We define the error $\mathbf{e}_m := A^{-1}\mathbf{b} - \mathbf{x}_m$ and the residual $\mathbf{r}_m := \mathbf{b} - A\mathbf{x}_m$. Then

$$\mathbf{r}_m = \mathbf{b} - A(\mathbf{x}_0 + q_{m-1}(A)\mathbf{r}_0) = (I - Aq_{m-1}(A))\mathbf{r}_0$$

and

$$\mathbf{e}_m = A^{-1}\mathbf{r}_m = (I - Aq_{m-1}(A))\mathbf{e}_0.$$

With the residual polynomial $p_m(z) = 1 - zq_{m-1}(z)$, so that $p_m(0) = 1$, these two equations can be written as $\mathbf{r}_m = p_m(A)\mathbf{r}_0$ and $\mathbf{e}_m = p_m(A)\mathbf{e}_0$. Therefore, for any consistent pair of matrix and vector norms on $\mathbb{C}^n$

$$\|\mathbf{r}_m\| \leq \|p_m(A)\|\|\mathbf{r}_0\| \quad \text{and} \quad \|\mathbf{e}_m\| \leq \|p_m(A)\|\|\mathbf{e}_0\| \qquad (m \geq 1),$$

and the aim is to choose polynomials p_m, in the set

$$\Pi_m = \{ \text{ polynomials of degree } m \mid p_m(0) = 1\},$$

such that $\|p_m(A)\|$ is as small as possible. For our residual polynomial, we will choose

$$p_m(z) = \frac{F_m(z)}{F_m(0)}.$$

4 Implementation

We will use the number of vector operations (a vector operation is n scalar multiplications and n scalar additions, where n is the dimension of the system) as an indication of the speed of convergence of the method (Nachtigal et al. 1992 and Starke and Varga, 1993). If we let l denote the average number of elements per row in the matrix A, then a matrix–vector multiplication costs l vector operations.

Firstly we must consider the cost of Arnolid's method, that is how many vector operations it takes to get m_1 eigenvalue estimates for the matrix A. For the j^{th} step, $j = 1, \ldots, m_1$, of Arnoldi's method, calculating $\mathbf{w}_j = A\mathbf{v}_j$ involves l vector operations, calculating $h_{i,j} = \mathbf{v}_i^T\mathbf{w}_j \quad$ for $i = 1, \ldots j$ involves j vector operations, computing $\widehat{\mathbf{v}}_{j+1} = \mathbf{w}_j - \sum_{i=1}^{j} h_{i,j}\mathbf{v}_i$ requires j vector operations, and finally the norm involves

one vector operation. So in total the number of vector operations involved in Arnoldi's method is

$$\sum_{j=1}^{m_1}(l+j+j+1) = m_1\,[l+2+m_1]\,.$$

Secondly we consider how many vector operations are involved in implementing a polynomial iterative method using a Faber polynomial of degree m. As in Starke and Varga (1993) we will implement the iteration polynomial, $q_{m-1}(z) = (F_m(0) - F_m(z))/F_m(0)z = \alpha_{0,m} + \alpha_{1,m}z + \ldots + \alpha_{m-1,m}z^{m-1}$ in a Horner-type iteration. This requires $m(l+1)$ vector operations. The Horner iteration is of the form

$$\mathbf{w}_0 = \alpha_{m-1,m}\mathbf{r}_{\text{old}},$$

$$\mathbf{w}_j = A\mathbf{w}_{j-1} + \alpha_{m-1-j,m}\mathbf{r}_{\text{old}}, \qquad j = 1,\ldots,m-1,$$

$$\mathbf{x}_{\text{new}} = \mathbf{x}_{\text{old}} + \mathbf{w}_{m-1},$$

$$\mathbf{r}_{\text{new}} = \mathbf{b} - A\mathbf{x}_{\text{new}}.$$

In the examples, we neglect the work involved in choosing the particular sector of the annulus, as this should be negligible compared to the work involved in the iterations.

5 Examples

We now consider three examples of nonsymmetric matrices, the first two of which are examples 6.2 and 6.3 of Starke and Varga's paper (1993). Unless otherwise stated the degree of the iteration polynomial will be the same as the number of eigenvalue estimates taken. The curve which starts furthest to the left in the figures will be the one with the least number of eigenvalue estimates.

Example 1

We consider discretising the boundary value problem

$$\begin{aligned} -\Delta u + \tau u_x &= f(x,y), \quad (x,y) \in S \\ u(x,y) &= g(x,y), \quad (x,y) \in \partial S, \end{aligned}$$

by central differences on the unit square $S := (0,1) \times (0,1)$ with boundary ∂S. This leads to solving a system of equations

$$(B \otimes I_N + I_N \otimes C)\mathbf{x} = \mathbf{b}$$

where $N = 32$,

$$B = \begin{pmatrix} 2 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{pmatrix} \quad \text{and} \quad C = \begin{pmatrix} 2 & -1+\mu & & \\ -1-\mu & \ddots & \ddots & \\ & \ddots & \ddots & -1+\mu \\ & & -1-\mu & 2 \end{pmatrix}$$

with $\mu = \tau(h/2)$, where h is the meshsize of the discretisation.

Following Starke and Varga we consider this example, with different initial estimates $\mathbf{x}_0$ and right hand sides $\mathbf{b}$. The results are shown in Figures 3 to 6. In all cases our method does at least as well as Starke and Varga's. When we consider non random $\mathbf{x}_0$ and $\mathbf{b}$ our method seems to converge faster than Starke and Varga's. For example when we have $\mathbf{b} = (1, \ldots, 1)^T$, $\mathbf{x}_0 = \mathbf{0}$ (Figure 6), our method using 32 eigenvalue estimates and a polynomial of degree 32 converges to 10^{-11} as fast as Starke and Varga's ARNOLDI/FABER(40) does, and certainly faster than their ARNOLDI/FABER(32) does.

Example 2

The Grcar matrix example. In this example we consider the matrix

$$A = \begin{pmatrix} \frac{9}{10} & 1 & 1 & 1 & & & \\ -1 & \frac{9}{10} & 1 & 1 & 1 & & \\ & -1 & \frac{9}{10} & 1 & 1 & 1 & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots \end{pmatrix} \in \mathbb{R}^{1024 \times 1024}$$

This is example 6.3 in Starke and Varga's paper, it is a shifted version of a matrix in a paper by Trefethen (1992), which originated in a paper by J. Grcar (1989). Starke and Varga chose it to illustrate that their method would even work if some of the spectrum of the matrix was situated in the left–half plane. Starke and Varga also point out that it is surprising their method would even work at all because some of the spectrum of the matrix is not included in their polygonal region.

The results for this matrix with a random $\mathbf{b}$ and $\mathbf{x}_0$ are shown in Figure 7. Our method with 24 eigenvalue estimates converges whereas Starke and Varga's did not. However, it is interesting that with 48 eigenvalue estimates our method and Starke and Varga's are comparable.

Example 3

Finally we consider the matrix

$$A = \begin{pmatrix} 1 & 1 & & & & & \\ & -1 & 1 & & & & \\ & & 1 & 1 & & & \\ & & & -1 & 1 & & \\ & & & & \ddots & \ddots & \end{pmatrix} \in \mathbb{R}^{200 \times 200}$$

This is a matrix whose eigenvalues are real and situated on both sides of the origin, so Starke and Varga's method (and many other iterative methods) cannot be used. Once again we use a random $\mathbf{b}$ and $\mathbf{x}_0$. The results using an iteration polynomial of degree 16 with 16 and 32 eigenvalue estimates are shown in Figure 8. Although the method diverges initially it eventually converges, and because our calculations are done in double precision the method only converges to 10^{-2} and 10^{-5} respectively.

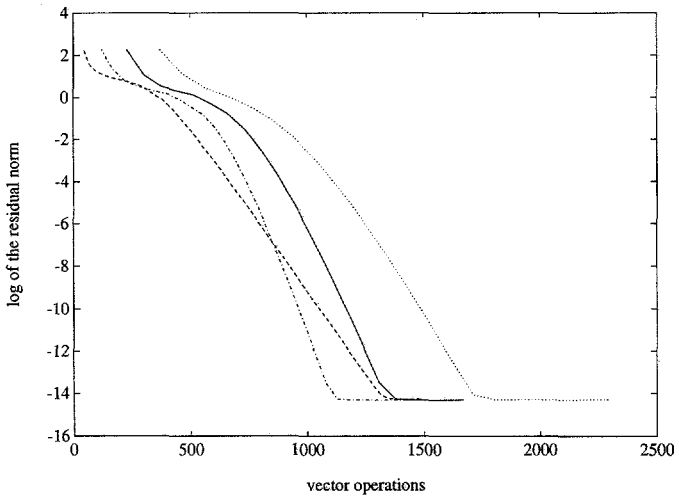


Figure 3. Example 1, $\mu = 2$, random starting vectors, eigenvalue estimates 4,8,12,16.

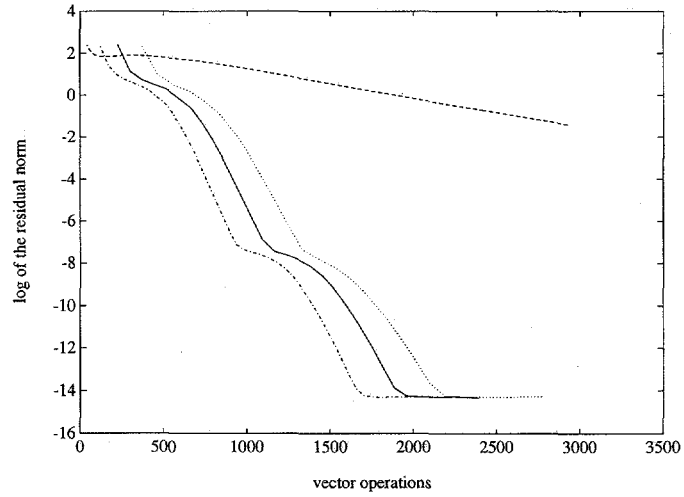


Figure 4. Example 1, $\mu = 4$, random starting vectors, eigenvalue estimates 4,8,12,16.

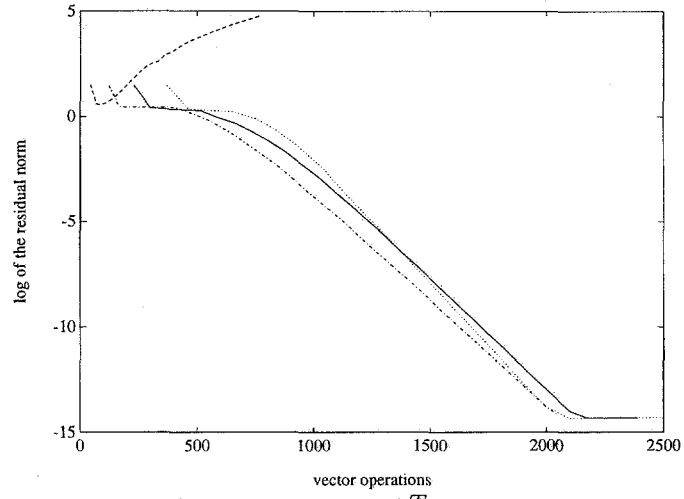


Figure 5. Example 1, $\mu = 2$, $\mathbf{b} = (-1, 1, \ldots, -1, 1)^T$, $\mathbf{x}_0 = \mathbf{0}$, eigenvalue estimates 4,8,12,16.

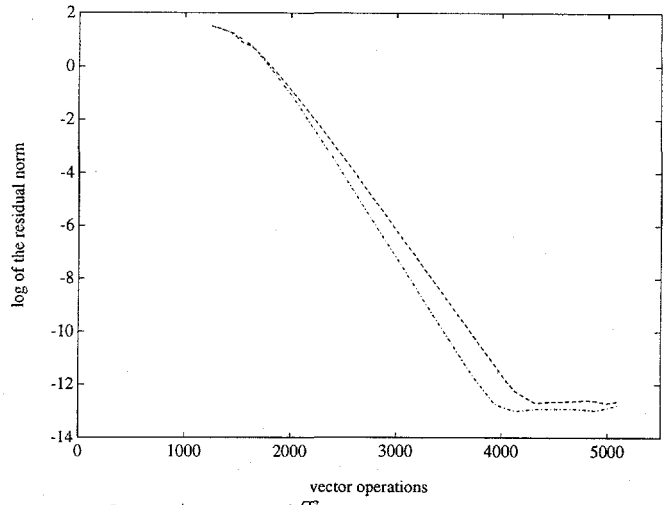


Figure 6. Example 1, $\mu = 2$, $\mathbf{b} = (1, \ldots, 1)^T$, $\mathbf{x}_0 = \mathbf{0}$, eigenvalue estimates 32, degree 32,16.

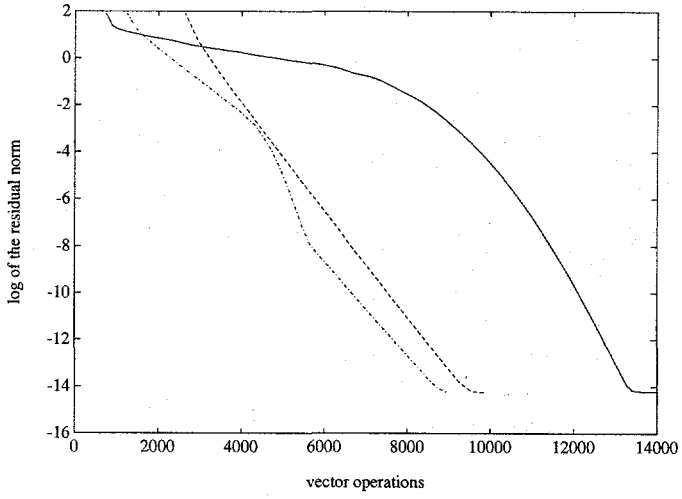


Figure 7. Example 2, random starting vectors, eigenvalue estimates 24,32,48.

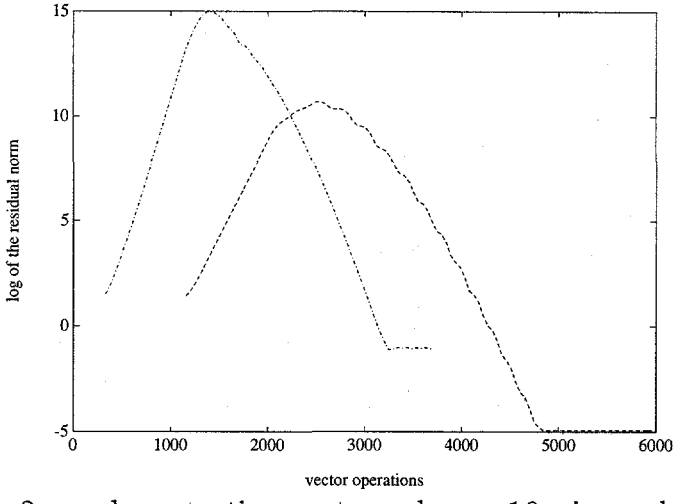


Figure 8. Example 3, random starting vectors, degree 16, eigenvalue estimates 16,32.

References

1. W.E. Arnoldi, *The principle of minimized iterations in the solution of the matrix eigenvalue problem*, Quart. Appl. Math. **9** (1951), 17-29.
2. J.P. Coleman and N.J. Myers, *The Faber polynomials for annular sectors*, Preprint (1993)
3. J.F. Grcar, *Operator Coefficient Methods for Linear Equations*, Sandia Report: SAND 89-8691, 1989.
4. N.M. Nachtigal, S.C. Reddy, L.N. Trefethen, *How fast are nonsymmetric matrix iterations?*, SIAM J. Matrix Anal. Appl. **13** (1992), 778-795.
5. G. Starke and R.S. Varga, *A hybrid Arnoldi-Faber iterative method for nonsymmetric systems of linear equations*, Numer. Math. **64** (1993).
6. L.N. Trefethen, *Pseudospectra of matrices*, In D.F. Griffiths and G.A. Watson, *Numerical Analysis 1991*, Longman, 234-266, 1992.

SUBSPACE ORTHOGONALIZATION FOR SUBSTRUCTURING PRECONDITIONERS FOR NONSYMMETRIC SYSTEMS OF LINEAR EQUATIONS

GERHARD STARKE*

Abstract. For nonselfadjoint elliptic boundary value problem which are preconditioned by a substructuring method, i.e., nonoverlapping domain decomposition, we introduce and study the concept of subspace orthogonalization. In subspace orthogonalization variants of Krylov methods the computation of inner products and vector updates, and the storage of basis elements is restricted to a (presumably small) subspace, in this case the edge and vertex unknowns with respect to the partitioning into subdomains. We investigate subspace orthogonalization for two specific iterative algorithms, the generalized minimal residual algorithm (GMRES) and the full orthogonalization method (FOM). This is intended to eliminate certain drawbacks of the Arnoldi-based Krylov subspace methods mentioned above. Above all, the length of the Arnoldi recurrences grows linearly with the iteration index which is therefore restricted to the number of basis elements that can be held in memory. Restarts become necessary and this often results in much slower convergence. The subspace orthogonalization methods, in contrast, require the storage of only the edge and vertex unknowns of each basis element which means that one can iterate much longer before restarts become necessary. Moreover, the computation of inner products is also restricted to the edge and vertex points which avoids the disturbance of the computational flow associated with the solution of subdomain problems. We view subspace orthogonalization as an alternative to restarting or truncating Krylov subspace methods for nonsymmetric linear systems of equations. Instead of shortening the recurrences, we restrict them to a subset of the unknowns which has to be carefully chosen in order to be able to extend this partial solution to the entire space. We discuss the convergence properties of these iteration schemes and its advantages compared to restarted or truncated versions of Krylov methods applied to the full preconditioned system. Subspace orthogonalization can be applied to general elliptic problems divided into substructures and, in a purely algebraic framework, to arbitrary systems of linear equations decoupled by node separators. In our computational experiments, however, we will focus on a specific preconditioner for elliptic boundary value problems in two dimensions which was introduced and analyzed in a recent paper by Cai, Gropp and Keyes.

* Institut für Praktische Mathematik, Universität Karlsruhe, Englerstrasse 2, D–76128 Karlsruhe, Germany. E-mail **starke@ipmsun1.mathematik.uni-karlsruhe.de**

Wednesday, April 6

Parallel Computation I
Chair: Howard Elman
Room B

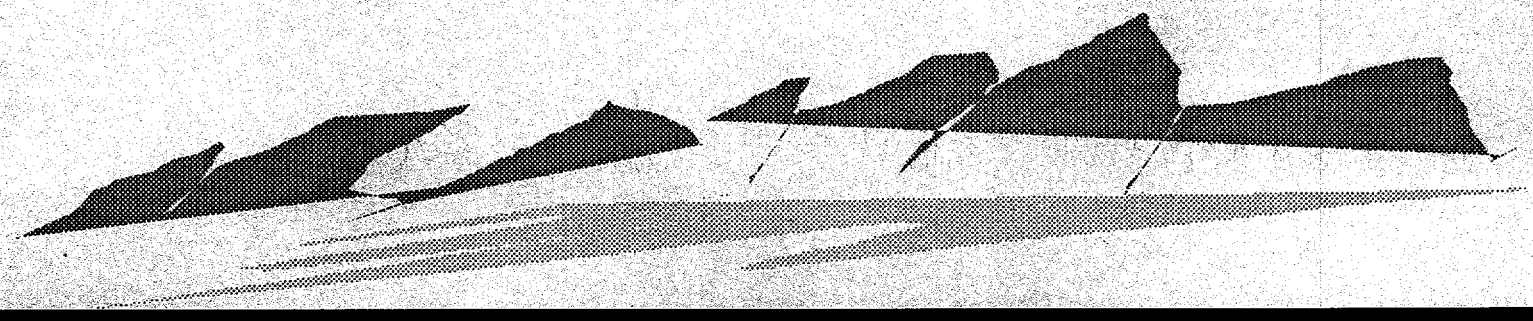
8:00 - 8:25 Wayne Joubert
PCG: A Software Package for the Iterative Solution of Linear Systems on Scalar, Vector Parallel Computers

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



Wayne Joubert
(505) 665-7374
e-mail: wdj@lanl.gov

PCG: A Software Package for the Iterative Solution of Linear Systems on Scalar, Vector and Parallel Computers

A great need exists for high performance numerical software libraries transportable across parallel machines. This talk concerns the PCG package, which solves systems of linear equations by iterative methods on parallel computers. The features of the package are discussed, as well as techniques used to obtain high performance as well as transportability across architectures. Representative numerical results are presented for several machines including the Connection Machine CM-5, Intel Paragon and Cray T3D parallel computers.

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

G. F. Carey
The University of Texas
Department of Aerospace Engineering
and Engineering Mechanics
Austin, TX 78712

MIGRATION OF VECTORIZED ITERATIVE SOLVERS TO DISTRIBUTED MEMORY ARCHITECTURES

CLAUDE POMMERELL* AND ROLAND RÜHL†

1. Introduction. Both necessity and opportunity motivate the use of high-performance computers for iterative linear solvers. Necessity results from the size of the problems being solved—smaller problems are often better handled by direct methods. Opportunity arises from the formulation of the iterative methods in terms of simple linear algebra operations, even if this "natural" parallelism is not easy to exploit in irregularly structured sparse matrices and with good preconditioners.

As a result, high-performance implementations of iterative solvers have attracted a lot of interest in recent years. Most efforts are geared to vectorize or parallelize the dominating operation—structured or unstructured sparse matrix-vector multiplication, or to increase locality and parallelism by reformulating the algorithm—reducing global synchronization in inner products or local data exchange in preconditioners.

Target architectures for iterative solvers currently include mostly vector supercomputers and architectures with one or few optimized (e.g., super-scalar and/or super-pipelined RISC) processors and hierarchical memory systems. More recently, parallel computers with physically distributed memory and a better price/performance ratio have been offered by vendors as a very interesting alternative to vector supercomputers. However, programming comfort on such distributed memory parallel processors (DMPPs) still lags behind.

In this paper, we are concerned with iterative solvers and their changing computing environment. In particular, we are considering migration from traditional vector supercomputers to DMPPs. Application requirements force us to use flexible and portable libraries. We want to extend the portability of iterative solvers rather than reimplementing everything for each new machine, or even for each new architecture.

Several research groups have defined languages and implemented compilers that increase DMPP programming comfort and allow portable coding at least of applications requiring highly structured computations, like dense matrix linear algebra, or finite difference and finite element methods applied to regular grids. The definition of High Performance Fortran (HPF) standard [5] is a try to standardize various efforts based on Fortran language extensions. However, HPF only provides minimal support for the parallelization of more irregular computations like linear algebra on general sparse matrices, or finite element or finite volume methods applied to irregularly refined grids. For the efficient parallelization of such applications, a high-level language DMPP compiler must support extensive run-time analysis, and the language—in contrast to HPF—must include constructs to dynamically distribute data and control flow.

2. Porting PILS with Oxygen. PILS [9] is a Package of Iterative Linear Solvers targeted to the solution of very large, irregularly sparse, unsymmetric, ill-conditioned systems of linear equations. It is integrated into a number of applications, including several semiconductor device simulators, where the solution of hundreds of ill-conditioned linear systems with an irregular sparsity structure dominates the

* AT&T Bell Laboratories, 600 Mountain Avenue, Murray Hill, NJ 07974-0636, USA
† CSCS–ETH, Swiss Scientific Computing Center, CH-6928 Manno, Switzerland

overall execution time [3]. PILS and its client applications have been used regularly over three years now, at several dozens of academic and industrial sites.

Not all linear systems arising in the client applications can be solved efficiently by only one preconditioned method. PILS therefore includes a large number of iterative methods, preconditioners, and other variants for iterative solvers, and combines them in a flexible and automatically adapting way. PILS runs on RISC workstations and vector supercomputers from all major manufacturers. Sparse matrices are stored in colored jagged diagonals based on a partitioning by matchings [13, 7]. This data structure is optimized to achieve a high level of vectorization on regular and transposed matrix-vector multiplications and the solution of sparse triangular systems for incomplete factorization preconditioners. The operations are coded in Fortran with compiler directives in a way that their optimization for a given pipelined computer is easy for the manufacturer's vectorizing compiler. The data structures themselves, as well as all other tasks that do not require vectorization for efficiency, are handled in C++ code.

In the present project, we extended the portability of PILS to DMPPs, by using the parallelizing Fortran compiler Oxygen [12]. Oxygen compiles for a variety of DMPP platforms, including the Intel Paragon and iWARP, the Parsytec SC256 [11], the Fujitsu AP1000 [4], and Thinking Machines' CM5 [14]. The latter two machines were used to evaluate our strategy quantitatively.

In this project, Oxygen's capabilities for automatic parallelism detection [6] were not used. That is, Oxygen was used to compile sequential Fortran enhanced with user-specified parallelization directives. This input language features a global name space much like HPF. In contrast to compilers of languages which had some influence on the HPF definition (e.g., Kali, Crystal, Fortran D, Superb, Arf), and also in contrast to the first HPF compilers commercially available (e.g., the HPF subset compiler from Applied Parallel Research), Oxygen includes several features that make it especially well suited for supporting the parallelization of irregular computations in general and PILS in particular:

- Oxygen directives include constructs to dynamically distribute data and control-flow in parallel programs.
- Run-time analysis supports a global name space even in program segments that include arbitrarily nested dependences on elements of distributed arrays. This mechanism is more general than the generation of inspector/executor pairs supported by other systems [2].
- Not only remote fetches, but also *remote updates* of distributed data are allowed. That is, we do not restrict ourselves to the "owner-computes rule".
- All processors in Oxygen-generated parallel programs execute duplicates of the sequential part, and synchronize only implicitly through local communications as required by data exchanges. Many of the above mentioned compilation systems use collective communication routines which implies at least one global synchronization at the end of each parallel loop.

Some of the above features are supported also by few other experimental systems (for instance, remote updates by Arf). However, for the DMPP parallelization of PILS, *all* features were crucial.

The irregular sparsity structure of the matrices is run-time information that Oxygen translates into message-based communication at the point where this information is used in the algorithm. Remote updates make transposed matrix operations just as transparent and efficient as regular matrix operations. Asynchronous overlapping of

different phases of the computation, particularly during different colors in the solution of sparse triangular systems within incomplete factorization preconditioners, reduces total execution time by ten percent or more in comparison to a version with global synchronizations between parallel loops.

We chose to execute all sequential parts of the solver software—including the construction of data structures and the interface to client applications—on the host of the parallel machine. The two parts of the package communicate via a remote procedure call interface, enhanced by a software cache for distributed data (such as matrices and vectors).

This separation has the consequence that the client application does not see and does not need to know whether the version of the solver library it is using runs sequentially on the host, or in parallel on the attached DMPP. Thus, the client application does not need any adjustment. Furthermore, preconditioners based on approximate factorization with numerical dropping, specialized for certain very ill-conditioned linear systems, run on the host without any additional performance penalty for their lack of parallelism. The separation of sequential and parallel parts of the code was actually present in PILS as the separation of data structuring written in C++ and vectorizable parts written in Fortran.

Mapping neighboring vertices in the sparse graph of the matrix to the same or neighboring processors is crucial in order to achieve locality in the communication patterns during matrix-vector multiplication. Although the PILS code was originally targeted to vector computers, the only essential addition to it, for efficient DMPP parallelization with Oxygen, was the implementation of a two-dimensional geometric mapping heuristic [8].

3. Experimental results. Table 1 summarizes the characteristics of the test problems we used to evaluate our approach. All problems are extracted from real simulations within several semiconductor device simulators, and selected to display a typical variety of problem sizes and complexities within typical large device simulations.

Table 2 reports benchmarks using Bi-CGSTAB as the iterative solution method, preconditioned by a D-ILU preconditioner in split position. Due to the non-deterministic summation sequence in the manufacturer-provided global reduction routines, the number of iterations for convergence often varies slightly from one run to another; therefore, we expressed speedup in Table 2 only as the time for one serial iteration over the time for one parallel iteration. The CM5 achieves less speedup than the AP1000, because of a difference in *communication/computation speed ratio* [1].

Table 3 details experiments with several other variants of iterative solvers in PILS, measured on an 8×8 AP1000. Several of these variants are required for different kinds of linear systems occurring in real problems.

More measurements and further details on the parallelization strategy can be found in [10].

REFERENCES

[1] M. Annaratone, C. Pommerell, and R. Rühl, *Interprocessor communication speed and performance in distributed-memory parallel processors*, in Proc. 16th Symposium on Computer Architecture, Jerusalem, June 1989, IEEE-ACM, pp. 315–324.

TABLE 1
Problem characteristics.

	LDDH	DR15E	BIPOL3D20KH	BP25E	DR15C	BP25C
Grid dimension	2-D	3-D	3-D	3-D	3-D	3-D
# PDEs	1	1	1	1	3	3
# unknowns	2674	15564	20412	25642	46692	76926
# nonzeros	18614	143710	263920	234436	986042	1618414
matrix density (nz/row)	7.0	9.2	13.0	9.1	21.1	21.0

TABLE 2
Execution times per iteration in seconds and speedup of the PILS Fortran code. Table entries denoted with n.a. (not available) could not be filled due to the memory requirements of the largest problems. All parallel times used in this paper stem from real measurements. However, some of the problems were too large to fit on a single PE. In these cases, we extrapolated serial time from measurements on SparcStations with comparable performance characteristics. Speedup numbers based on such extrapolations are printed in italics.

machine	#processors	Problem LDDH	DR15E	BIPOL3D20KH	BP25E	DR15C	BP25C
		serial execution times (s)					
CM5	1	0.14	0.76	1.23	1.23	*4.3*	*7.1*
AP1000	1	0.20	1.46	*2.4*	*2.5*	*8.6*	*14.2*
SparcStation 1+	1	0.16	1.15	1.89	1.98	6.72	n.a.
SparcStation 2	1	0.09	0.69	1.13	1.15	4.03	6.68
		parallel execution times (s)					
CM5	32	0.044	0.126	0.247	0.502	0.552	1.913
CM5	64	0.056	0.101	0.165	0.295	0.444	0.976
AP1000	16	0.035	0.188	0.326	0.474	1.059	n.a.
AP1000	32	0.025	0.104	0.182	0.295	0.571	n.a.
AP1000	64	0.019	0.067	0.109	0.176	0.346	0.750
AP1000	128	0.017	0.045	0.072	0.127	0.213	0.500
		Speedup					
CM5	32	3.2	6.0	5.0	2.5	*7.8*	*3.7*
CM5	64	2.5	7.5	7.5	4.2	*9.7*	*7.3*
AP1000	16	5.7	7.8	*7.4*	*5.3*	*8.1*	n.a.
AP1000	32	8.0	14.0	*13.2*	*8.5*	*15.1*	n.a.
AP1000	64	10.5	21.8	*22.0*	*14.2*	*24.9*	*18.9*
AP1000	128	11.7	32.4	*33.3*	*19.7*	*40.4*	*28.4*

TABLE 3

Timing of various variants of methods, for one particular problem on an 64-processor AP1000. The total solution time is the sum of the PE time, the host communication time, and the matrix load time. The latter is 44 seconds with ILU preconditioning, and 22 seconds for all the other variants. The PE time is compared to performance on a SparcStation 1+ in the rightmost column.

Iterative Method	*Precon-ditioner*	*Iter-ations*	*Time (seconds) on slowest PE*	*Time (seconds) for host-PE protocol*	*PE speedup over SS1+*
reference					
Bi-CGSTAB	split D-ILU	65	23.1	6.4	18.9
other methods					
BiCG	split D-ILU	130	45.9	10.7	19.4
CGS	split D-ILU	78	27.2	7.5	19.9
GMRES(10)	split D-ILU	388	87.4	42.1	19.3
GMRES(∞)	split D-ILU	89	41.6	37.4	24.0
other preconditioners					
Bi-CGSTAB	right D-ILU	67	33.4	8.5	17.8
Bi-CGSTAB	split ILU	61	38.0	5.8	18.1
Bi-CGSTAB	split SSOR	69	24.6	7.3	17.9
nested iterative solvers					
GCR(∞)	GMRES	17	43.0	21.1	19.2
GCR(∞)	Bi-CGSTAB	11	41.4	15.5	17.7

[2] K. CROWLEY, J. SALTZ, R. MIRCHANDANEY, AND H. BERRYMAN, *Run-time scheduling and execution of loops on message passing machines*, Tech. Rep. 89-7, ICASE, NASA Langley Research Center, January 1989.

[3] G. HEISER, C. POMMERELL, J. WEIS, AND W. FICHTNER, *Three dimensional numerical semiconductor device simulation: Algorithms, architectures, results*, IEEE Trans. Comput.-Aided Design Integrated Circuits, 10 (1991), pp. 1218–1230.

[4] H. ISHIHATA ET AL., *An architecture of highly parallel computer AP1000*, in Pacific Rim Conference on Communications, Computers and Signal Processing, IEEE, May 1991, pp. 13–16.

[5] D. B. LOVEMAN, *High performance Fortran*, IEEE Parallel & Distributed Technology, (1993).

[6] M. NEERACHER AND R. RÜHL, *Automatic parallelization of LINPACK routines on distributed memory parallel processors*, in Proc. IEEE International Parallel Processing Symposium, New Port Beach, California, April 1993.

[7] C. POMMERELL, *Solution of Large Unsymmetric Systems of Linear Equations*, PhD thesis, ETH-Zürich, 1992. publ. by Hartung-Gorre Verlag, Konstanz, Germany.

[8] C. POMMERELL, M. ANNARATONE, AND W. FICHTNER, *A set of new mapping and coloring heuristics for distributed-memory parallel processors*, SIAM J. Sci. Stat. Comput., 13 (1992), pp. 194–226.

[9] C. POMMERELL AND W. FICHTNER, *PILS: An iterative linear solver package for ill-conditioned systems*, in Proc. Supercomputing '91, Albuquerque, Nov. 1991, ACM-IEEE, pp. 588–599.

[10] C. POMMERELL AND R. RÜHL, *Compiler assisted distributed memory parallelization of an iterative solver for irregular sparse linear systems*, Numerical Analysis Manuscript 93–12, AT&T Bell Laboratories, Apr. 1993. Also available as Technical Report CSCS-TR-93-03, Swiss Scientific Computing Center.

[11] R. RÜHL, *Evaluation of compiler generated parallel programs on three multicomputers*, in Proc. ACM International Conference on Supercomputing, Washington, July 1992.

[12] R. RÜHL, *A Parallelizing Compiler for Distributed-Memory Parallel Processors*, PhD thesis, ETH-Zürich, 1992. publ. by Hartung-Gorre Verlag, Konstanz, Germany.

[13] Y. SAAD, *Krylov subspace methods on supercomputers*, SIAM J. Sci. Stat. Comput., 10 (1989), pp. 1200–1232.

[14] *CM-5 technical summary*, tech. rep., Thinking Machines Corporation, October 1991.

P_SPARSLIB: a Parallel Sparse Iterative Solution Package.

Youcef Saad
University of Minnesota
Department of Computer Science
Minneapolis, MN 55455

December 14, 1993

Abstract

Iterative methods are gaining popularity in engineering and sciences at a time where the computational environment is changing rapidly. P_SPARSLIB is a project to build a software library for sparse matrix computations on parallel computers. The emphasis is on iterative methods and the use of distributed sparse matrices, an extension of the domain decomposition approach to general sparse matrices.

One of the goals of this project is to develop a software package geared towards specific applications. For example, we will test the performance and usefulness of P_SPARSLIB modules on linear systems arising from CFD applications. Equally important is the goal of portability. In the long run, we wish to ensure that our package is portable on a variety of platforms, including SIMD environments and shared memory environments.

Currently, we are using the CM-5 as our development platform and we are focussing our current efforts on message-passing distributed memory environments. Thus, we have recently implemented most of the important iterative solvers such as GMRES, BiCG, BiCGSTAG, TFQMR, DQGMRES, etc., as well as a few of the standard parallel preconditioners, such as multicolor (block) SOR, SSOR, and the wavefront (block) ILU(0). We have also developed a number of (new) graph partitioning algorithms which allow to partition a given graph automatically.

The concept of distributed sparse matrices exploits partitionings of the adjacency graph into subgraphs. Similarly to the PDE framework, the key idea is to recover the global solution from the separate solutions on the nodes associated with each subgraph. We will first present a number of tools that are devoted to implementing these 'graph decomposition' methods. These include graph partitioners, coloring algorithms, and a few other preprocessing algorithms. We will then describe a few preconditioning techniques designed for distributed sparse matrices. These techniques include block SOR, multicolor Block ILU, block Jacobi with overlap, as well as a preconditioning technique based on the Schur complement approach.

Portable, Parallel, Reusable Krylov Space Codes

Barry Smith
Department of Mathematics
405 Hilgard Ave.
Los Angeles, CA 90024-1555
bsmith@math.ucla.edu

William Gropp
MCS Division
Argonne National Laboratory
9700 South Cass Ave.
Argonne, IL 60439-4844
gropp@mcs.anl.gov

Krylov space accelerators are an important component of many algorithms for the iterative solution of linear systems. Each Krylov space method has it's own particular advantages and disadvantages, therefore it is desirable to have a variety of them available all with an identical, easy to use, interface.

A common complaint application programmers have with available software libraries for the iterative solution of linear systems is that they require the programmer to use the data structures provided by the library. The library is not able to work with the data structures of the application code. Hence, application programmers find themselves constantly recoding the Krylov space algorithms.

The Krylov space package (KSP) is a data-structure-neutral implementation of a variety of Krylov space methods including preconditioned conjugate gradient, GMRES, BiCG-Stab, transpose free QMR and CGS. Unlike all other software libraries for linear systems that we are aware of, KSP will work with any application codes data structures, in Fortran or C. Due to it's data-structure-neutral design KSP runs unchanged on both sequential and parallel machines. KSP has been tested on workstations, the Intel i860 and Paragon, Thinking Machines CM-5 and the IBM SP1.

We will discuss the design philosophy of KSP and how we are able to provide an object oriented library to pre-existing applications. To illustrate the power of this approach we mention an application to magnetostatics that involves solving a large dense, (but well conditioned) linear system in the inner loop. By using the KSP it was a minor change to the application code from using direct methods with LAPACK to Krylov space iterative methods.

KSP is part of our Portable, Extensible, Toolkit for Scientific computation (PETSC). PETSc is a large toolkit of software for portable, parallel (and serial) scientific computation. All of PETSc is available by anonymous ftp from info.mcs.anl.gov in the directory pub/pdetools.

Wednesday, April 6

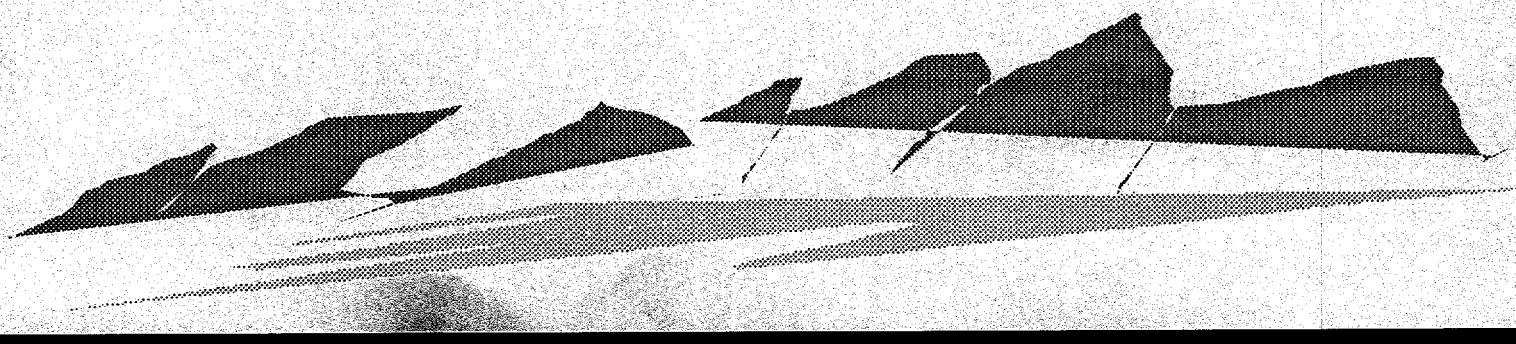
Nonsymmetric Solvers II
Chair: Tom Manteuffel
Room A

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



Minimal Residual Method Stronger than Polynomial Preconditioning

Two popular methods for solving symmetric and nonsymmetric systems of equations are the minimal residual method, implemented by algorithms such as GMRES, and polynomial preconditioning methods. In this study results are given on the convergence rates of these methods for various classes of matrices. It is shown that for some matrices, such as normal matrices, the convergence rates for GMRES and for the optimal polynomial preconditioning are the same, and for other matrices such as the upper triangular Toeplitz matrices, it is at least assured that if one method converges then the other must converge. On the other hand, it is shown that matrices exist for which restarted GMRES always converges but any polynomial preconditioning of corresponding degree makes no progress toward the solution for some initial error. The implications of these results for these and other iterative methods are discussed.

Vance Faber
Los Alamos National Laboratory
Group C-3, MS B265
Los Alamos NM 87545

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

Emanuel Knill
Los Alamos National Laboratory
Group C-3, MS B265
Los Alamos NM 87545

Thomas Manteuffel
University of Colorado at Boulder
Boulder CO

Peaks, plateaus, numerical instabilities,and achievable accuracy in Galerkin and norm minimizing procedures for solving $Ax = b$

Jane Cullum*

December 14, 1993

Abstract

Plots of the residual norms generated by Galerkin procedures for solving $Ax = b$ often exhibit strings of irregular peaks. At seemingly erratic stages in the iterations, peaks appear in the residual norm plot, intervals of iterations over which the norms initially increase and then decrease. Plots of the residual norms generated by related norm minimizing procedures often exhibit long plateaus, sequences of iterations over which reductions in the size of the residual norm are unacceptably small. In an earlier paper [1] we discussed and derived relationships between such peaks and plateaus within corresponding Galerkin/Norm Minimizing pairs of such methods.

In this paper, through a set of numerical experiments, we examine connections between peaks, plateaus, numerical instabilities, and the achievable accuracy for such pairs of iterative methods. Three pairs of methods, GMRES/Arnoldi, QMR/BCG, and two bidiagonalization methods are studied.

*IBM T.J. Watson Research Center, Yorktown Heights, NY 10598

References

[1] Jane Cullum and Anne Greenbaum (1993). Residual Relationships within Three Pairs of Iterative Algorithms, for Solving $Ax = b$, IBM Research RC 18672, January 1993, IBM T.J. Watson Research Center, Yorktown Heights, New York, 10598. Submitted to SIAM J. Mat. Anal. Applics.

COMPUTATIONAL TRIGONOMETRY

Karl Gustafson

Department of Mathematics
University of Colorado
Boulder, Colorado 80309–0395

Abstract. By means of my earlier theory of antieigenvalues and antieigenvectors, a new computational approach to iterative methods is presented. This enables an explicit trigonometric understanding of iterative convergence and provides new insights into the sharpness of error bounds. Direct applications to Gradient descent, Conjugate gradient, GCR(k), Orthomin, CGN, GMRES, CGS, and other matrix iterative schemes will be given.

Keys to this theory are my minmax theorem [4] for bounded strongly accretive operators

$$\sup_{\|x\|=1} \inf_{-\infty<\epsilon<\infty} \|(\epsilon B - I)x\|^2 = \inf_{-\infty<\epsilon<\infty} \sup_{\|x\|=1} \|(\epsilon B - I)x\|^2,$$

and my nonlinear Euler equation [5,7]

$$2\|Au\|^2\|u\|^2(\mathrm{Re}\, A)u - \|u\|^2 \mathrm{Re}\,\langle Au, u\rangle A^*Au - \|Au\|^2 \mathrm{Re}\,\langle Au, u\rangle u = 0,$$

which is generally satisfied by both the first antieigenvectors and all of the eigenvectors of A. This new theory will be seen to be a significant extension of the Rayleigh–Ritz theory variationally characterizing eigenvalues and as such will comprise a new spectral theory which will have numerous uses.

[1] K. GUSTAFSON, *The angle of an operator and positive operator products*, Bull. Amer. Math. Soc. **74** (1968), 488–492.
[2] K. GUSTAFSON, *Positive (noncommuting) operator products and semigroups*, Math. Zeit. **105** (1968), 160–172.
[3] K. GUSTAFSON, *A note on left multiplication of semigroup generators*, Pac. J. Math. **24** (1968), 463–465.
[4] K. GUSTAFSON, *A min-max theorem*, Notices Amer. Math. Soc. **15** (1968), p. 799.
[5] K. GUSTAFSON, *Antieigenvalue inequalities in operator theory*, in Inequalities III, (O. Shisha, ed.), Academic Press, New York, 1972, 115–119.
[6] K. GUSTAFSON, *Operator trigonometry*, Linear and Multilin. Alg., to appear.
[7] K. GUSTAFSON, *Antieigenvalues*, Linear Alg. Appl., to appear.

Typeset by $\mathcal{AMS}$-TEX

Convergence of Arnoldi method

Olavi Nevanlinna
Insitute of Mathematics
Helsinki University of Technology
02150 Espoo, Finland
e-mail: Olavi.Nevanlinna@hut.fi

This note summarizes some results on (a monitored version of) the Arnoldi method in Hilbert spaces. The interest in working in infinite dimensional spaces comes partly from the fact that only then we can have meaningful asymptotical statements (which hopefully give some light to the convergence of Arnoldi in large dimensional problems with iteration indices far less than the dimension).

1.The Set-up.

Let H be a separable Hilbert space and $A \in L(H)$ a bounded linear operator in H. The computational task is to find the spectrum of A, $\sigma(A)$. By "Arnoldi method" we mean the following: Select a nonvanishing initial vector $b \in H$ and orthogonalize the Krylov subspaces $K_n(A,b)$:=span$\{b, Ab, \dots, A^{n-1}b\}$. Let $\{p_j\}$ denote the *monic* polynomials of degree j such that the vectors $v_{j+1} := p_j(A)b$ with $j = 1, \dots, n$ form a basis for $K_n(A,b)$. Let P_n be the orthogonal projection in H onto $K_n(A,b)$ and put A_n for P_nA when restricted into $K_n(A,b)$.

Proposition 1. *The spectrum of A_n and the set of zeros of p_n are equal.*

In general, the limit set of the zeros of the polynomials and the spectrum of A need not have much in common. Our remarks will however concern a *monitored version* of Arnoldi where we only look at those indeces j for which the ratio $||p_j(A)b||/||p_{j-1}(A)b||$ tends to 0. To that end put

$$\beta_j := ||p_j(A)b|| \tag{1.1}$$

and consider any sequence $J := \{j_m\}$ such that

$$\frac{\beta_{j_m}}{\beta_{j_m-1}} \to 0. \tag{1.2}$$

In other words, if you write down the Hessenberg form with vectors $v_{j+1}/||v_{j+1}||$ as the basis, then these ratios show up on the $j+1, j-$diagonal. Restricting to the index set J thus means that the operator A has an "almost invariant" subspace of dimension j_m and, consequently, the spectrum of A and that of A_{j_m} are related. The existence of J so that (1.2) holds, is guaranteed for all *quasialgebraic* operators, see [N1], [N2], and thus in particular whenever the spectrum is at most countable.

Definition 1. Given a sequence $\{E_j\}$ of compact sets we denote by $\omega\{E_j\}$ the set of z such that there exists an increasing sequence n_k and $\lambda_k \in E_{n_k}$ such that $\lambda_k \to z$.

In particular $\omega\{\sigma(A_{j_m})\}$ is the set of all accumulation points of the zeros of $\{p_{j_m}\}$.

Typeset by $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX

Proposition 2. *If (1.2) holds, then* $\omega\{\sigma(A_{j_m})\} \subset \sigma(A)$.

Proposition 1 is essentially in Householder's book (or in this form in [N2]). Proposition 2 is also simple, see Proposition 5.4 in [N2].

2. Example.

Let S be the "bilateral" shift in $l_2(\mathbb{Z})$

$$Se_k = e_{k+1}.$$

Starting from $b := e_0$ gives $p_j(\lambda) = \lambda^j$ and in particular $\beta_j \equiv 1$. Notice that along any subsequence the zeros of p_j always stay (and accumulate) at the origin which lies with distance 1 from the spectrum of S (which is the unit circle).

Let $K(A,b)$ be the closure of span$\{b, Ab, A^2b, \dots\}$ and denote by $A_{[b]}$ the restriction of A to the invariant subspace $K(A,b)$. Then in this example $\sigma(S_{[e_0]})$ is the closed unit disc while $\sigma(S)$ only contains the unit circle.

3. Effect of initial vectors.

The example above shows that $\sigma(A_{[b]})$ can be properly larger than $\sigma(A)$.

Almost a tautology. *From the Krylov information* $\{b, Ab, A^2b, \dots\}$ *we can only try to compute* $\sigma(A_{[b]})$.

Notice, however Proposition 2 which is about $\sigma(A)$ and not about $\sigma(A_{[b]})$.

Proposition 3. *The set* H_0 *of initial vectors* b *such that*

$$\partial\sigma(A) \subset \sigma(A_{[b]}) \tag{3.1}$$

does not hold, is of first (Baire) category.

Combining these facts, we can only try to compute $\sigma(A_{[b]})$, but on the other hand, apart from an exceptional set of initial vectors H_0 the true spectrum will be included.

Proposition 4. *If* $\sigma(A)$ *is totally disconnected, then for all* b

$$\sigma(A_{[b]}) \subset \sigma(A).$$

These facts are discussed in [N2], in particular Proposition 3 follows from a result of Vrbova.

4. Approximation of isolated components.

Let E be an isolated component of $\sigma(A_{[b]})$. Then the following holds.

Proposition 5. *Let* U *be an open set such that* $E \subset U$ *and* $U \cap \sigma(A_{[b]}) = E$. *If (1.2) holds, then there exists* k *such that for* $j_m > k$.

$$\sigma(A_{j_m}) \cap U \neq \emptyset.$$

This is a special case of Theorem 4.1 (ii) in [N and V].

5. Main theorem.

Let us put $\|(\lambda - T)^{-1}\| = \infty$ for $\lambda \in \sigma(T)$.

Definition 2. Given a sequence $\{T_n\} \subset L(H)$ we set (in [N2])

$$\Lambda_i\{T_n\} := \{\lambda |\ \liminf \|(\lambda - T_n)^{-1}\| = \infty\}$$

and

$$\Lambda_s\{T_n\} := \{\lambda |\ \limsup \|(\lambda - T_n)^{-1}\| = \infty\}.$$

If these sets agree, then we say that $\Lambda\{T_n\}$ exists (and it is defined by $\Lambda\{T_n\} := \Lambda_i\{T_n\}$.

The main result is the following

Theorem. *If (1.2) holds, then $\Lambda\{A_{j_m}\}$ exists and*

$$\Lambda\{A_{j_m}\} = \sigma(A_{[b]}).$$

This follows from results in [N and V]. The computation of $\Lambda\{A_{j_m}\}$ is shortly discussed in [N2].

References.

[N1] Olavi Nevanlinna: Convergence of Iterations for Linear Equations, Birkhäuser, 1993

[N2] Olavi Nevanlinna: Hessenberg Matrices in Krylov Subspaces and the Computation of the Spectrum, to appear in J.Numer.Funct.Anal.Optim.; a former version available as technical report June 1993

[N and V] Olavi Nevanlinna and Gennadi Vainikko: Manuscript

Wednesday, April 6

Parallel Computation II
Chair: Howard Elman
Room B

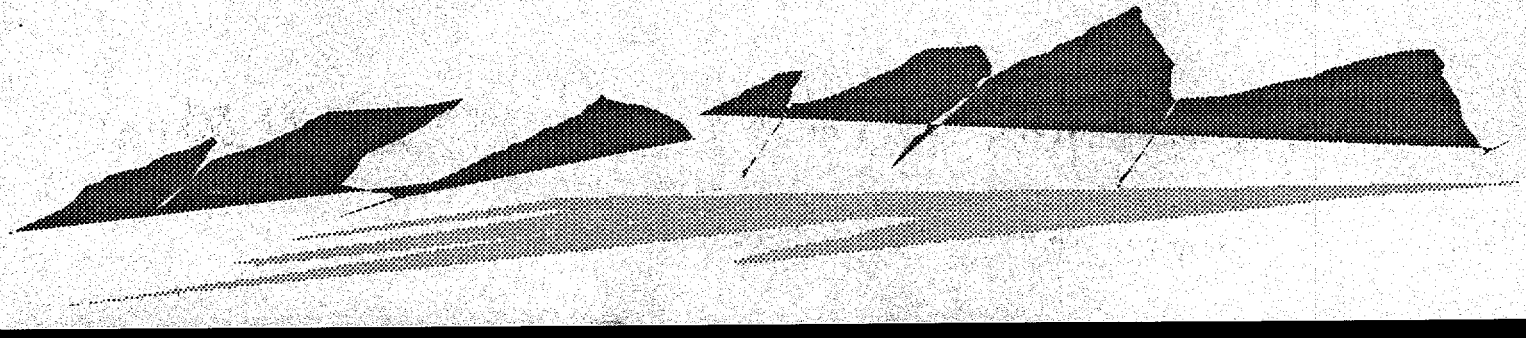
10:15 - 10:40 Anne E. Trefethen
The Conjugate Gradient NAS Parallel Benchmark on the IBM SP1

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



The Conjugate Gradient NAS Parallel Benchmark on the IBM SP1

Anne E. Trefethen* Tong Zhang†

The NAS Parallel Benchmarks are a suite of eight benchmark problems developed at the NASA Ames Research Center. They are specified in such a way that the benchmarkers are free to choose the language and method of implementation to suit the system in which they are interested. In this presentation we will discuss the Conjugate Gradient benchmark and its implementation on the IBM SP1. The SP1 is a parallel system which is comprised of RS/6000 nodes connected by a high performance switch. We will compare the results of the SP1 implementation with those reported for other machines. At this time, such a comparison shows the SP1 to be very competitive.

*Speaker. Cornell National Supercomputing Facility, Cornell University, Ithaca, NY 14853 (aet@tc.cornell.edu)

†Department of Mathematics, Cornell University, Ithaca, NY 14853

Advancements and Performance of Iterative Methods in Industrial Applications Codes on CRAY Parallel/Vector Supercomputers

Gene Poole & Mike Heroux
Engineering Applications Group
Cray Research Inc.
655e Lone Oak Drive
Eagan, MN 55121

gene.poole@cray.com
mike.heroux@cray.com

This paper will focus on recent work in two widely used industrial applications codes with iterative methods. The ANSYS program, a general purpose finite element code widely used in structural analysis applications, has now added an iterative solver option. Some results are given from real applications comparing performance with the tradition parallel/vector frontal solver used in ANSYS. Discussion of the applicability of iterative solvers as a general purpose solver will include the topics of robustness, as well as memory requirements and CPU performance. The FIDAP program is a widely used CFD code which uses iterative solvers routinely. A brief description of preconditioners used and some performance enhancements for CRAY parallel/vector systems is given. The solution of large-scale applications in structures and CFD includes examples from industry problems solved on CRAY systems.

A Component Analysis Based on Serial Results for Analyzing Performance of Parallel Iterative Programs

Shu-Mei C. Richman
Dalhousie University, Canada; (902)494 – 2572; srichman@cs.dal.ca

Abstract. This research is concerned with the parallel performance of iterative methods for solving large, sparse, nonsymmetric linear systems. Most of the iterative methods are first presented with their time costs and convergence rates examined intensively on sequential machines, and then adapted to parallel machines. The analysis of the parallel iterative performance is more complicated than that of serial performance, since the former can be affected by many new factors, such as data communication schemes, number of processors used, and ordering and mapping techniques. Although we are able to summarize results from data obtained after examining certain cases by experiments, there are two questions remain: (1.) How to explain the results obtained? (2.) How to extend the results from the certain cases to general cases ?

To answer these two questions quantitatively, we introduce a tool called *component analysis based on serial results*. This component analysis is introduced because the iterative methods consist mainly of several basic functions such as linked triads, inner products, matrix-vector multiplications, and triangular solves, which have different intrinsic parallelisms and are suitable for different parallel techniques. We first express the parallel performance of each iterative method as a weighted sum of the parallel performance of the basic functions that are the components of the method. Then, we separately examine the performance of basic functions and the weighting distributions of iterative methods, from which we obtain two independent sets of information when solving a given problem. In this component approach, all the weightings require only serial costs not parallel costs, and each iterative method for solving a given problem is represented by its unique weighting distribution. The information given by the basic functions is independent of iterative method, while that given by weightings is independent of parallel technique, parallel machine and number of processors. So, these two independent sets of information allow us to give answers quantitatively to the above two questions.

Performance analysis of high quality parallel preconditioners applied to 3D finite element structural analysis.

Lilia Kolotilina, Andy Nikishin, Alex Yeremin
Russian Academey of Sciences and Elegant Mathematics, Inc.
Michael A. Heroux*, Qasim Sheikh
Cray Research, Inc.

1 Abstract

The solution of large systems of linear equations is a crucial bottleneck when performing 3D finite element analysis of structures. Also, in many cases the reliability and robustness of iterative solution strategies, and their efficiency when exploiting hardware resources, fully determine the scope of industrial applications which can be solved on a particular computer platform. This is especially true for modern vector/parallel supercomputers with large vector length and for modern massively parallel supercomputers.

Preconditioned iterative methods have been successfully applied to industrial class finite element analysis of structures. The construction and application of high quality preconditioners constitutes a high percentage of the total solution time. Parallel implementation of high quality preconditioners on such architectures is a formidable challenge. Two common types of existing preconditioners are the implicit preconditioners and the explicit

*Corresponding Author: 655 Lone Oak Drive, Eagan, MN 55121 USA, email: mike.heroux@cray.com, Phone: (612) 683-5628, Fax: (612) 683-3099

preconditioners. The implicit preconditioners (e.g. incomplete factorizations of several types) are generally high quality but require solution of lower and upper triangular systems of equations per iteration which are difficult to parallelize without deteriorating the convergence rate. The explicit type of preconditionings (e.g. polynomial preconditioners or Jacobi-like preconditioners) require sparse matrix-vector multiplications and can be parallelized but their preconditioning qualities less than desirable.

We present results of numerical experiments with Factorized Sparse Approximate Inverses (FSAI) for symmetric positive definite linear systems. These are high quality preconditioners that possess a large resource of parallelism by construction without increasing the serial complexity.

Let A be an $n \times n$ symmetric positive definite matrix and let $A = LL^T$ be its Cholesky decomposition. We compute a matrix G_L with a given fixed sparsity pattern of off-diagonal entries such that G_L approximates L^{-1} and minimizes the Frobenius matrix norm

$$\|I - XL_A\|_F = \sqrt{tr[(I - XL_A)(I - XL_A)]},$$

over all matrices X with the given sparsity pattern. The construction of G requires solution of n linearly independent systems each of size equal to the number of nonzeros in a given row of G. Thus the construction of G is highly parallel. Also, no knowledge of elements of L is required during the construction phase. The application of the preconditioner requires multiplication by lower and upper triangular sparse matrices.

We solve 3D equilibrium equations for linear elastic orthotropic materials approximated by the p-version of the finite element methods using the FSAI preconditioners and the well known block SSOR preconditioner to compare the convergence properties and serial cost of these two methods.

The FSAI preconditioner with a stable version of block CG algorithm is used to solve industrial class problems extracted from NASTRAN and a proprietary finite element package for tire design. We show that these new parallel preconditioners can solve rather difficult industrial class problems without increasing serial complexity. The performance of these new preconditioners is compared with the performance of well known highly optimized direct solvers as well. The numerical experiments are performed on Cray Y-MP and Cray C-90.

Wednesday, April 6

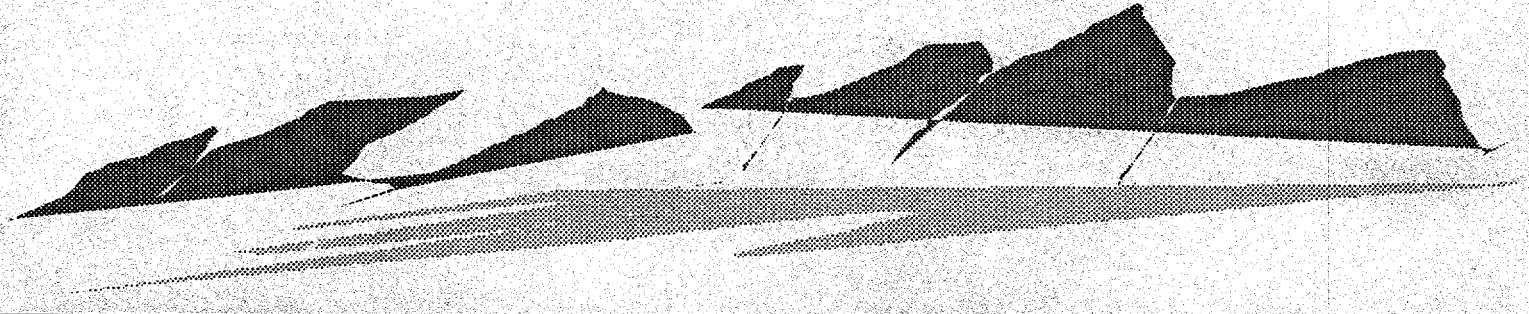
Iterative Methods: Theory
Chair: Roland Freund
Room A

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



ABSTRACT FOR COPPER MOUNTAIN CONFERENCE

TITLE: RECENT ADI ITERATION ANALYSIS AND RESULTS

AUTHOR: Eugene L. Wachspress

DATE: November 9, 1993

Some recent ADI iteration analysis and results are discussed. Discovery that the Lyapunov and Sylvester matrix equations are model ADI problems [1] stimulated much research on ADI iteration with complex spectra. The ADI rational Chebyshev analysis parallels the classical linear Chebyshev theory. Two distinct approaches have been applied to these problems. First, parameters which were optimal for real spectra were shown to be nearly optimal for certain families of complex spectra. In the linear case these were spectra bounded by ellipses in the complex plane [2]. In the ADI rational case these were spectra bounded by "elliptic-function regions" [3]. The logarithms of the latter appear like ellipses, and the logarithms of the optimal ADI parameters for these regions are similar to the optimal parameters for linear Chebyshev approximation over superimposed ellipses. W.B. Jordan's bilinear transformation of real variables to reduce the two-variable problem to one variable was generalized [4] into the complex plane. This was needed for ADI iterative solution of the Sylvester equation.

For more general spectral regions, there are classical algorithms for finding asymptotically optimal parameters for the linear Chebyshev problem [5]. This theory was generalized by Starke [6] to the rational ADI problem. This work stimulated further research by Istace and Thiran [7], who applied nonlinear optimization techniques to computation of optimal parameters for any specified number of iterations.

Another development which is less well analyzed deals with three-variable problems with real spectra. Several approaches have been used here, one of which was discussed in a seminal ADI iteration paper [8]. A recent attempt has been made to treat two variables jointly with an inner ADI iteration. The effect of this inner iteration on the composite scheme has been analyzed, and numerical studies have been performed to support the theory [9]. This approach appears to be well suited for model ADI preconditioning where relatively modest error reduction is required of the ADI iteration for each ADI-CG cycle.

REFERENCES

1. N.S. Ellner and E.L. Wachspress, "New ADI model problem applications," Proc. FJCC, Dallas, Texas 1986, IEEE CS Press, pp. 528-534.

2. E.E. Wrigley, "Accelerating the Jacobi method for solving simultaneous equations by Chebyshev extrapolation when the eigenvalues of the iteration matrix are complex," Comp. J. vol 6, pp. 169-176, 1963.

3. E.L. Wachspress, "The ADI minimax problem for complex spectra," in Iterative Methods for Large Linear Systems, eds. D. Kincaid and L. Hayes, Academic Press, 1990, pp. 251-271.

4. E.L. Wachspress, "Optimum parameters for two-variable ADI iteration," Ann. Nucl. Energy, vol 19, no. 10-12, pp. 765-778, 1992.

5. T. Manteuffel, "The Tchebychev iteration for nonsymmetric linear systems," Numerische Math. vol 28, pp 307-327, 1979

6. G. Starke, "Rational minimization problems in connection with

optimal ADI-parameters for complex domains," PhD thesis, University of Karlsruhe (1989).

7.M.P. Istace and P.M. Thiran, "On the third and fourth Zolotarev problems in the complex plane," Mathematics of Computation, 1993

8. J. Douglas, Jr., "Alternating direction methods for three space variables," Numerische Math. 4 (1962) pp. 41-63

9. E.L. Wachspress, "Three-variable ADI iteration," CAMWA: In process.

A Sparse Matrix Iterative Method for Efficiently Computing Multiple Simultaneous Solutions

W. E. Boyse and A. A. Seidl
Lockheed Palo Alto Research Laboratories
3241 Hanover Street
Palo Alto, CA 94304-1191

November 23, 1993

1 **Abstract** We consider the solution of large sparse complex symmetric indefinite systems of equations where multiple solutions are required. This type of problem occurs in calculating monostatic radar cross sections in electromagnetic scattering using the finite element method.

The Quasi Minimum Residual (QMR) [3] method, ideally suited for these matrices, is generalized using the block Lanczos algorithm to solve blocks of solutions simultaneously. The algorithm is presented and a natural convergence criterion is proposed which is shown to be as effective as the usual equation residual in monitoring convergence.

A preconditioner is used in conjunction with this iterative method to accelerate the convergence. This preconditioner, denoted IC(T) [5, 6, 1] is computed by retaining the largest elements in each row of the factor as it is computed row by row. Thus, the nonzero structure of the imcomplete factor is determined by the numerics and is not restricted to the structure of the original matrix as are "classical" IC preconditioners. The definition of "largest" may also be adjusted to alter the degree of "completeness" of the incomplete factor.

The performance of block QMR iterative method and preconditioning strategy is evaluated on a large finite element problem. The matrix used is real symmetric indefinite and of rank 270,000. This example shows that the IC(T) preconditioner is superior to none or the IC preconditioner in terms of wall clock time to solution.

The block solution capability of the iterative method is evaluated on 1, 2, 4, and 8 simultaneous solution. There is significant convergence acceleration [2] accompanying multiple simultaneous solutions, as predicted and observed by [4] in the context of eigenvalue problems. This block solution method also provides another degree of parallelism which can be exploited in the numerical implementation.

References

[1] W. E. Boyse and A. A. Seidl, *Convergence acceleration by preconditioning a block quasi minimum residual method for a finite element formulation of electromagnetic scattering*, in IEEE AP-S / URSI Symposium Digest, 1992.

[2] ——, *An iterative solver for multiple right hand sides based on the block lanczos algorithm*, in IEEE AP-S / URSI Symposium Digest, 1992.

[3] R. W. Freund, *Conjugate gradient type methods for linear systems with complex symmetric coefficient matrices*, SIAM J. Sci. Stat. Comput., 13 (1992).

[4] D. P. O'Leary, *The block conjugate gradient algorithm and related methods*, Linear Algegra Appl., (1980).

[5] Y. Saad, *ILUT: a dual threshold incomplete LU factorization*, Research Report UMSI 92/38, University of Minnesota Supercomputer Institute, Minneapolis, MN, March 1992.

[6] D. P. Young, R. G. Melvin, F. T. Johnson, J. E. Bussoletti, L. B. Wigton, and S. S. Samant, *Application of sparse matrix solvers as effective preconditioners*, SIAM J. Sci. Stat. Comput., 10 (1989), pp. 1186–1199.

On the Effects of Using the GTH Method in the Iterative-Aggregation Disaggregation Technique

Tuğrul Dayar and William J. Stewart

Computer Science Department
North Carolina State University
Raleigh, NC 27695

Abstract. The iterative aggregation-disaggregation (IAD) technique is an effective method for solving finite nearly completely decomposable (NCD) Markov chains. Small perturbations in the transition probabilities of these chains lead to a considerable change in the stationary vector. Therefore, NCD Markov chains are referred to as being ill-conditioned. During an IAD step, this undesirable condition is inherited by the coupling matrix and one confronts the problem of finding the stationary vector of a stochastic matrix which has weighty diagonal elements close to one. In this paper, we investigate the effects of using the Grassmann-Taksar-Heyman (GTH) method to solve the coupling matrix formed in the aggregation step. Then, we extend the idea in such a way that this direct method can be incorporated into the disaggregation step. Finally, we discuss various implementation issues, demonstrate the effect of using the GTH method in the IAD algorithm on various examples, and elaborate on the conditions under which it should be applied.

1 Introduction

Nearly completely decomposable (NCD) Markov chains are irreducible stochastic matrices that can be ordered so that the transition probabilities have a block structure in which the non-zero elements of the off-diagonal blocks are small compared to those of the diagonal blocks. Such matrices often arise in queueing network analysis, large scale economic modeling, and computer systems performance evaluation. The problem is to find a nontrivial solution to the system

$$\pi P = \pi, \qquad \|\pi\|_1 = 1 \tag{1.1}$$

where P is a $(n \times n)$ irreducible stochastic matrix and π is the $(1 \times n)$ stationary vector. For a formal definition and notation of NCD Markov chains please refer to [3].

NCD chains that appear in applications are quite large and sparse, possibly having more than thousands of states. For such large chains, direct methods cause an immense fill-in during the triangularization phase of the solution process, and due to storage requirements they are not recommended. Additionally, the ill-conditioned nature of these chains generally invalidate the feasibility of other iterative methods. Aggregation is often necessary to make computation tractable with available resources, and in order to circumvent the described problems, IAD algorithms have been developed (see [2] for example).

The idea in IAD methods is to observe the system in isolation in each one of the diagonal blocks as if the system is completely decomposable, and to compute the stationary vector of each diagonal block. However, there are two problems with this approach. First of all, since the diagonal blocks are substochastic, the off-diagonal weights must somehow be incorporated into the diagonal blocks. Secondly, the probabilities obtained through the above approach are conditional, and this condition has to be removed by weighing each probability subvector by

the probability of being in that group of states. Only if these two problems are overcome, can one form the stationary vector of the Markov chain by weighing the subvectors and pasting them together.

A stochastic complement is the transition probability matrix of a smaller irreducible Markov chain obtained by observing the original process in the corresponding block of states. Hence, by finding an approximation to each stochastic complement, we can get around the first problem (see [3] for details).

To determine the probability of being in a certain block of states, one needs to form the irreducible coupling matrix which shrinks each block down to a single element. The stationary vector of the coupling matrix gives the stationary probability of being in each one of the block of states. So, we essentially compute the weighing factors mentioned before.

In the next section, we discuss how a modified version of Gaussian elimination (GE) may be used to enforce stability in the solution of the coupling matrix. In §3, we extend the idea so that it can be used in a non-singular system of equations with a substochastic coefficient matrix. §4 discusses certain implementation issues, and §5 is about numerical experiments with the IAD algorithm on NCD chains.

2 Solving the Coupling Matrix

The coupling matrix is an irreducible stochastic matrix of order N (= number of NCD blocks) Our goal is to solve the singular system $\xi C = \xi$ subject to $\|\xi\|_1 = 1$. Being an irreducible stochastic matrix, the coupling matrix has a unique eigenvalue that is equal to 1. All other $(N-1)$ eigenvalues are close to 1. The distance of these other eigenvalues to 1, obviously depends on the off-diagonal weights in C.

A careful inspection reveals that one can solve the equivalent system

$$(I - C^T)\xi^T = 0, \qquad \|\xi\|_1 = 1 \tag{2.1}$$

more effectively. $(I - C^T)$ is a singular M-matrix and we are seeking the unique null vector that has a unit-norm of 1. For such a matrix, the pivot element at each step of the direct solution method is bounded by 1 and consequently, there is no need for pivoting if column diagonal dominance is preserved throughout the computation.

Since C has a subdominant eigenvalue close to 1, iterative methods tend to have a slow convergence rate when applied to the above system. On the other hand, certain stability issues need to be addressed if direct methods are used. As shown in [4], ordinary GE is not stable in the presence of rounding errors on a coupling matrix having very weighty diagonal elements. Hence, it is necessary to apply some sort of pivoting strategy in order to proceed with GE. Otherwise, the algorithm breaks down to the contrary of general belief.

Our motivation in trying to come up with a remedy for the above situation is the GTH algorithm in [1] and the direct method in [4]. The original GTH algorithm emerges from probabilistic arguments, and the idea is to calculate the stationary vector of a Markov chain using only nonnegative numbers and avoiding subtraction operations. It has been shown that this algorithm achieves significantly greater accuracy than other algorithms described in the literature. Interestingly, the inspiration for the algorithm presented in [4], which is specifically for the solution of NCD chains, has been the GTH algorithm.

Let

$$A = [a_{ij}] = I - C^T.$$

An observation made in the above papers is that, if A is partitioned in the form

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \Longrightarrow \begin{pmatrix} U_{11} & \tilde{A}_{12} \\ 0 & \tilde{A}_{22} \end{pmatrix},$$

provided A_{11} is a nonsingular M-matrix, then the Schur complement of A_{11}, (i.e., the result of performing Gaussian elimination through the block A_{11}, where U_{11} is upper-triangular), $\tilde{A}_{22}$, is a singular M-matrix having 0 column sums just as the initial matrix A. The properties of a singular M-matrix coupled with the GTH idea of avoiding subtractions and negative numbers, suggests the following modification to GE. At each step of GE, rather than calculating the pivot element in the usual way, one can correct the pivot by replacing it with the negated sum of the off-diagonal elements in the unreduced part of the same column as the pivot in the matrix under consideration. When one mentions the GTH method, it is this approach used in calculating the pivot elements that is implied.

3 Using the GTH Method in the Disaggregation Step

In a given iteration, the disaggregation step of the IAD method uncouples the NCD chain to obtain a new estimate for the stationary vector. To achieve a yet better approximation for $\pi = (\pi_1, \pi_2, \cdots, \pi_N)$, at the $(k+1)^{st}$ iteration, one solves for $(\pi_i^{(k+1)})^T$, in

$$(I - P_{ii}^T)(\pi_i^{(k+1)})^T = {b_i}^T, \tag{3.1}$$

where $\pi_i^{(k+1)}$ is the $(k+1)^{st}$ approximation to π_i and $b_i = \sum_{j<i} \pi_j^{(k+1)} P_{ji} + \sum_{j>i} z_j^{(k+1)} P_{ji}$ for $i = 1, 2, \cdots, N$.

Here, P_{ij} is the i^{th} row, j^{th} column block of dimension $(n_i \times n_j)$. In what follows, P_{i*} (P_{*i}) is the i^{th} row (column) of blocks with P_{ii} removed; e is a column vector of 1's and its length is determined in the context it is used. Clearly, P_{ii} is a strictly substochastic matrix of order n_i, and $b_i \neq 0$, implying a nonsingular system of equations. If the matrix in the above linear system had been stochastic, then we could clearly employ the same technique utilized in solving the coupling matrix. However, by adding one more equation and augmenting the matrix with b^T, we can put the system into the form

$$W_i(\pi_i^{(k+1)}, \hat{\pi}_i^{(k+1)})^T = 0, \tag{3.2}$$

where

$$W_i = \begin{array}{c} \begin{matrix} n_i & 1 \end{matrix} \\ \begin{pmatrix} I - P_{ii}^T & b^T \\ w & \hat{w} \end{pmatrix} \begin{matrix} n_i \\ 1 \end{matrix} \end{array},$$

$w = (P_{i*}e)^T$, $\hat{w} = -be$, and $\hat{\pi}_i^{(k+1)}$ is introduced so that the solution vector has as many columns as the coefficient matrix. In other words, $(w, \hat{w})$ sums up the columns of W_i to 0. The values of $\hat{w}$ and $\hat{\pi}_i^{(k+1)}$ are irrelevant, because just as in (2.1), we have a singular system with $(n_i + 1)$ equations, and after W_i is reduced to upper-triangular form, the last row will be all 0's. Hence, the reduction needs to be carried out for only n_i steps. What needs to be done for the computation of $(\pi_i^{(k+1)})^T$ is, to use the first n_i elements of column $(n_i + 1)$ in the upper-triangular matrix as the right-hand side in the back substitution phase.

It is not possible to put GE to use in the disaggregation step in problems, where $\|P_{i*}\|_\infty$ (degree of coupling) is less than ϵ (machine epsilon). On the other hand, GTH computes row $(n_i + 1)$ in W_i by using the non-zero elements in P_{i*}. Hence, GTH may be applied to solve such blocks.

4 Implementation Considerations

As mentioned before, NCD chains confronted in real-life applications are generally large and sparse. This property necessitates the design and employment of sparse storage schemes, which essentially store only the non-zero elements in the matrix that is operated on.

So far, we considered the application of GTH (and GE, for that matter) to the systems (2.1) and (3.1). We wrote (3.1) in the form (3.2) so that its coefficient matrix is a singular M-matrix with 0 row sums just as in (2.1). Now, let us write the alternative non-transposed system of equations that may be solved. The system that corresponds to (2.1) is

$$\xi A^T = 0, \qquad \|\xi\|_1 = 1. \tag{4.1}$$

Similarly, the equivalent of (3.2) is

$$(\pi_i^{(k+1)}, \hat{\pi}_i^{(k+1)}) W_i^T = 0. \tag{4.2}$$

Define

Scheme 1: The transposed system of equations (2.1) and (3.2).
Scheme 2: The non-transposed system of equations (4.1) and (4.2).

Unless otherwise specified, by reductions we mean row-reductions. Our last assumption is that the coefficient matrices are supplied to both schemes in the non-transposed version. This is fairly reasonable, since the matrices are generated by following the possible transitions from a given state implying a row-wise generation.

The advantage of reducing the coefficient matrices in scheme 1 rather than the ones in scheme 2 to upper-triangular form is twofold:

- The pivot element at each step of the reduction is bounded by 1.
- Only the upper-triangular matrix needs to be stored during the reduction process.

Although, the pivots in scheme 2 are not necessarilly bounded by 1, the growth factor still cannot be greater than 1. However, both the upper-triangular matrix and the lower-triangular matrix, which encompasses the multipliers, have to be stored during the triangularization process. On the other hand, scheme 1 calls for the transposition of the coefficient matrix before executing GE or GTH. In the following section, we talk about issues related to numerical experiments.

5 Numerical Results

We have experimented with the IAD algorithm in sparse storage. All routines used are part of the software package MARCA (Markov Chain Analyzer) (see [5]). The routines are written in FORTRAN and compiled in both double and quadruple precision floating-point arithmetic. We have experimented with the software package on a SUN SPARC station 2. For each problem solved, the residual and the relative error in the solution are computed. We would like to see how GE and GTH behave comparatively for problems both GE and GTH may be employed in the aggregation and the disaggregation steps of the iterative algorithm. Therefore, we have experimented with both scheme 1 and scheme 2 type implementations in MARCA.

We have worked on three problems. Although the order of the matrices considered in the first two problems are quite small, we think it is instructive to examine the effects of using the IAD algorithm on such problems. The third problem investigated is a real-life example and

had been studied with different parameters in the past. The transition probility matrix for this problem is large, and close to the identity matrix. All three problems will be discussed at the conference.

In all three problems, both IAD with GE and IAD with GTH have converged to the stopping criterion with a residual in the order of machine epsilon. For each problem, the number of iterations taken by both methods are the same. However, the relative error in IAD with GE is much larger than that of IAD with GTH in the first two small problems where GTH has a relative error almost in the order of machine epsilon. Furthermore, it is not very clear how good a measure relative error is for matrices having stationary probabilities in the order of machine epsilon, as in the third problem. One other observation regarding the results of problem 3 is the difference between execution times for scheme 1 and scheme 2 type implementations.

6 Conclusion

Our interest lies in the computation of the stationary vector of ill-natured NCD Markov chains. The GTH method, which avoids subtractions, is a much more stable version of GE. For that reason, it is a good candidate to be used in the two-level IAD algorithm. We have experimented on several problems, applying this idea versus GE in the IAD technique. The GTH way of calculating the pivot element by taking the negated sum of the off-diagonal elements in the unreduced part of the pivot column proves to be valuable for singular M-matrices, and it is shown to be quite effective on the problems of interest.

In conclusion, ordinary GE should definitely be avoided in both steps of the iterative IAD algorithm when solving NCD chains with a degree of coupling less than machine epsilon. However, if just a rough approximation to the stationary vector of a large NCD Markov chain is sought in a short time, IAD with GE may be used. On the contrary, if relative error in the stationary vector of the NCD chain is deemed as of utmost importance, then IAD with GTH has to be recommended. Examination of several sparse storage formats for both GTH and GE has indicated one disadvantage of the GTH method. The time to execute the IAD algorithm on large irreducible NCD Markov chains tends to be longer when the GTH method is used in the aggregation and disaggregation steps of the iterative solver. Memory requirement of the GTH algorithm is generally slightly higher than that of GE; nevertheless this mostly depends on the sparse storage format chosen.

References

[1] W. K. Grassmann, M. I. Taksar and D. P. Heyman, Regenerative Analysis and Steady State Distributions for Markov Chains, *Oper. Res.* **33**, (1985), 1107–1116.

[2] J. R. Koury, D. F. McAllister and W. J. Stewart, Iterative Methods for Computing Stationary Distributions of Nearly Completely Decomposable Markov Chains, *SIAM J. Alg. Disc. Meth.* **5**, (1984), 164–186.

[3] C. D. Meyer, Stochastic Complementation, Uncoupling Markov Chains, and the Theory of Nearly Reducible Systems, *SIAM Review* **31**, (1989), 240–272.

[4] G.W. Stewart and G.Zhang, On a Direct Method for the Solution of Nearly Uncoupled Markov Chains, *Numer. Math.* **59**, (1991), 1–11.

[5] W. J. Stewart, MARCA: Markov Chain Analyzer, A Software Package for Markov Modeling in *Numerical Solution of Markov Chains*, Marcel Dekker, Inc., New York, (1991), 37–61.

On the interplay between inner and outer iterations for a class of iterative methods.

Eldar Giladi
Scientific Computing and Computational Mathematics Program
Stanford University, Stanford CA 94305.

1 Introduction

Iterative algorithms for solving linear systems of equations often involve the solution of a subproblem at each step. This subproblem is usually another linear system of equations. For example, a preconditioned iteration involves the solution of a preconditioner at each step. In this paper, we consider algorithms for which the subproblem is also solved iteratively. The subproblem is then said to be solved by "inner iterations", while the term "outer iteration" refers to a step of the basic algorithm. The cost of performing an outer iteration is dominated by the solution of the subproblem, and can be measured by the number of inner iterations. A good measure of the total amount of work needed to solve the original problem to some accuracy ϵ is then, the total number of inner iterations. To lower the amount of work, one can consider solving the subproblems "inexactly" i.e. not to full accuracy. Although this diminishes the cost of solving each subproblem, it usually slows down the convergence of the outer iteration.

It is therefore interesting to study the effect of solving each subproblem inexactly on the total amount of work. Specifically, we consider strategies in which the accuracy to which the inner problem is solved, changes from one outer iteration to the other. We seek the "optimal strategy", that is, the one that yields the lowest possible cost.

In this paper, we develop a methodology to find the optimal strategy, from the set of slowly varying strategies, for some iterative algorithms. We apply this methodology to the Chebychev iteration (CI) and show that for CI, a strategy in which the inner-tolerance remains constant is optimal. We also estimate this optimal constant. Then, we discuss generalizations to other iterative procedures.

The convergence rate of the constant strategy CI has been previously studied by Golub and Overton [1]. We shall use some of their results in this work. Other papers which study the use of inner and outer iterations include Nicolaides [2], Pereyra [3], Nichols [4], Dembo et al [5], and a recent paper by Elman and Golub [6].

1.1 Problem Statement

To formalize this problem, we let $\boldsymbol{\delta} = \{\delta_j\}_{j=0}^{\infty}$, be a sequence of tolerance values. The j'th component of $\boldsymbol{\delta}$, δ_j, is the relative error in the residual, required in the solution of the subproblem at outer iteration j. Therefore $\delta_j \in (0,1)$ and the number of inner iterations at step j is approximately $\lceil \frac{-\log \delta_j}{-\log \rho} \rceil$. In this estimate, ρ is the convergence factor of the method which is used in the solution of the subproblem. We then define $N(\epsilon,\boldsymbol{\delta})$ to be the number of outer iterations needed to reduce the initial error by a factor of ϵ for strategy $\boldsymbol{\delta}$. It follows that the total number of inner iterations required to achieve this accuracy ϵ is proportional to

$$\mathbf{C}(\epsilon, \boldsymbol{\delta}\ ) = \sum_{j=0}^{N(\epsilon,\delta)-1} -\log \delta_j. \tag{1}$$

Our objective is to minimize $\mathbf{C}(\epsilon, \boldsymbol{\delta}\ )$ with respect to $\boldsymbol{\delta}$.

In this minimization, it seems reasonable to consider slowly varying monotonically decreasing strategies. We examine such strategies due to the following heuristic argument. When $\delta_j = \hat{\delta}$ for all j and $\hat{\delta} << 1$, we are solving the inner problem to a very high accuracy. Then, the amount of work required at each step, is very large. Although the number of outer iterations in this case is small, we still do a lot of work and hence $\mathbf{C}(\epsilon, \delta)$ is large. On the other hand if $\hat{\delta} \sim 1$, each subproblem requires only a few inner iterations. However, a severe slow down or even divergence of the algorithm can occur. Thus, the number of outer iterations $N(\epsilon, \boldsymbol{\delta})$ in this case is very large, and $\mathbf{C}(\epsilon, \boldsymbol{\delta}\ )$ is again high. Between these two extremes, one can consider a variable strategy $\boldsymbol{\delta}$, for which δ_0 is not too small, but δ_j decreases slowly to zero as $j \to \infty$. For such a strategy, $-\log \delta_j$ in (1), is relatively small for many values of j. Furthermore, since δ_j is decreasing, we expect the number of outer iterations to be smaller than it is if the tolerance is fixed at δ_0. Hence, we hope that the use of a low accuracy initially and a higher accuracy as the algorithm proceeds, is more cost effective.

In view of the heuristic arguments above, we consider the set $\mathcal{S}$ of slowly varying strategies

$$\mathcal{S} = \left\{ \boldsymbol{\delta} \ \mid \delta_k = \delta(\beta k),\ \ 0 < \beta << 1,\ \ \frac{\delta'}{\delta} = O(1),\ \ \forall x \geq 0 : \delta(0) \geq \delta(x) \right\}. \tag{2}$$

In (2), the function $\delta(x)$ is assumed to be continuously differentiable and δ' denotes it's derivative. The condition $\frac{\delta'}{\delta} = O(1)$ insures that $\delta(\beta k)$ varies slowly as a function of k. The last condition implies that the initial tolerance is the lowest. However, the strategies in $\mathcal{S}$ are not necessarily monotonically decreasing.

Since $\boldsymbol{\delta}$ depends on the function $\delta(x)$, we interchange between the two notations from here on. Furthermore, to make the problem easier to analyze, we use the fact that $\sum_{j=0}^{N(\epsilon,\delta)-1}(-\log \delta_j) \approx -\int_0^{N(\epsilon,\delta)} \log \delta(\beta t)dt$, and redefine the cost

$$C(\epsilon, \delta) = -\int_0^{N(\epsilon,\delta)} \log \delta(\beta t)dt. \tag{3}$$

We can now restate the problem as follows. Find $\delta^* \in \mathcal{S}$ such that

$$\mathcal{C}(\epsilon, \delta^*) = \min_{\delta \in \mathcal{S}} \mathcal{C}(\epsilon, \delta). \tag{4}$$

The outline of this paper is as follows. We first solve problem (4) for the Chebychev iteration. In Section 2, we review the method and describe previous results. In Section 3, we obtain an approximation to the error bound for the variable strategy CI. Using this bound, in Section 4, we show that for CI, the optimal strategy is δ = constant. We also estimate the optimal constant. Finally, in Section 5, we generalize this approach to other algorithms that satisfy a certain set of conditions. In appendices 1 and 2 we present a few numerical calculations that demonstrate the accuracy of our results from sections 3 and 4.

2 Review of the Chebychev Iteration

In this section, we first describe the Chebychev iteration (Manteuffel [7]). Then, we describe the constant and variable strategy variants of CI. Finally, we review some results which are relevant to this work.

The Chebychev iteration can be used to solve the real $n \times n$ system of linear equations

$$Ax = b \tag{5}$$

using the splitting

$$A = M - (M - A). \tag{6}$$

It requires that the spectrum of $M^{-1}A$ be contained in an ellipse, symmetric about the real axis, in the open right half of the complex plane. We denote the estimates of the foci of such an ellipse by l and u. Furthermore, we assume here that $M^{-1}A$ is diagonalizable.

The exact Chebychev method is defined by

$$x_1 = x_0 + \alpha z_0 \tag{7}$$

$$x_{k+1} = x_{k-1} + \omega_{k+1}(\alpha z_k + x_k - x_{k-1}), \qquad k = 1, 2, \ldots \tag{8}$$

where

$$Mz_k = r_k, \quad r_k = b - Ax_k, \tag{9}$$

$$\alpha = \frac{2}{l+u}, \qquad \mu = \frac{u+l}{u-l} \tag{10}$$

$$\omega_{k+1} = 2\mu \frac{c_k(\mu)}{c_{k+1}(\mu)}. \tag{11}$$

In (7), the initial iterate x_0 is given, and in (11), c_k denotes the Chebychev polynomial of degree k.

The constant strategy version of the method is obtained by replacing the equation for z_k in (9) by

$$Mz_k = r_k + q_k, \qquad \| q_k \| \leq \hat{\delta} \| r_k \|, \quad \hat{\delta} \in (0,1). \tag{12}$$

Then, the variable strategy scheme is obtained by replacing $\hat{\delta}$ by δ_k in (12).

If we denote the error at step k by

$$e_k = x - x_k, \tag{13}$$

and define,

$$K = I - \alpha M^{-1}A, \quad K = V\Sigma V^{-1}, \quad \Sigma = \text{Diag}(\sigma_j), \tag{14}$$

then, it can be shown that provided $\mu\sigma_j \neq \pm 1$,

$$\frac{\| V^{-1}e_k \|}{\| V^{-1}e_0 \|} \leq \frac{\rho^k \tau(k,\delta)}{|\cosh(k\cosh^{-1}(\mu))|}. \tag{15}$$

In (15), $\tau(k,\delta)$ satisfies the recurrence equation

$$\tau(k+1,\delta) - 2(1+\Delta\delta_k)\tau(k,\delta) + \tau(k-1,\delta) = 0, \tag{16}$$

with initial conditions

$$\tau(0,\delta) = 1, \qquad \tau(1,\delta) = 1 + 2\Delta\delta_0. \tag{17}$$

The constant Δ in (16) is given by

$$\Delta = \frac{\alpha|\mu| \| V^{-1}M^{-1} \| \| AV \|}{\rho}, \quad \rho = \max_j |e^{\theta_j}|, \quad \theta_j = \cosh^{-1}(\mu\sigma_j). \tag{18}$$

Equations (15) - (17) were obtained in an equivalent form by Golub and Overton [1] for the constant strategy case.

3 Asymptotic Approximation of the Error Bound for Variable Tolerance Chebychev Iteration

In this section, we complete the error analysis of CI for the case where the tolerance of the inner iteration varies slowly. In order to do so, we evaluate the right hand side of (15). Therefore, we first seek a solution to equation (16) for $\tau(k,\delta)$ when the strategy $\delta_k = \delta(\beta k)$ belongs to $\mathcal{S}$. Since we do not know how to solve that equation for an arbitrary strategy $\delta(\beta k)$, we derive the asymptotic expansion of it's solution for small β. To emphasize the fact that $\tau(k,\delta)$ now depends on the parameter β, we denote it by $\tau(k,\delta,\beta)$. We calculate only the leading order of the expansion for τ.

To simplify the analysis we assume that the function $\delta(x)$ is constant on $[0,\beta]$. This assumption is not very restrictive since it only requires that we change the value of δ_0 to

equal δ_1. Moreover, since δ_k is slowly varying the impact on the cost of this change is negligible.

The method we use is similar to the W.K.B method (Bender and Orszag [8]) for linear ordinary differential equations with a small parameter, and the ray method (Keller [9]) for linear partial differential equations with a small parameter. These methods have recently been adapted to linear difference equations with small parameters (see Giladi and Keller [10], Knessel [11], and references therein).

Let us now obtain an approximate solution to equation (16) when $\delta_k = \delta(\beta k)$ belongs to $\mathcal{S}$. Since we are looking for an asymptotic expansion of $\tau(k, \delta, \beta)$ for small β, we introduce in this equation the new scaled independent variable $x \equiv \beta k$. Then, we write $\tau(k, \beta, \delta)$ in the form

$$\tau(k, \delta, \beta) = R(x, \delta, \beta). \tag{19}$$

When (19) is used in (16), it leads to the following equation for R

$$R(x + \beta, \delta, \beta) = 2(1 + \Delta\delta(x))R(x, \delta, \beta) - R(x - \beta, \delta, \beta). \tag{20}$$

In order to solve (20), we recall that homogeneous linear difference equations with constant coefficients have solutions which are exponentials. The solutions are not exponentials though, in the case of variable coefficients. However, their asymptotic expansions, with respect to a small parameter, are generally exponentials multiplied by power series in the parameter. Therefore, we seek the asymptotic expression for $R(x, \delta, \beta)$ for small β, in the form

$$R(x, \beta, \delta) \sim e^{\psi(x,\delta)/\beta}[K(x, \delta) + \beta K_1(x, \delta) + \beta^2 K_2(x, \delta) + \ldots]. \tag{21}$$

The functions $\psi(x, \delta)$, $K(x, \delta)$, $K_1(x, \delta)\ldots$ are to be determined to make R satisfy (20).

Substitution of (21) into (20), and multiplication by $e^{-\psi/\beta}$ yields

$$e^{(\psi(x+\beta,\delta)-\psi(x,\delta))/\beta}(K(x + \beta, \delta) + \beta K_1(x + \beta, \delta) + \ldots) =$$

$$2(1 + \Delta\delta(x))[K(x, \delta) + \beta K_1(x, \delta) + \ldots] -$$
$$e^{-(\psi(x)-\psi(x-\beta))/\beta}(K(x - \beta, \delta) + \beta K_1(x - \beta, \delta) + \ldots). \tag{22}$$

We now express each side of (22) in powers of β, assuming that $\psi(x + \beta, \delta)$, $\psi(x - \beta, \delta)$, $K(x + \beta, \delta)$, etc.. can be expanded in Taylor series in powers of β. Then, we equate the coefficients of each power of β on the left hand side of (22), to the same power of β on the right hand side. The coefficients of β^0 and of β^1, yield the following equations for $\psi(x, \delta)$ and $K(x, \delta)$ respectively,

$$e^{2\psi_x} - 2(1 + \Delta\delta(x))e^{\psi_x} + 1 = 0, \tag{23}$$

$$\tanh(\psi_x)K_x + \frac{\psi_{xx}}{2}K = 0. \tag{24}$$

Although (23) is a non-linear ordinary differential equation for $\psi(x,\delta)$, it is a quadratic in e^{ψ_x} and can easily be solved for ψ_x, with the result

$$\psi_x(x,\delta) = \log\left(1+\Delta\delta(x) \pm \sqrt{(1+\delta(x)\Delta)^2-1}\right) = \pm\cosh^{-1}(1+\Delta\delta(x)). \tag{25}$$

Integrating (25) yields, with a a constant of integration

$$\psi(x,\delta) = \pm\int_0^x \cosh^{-1}(1+\Delta\delta(t))dt + a. \tag{26}$$

We note that ψ_x in (25) depends on x only through $\delta(x)$. Hence, we introduce the function $\Phi(\delta)$ defined as

$$\Phi(\delta) \equiv \psi_x(x,\delta) = +\cosh^{-1}(1+\Delta\delta). \tag{27}$$

Then, (26) can be rewritten as

$$\psi(x,\delta) = \pm\int_0^x \Phi(\delta(t))dt + a. \tag{28}$$

As we shall see in Section 4, the properties of Φ are very important in determining the optimal strategy for CI.

We now solve equation (24) for K. A slight manipulation of that equation yields

$$\frac{K_x}{K} = -\frac{\cosh(\psi_x)}{\sinh(\psi_x)}\frac{\psi_{xx}}{2}. \tag{29}$$

Integrating (29), leads to the solution for K, with b a constant of integration

$$K(x,\delta) = \frac{b}{\sqrt{|\sinh(\psi_x(x,\delta))|}}. \tag{30}$$

Now, we use expression (25) for ψ_x in (30) to obtain

$$K(x,\delta) = \frac{b}{((1+\Delta\delta(x))^2-1)^{1/4}}. \tag{31}$$

To obtain the leading order of $\tau(k,\delta,\beta)$, we substitute expression (28) for ψ in expression (21) for R. Then, we use the result in (19) and substitute $x \equiv \beta k$ to find

$$\tau(k,\delta,\beta) \sim K(\beta k,\delta)(Ae^{\frac{1}{\beta}\int_0^{\beta k}\Phi(\delta(t))dt} + Be^{-\frac{1}{\beta}\int_0^{\beta k}\Phi(\delta(t))dt}), \tag{32}$$

where $\Phi(\delta)$ is defined in (27) and $K(x,\delta)$ is given by (31). The constants A and B are determined to make right hand side of (32) satisfy the initial conditions (17) for τ. Therefore, we obtain

$$A = \frac{1}{2\sinh[\frac{1}{\beta}\int_0^\beta \Phi(\delta(t))dt]}\left(\frac{1+2\Delta\delta(0)}{K(\beta,\delta)} - \frac{e^{-\frac{1}{\beta}\int_0^{\beta k}\Phi(\delta(t))dt}}{K(0,\delta)}\right), \qquad B = \frac{1}{K(0,\delta)} - A. \tag{33}$$

Since $\delta(x)$ is constant on $[0,\beta]$ we see from equation (31) that $K(0,\delta) = K(\beta,\delta)$. Furthermore, we may replace $\frac{1}{\beta}\int_0^\beta \Phi(\delta(t))dt$ by $\Phi(\delta(0))$. Then, we substitute (33) in (32) to obtain after some manipulation the leading order of τ

$$\tau(k,\delta,\beta) \sim \frac{K(\beta k,\delta)}{K(0,\delta)}\left[\frac{2}{1+e^{-\Phi(\delta(0))}}\sinh\left(\frac{1}{\beta}\int_0^{\beta k}\Phi(\delta(t))dt\right) + e^{-\frac{1}{\beta}\int_0^{\beta k}\Phi(\delta(t))dt}\right]. \tag{34}$$

When $\delta(x) \equiv \hat{\delta}$ is constant for all $x \geq 0$, (25) implies that $\psi_{xx} = 0$. In this case, we see from (24) that K is also a constant. Hence, the right hand side of (34) simplifies to

$$\frac{2}{1+e^{-\Phi(\hat{\delta})}}\sinh(k\Phi(\hat{\delta})) + e^{-k\Phi(\hat{\delta})}. \tag{35}$$

Note that expression (35) is the exact solution of (16) and (17) when $\delta_k \equiv \hat{\delta}$ is a constant. It agrees with the solution obtained by Golub and Overton.

After a few outer iterations, the exponentially decaying term in (34) can be neglected. Furthermore, we perform the change of variable $s = t/\beta$ in (34) and thus introduce the function

$$\sigma(k,\delta) = \frac{K(\beta k,\delta)}{K(0,\delta)}\frac{2}{1+e^{-\Phi(\delta(0))}}\sinh\left(\int_0^k \Phi(\delta(\beta s))ds\right). \tag{36}$$

Then, we approximate the bound for the error in the right hand side of (15) with

$$B(k,\delta) = \frac{\sigma(k,\delta)\rho^k}{|\cosh(k\cosh^{-1}(\mu))|}. \tag{37}$$

We now briefly discuss the validity of the approximation (36). When $\delta \equiv \hat{\delta}$ is constant, expression (36) is exact up to an exponentially decaying term and it is very accurate after a few iterations. When δ is not a constant, the approximation is based on expression (34), which is valid for $\beta << 1$ and $\beta k = O(1)$. Therefore, the accuracy decreases as the number of outer iterations $k \to \infty$, and for a fixed k, increases as $\beta \to 0$. We are currently attempting to derive error bounds for this expansion. In Appendix 1, we present a few numerical calculations that demonstrate it's accuracy for a few variable strategies in $\mathcal{S}$. As we shall see, even for large values of k, it is very accurate. For most practical purposes this accuracy is sufficient. In the next section, we find the optimal strategy for CI based on the bound (37). We show that a limited region of validity is not a problem in this case.

4 Constant strategy is optimal

In this section, we show that the optimal strategy for the Chebychev iteration is to choose δ equal to a constant. Our conclusion is based on the approximate error bound (37), which we derived in the previous section. The numbers $N(\epsilon,\delta)$ and $\mathcal{C}(\epsilon,\delta)$ in (3), are hard to

determine precisely. Therefore, we introduce the quantities $N_B(\epsilon,\delta)$ and $C_B(\epsilon,\delta)$, which are the number of outer iterations required to reduce the error bound (37) to ϵ and the associated cost, respectively. Based on the following theorem, we shall conclude that a constant strategy is optimal for CI.

Theorem 1 *Assume that a linear system of equations is solved to accuracy ϵ, using the Chebychev iteration, with some variable strategy δ. Then, there exists a constant strategy $\hat{\delta}(\delta,\epsilon)$, for which the cost*

$$C_B(\epsilon,\hat{\delta}) \leq C_B(\epsilon,\delta).$$

Proof: Given the variable strategy δ and the accuracy ϵ, used in the solution of the linear system, we define the associated constant strategy $\hat{\delta}(\delta,\epsilon)$

$$\hat{\delta}(\delta,\epsilon) = \Phi^{-1}\left(\frac{\int_0^{N_B(\epsilon,\delta)} \Phi(\delta(\beta t))dt}{N_B(\epsilon,\delta)}\right). \tag{38}$$

The function Φ in (38) is defined in (27).

In Lemma 1, we show that $N_B(\epsilon,\hat{\delta}) \leq N_B(\epsilon,\delta)$. Therefore,

$$C_B(\epsilon,\hat{\delta}) = -\int_0^{N_B(\epsilon,\hat{\delta})} \log\hat{\delta}dt = -\log\hat{\delta}N_B(\epsilon,\hat{\delta}) \leq -\log\hat{\delta}N_B(\epsilon,\delta). \tag{39}$$

In Lemma 2, we show that

$$-\log\hat{\delta}N_B(\epsilon,\delta) \leq C_B(\epsilon,\delta). \tag{40}$$

Using (40) in the right hand side of (39), proves the theorem.

Lemma 1

$$N_B(\epsilon,\hat{\delta}) \leq N_B(\epsilon,\delta) \tag{41}$$

Proof: By definition of $N_B(\epsilon,\delta)$ the bound for the error $B(k,\delta)$ in (37) satisfies

$$B(N_B(\epsilon,\delta),\delta) \leq \epsilon.$$

Therefore, to prove (41) it is sufficient to show that after $N_B(\epsilon,\delta)$ outer iterations, the bound for the error associated with the variable strategy is greater than the one associated with the constant strategy. Hence we need to show

$$B(N_B(\epsilon,\delta),\delta) \geq B(N_B(\epsilon,\delta),\hat{\delta}). \tag{42}$$

We see from (37) that (42) is equivalent to the following inequality

$$\sigma(N_B(\epsilon,\delta),\delta) \geq \sigma(N_B(\epsilon,\delta),\hat{\delta}), \tag{43}$$

where σ is defined in (36). To prove (43) we begin by rewriting expression (36) for $\sigma(k,\delta)$ with $k = N_B(\epsilon,\delta)$,

$$\sigma(N_B(\epsilon,\delta),\delta) = \frac{2}{1+e^{-\Phi(\delta(0))}} \frac{K(\beta N_B(\epsilon,\delta),\delta)}{K(0,\delta)} \sinh\left(\int_0^{N_B(\epsilon,\delta)} \Phi(\delta(\beta t))dt\right). \tag{44}$$

Then, we note that Φ is monotonically increasing from (27) and that for all non-negative x: $\delta(0) \geq \delta(x)$ from (2). Therefore,

$$\Phi(\hat{\delta}) = \frac{\int_0^{N_B(\epsilon,\delta)} \Phi(\delta(\beta t))dt}{N_B(\epsilon,\delta)} \leq \Phi(\delta(0)), \tag{45}$$

$$\frac{2}{1+e^{-\Phi(\delta(0))}} \geq \frac{2}{1+e^{-\Phi(\hat{\delta})}}. \tag{46}$$

Furthermore, we see that $K(\beta N_B(\epsilon,\delta),\delta)/K(0,\delta) \geq 1$ from equation (31). Using this and (46) in the right hand side of (44) we obtain

$$\sigma(N_B(\epsilon,\delta),\delta) \geq \frac{2}{1+e^{-\Phi(\hat{\delta})}} \sinh\left(N_B(\epsilon,\delta)\Phi\Phi^{-1}\left(\frac{\int_0^{N_B(\epsilon,\delta)} \Phi(\delta(\beta t))dt}{N_B(\epsilon,\delta)}\right)\right) = \sigma(N_B(\epsilon,\delta),\hat{\delta}). \tag{47}$$

Lemma 2

$$-\log \hat{\delta} N_B(\epsilon,\delta) \leq \mathcal{C}_B(\epsilon,\delta)$$

Proof: From the definition of Φ in equation (27), we find that $\Phi^{-1}(x) = \frac{\cosh(x)-1}{\Delta}$. Therefore,

$$\frac{d^2}{dx^2}\left(-\log \Phi^{-1}(x)\right) = (\cosh(x)-1)^{-1} > 0$$

and $-\log \Phi^{-1}(x)$ is stricly convex on the interval $\{\Phi(\delta(x)) \mid 0 \leq x \leq N_B(\epsilon,\delta)\}$. It follows from Jensen's inequality that

$$-\log \hat{\delta} = -\log \Phi^{-1}\left(\frac{\int_0^{N_B(\epsilon,\delta)} \Phi(\delta(\beta t))dt}{N_B(\epsilon,\delta)}\right) \leq \frac{-\int_0^{N_B(\epsilon,\delta)} \log \Phi^{-1}\Phi(\delta(\beta t))dt}{N_B(\epsilon,\delta)} = \frac{\mathcal{C}_B(\epsilon,\delta)}{N_B(\epsilon,\delta)}. \tag{48}$$

Multiplying (48) by $N_B(\epsilon,\delta)$ proves the lemma.

Remarks

1. The proof is not affected by the limited region of validity of the approximation (37). Since if the approximation is highly accurate for iterations $k \leq M$, we first modify δ to be a constant on the interval $[0,M]$ as we did above. We denote this new strategy by δ_1. For δ_1 the approximation (37) is exact, up to an exponentially decaying term, on the interval $[0,M]$. Therefore, the region of high accuracy of (37) for δ_1 is $[0,2M]$. We can now proceed by induction until we have a constant strategy on $[0,N_B(\epsilon,\delta)]$ which is more efficient then the original strategy.

2. Assuming the error bound (37) is equally sharp for any strategy $\delta \in \mathcal{S}$, we conclude from Theorem 1 that the optimal strategy for CI is a constant.

We now show how to estimate the optimal constant $\hat{\delta}$ for CI. We note from (37) that for any iteration N

$$B(N,\hat{\delta}) \leq 2\frac{\rho^N \sinh(N\Phi(\hat{\delta}))}{|\cosh(N\cosh^{-1}(\mu))|} \approx 2e^{N[\log\rho+\Phi(\hat{\delta})-\mathrm{Re}(\cosh^{-1}(\mu))]}. \tag{49}$$

Then, by equating (49) to some accuracy ϵ and using (27), we obtain

$$N_B(\epsilon,\hat{\delta}) \approx \frac{\log 2 - \log\epsilon}{\mathrm{Re}(\cosh^{-1}(\mu)) - \log\rho - \cosh^{-1}(1+\Delta\hat{\delta})}. \tag{50}$$

An estimate of the cost is then

$$\mathcal{C}_B(\epsilon,\hat{\delta}) \approx \frac{-\log\hat{\delta}(\log 2 - \log\epsilon)}{\mathrm{Re}(\cosh^{-1}(\mu)) - \log\rho - \cosh^{-1}(1+\Delta\hat{\delta})}. \tag{51}$$

The right hand side of (51) can be easily minimized with respect to $\hat{\delta}$ using a standard minimization technique. The original variational problem (4) is reduced to a simple optimization problem.

The determination of the optimal constant depends on our knowledge of the parameters μ and ρ. These are often determined adaptively while solving the system [12]. We are currently studying the algorithmic aspects of finding the optimal parameter.

5 Generalization to Other Iterative Procedures

In this section, we consider a class of iterations which satisfy the following condition:

- There exists a bound on the error, $B(N,\delta)$, which can be written as

$$B(N,\delta) = F(N)\tau(N,\delta,\beta). \tag{52}$$

  In (52), F is known and $\tau(N,\delta,\beta)$ satisfies a recurrence equation of the form

$$\sum_{j=0}^{m} A_j(\delta_j)\tau(k-j,\delta,\beta) = 0, \quad \delta_j = \delta(\beta j), \quad \beta << 1, \tag{53}$$

  with appropriate initial conditions. Note that the coefficients A_j in (53) depend on j only through δ_j.

The Chebychev and Richardson iterations satisfy the above condition. We are currently attempting to identify other m-term linear iterations as candidates for this class. We note that the bound (15) has the form (52) where

$$F(k) = \frac{\rho^k}{|\cosh(k \cosh^{-1}(\mu))|}. \tag{54}$$

while equation (16) is in the form (53). The asymptotic methods described in Section (3) can be applied to equations of the form (53).

If the recurrence (53) has three terms for some iterative procedure, the approximate solution has a form similar to (34). Then, the proof of Section 4 can be used to show that a constant strategy is optimal, provided the following criteria are satisfied:

1. $\Phi(\delta)$ is monotonically increasing in δ.
2. $-\log \Phi^{-1}$ is convex.
3. $K(\delta(x))$ is monotonically decreasing in δ.

When it is not determined that the optimal strategy is constant, it is hard to obtain $N_B(\epsilon, \delta)$ from $B(N, \delta)$ in (37), without making further assumptions on δ. In this case, we propose to restrict the minimization to a subclass of $\mathcal{S}$. One possible example is the three parameter family $\delta(\beta k) = \frac{A}{B(1+(\beta k)^\gamma)}$ where $1 \leq \gamma \leq 2$ and $A, B > 0$. Then, we equate $B(N, \delta)$ to some accuracy ϵ and manipulate the result to obtain approximations to $N_B(\epsilon, \delta)$ and $\mathcal{C}_B(\epsilon, \delta)$, as we did at the end of Section 4. This reduces (4) to an optimization problem in the parameters of the strategy. Many methods exist to solve such a problem (Gill, Murray, Wright [13]).

6 Acknowledgements

I would like to thank professor Gene Golub for introducing me to this problem. I would like to thank my advisor, professor J.B.Keller, for teaching me the asymptotic analysis tools which were invaluable for this project. Part of this project was done in the summer of 1993, while I was a summer student at Schlumberger Doll Research (SDR). I would like to thank Dr R.Burridge of SDR for enabling me to spend time on this work.

Appendix 1

In this appendix, we present a few numerical calculations that demonstrate the accuracy of the expansion derived in section 3. In each calculation, we solve equation (16) for $\tau(k, \delta, \beta)$ by iteration for all $0 \leq k \leq 2000$. Then, we compute the approximate solution $\sigma(k, \delta)$ in (36) for each k which is a multiple of 10 and is less than 2000. We present the relative error in this approximation.

For these calculations, we use in equation (16) strategies from the three parameter family $\delta_k = \frac{A}{B(1+(\beta k)^\gamma)}$, $k \geq 1$, $\delta_0 = \delta_1$. The value of the parameter A is fixed at 1. The parameters B and γ and the value of β vary from one experiment to the other. The value of Δ in equation (16) is set to 37. We also experimented with larger values of Δ and obtained similar results.

In table 1, we present the maximum with respect to k, of the absolute value of the relative error in percent. Each entry in this table corresponds to a calculation with a different strategy. The strategy is determined by the parameters B and β. All strategies in this table have γ set to 1. Table 2 is analogous to table 1 but there $\gamma = 1.5$.

$B \backslash \beta$	.1	.01
1.01	0.74	0.05
1.10	0.74	0.05
1.50	0.74	0.05
2.00	0.73	0.05
5.00	0.72	0.07
10.00	0.71	0.07
100.00	0.70	0.11

Table 1: Maximal relative error in (%).

$B \backslash \beta$	.1	.01
1.01	0.44	0.06
1.10	0.44	0.06
1.50	0.46	0.05
2.00	0.48	0.05
5.00	0.59	0.05
10.00	0.72	0.05
100.00	1.64	0.13

Table 2: Maximal relative error in (%).

Appendix 2

In this appendix, we present a few numerical experiments that validate the analysis of section 4. In each experiment, we solve a linear system with CI and some variable strategy δ. The system is solved to some accuracy ϵ. Then, we solve the same system with the associated constant strategy $\hat{\delta}(\delta, \epsilon)$. Strategy $\hat{\delta}$ is defined in (38) with $N_B(\epsilon, \delta)$ replaced by $N(\epsilon, \delta)$. We check if the predictions of Lemma 1 and Theorem 1 hold in practice. The accuracy of these predictions depends on the convergence factor ρ of the method used for the inner-iteration. If ρ is close to 1, the relative error between $\lceil \frac{-\log \delta_j}{-\log \rho} \rceil$ and $\frac{-\log \delta_j}{-\log \rho}$ is usually small. In this case, the cost in (1) and in (3) is truly proportional to the total number of inner iterations. Furthermore, since $\hat{\delta}$ assumes in practice only the value ρ^k for some $k = 1, 2, 3 \ldots$, it is better approximated when ρ is close to 1. Hence, we expect that the analysis will closely fit the experiments whenever ρ is close to 1. When ρ is small, some fluctuations around the predicted behavior is expected. Our experiments cover both cases of ρ.

In these experiments, we solve the symmetric system

$$Ax = b, \tag{55}$$

arising from the central difference discretization of the equation

$$\frac{d^2 f(x)}{dx^2} + (\sin(10x) + 1) * Cf(x) = g(x), \tag{56}$$

in the interval $[0, 1]$. The boundary conditions are homogeneous . The right hand side b in (55) is chosen at random. The splitting matrix M is obtained from the discretization of the equation

$$\frac{d^2 f(x)}{dx^2} + Cf(x) = g(x), \tag{57}$$

with the same boundary conditions. The number of interior points is $N = 100$ and the accuracy $\epsilon = 10^{-12}$. All initial iterates are 0.

In all our experiments, we use strategies from the family $\delta_k = \frac{A}{B(1+(\beta n)^\gamma)}$, $k \geq 1$, $\delta_0 = \delta_1$. The values of γ and A are fixed at 1. The parameter B and the value of β vary from one experiment to the other. For each variable strategy δ, the associated constant strategy $\hat{\delta}$ is computed from (38) with $N_B(\epsilon, \delta)$ replaced by $N(\epsilon, \delta)$. In the right hand side of (38) Φ depends on Δ and we evaluate Δ exactly. However, we note that the expression for $\hat{\delta}$ is not very sensitive to fluctuations in the value of Δ.

In the first series of experiments we set $C = .05$ in (56) and (57).

We use two methods for the inner iteration. The symmetric Gauss Seidel, with convergence factor $\rho = .905$, and S.S.O.R, with a smaller convergence factor $\rho = .643$. In the S.S.O.R iteration, the relaxation parameter ω is the optimal parameter ω^* of S.O.R. In each experiment, we log the number of outer iterations and the total number of inner iterations for the variable and constant strategy cases.

Tables 3 and 4 correspond to the case where the inner iteration is the symmetric Gauss Seidel. In table 3 we report the difference in the total number of inner iterations between the variable strategy case and the associated constant strategy case. All entries in the table are in (%) that is, each entry is computed by

$$\frac{\mathrm{N}_{in}(\epsilon,\delta) - \mathrm{N}_{in}(\epsilon,\hat{\delta})}{\mathrm{N}_{in}(\epsilon,\hat{\delta})} * 100. \tag{58}$$

The number $\mathrm{N}_{in}(\epsilon,\delta)$ in (58) is the total number of inner iterations performed when solving the system to accuracy ϵ with strategy δ.

Each entry in table 3 corresponds to a different strategy. The strategy is determined by the parameters B and β. Note that not all strategies are slowly varying. We also note that all entries in this table are positive. Hence, there is agreement with Theorem 1.

$B\backslash\beta$	.1	.5	1	2
1.01	3.62	3.59	5.44	1.62
1.10	3.34	4.92	4.79	3.77
1.50	4.06	3.65	3.86	3.64
2	3.19	4.11	3.75	3.23
5	1.82	2.94	3.41	3.42
10	1.45	2.72	2.66	2.71
100	0.73	1.67	2.10	2.03

Table 3: Difference in number of inner iterations in (%).

In table 4, we present the difference in the number of outer iterations between the variable strategy case and the constant strategy case. We see that all entries except one are non-negative. Hence, there is very good agreement with Lemma 1.

$B\backslash\beta$	.1	.5	1	2
1.01	1	0	1	−1
1.10	1	1	0	0
1.50	2	0	0	0
2	1	0	0	0
5	0	0	0	0
10	0	0	0	0
100	0	0	0	0

Table 4: Difference in number of outer iterations.

Tables 5 and 6 are analogous to tables 3 and 4 respectively. They correspond to the case where the inner iteration is solved with S.S.O.R. The negative entries in these tables do not agree with the predictions of our model. However, the magnitude of these fluctuations is small. These fluctuations are expected in this case since ρ is small.

$B\backslash\beta$	.1	.5	1	2
1.01	−1.18	0.96	2.33	5.98
1.10	0.78	2.24	1.75	7.55
1.50	6.25	−2.04	2.17	3.46
2	−1.75	1.75	7.07	−0.91
5	−2.33	2.96	2.73	−0.40
10	1.09	3.83	0.20	3.66
100	1.28	−0.74	4.06	3.88

Table 5: Difference in number of inner iterations in (%).

$B\backslash\beta$	.1	.5	1	2
1.01	6	1	0	0
1.10	5	0	−1	0
1.50	−1	1	1	0
2	1	−1	0	0
5	1	0	−1	0
10	0	−1	0	0
100	0	0	1	0

Table 6: Difference in number of outer iterations.

In the last set of experiments we set $C = .001$ in (56) and (57) and use the symmetric Gauss Seidel method for the inner iteration. In this case, the convergence factor is $\rho = .973$. As tables 7 and 8 demonstrate, we have perfect agreement with the predictions of Lemmas 1 and 2.

$B\backslash\beta$	.1	.5	1	2
1.01	2.81	2.24	2.61	3.04
1.10	1.18	1.81	2.56	3.03
1.50	1.01	2.27	2.65	3.02
2	0.92	2.27	2.86	3.22
5	4.06	2.64	3.33	3.55
10	1.30	2.82	3.24	3.81
100	1.36	0.96	0.87	1.29

Table 7: Difference in number of inner iterations in (%).

$B\backslash\beta$	.1	.5	1	2
1.01	1	0	0	0
1.10	0	0	0	0
1.50	0	0	0	0
2	0	0	0	0
5	1	0	0	0
10	0	0	0	0
100	0	0	0	0

Table 8: Difference in number of outer iterations..

We are currently performing further experimentation with two dimensional problems. We are also experimenting with the optimal constant.

References

[1] G.H. Golub and M.L. Overton. The convergence of inexact chebyshev and richardson iterative methods for solving linear systems. *Numer. Math.*, 53:571–593, 1988.

[2] R.A.Nicolaides. On the local convergence of certain two step iterative procedures. *Numer. Math.*, 24:95–101, 1975.

[3] V.Pereyra. Accelerating the convergence of discretization algorithms. *SIAM J. Numer. Anal.*, 4:508–533, 1967.

[4] N.K.Nichols. On the convergence of two-stage iterative process for solving linear equations. *SIAM J. Numer. Anal.*, 10:460–469, 1973.

[5] S.C.Eisenstat R.S.Dembo and T.Steihaug. Inexact newton methods. *SIAM J. Numer. Anal.*, 19:400–408, 1982.

[6] H. C. Elman and G. H. Golub. Inexact and preconditioned uzawa algorithms for saddle point problems. Technical report, Stanford University, June 1993.

[7] T. A. Manteuffel. The tchebyshev iteration for nonsymmetric linear systems. *Numer. Math.*, 28:307–327, 1977.

[8] C. M. Bender and S.Orszag. *Advanced Mathematical Methods for Scientists and Engineers.* Mc. Graw Hill, 1978.

[9] J.B.Keller. Rays, waves and asymptotics. *Bull. Am. Math. Soc.*, 84:727–750, 1978.

[10] E. Giladi and J. B. Keller. Eulerian number asymptotics. *To appear in Proceedings of the London Royal Society, Series A*, 1994.

[11] C. Knessl. The wkb approximation to the g/m/m queue. *SIAM J. Appl. Math.*, 51:1119–1133, 1991.

[12] T. A. Manteuffel. Adaptive procedure for estimation of parameters for the nonsymmetric tchebychev iteration. *Numer. Math.*, 31:187–208, 1978.

[13] W.Murray P.Gill and M.Wright. *Practical Optimization.* Academic Press, 1981.

Wednesday, April 6

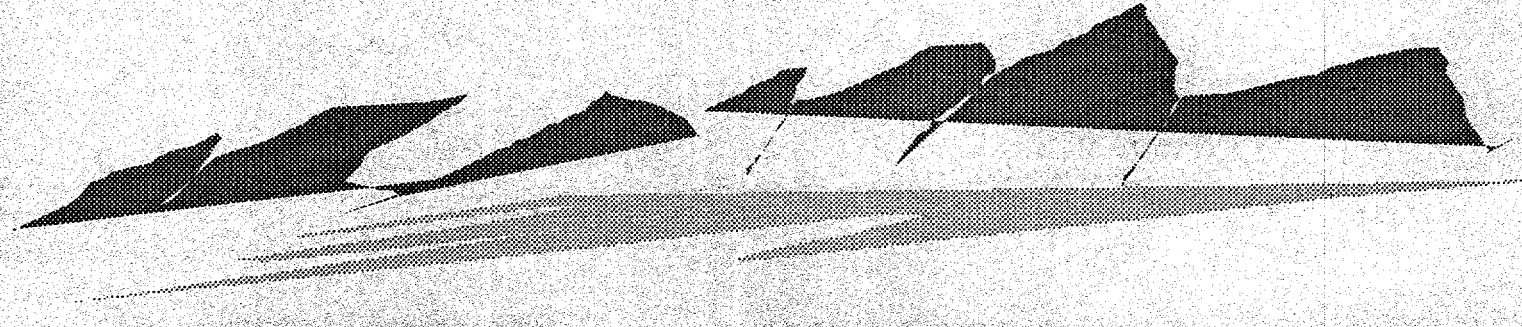
Software and Programming Environments
Chair: Steve Ashby
Room B

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



Object-oriented Design of Preconditioned Iterative Methods*

Extended abstract for CCIM '94

Are Magnus Bruaset**

January 3, 1994

In this talk we discuss how object-oriented programming techniques can be used to develop a flexible software package for preconditioned iterative methods. The ideas we describe have been used to implement the linear algebra part of Diffpack, which is a collection of C++ class libraries that provides high-level tools for the solution of partial differential equations. In particular, this software package is aimed at rapid development of PDE-based numerical simulators, primarily using finite element methods.

Over the past few years there has been a growing interest in numerical software engineering based on object-oriented principles. The experiences made in the computer science community suggests that this approach can lead to considerable savings in cost of development and maintenance for numerical applications as well. Object-oriented languages, e.g. C++, encourage modular programming in that subroutines and the data items on which they operate can be grouped together in a *class*. Moreover, details that are specific to the actual implementation of a class can be marked as private. Such data hiding permits a high level of transparency, directing the user to focus on the public interface of the class rather than irrelevant details. In this way it is simple to replace the private parts of a class with a new implementation without changes in the application program, or even offering the user a uniform interface to a range of alternative methods. The class concept also permits the programmer to derive new classes from old ones, in that data members and functionality can be inherited. This makes it easier to separate generic code segments from problem-specific parts of the program, thus reducing redundancy and encouraging reuse of existing software. In general, an object-oriented design can lead

*This work was supported by the Royal Norwegian Council for Scientific and Industrial Research (NTNF) through program no. STP.28402: *Toolkits in industrial mathematics at the Center for Industrial Research (SI).*

**SINTEF SI, P.O. Box 124 Blindern,N-0314 Oslo, Norway. The author's email address is `Are.Magnus.Bruaset@si.sintef.no`.

to higher flexibility and provide a better fundament for future expansions than what is possible with traditional functional programming. However, for most scientific computing applications special care should be taken to ensure that the gained flexibility still permits a high level of computational efficiency.

By its nature, linear algebra is object-oriented and provides evident building blocks like matrices and vectors with associated operations, e.g. matrix factorizations and dot products. From a computational point of view, the picture gets a bit more complicated when we want to represent matrices with different structural properties, such as dense and sparse formats. However, the use of inheritance and mechanisms for data hiding allows an elegant solution to this problem. In particular, we will show how a wide range of matrix storage formats are implemented in Diffpack in a way that is totally transparent to iterative solvers. Moreover, we discuss an object-oriented framework for implementation of preconditioners and Krylov subspace methods, which makes it very easy to implement new solvers.

Matrix formats. Any specific matrix format in Diffpack is derived from the abstract base class `Matrix` which defines the member functions that are in common for all types of matrices, such as the matrix by vector product. The purpose of this design is that any code segment that operates on a `Matrix` can be applied to any matrix format, e.g. a general sparse matrix, without knowledge of its internal representation. Consequently, the actual matrix format may be chosen interactively rather than at compile-time. At present, Diffpack supports several matrix formats: dense, banded, diagonal, tridiagonal, structured sparse and general sparse matrices as well as point operators. The latter format represents a sparse matrix as a computational molecule, thus providing the functionality needed for iterative methods at very low storage cost. It is easy to add new matrix formats to the package without changes in previously defined classes.

Solving linear systems. The linear system $Ax = b$ is conveniently represented by its own class `LinEqSystemStd`. If this system is to be solved by a preconditioned iterative method, the actual preconditioners can be implemented as additional data members of a derived class representing the preconditioned system. The left and right preconditioners C_L and C_R are then instanced by objects from the `Precond` hierarchy which provides a generic interface to preconditioners whether represented by a preconditioning matrix or by a self-contained algorithm, such as a multigrid sweep. For example, we may have a C_L of type `PrecRILU` and a C_R of type `PrecNone`, representing a relaxed incomplete LU factorization applied to the left and no right preconditioner ($C_R = I$).

Once we have established a system, preconditioned or not, it may be solved by a direct or iterative method implemented as a derivation of the base class `LinEqSolver`. Regardless of the chosen method, the solver is invoked simply by a call to the current member function `solve`. For direct methods, we restrict the

coefficient matrix to be of a type that know how to factorize itself, e.g. a banded matrix. Similar restrictions may also be present for certain types of preconditioners. The `LinEqSolver` hierarchy have several levels providing a suitable environment for implementing stationary iterations and Krylov subspace methods. In particular, when implementing a new solver, we derive a new class which defines the corresponding `solve` function. The framework provided by its base class automatically gives access to a wide range of convergence monitors. These objects, which can be instructed to act as stopping criteria for the iteration or merely to monitor the convergence history for later graphing, are implemented as a separate collection of classes. Several monitor objects can be combined at run-time to achieve compound convergence criteria, using serial and/or parallel couplings. In order to communicate information such as residual vectors or precomputed inner products from the iterative solver to a convergence monitor, the framework supplies a communication block. This concept allows exchange of data between the solver and external modules without redundant storage, e.g. by letting a convergence monitor access the residual vector through a pointer rather than passing the vector itself. In fact, the communication block also allows coupling to other external objects than convergence monitors, e.g. an object that computes eigenvalue estimates on basis of parameters from an iteration such as CG or BiCGSTAB. These estimates are put back into the same communication block and are thus available to convergence monitors, preconditioners or iterations that rely on spectral information.

The suggested design is a serious attempt to offer developers a robust framework for implementation of linear solvers based on modern programming paradigms and tools. It is believed that the availability of object-oriented programming environments like the one proposed here can give significant contributions to future developments of linear algebra software.

Availability. It has been decided to release the Diffpack system as a public domain package. According to current plans, the release will take place in the late summer of 1994.

Submitted to: "Colorado Conference on Iterative Methods"

Voila

a

Visual Object-Oriented Iterative Linear Algebra Problem Solving Environment

---

H. Carter Edwards
phone: (512) 471-4710
email: carter@slave.ae.utexas.edu

Dr. Linda J. Hayes
phone: (512) 471-4208
email: lhayes@ticom.ae.utexas.edu

Computational and Applied Mathematics Program,
Department of Aerospace Engineering and Engineering Mechanics,
Mail Code 60600, University of Texas, Austin, TX 78712-1085

Overview

Application of iterative methods to solve a large linear system of equations currently involves writing a program which calls iterative method subprograms from a large software package. These subprograms have complex interfaces which are difficult to use and even more difficult to program. A problem solving environment specifically tailored to the development and application of iterative methods is needed. This need will be fulfilled by **Voila**, a problem solving environment which provides a visual programming interface to object-oriented iterative linear algebra kernels.

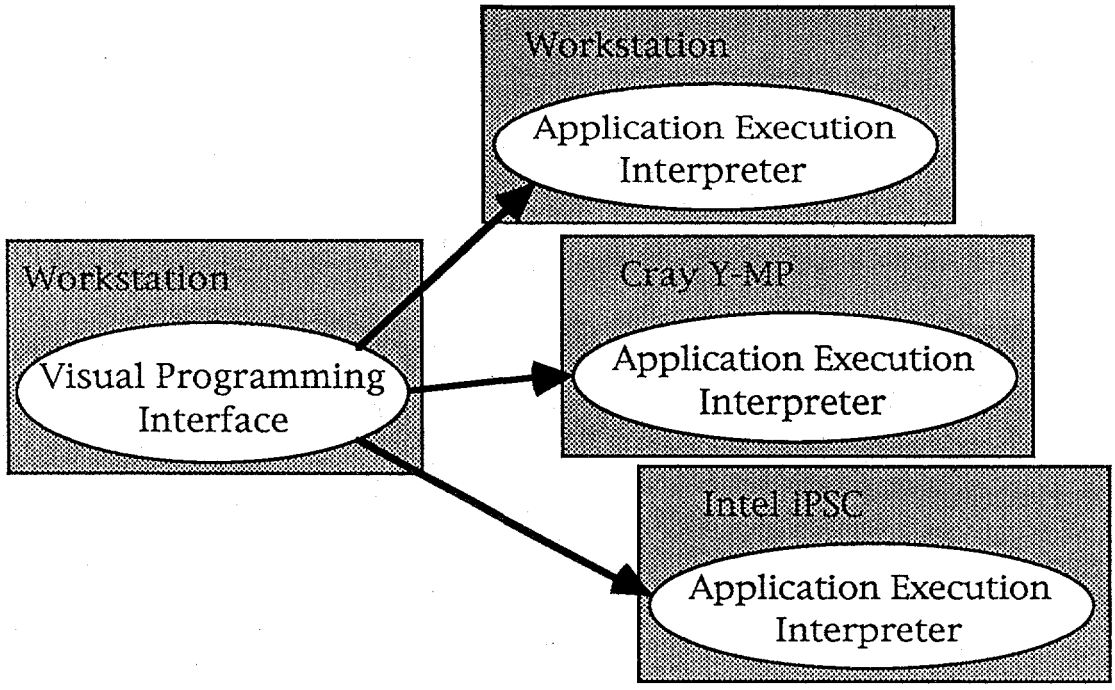
Voila will provide several quantum improvements over current iterative method problem solving environments. First, programming and applying iterative methods is considerably simplified through Voila's visual programming interface. Second, iterative method algorithm implementations are independent of any particular sparse matrix data structure through Voila's object-oriented kernels. Third, the compile-link-debug process is eliminated as Voila operates as an interpreter.

Specifications for Voila's visual programming language, object-oriented kernels, and interpreter interface will be presented in the full paper. Implementation of Voila is in progress on a Sun SPARC station where all components of Voila will be integrated and verified. Voila will then support additional implementations of the interpreter and associated object-oriented kernels on other supercomputer platforms.

Problem Solving Environment

Voila is a problem solving environment tailored to algorithms which solve large sparse systems of linear equations. Most of these algorithms are based upon iterative application of basic linear algebra operations. Voila is an environment for expediently implementing and applying these iterative algorithms.

The Voila environment is partitioned into two primary components, the visual programming interface and application execution interpreter. The visual programming interface is hosted on a workstation supporting X Windows. The interpreter will be hosted on multiple classes of computers, from workstations to supercomputers. This partitioning allows users to select the computational host most appropriate for executing their particular application.



Visual Programming

Visual programming in Voila will be very expressive and intuitive for both novice and experienced users. Coding through Voila's visual programming interface involves selecting icons from a library and wiring the icons together in a data flow diagram. Voila's visual programming language has two significant departures from conventional data flow diagrams. First, the lines which connect icons represent objects as opposed to passive data. Second, control flow is implicitly expressed in the diagrams via composite subdiagrams. In Voila's data flow diagram

paradigm icons represent either operations performed by objects or operations performed by macros. A macro is a composite subdiagram which is independently coded in Voila and stored for reuse in Voila's library.

Novice users implementing their applications will be assisted by Voila with interactive macro browsing and automatic interface checking. Macros for iterative methods will include system builders, preconditioners, methods, accelerators, and parameter estimators. As a user selects macros from the library Voila will assist the user by noting correct interfaces and disallowing incompatible interfaces.

Object-Oriented Kernels

The kernels of Voila's visual programming language will include object classes for iterative linear algebra. These intrinsic classes include the usual scalars and vectors; however, the significant innovation in Voila's classes is the class hierarchy for linear operators. The root of this class hierarchy is an abstract base class from which all specific linear operator classes are derived. The linear operator base class defines the system properties and vector operations required by iterative method algorithms. It is through this class hierarchy that iterative algorithms are made independent of any specific data structure for sparse systems.

The derived linear operator classes can include conventional sparse systems such as diagonal matrices, permutation matrices, banded matrices, or common sparse matrix data structures. The derived classes may also include composite matrix data structures. These composite matrices could include assemblies of row/column blocked submatrices, band blocked submatrices, or summations of mapped matrices.

Iterative algorithms such as system builders, preconditioners, methods, accelerators, and parameter estimators will be programmed by experienced users and stored in Voila as macros. These macros will utilize linear operator classes which are as close to the base class as the algorithm permits. This allows the iterative algorithms to be instantiated with a matrix class which is optimal for a particular application or computational environment.

Interpreter

Visually programmed applications are executed by Voila's interpreter. While the visual programming task is performed entirely on a user's workstation, execution of the application will typically be performed on a supercomputer. The programming and execution tasks have been separated to place each task on the most appropriate hardware.

Voila's approach of using an interpreter as opposed to a code generation scheme is motivated by the observation that the majority of the compute time for solving large sparse (or dense) linear systems is spent in the linear algebra kernels. In comparison, the compute time required to sequence these kernels is trivial. The additional compute time incurred interpreting a kernel based algorithm, as opposed to compiling and linking the same algorithm with the kernels, is not considered to be significant.

Multilevel Adaptive Solution Procedure for Material Nonlinear problems in Visual Programming Environment

D. Kim
Graduate Assistant
dongkim@lictor.acsu.buffalo.edu

R. Ghanem
Assistant Professor
ghanem@venus.eng.buffalo.edu

Department of Civil Engineering,
State University of New York, Buffalo, NY 14260
Tel:+01 716 645 3938, Fax:+01 716 645 3948

Abstract

Multigrid solution technique to solve a material nonlinear problem in a visual programming environment using the finite element method is discussed.

The nonlinear equation of equilibrium is linearized to incremental form using Newton-Rapson technique, then multigrid solution technique is used to solve linear equations at each Newton-Rapson step.

In the process, adaptive mesh refinement, which is based on the bisection of a pair of triangles, is used to form grid hierarchy for multigrid iteration.

The solution process is implemented in a visual programming environment with distributed computing capability, which enables more intuitive understanding of solution process, and more effective use of resources.

ITPACK Project: Past, Present, and Future*

David R. Kincaid, Linda J. Hayes, David M. Young
Center for Numerical Analysis
The University of Texas at Austin

February 28, 1994

Abstract

A status report is given on past, present, and future work in the development of research-oriented software packages. Our current and future work involves the following areas of research and associated software development:

- Parallel Iterative Software Using Kernels
- Visual Programming
- Research on Iterative Algorithms
- Objective-oriented Design

The overall objective of this research is to advance iterative methods software technology.

Introduction

In solving partial differential equations using finite difference or finite element methods, the most computationally intensive part is the solution of large sparse systems of linear and nonlinear equations. Such problems arise in many important engineering application areas such as oil reservoir simulation, numerical weather prediction, nuclear reactor simulation, etc. To obtain accurate numerical solutions, it is usually necessary to have a very fine mesh and/or a large number of elements. In either case, one is ultimately faced with solving systems of immense size with only a relatively few nonzeros entries per row. Iterative

*This work was supported, in part, by the National Science Foundation under Grant ASC–8917592, the Department of Energy under Grant DE–FG05–93ER25183, Cray Research, Inc., under Grant LTR DTD, and Texas Advanced Research Program under Grant TARP-212 with The University of Texas at Austin.

methods are ideally suited for such problems. Nevertheless, the resulting solution techniques can require an extensive amount of computer time and computer memory even on a uniprocessor supercomputer. The use of iterative methods, new software environments, and/or multiprocessor computers may be the only way to solve many important applications involving 3-D problems.

Past

The Center for Numerical Analysis has an active program of research on iterative algorithms for solving sparse linear systems of algebraic equations with special emphasis on systems arising in the numerical solution of partial differential equations. An important aspect of this work, known as the *ITPACK Project*, has resulted in the development of several research-oriented software packages. `ITPACKV 2D` is a package of seven iterative algorithms for solving sparse linear systems with symmetric positive definite or midly nonsymmetric coefficient matrices. The iterative algorithms in this package have been vectorized and parallelized for efficient use on vector and parallel computers such as the Cray Y-MP, the Alliant, the Sequent Symmetry, etc. By employing *wave-front* techniques, the modified software has resulted in an improved performance of several of the iterative algorithms that could not be vectorized previously. Recently, a parallel version of `ITPACKV` was written and tested on the Cray Y-MP and it resulted in a significant increase in the speed of these solvers. Since there is substantial scalar (uniprocessor) work in these algorithms for estimating parameters, the initial results indicate the need for the development of new algorithms with parallelization in mind from the beginning.

The latest and by-far the most powerful ITPACK package is called `NSPCG` since it contains many NonSymmetric Preconditioned Conjugate Gradient procedures. This package allows for the use of a wide variety of iterative algorithms suitable for use on vector supercomputers and for both symmetric and nonsymmetric systems.

Both `ITPACKV 2D` and `NSPCG` are now part of the Cray Applications Library of Software. Moreover, the `ITPACKV 2C` package was incorporated into the `ELLPACK` package for solving elliptic partial differential equations.

Present

Our current work calls for research, development, and implementation of iterative basic linear algebra subprograms (Iterative BLAS) for use with iterative algorithms for solving large sparse linear systems. Special emphasis will be placed on coding and testing of scalar and parallel versions of iterative algorithms using these subprograms for solving systems with either symmetric or nonsymmetric coefficient matrices on multiprocessor computers.

A primary goal of the research is the development of software and the testing of it on large systems arising from applications. Techniques for optimizing software on parallel computers will be investigated. The goal is to find, develop, and test software for computational kernels that are common to a large class of iterative methods. The main objective is to develop fundamental operations for various iterative algorithms using these kernels so that they parallelize regardless of the particular sparse matrix storage scheme being used or the particular parallel computer architecture being used—without jeopardizing the convergence properties of the underlying procedures.

This research will be part of an international effort to establish standard computational kernels to be used in developing efficient and portable implementations of iterative algorithms for sparse matrix problems on high performance computers. We plan to consult and work with others involved in similar projects. Some of the problems that have to be overcome is keeping the interface for the suite of codes simple yet functional and flexible for many different data structures and for various iterative algorithms. This is particularly difficult to achieve on different parallel architectures.

Future

The objective of our future research is to advance iterative method software technology, specifically in regard to ITPACK, in the following four areas:

- Parallel Iterative Software Using Kernels
- Visual Programming
- Research on Iterative Algorithms
- Object-oriented Design

The current programming environment for the application of iterative methods will be changed in several ways. First, computational kernels will be implemented for use in parallel environments. Second, visual programming will be introduced to provide a user friendly interface. Third, basic research will continue on iterative methods for nonsymmetric systems and for parallel computer architectures. Finally, a ITPACK library will be designed to support object-oriented design concepts.

Visual Programming

Visual programming would be a natural interface for ITPACK on a workstation or workstation-supercomputer network. Major components of the package could be developed as modules containing an integrated set of computational kernels. The user could select individual modules from a menu and connect

them together by drawing lines on a visual display. The software would then construct a program containing these modules. Some of the modules might be for the basic method, accelerator, preconditioner, adaptive procedure, stopping test, and graphical output. It would be relatively easy to experiment with new methods and combinations of old methods. The programmer/analyst could tailor the code to the particular application being investigated. The addition of new modules would be a simple matter of adding a new entry in the menu. This interface will allow ITPACK users to program in ITPACK by "wiring" icons together and defining parameters through context sensitive menus. Development of the visual programming interface begins with designing the visual language from the entity-relationship (ER) model of the iterative method domain to insure compatibility. (Entity-relationship modeling provides a mechanism to concisely represent knowledge about a domain.) The interface is realized through the development of an interactive editor and a translator.

Parallel Iterative Software Using Kernels

This part of our future work is for research, development, and implementation of iterative basic linear algebra subprograms (Iterative BLAS) for use with iterative algorithms for solving large sparse linear systems. Special emphasis will be placed on coding and testing of parallel versions of iterative algorithms using these subprograms for solving systems with either symmetric or nonsymmetric coefficient matrices on multiprocessors. The goal of this project is research in the theory of parallel iterative methods and the development of parallel ITPACK-type software. Techniques for optimizing software on parallel computers will be investigated. The goal is to find, develop, and test software for computational kernels that are common to a large class of iterative methods. The main objective is to develop fundamental operations for various iterative algorithms using these kernels so that they parallelize regardless of the particular sparse matrix storage scheme being used for the matrix problems or the particular parallel computer architecture being used.

Research on Iterative Algorithms

One of the goals of this project is to develop algorithms and programs that make maximum use of the potential of parallel computers for solving large sparse systems of linear algebraic equations by iterative methods. To achieve this goal one must consider both "fine grain" and "coarse grain" parallelization. Fine grain parallelism at the basic linear algebra level is needed to insure that each iteration is carried out efficiently and this involves the sparse storage structure and the manipulation of vectors and sparse matrices. In addition to the fine grain parallelism it is necessary to use algorithms that are rapidly convergent and that also have "high level" or "coarse grain" parallelism. We have been carrying on research on iterative algorithms for many years with the primary

focus on methods for sequential and, more recently, for vector computers. We now plan to continue our research on iterative algorithms, with the focus more on algorithms for parallel computers. We plan to emphasize the following areas: the iterative solution of nonsymmetric linear systems; rational iterative methods; and adaptive iterative methods.

Object-oriented Design

The object-oriented kernels will be used to develop a kernel-based interpretive program. This "number crunching" program will interpret a text based macro language to declare kernel objects and perform operations on those objects. The program will provide a MATLAB-type of capability but with object-oriented kernels for both traditional linear algebra and iterative algorithms. The macro language will be constructed using the ER model of the iterative method domain. This will insure consistency between the macro language and the object-oriented kernels. Iterative algorithms from current versions of ITPACK will be implemented as modules in the macro language of the interpretive number cruncher program. The use of an interpretive program allows iterative algorithms to be developed, tested, and applied to solve problems without having to compile and link a new program. This approach will significantly reduce the effort in implementing new algorithms by eliminating the cumbersome code-compile-link-debug loop.

Summary

Many years of research have gone into the development of the **ITPACKV** and **NSPCG** packages for solving large sparse symmetric and nonsymmetric systems of linear algebraic equations by various iterative methods. The proposed research would build on this experience in using iterative methods and would expand both the theory and applications into the new areas of scientific computing.

Wednesday, April 6

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

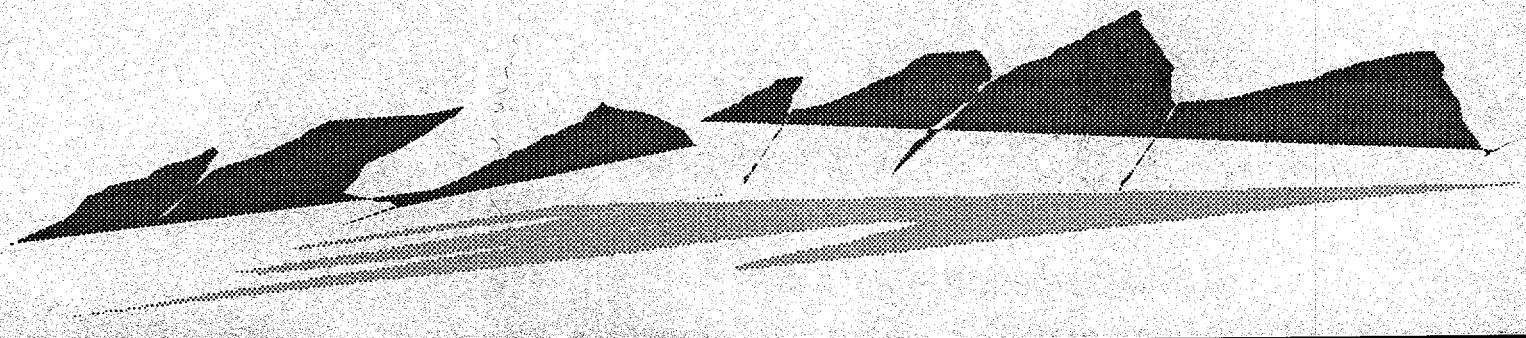
David M. Young
Origins of the ITPACK Project

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



Wednesday Evening's Workshop

Recent Progress and Advances in Iterative Software (including Parallel Aspects)

Organizer: Graham Carey (with D. M. Young, D. Kincaid and W. Joubert)

Abstract

The purpose of the workshop is to provide a forum for discussion of the current state of iterative software packages. Of particular interest is software for large scale engineering and scientific applications, especially for distributed parallel systems. However, we will also review the state of software development for conventional architectures. This workshop will complement the other proposed workshops on iterative BLAS kernels and applications.

The format for the workshop is as follows: To provide some structure, there will be brief presentations, each of less than five minutes duration and dealing with specific facets of the subject. These will be designed to focus the discussion and to stimulate an exchange with the participants.

Issues to be covered include: The evolution of iterative packages, current state of the art, the parallel computing challenge, applications viewpoint, standards, and future directions and open problems.

Speakers

- David M. Young, University of Texas at Austin
  "Origins of the ITPACK Project"
- David Kincaid, University of Texas at Austin
  "Recent Developments on ITPACK"
- Graham Carey, University of Texas at Austin
  "Design Considerations for a Portable Parallel Package"
- Wayne Joubert, Los Alamos National Laboratory
  "Adapting Iterative Software Libraries to Parallel Environments"
- Rossen Parashkevov, University of Colorado at Denver
  "Operator-based Iterative Tools"

Audience participation is strongly encouraged!