0 0 0 0 3 1 0 7 5 3 8

LBL-8158 c.1
UC-32
CONF-780878

NRCC NATIONAL RESOURCE FOR COMPUTATION IN CHEMISTRY

NUMERICAL ALGORITHMS IN CHEMISTRY: ALGEBRAIC METHODS

RECEIVED
LAWRENCE
BERKELEY LABORATORY

APR 24 1979

LIBRARY AND
DOCUMENTS SECTION

Report
on the Workshop
August 9-11, 1978

For Reference

Not to be taken from this room

LBL-8158 c.1

LAWRENCE BERKELEY LABORATORY
UNIVERSITY OF CALIFORNIA, BERKELEY

Prepared for the U.S. Department of Energy under Contract W-7405-ENG-48

LEGAL NOTICE

This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Department of Energy, nor any of their employees, nor any of their contractors, subcontractors, or 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.

Printed in the United States of America
Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road
Springfield, VA 22161
Price Code: A08

0 0 0 0 5 1 0 7 6 3 9

REPORT

ON THE WORKSHOP

NUMERICAL ALGORITHMS IN CHEMISTRY: ALGEBRAIC METHODS

Sponsored by the

NATIONAL RESOURCE FOR COMPUTATION IN CHEMISTRY

Lawrence Berkeley Laboratory
Berkeley, California 94720

August 9-11, 1978

Edited by: Cleve Moler and I. Shavitt

CONTENTS

Workshop Participants 1

Foreword
William A. Lester, Jr. 5

Part I. Eigenvalue Problems

Matrix Eigenvalue Problem
I. Shavitt 7

Origin and Structure of the H-Matrix
E. R. Davidson 11

Expansion Methods for Eigenvectors of Large Matrices
E. R. Davidson 15

The Use of Variation-Perturbation Techniques in CI-Calculations
B. Roos 26

History of Relaxation Algorithms and Theoretical Chemistry
R. C. Raffenetti 29

The Simultaneous Expansion Method
B. Liu 49

Power Methods and Lanczos Methods
C. Van Loan 54

The Generalized Eigenvalue Problem in Quantum Chemistry
N. Beebe and C. Moler 63

Feler's Method for Finding Eigenvalues and Eigenvectors
N. Beebe 66

Bibliography on the Large Matrix Eigenvalue Problem in
Quantum Chemistry and in Related Fields 68

Reflections on the NRCC Conference
B. Parlett 73

Recommendations for Work by NRCC 74

Part II. Linear Systems of Equations

Summary of Discussion on the Solution of Large Linear Systems
and Recommendations for Work by the NRCC
I. S. Duff 75

Linear Equation Systems in Bound State and Scattering Problems
R. K. Nesbet 79

Direct Methods for Solution of $A\underset{\sim}{x} = \underset{\sim}{b}$
I. S. Duff 89

Direct Methods for Solving Sparse Systems of Equations
S. Eisenstat 100

Remarks on Iterative Methods for the Solution of Large Systems of Linear Algebraic Equations
O. Widlund 112

Modified Gram-Schmidt
C. Moler 124

The SYMMLQ Algorithm
A. K. Cline 126

Part III. Integral Transformations

Four Index Integral Transformation
S. Elbert 129

Outline of Yoshimine's Two-Pass Sorting Scheme 142

Storage Utilization and Sorting
M. Yoshimine 144

A Different Approach to Integrals and Integral Transformations
N. Beebe and J. Linderberg 158

PARTICIPANTS

of the

WORKSHOP ON NUMERICAL ALGORITHMS IN CHEMISTRY: ALGEBRAIC METHODS

University of California at Santa Cruz

August 9-11, 1978

Dr. Nelson Beebe
Quantum Theory Project
University of Florida
Gainesville, FL

Dr. Charles S. Bender
Lawrence Livermore Laboratory
P.O. Box 808, L-259
Livermore, CA 94550

Dr. David Ceperly
Lawrence Berkeley Laboratory
National Resource for
Computation in Chemistry
University of California, Berkeley
Berkeley, CA 94720

Prof. Alan K. Cline
Computer Science Dept.
University of Texas
Austin, TX

Prof. Ernest R. Davidson
Dept. of Chemistry
University of Washington
Seattle, WA 98195

Dr. G. H. F. Diercksen
Max-Planck Institut für Physik
und Astrophysik
Föhringer Ring 6
8000 München, W.Germany

Dr. David Dion
Lawrence Berkeley Laboratory
National Resource for Computation
in Chemistry
University of California, Berkeley
Berkeley, CA 94720

Dr. Ian Duff
Computer Science & Systems Division
A.E.R.E. Harwell
Didcot, Oxford OX11 ORA England

Dr. Michel Dupuis
IBM Research Laboratory K34/281
5600 Cottle Road
San Jose, CA 95193

Prof. Stanley Eisenstat
Computer Science Dept.
Yale University
New Haven, CT

Dr. Stephen Elbert
Dept. of Chemistry
Iowa State University
Ames, IA

Prof. Gene H. Golub
Computer Science Dept.
Serra House
Stanford University
Stanford, CA 94305

Dr. Stanley Hagstrom
Lawrence Berkeley Laboratory
National Resource for
Computation in Chemistry
University of California, Berkeley
Berkeley, CA 94720

Dr. Jurgen Hinze
Universität Bielefeld
Fakultät für Chemi
Postfach 8640
4800 Bielefeld 1
Universitätsstrasse, W.Germany

Dr. Joyce K. Kaufman
Dept. of Chemistry
Johns Hopkins University
Remsen Hall - Dunning Hall
Baltimore, MD 21218

Dr. Harry F. King
State University of New York
at Buffalo
Acheson Hall
Buffalo, NY 14214

Dr. Stephen R. Langhoff
Ames Research Center, NASA
Moffett Field, CA 94035

Dr. William A. Lester, Jr.
Lawrence Berkeley Laboratory
National Resource for
Computation in Chemistry
University of California, Berkeley
Berkeley, CA 94720

Mr. Randy Leveque
Computer Science Dept.
Serra House
Stanford University
Stanford, CA 94305

Dr. Bowen Liu
IBM Research Laboratory
5600 Cottle Road
San Jose, CA 95193

Prof. Cleve Moler
Dept. of Mathematics
University of New Mexico
Albuquerque, NM 87131

Dr. Joseph R. Murdoch
Dept. of Chemistry
University of California, Los Angeles
Los Angeles, CA 90024

Mr. Steve Nash
Computer Science Dept.
Serra House
Stanford University
Stanford, CA 94305

Dr. Robert K. Nesbet
IBM Research Laboratory
5600 Cottle Road
San Jose, CA 95193

Prof. Beresford Parlett
Computer Science Dept.
University of California, Berkeley
Berkeley, CA 94720

Dr. Richard C. Raffenetti
Chemistry Division
Argonne National Laboratory
Argonne, IL 60439

Dr. Bjorn Roos
Institute of Theoretical Chemistry
University of Stockholm
Stockholm, Sweden

Dr. Isaiah Shavitt
Battelle Columbus Laboratories
505 King Avenue
Columbus, OH 43201

Dr. Ron Shepard
University of Utah
Chemistry Building, Box 77
Salt Lake City, UT 84112

Dr. Dale Spangler
Lawrence Berkeley Laboratory
National Resource for
Computation in Chemistry
University of California, Berkeley
Berkeley, CA 94720

Prof. Richard Underwood
CIS Dept.
Ohio State University
Columbus, OH

Prof. Charles Van Loan
Computer Science Dept.
Cornell University
Ithaca, NY

Prof. Olof Widlund
Courant Institute of Mathematical Science
251 Mercer Street
New York, NY 10012

Prof. Edward Wilson
Dept. of Civil Engineering
781 Davis Hall
University of California, Berkeley
Berkeley, CA 94720

Dr. Megumu Yoshimine
IBM Research Laboratory
5600 Cottle Road
San Jose, CA 95193

FOREWORD

The National Resource for Computation in Chemistry (NRCC) was established as a division of Lawrence Berkeley Laboratory (LBL) in October, 1977. The functions of the NRCC may be broadly categorized as follows: (1) to make information on existing and developing computational methodologies available to all segments of the chemistry community, (2) to make state-of-the-art computation facilities [both hardware and software] accessible to the chemistry community, and (3) to foster research and development of new computational methods for application to chemical problems.

Workshops are one facet of the NRCC's program for both obtaining and making available information on new developments in computationally-oriented sub-disciplines of chemistry. Numerical algorithms underlie all aspects of the NRCC's program; a focus on algebraic methods in the first year of workshop activity is appropriate because of its continuing importance in chemical applications. It was planned that the workshop include not only chemists who have pioneered algebraic methods, but also numerical analysts and computer scientists expert in the subject area. This workshop was therefore organized by a numerical analyst along with a computational chemist. We are indebted to Dr. Isaiah Shavitt, Battelle Columbus Laboratories, and Professor Cleve Moler, Department of Mathematics, University of New Mexico, for organizing the workshop and for their considerable efforts in editing this volume.

In order to promote maximum interaction among the attendees of disparate disciplines, the workshop was held at the University of California, Santa Cruz, in the commodious facilities of Oakes College. The workshop focused on three topics: eigenvalue problems, linear systems of equations, and the four-index transformation of quantum chemistry. At the conclusion of the presentations by invited speakers in each area, participants from the three areas met separately to develop recommendations on the role the NRCC could uniquely serve to advance the field. These recommendations are listed on pages 74 and 77.

At the time of preparation of this volume, the NRCC has begun to implement the first recommendation of acquiring the most important programs available in the subject areas. This effort is being coordinated by Professor Moler, in his recently assumed capacity as a consultant to the NRCC, in collaboration with the Applied Mathematics Department of Lawrence Berkeley Laboratory.

The present volume attempts to present a timely and succinct digest of the contribution of each speaker. Extended annotated bibliographies serve as a guide to the open literature of various areas.

A companion workshop ("Post Hartree-Fock: Configuration Interaction") held at Lawrence Berkeley Laboratory on August 14-16, 1978, was concerned with the numerical methods for the study of the electron correlation problem in molecules. The proceedings of this workshop are available upon request from the NRCC.

The NRCC is jointly funded by the Department of Energy and the National Science Foundation.

— William A. Lester, Jr.
Director, NRCC

MATRIX EIGENVALUE PROBLEM: THE NATURE OF THE PROBLEM

I. Shavitt

Battelle Columbus Laboratories
Columbus, Ohio 43201

MATRIX NOTATION AND TERMS AS USED IN THEORETICAL CHEMISTRY

$\underset{\sim}{A}$ = matrix with elements A_{ij} (or a_{ij})

$\tilde{\underset{\sim}{A}}$ = transpose of $\underset{\sim}{A}$

$\underset{\sim}{A}^*$ = complex conjugate of $\underset{\sim}{A}$

$\underset{\sim}{A}^\dagger = \tilde{\underset{\sim}{A}}^*$ = (Hermitian) adjoint of $\underset{\sim}{A}$

$\underset{\sim}{A}^\dagger = \underset{\sim}{A}$ ==> Hermitian matrix

$\underset{\sim}{A}^\dagger = \underset{\sim}{A}^{-1}$ ==> unitary matrix

$\underset{\sim}{x}$ = column vector with elements x_i

$\underset{\sim}{x}^\dagger$ = row vector with elements x_i^*

In most problems all quantities are real,

$$\underset{\sim}{A}^* = \underset{\sim}{A}\,, \quad \underset{\sim}{x}^* = \underset{\sim}{x}$$

THE MATRIX EIGENVALUE PROBLEM IN QUANTUM CHEMISTRY

The general form is

$$\underset{\sim}{A}\,\underset{\sim}{x}_i = \lambda_i \underset{\sim}{B}\,\underset{\sim}{x}_i\,,$$

where $\underset{\sim}{A}$, $\underset{\sim}{B}$ are Hermitian (and usually real) and $\underset{\sim}{B}$ is positive definite.

In most applications $\underset{\sim}{B} = \underset{\sim}{1}$ or $\underset{\sim}{B}$ is block diagonal (in small blocks), so that a transformation to an orthogonal representation is easily carried out. Thus in most cases we are only interested in

$$\underset{\sim}{A}\,\underset{\sim}{x}_i = \lambda_i \underset{\sim}{x}_i$$

The roots (eigenvalues) are numbered so that

$$\lambda_1 \le \lambda_2 \le \ldots \le \lambda_n \ .$$

CHARACTERISTICS OF THE PROBLEM

- The order n is very large, typically $10^3 - 10^5$.
- $\underset{\sim}{A}$ is <u>randomly</u> sparse, with the fraction of nonzero elements 5 - 25 % in most cases.
- $\underset{\sim}{A}$ is diagonally dominant in most applications.
- Only the lowest root λ_1, or just a few lowest roots λ_i $(i = 1,2,\ldots,k)$, and the corresponding vectors, are required ($k \le 10$ in most cases, but in some applications k may be as large as 100).
- Reasonable initial approximations are easily generated.
- Only nonzero elements A_{ij} $(i \ge j)$ are usually stored (or computed).
- In many cases the nonzero elements are ordered by rows of the lower triangle,

  A_{ij} precedes $A_{i'j'}$ if $i < i'$
  or if $i = i'$, $j < j'$.

  It would be very inconvenient if we would have to store <u>both</u> A_{ij} and A_{ji}, ordered by rows of the square matrix.
- In some applications the nonzero elements of $\underset{\sim}{A}$ are available in essentially random order, and it would be impractical to have them reordered.

PREFERRED APPROACHES

- Iterative, starting with generated initial approximations ("trial vectors").
- The original matrix should not have to be modified (to preserve sparsity, avoid re-writing the matrix, and reduce error accumulation).
- Only a small section (e.g., row) of the matrix should be required in central memory at one time.

CLASSIFICATION OF METHODS

- Gradient methods -

  Search for minimum of the Rayleigh quotient $(\underset{\sim}{v}^{\dagger}\underset{\sim}{A}\underset{\sim}{v})/(\underset{\sim}{v}^{\dagger}\underset{\sim}{v})$ by steepest descent (Hestenes, Karush) or by conjugate gradient methods. Relatively slow.

- Relaxation methods -

  Modify one element of the trial vector(s) at a time, either to reduce the residuals (Cooper, Nesbet) or to minimize the Rayleigh quotient. Can use over-relaxation (Ruhe, Schwarz).

- Expansion methods -

  Expand the required vectors in a gradually increasing set of generated vectors. This set can be a sequentially orthogonalized Krylov sequence $\underset{\sim}{A}^{i}\underset{\sim}{v}$ $(i = 0,1,2,\ldots)$ (Lanczos), or can be generated by a relaxation method (Davidson) or by perturbation theory (Roos, Siegbahn, Pople).

MULTIROOT TREATMENT

- Sequential -

  Implicit modification of $\underset{\sim}{A}$ to shift previously found roots above next root sought (Shavitt et al., has difficulties for closely spaced roots), or including previously found eigenvectors in the expansion set (Davidson).

- Simultaneous relaxation -

  Concurrent iterations for k roots, interspersed with diagonalization of $k \times k$ "interaction" matrix (Raffenetti, Shavitt)

- Global -

  Lanczos method; extreme roots converge first, but explicit orthogonalization is required because of round-off error accumulation.

ORIGIN AND STRUCTURE OF THE H-MATRIX

E. R. Davidson

University of Washington
Seattle, Washington

Let $f_i(\vec{r})$ be a finite basis (L^2) in Cartesian 3-space, and $\{\alpha,\beta\}$ be the basis in spin space. Let

$$\phi_j = \sum_{i=1}^{N} C_{ij} f_i \qquad \underline{\text{DIM}} \ N$$

Solve model "independent particle" (Roothaan-Hartree-Fock) problem

$$\Phi = \begin{vmatrix} \phi_1(\vec{r}_1)\alpha(\xi_1) & \phi_1(\vec{r}_2)\alpha(\xi_2) & \cdots & \phi_1(\vec{r}_n)\alpha(\xi_n) \\ \phi_1(\vec{r}_1)\beta(\xi_1) & & & \vdots \\ \vdots & & & \vdots \\ \phi_{n/2}(\vec{r}_1)\beta(\xi_1) & \cdots & \cdots & \phi_{n/2}(\vec{r}_n)\beta(\xi_n) \end{vmatrix}$$

("Slater determinant")

$$\delta \frac{\langle \Phi \, H \, \Phi \rangle}{\langle \Phi \Phi \rangle} = 0$$

$$F\phi_j = \varepsilon_j \phi_j$$

where

$$F = -\tfrac{1}{2} \nabla^2 - \sum_A \frac{Z_A}{r_A} + v_{eff} = h + v_{eff}$$

<u>Matrix generation</u>

$$H = \sum_i \underbrace{\left\{ -\tfrac{1}{2} \nabla_i^2 - \sum_A^K \frac{Z_A}{r_{Ai}} \right\}}_{h(i)} + \sum_{i<j}^{n} \frac{1}{r_{ij}}$$

$$H\psi = E\psi$$

$$\Psi = \Psi(r_1, \xi_1, r_2, \xi_2, \ldots)$$

r_1 = position 3-dimension

ξ_1 = $\pm\frac{1}{2}$, discrete spin variable

$$\sum_{\xi}^{(n)} \int_{(n)} d\vec{r}\, |\Psi|^2 = 1$$

$$P_{ij}\,\Psi = -\Psi \qquad (P_{ij} \text{ is spin-dependent boundary condition})$$

$$(f_i, F\, f_j) = -\tfrac{1}{2}(f_i, \nabla^2 f_j) - \sum_A Z_A (f_i, 1/r_A\, f_j) + \sum P_{k\ell}\left\{2[f_i f_j \| f_k f_\ell] - [f_i f_k \| f_j f_\ell]\right\}$$

where

$$P_k = \sum_j C_{kj}\, C_{\ell j}$$

and

$$[f_i f_j \| f_k f_\ell] = \langle f_i(r_1) f_k(r_2) | 1/r_{12} | f_j(r_1) f_\ell(r_2) \rangle \quad .$$

Solutions to model problem *almost* solve the true problem. Expand solution Ψ as combination of Slater determinants (or symmetry-adapted Slater determinant)

$$\Phi_I = A\left\{\phi_{i_1}\, \chi_{i_1},\ \Phi_{i_2}\, \chi_{i_2},\ \ldots\right\}$$

or

$$\Phi'_J = \sum g_{IJ}\, \Phi_I \qquad (\text{symmetry-determined } g_{IJ})$$

<u>Rules</u> (Slater-Condon):

$$(\Phi_I, \Phi_J) = 0 \qquad I \neq J$$

because all $(\phi_i, \phi_j) = \delta_{ij}$.

$$(\Phi_I, H\Phi_J) = 0 \quad \dim\ \{I\} \cap \{J\} < (n-2)$$

$$= a\left[\phi_{i_1}\phi_{j_1} \,\|\, \phi_{i_2}\phi_{j_2}\right] + b\left[\phi_{i_1}\phi_{j_2} \,\|\, \phi_{i_2}\phi_{j_1}\right]$$

if

$$\{I\} \cap \sim\big(\{I\} \cap \{J\}\big) = \{i_1, i_2\}$$

$$\{J\} \cap \sim\big(\{I\} \cap \{J\}\big) = \{j_1, j_2\}$$

$$= h_{ij} + \sum_{k \in \{I\} \cap \{J\}}^{n-1} \left(a[\phi_i \phi_j \| \phi_k \phi_k] - b[\phi_i \phi_k \| \phi_j \phi_k] \right)$$

if

$$\{I\} \cap \sim\left(\{I\} \cap \{J\}\right) = \{i\}$$

$$\{J\} \cap \sim\left(\{I\} \cap \{J\}\right) = \{j\}$$

$$= \sum n_i h_{ii} + \left(\sum_{k,\ell \in \{I\}} a[\phi_k \phi_k \| \phi_\ell \phi_\ell] - b[\phi_k \phi_\ell \| \phi_k \phi_\ell] \right)$$

$$\{I\} = \{J\}$$

Choice of Φ_K for inclusion in

$$\Psi_P \cong \sum_K C_{KP} \Phi_K$$

Pick *zero'th order space* from model (RHF) problem

$$\Psi_1 \sim \Phi_1, \quad \Psi_2 \sim \Phi_2, \quad \text{etc.}$$

add other Φ with large ($\geqslant 0.1$)

$$\left| \frac{(\Phi, H \Phi_J)}{H_{\phi\phi} - H_{JJ}} \right|, \quad J \leqslant P_{max}$$

Typical size for zero'th order space $\sim(1\text{-}5) \times P_{max} = \mathcal{P}$.

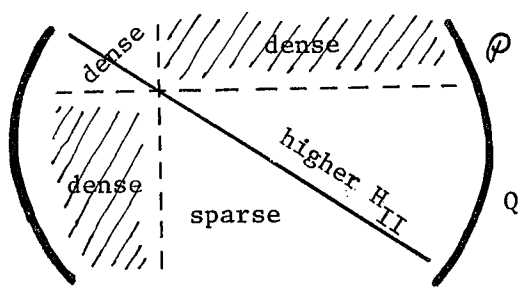
Pick *first order space* such that

$$\left[\left| \frac{(\Phi, H \Phi_J)}{H_{\phi\phi} - H_{JJ}} \right|, J \leqslant P_{max} \right] \gtrsim 10^{-3}$$

Note Φ must be a single or double substitution from $\{\Phi_J\}$ or else $(\Phi, H \Phi_J) = 0$.

$$H_{IJ} = (\Phi_I, H\Phi_J)$$

H =



Example, $\mathcal{P} = 1$

$$Q \equiv \left\{ ij \text{ from } \{I^o\} \longrightarrow pq \text{ from } \sim\{I^o\} \right\}$$

$$(n^2) \longrightarrow (N-n)^2 \text{ choices}$$

$$\dim Q \sim n^2(N-n)^2$$

(usual $N \gtrsim 3n$) $([ij \to pq], H[k\ell \to rs]) = 0$ unless $\dim\{ij\ pq\ k\ell\ rs\} \leqslant 4$ for $N \gg n$, $\sim N^2$ non-zero per row H_{QQ}, or $\sim 1/n^2$ = fraction non-zero, $n^2N^4 \sim$ total non-zero.

$\boxed{N^6 \text{ method}}$

EXPANSION METHODS FOR EIGENVECTORS OF LARGE MATRICES

E. R. Davidson

University of Washington
Seattle, Washington

$\underline{H}\underline{C}_p = E_p \underline{C}_p$

$N \times N$ Matrix $\underline{H}$

$N \gg 10^2$, $p \lesssim 10$

n^{th} Approximate

$$\underline{C}_p \doteq \underline{C}_p^{(n)} \equiv \sum_{i=1}^{K} \alpha_{ip} \underline{b}_i^{(n)} , \qquad p < K << N$$

$\{\underline{b}_i^{(n)}\}$ Set of Basic Vectors

$$h_{ij}^{(n)} = \left(\underline{b}_i^{(n)}, \; H \underline{b}_j^{(n)}\right)$$

$$s_{ij}^{(n)} = \left(\underline{b}_i^{(n)}, \; \underline{b}_j^{(n)}\right)$$

$$\underline{h}\, \underline{\alpha}_p = \lambda_p \underline{s}\, \underline{\alpha}_p$$

$$E_{N-p} \geqslant \lambda_{K-p} , \qquad \lambda_p \geqslant E_p$$

Cyclic Methods

$\{\underline{b}_i^{(n)}\}$ $n = 1,2, \ldots$ cycles through a complete set of vectors repeatedly

$$\boxed{p = 1, \qquad K = 2}$$

Nesbet; J. Chem. Phys. 43, 311 (1965)

Cooper; Quart. Appl. Math. 6, 179 (1948)

Shavitt; J. Comput. Phys. 6, 124 (1970)

$$\underline{b}_1^{(n)} = \underline{C}^{(n-1)} \qquad \left[\begin{array}{l} \underline{C}^{o} = \hat{e}_J \\ H_{JJ} = \inf H_{ii} \end{array} \right]$$

$$\underline{b}_2^{(n)} = \hat{e}_r$$

$$E^{(n)} = \left(\underline{C}^{(n)} \, , \quad \underline{H} \, \underline{C}^{(n)} \right)$$

$$h = \begin{pmatrix} E^{(n-1)} \, \| \underline{C}^{(n-1)} \|^2 & [\underline{H} \, \underline{C}^{(n-1)}]_r \\ [\underline{H} \, \underline{C}^{(n-1)}]_r & H_{rr} \end{pmatrix}$$

$$s = \begin{pmatrix} \| \underline{C}^{(n-1)} \|^2 & C_r^{(n-1)} \\ C_r^{(n-1)} & 1 \end{pmatrix}$$

$$\frac{\alpha_2}{\alpha_1} = \frac{[(\underline{H} - \lambda \mathbb{1}) \underline{C}^{(n-1)}]_r}{\lambda - H_{rr}} \doteq \frac{[(\underline{H} - E^{(n-1)} \, \mathbb{1}) \, \underline{C}^{(n-1)}]_r}{E^{(n-1)} - H_{rr}}$$

$$E^{(n)} \geqslant E^{(n+1)} \geqslant E$$

Fadeev and Fadeeva; "Computational Methods of Linear Algebra"

Shavitt, Bender, Pipano, Hosteny; J. Comput. Phys. 11, 90 (1973)

Solve 2×2 problem exactly,

$$(E^{(n-1)} - \lambda)(H_{rr} - \lambda) \, \| \underline{C}^{(n-1)} \|^2 = [(\underline{H} - \lambda \mathbb{1}) \, \underline{C}^{(n-\mathbb{1})}]_r^2$$

$$\boxed{p \neq 1 \qquad K = 2}$$

Shavitt, Bender, Pipano, Hosteny; J. Comput. Phys. 11, 90 (1973)

Root shift,

$$\tilde{\underline{H}} = \underline{H} + \sum_{j=1}^{p-1} \mu_j \, \underline{C}_j \, \underline{C}_j^T$$

so λ_p is the lowest root of $\tilde{\underline{H}}$.

Requires $\underline{C}_1 \cdots \underline{C}_{p-1}$ before $\underline{C}_p$. Error accumulates in higher roots. Rapidly convergent for diagonal dominant matrix -- slow otherwise. Trouble with near degeneracy.

$$\boxed{p \neq 1 \quad , \qquad K = p+1}$$

Raffenetti; J. Comput. Phys. (in press)

$$\underline{b}_1^{(n)} \cdots \underline{b}_p^{(n)} = \underline{C}_1^{(n-1)} \cdots \underline{C}_p^{(n-1)}$$

$$\underline{b}_{p+1}^{(n)} = \hat{e}_r$$

$$\boxed{p = 1 \quad , \qquad K > 2}$$

Chung and Bishop; preprint

Falk; Z. Agnew. Math. Mech. 53, 73 (1973)

Fadeev and Fadeeva; --

Davidson and Bender; unpublished, used in all diatomic calculations (1968)

Bender; Ph.D. Thesis, University of Washington (1968)

$$\underline{b}_1^{(n)} = \underline{C}_1^{(n-1)}$$

$$\underline{b}_2^{(n)} \cdots \underline{b}_K^{(n)} = \hat{e}_{r_1}, \hat{e}_{r_2}, \cdots$$

Variation — Perturbation Methods

$$\boxed{p = 1 \quad , \qquad K = 2}$$

$$\underline{b}_1^{(n)} = \underline{C}^{(n-1)}$$

$$\underline{H} = \underline{H}_o + \underline{V}$$

$$\underline{b}_2^{(n)} = \left(E^{(n-1)} \mathbb{1} - \underline{H}_o\right)^{-1} \underline{V}\, \underline{b}_1^{(n)}$$

or equivalently

$$\tilde{\underline{b}}_2^{(n)} = \underline{b}_2 - \underline{b}_1 = \left(E^{(n-1)} \mathbb{1} - \underline{H}_o\right)^{-1} \left(\underline{H} - E^{(n-1)} \mathbb{1}\right) \underline{b}_1^{(n)}$$

Dalgarno and Stewart; Proc. Roy. Soc. (London) 77, 467 (1961).

(b_1, b_2) or $(b_1, \tilde{b}_2)$ span same space; $\tilde{b}_2$ is same as Nesbet formula, if

all elements are varied simultaneously rather than sequentially, and if

$$\underline{H}_o = \begin{pmatrix} H_{11} & & 0 \\ & H_{22} & \\ & & \ddots \\ 0 & & & \ddots \end{pmatrix}$$

Note non-variational simultaneous Nesbet (i.e., iterative first order perturbation theory) often diverges. Variational must converge.

$$\boxed{p = 1 , \qquad K > 2}$$

Roos and Siegbahn; in "Methods of Electronic Structure Calculations," (H.F.Schaefer, editor, Plenum Press, 1978)

Rayleigh – Schrodinger Pert Series

Lowdin; J. Math. Phys. 6, 1341 (1965)

DuPont-Boudelet, Tillieu, Guy; J. Phys. Radium 21, 776 (1960)

Hirschfelder and Epstein; Adv. Quan. Chem., Vol. 1

$$\underline{H}_o \underline{b}_o = E_o \underline{b}_o \quad , \qquad \text{i.e.,} \quad \underline{b}_o = \underline{C}^{(0)}$$

$$\underline{b}_K - \underline{b}_{K-1} = (E_0 \mathbb{1} - \underline{H}_0)^{-1} \left\{ (\underline{H} - E_o \mathbb{1}) \underline{b}_{K-1} - \sum_{n=0}^{K-1} E_{K-n} \underline{b}^n \right\}$$

$$E_{2K-1} = \left(\underline{b}^{K-1}, (\underline{H} - \underline{H}_o) \underline{b}^{K-1} \right) - \sum_{m=1}^{K-1} \sum_{n=1}^{K-1} E_{2K-1-m-n} (\underline{b}^m, \underline{b}^n)$$

$$E_{2K} = \left(\underline{b}^K, (\underline{H} - \underline{H}_o) \underline{b}^{K-1} \right) - \sum_{m=1}^{K} \sum_{n=1}^{K-1} E_{2K-m-n} (\underline{b}^m, \underline{b}^n)$$

Usually

$$\underline{H}_o = \begin{pmatrix} H_{11} & & 0 \\ & H_{22} & \\ & & \ddots \\ 0 & & & \ddots \end{pmatrix}$$

$$E = \sum_{j=0}^{2K} E_j \quad , \qquad \underline{C} = \sum_{j=0}^{K} \underline{b}_j \qquad \text{is pert. result}$$

Seeger, Krishnan, Pople; J. Chem. Phys. 68, 2519 (1978)

$$\underline{b}_{K+1}^{(n+1)} = \left(E^{(n)}\,\mathbb{1} - \underline{H}_o\right)^{-1}\left(\underline{H} - E^{(n)}\right)\underline{C}^{(n)}$$

$$\underline{H}_o = \sum \underline{F}(i) + \left(\underline{C}^o, \left(\underline{H} - \Sigma \underline{F}(i)\right)\underline{C}^o\right)$$

same as Davidson method (see below) except for choice of H_o.

$$\boxed{p \neq 1\,, \qquad K \geqslant p+1}$$

Davidson; J. Comput. Phys. 17, 87 (1975)

Butscher, Kammer; J. Comput. Phys. 20, 313 (1976)

$$\underline{b}_{K+1}^{(n+1)} = \left(E_p^{(n)}\,\mathbb{1} - \underline{H}_o\right)^{-1}\left(\underline{H} - E_p^{(n)}\right)\underline{C}_p^{(n)}$$

$$\underline{b}_1^{(n+1)} \ldots \underline{b}_K^{(n+1)} \equiv \underline{b}_1^{(n)} \ldots \underline{b}_K^{(n)}$$

for $K = p+1, \ldots K_{max}$

i.e., Border $\underline{h}$ on each iteration until K gets too large or convergence is reached. Then truncate to

$$\underline{b}_1 \ldots \underline{b}_p = \underline{C}_1^{(n)} \ldots \underline{C}_p^{(n)}$$

and start building new $\underline{h}$.

Can solve directly for any p or for "root-homing" pattern search on vector.

Properties of Pert. Methods

A. Rapidly convergent for diagonal dominant matrices -- *otherwise can be slow*. Derived from Rayleigh-Schrodinger theory or Gauss-Seidel approximation to Newton-Raphson inverse iteration.

B. Most reasonable for $p \leqslant 5$, $K_{max} \leqslant 15$, $N > 10^3$.

C. Well adapted to direct CI methods since $\underline{H}\underline{C}$ can be formed without H.

D. In case B, time is dominated by forming HC, cost/root ~ # of non-zero H_{ij}.

E. No trouble with degenerate roots, or non-dominant roots.

F. Input/output demands are high.

G. Only two vectors are in high-speed memory at once.

H. May miss roots.

Davidson Variation - Perturbation

<u>Initialization</u>, given $\underline{C}_1^o \ldots \underline{C}_J^o$ $(J \geqslant P)$

$$\underline{b}_1 = \underline{C}_1^o / (\underline{C}_1^o, \underline{C}_1^o)^{\frac{1}{2}} \qquad * \underline{d}_1 = \underline{H}\, \underline{b}_1 \qquad h_{11} = (\underline{b}_1, \underline{d}_1)$$

$$\underline{f}_2 = \underline{C}_2^o - (\underline{b}_1, \underline{C}_2^o)\underline{b}_1 \qquad \underline{b}_2 = \underline{f}_2 / (\underline{f}_2, \underline{f}_2)^{\frac{1}{2}}$$

$$* \underline{d}_2 = \underline{H}\, \underline{b}_2 \qquad h_{i2} = (\underline{b}_i, \underline{d}_2) \qquad i = 1,2$$

$$\vdots$$

$$f_j = C_j^o - \sum_{\ell=1}^{j-1} (\underline{b}_\ell, \underline{C}_j^o)\underline{b}_\ell \qquad \underline{b}_j = f_j / (f_j, f_j)^{\frac{1}{2}}$$

$$* \underline{d}_j = \underline{H}\, \underline{b}_j \qquad h_{ij} = (\underline{b}_i, \underline{d}_j) \qquad i = 1 \ldots j$$

<u>or</u> in root-homing mode, given only C^o approximation to some root of unknown p.

<u>Iteration</u>

Solve $\underline{h}\, \underline{\alpha}_i = \lambda_i \underline{\alpha}_i$ $K \times K$ Matrix

Select $\underline{\alpha}_i$ of current interest

(track on vector, or on $i = p$ with tests for root switching)

Test α_{iK} for convergence of vector (or α_i for convergence of E)

Take $\bar{E} = \lambda_i$, $\underline{C}_i^{(K)} = \sum_j \alpha_{ji}\, \underline{b}_j$

$$\underline{q}_K = (H - \bar{E})\underline{C}_i^{(K)} \equiv \sum_j \alpha_{ji} [\underline{d}_j - \bar{E}\underline{b}_j]$$

$$f_{j,K+1} = (\bar{E} - H_{jj})^{-1} q_{j,K}$$

$$\underline{b}_{K+1} = \frac{\tilde{\underline{f}}_{K+1}}{(\tilde{\underline{f}}_{K+1}, \tilde{\underline{f}}_{K+1})^{\frac{1}{2}}}, \qquad \tilde{\underline{f}}_{K+1} = \prod_i^K (\mathbb{1} - \underline{b}_i \underline{b}_i^T)\underline{f}_K$$

$$* \underline{d}_{K+1} = \underline{H}\, \underline{b}_{K+1}, \qquad h_{i,K+1} = (\underline{b}_i, \underline{d}_{K+1}), \qquad i = 1 \ldots j$$

Note each iteration requires all $\underline{d}_j$ once, and each $\underline{b}_j$ twice -- large I/O demand.

*Time consuming $\underline{H} \cdot \underline{b}$ required once per iteration.

Truncation

If K is too large, or if starting on new root, truncate to K = J

$$\underline{b}_i \Leftarrow \Sigma\, \alpha_{ji}\, \underline{b}_j \qquad i = 1 \ldots J$$

$$\underline{d}_i \Leftarrow \Sigma\, \alpha_{ji}\, \underline{d}_j \qquad i = 1 \ldots J$$

Krylov Methods (Power Method)

Expand $C^{(n)}$ in sequence based on $H^j C$

$$\boxed{p = 1\,, \qquad K = 2}$$

Karush; Pacific J. Math. 1, 233 (1951)

Hestenes; "Simultaneous Linear Equations and the Determination of Eigenvalues", NBS

$$\underline{b}_1^{(n)} = \underline{C}^{(n-1)}$$

$$\underline{b}_2 = \underline{H}\, \underline{b}_1$$

or

$$\tilde{\underline{b}}_2 = \nabla \frac{(\underline{b}, \underline{H}\underline{b})}{(\underline{b}, \underline{b})}\Bigg|_{\underline{b}=\underline{b}_1} = a\left(\underline{H} - E^{(n-1)}\right) \underline{C}^{(n-1)}$$

Note $(\underline{b}_1, \underline{b}_2)$ and $(\underline{b}_1, \tilde{\underline{b}}_2)$ span the same vector space.

$$\boxed{p \geqslant 1\,, \qquad K \geqslant p}$$

Delos and Blinder; J. Chem. Phys. 47, 2784 (1977)

Expand in $\underline{b}_{j+1} = \underline{H}^j \underline{C}^o$

Extended gradient method:

Expand in

$$\underline{b}_{j+1}^j = \nabla \frac{(\underline{b}, \underline{H}\underline{b})}{(\underline{b}, \underline{b})}\Bigg|_{\underline{C}^{(j-1)}} = a\left(\underline{H} - E^{(n-1)}\right) \underline{C}^{(n-1)}$$

$$\underline{b}_\ell^j = \underline{b}_\ell^{j-1}\,, \qquad \ell \leqslant j$$

Note $(\underline{b}_1, \ldots \underline{b}_j)$ spans $(\underline{C}^o, \underline{H}\underline{C}^o, \ldots \underline{H}^{j-1}\, \underline{C}^o)$.

Lanczos; J. Res. Nat. Bur. Stand. 45, *255 (1950)*

Expand in sequentially orthogonalized $\underline{H}^j\underline{C}^o$ gives tri-diagonal $\underline{h}$, diagonal $\underline{s}$.

A. Usually slowly convergent for first root (unless $|E_1/E_2|$ is large). May then give several roots without too much additional work. Doesn't depend on diagonal dominance, but needs root ratios $|E_L/E_M|$ for $L \leqslant P$, $M > P$ to be *mostly* large.

B. Well adapted to direct $\underline{HC}$ methods.

C. Time dominated by $\underline{HC}$ for $P \ll K \ll N$. $\underline{H}$ for $\underline{C}^o = e_1$ is same as Householder tri-diagonal $\underline{H}$ but cost is much higher if $\underline{H}$ will fit into core.

D. The sequence $\underline{H}^j\underline{C}$ is nearly linearly dependent. Consequently there is a rapid dramatic loss of figures in $\underline{h}$ and $\underline{s}$. In principle, sequence should truncate with exact linear dependence at $n = N$. In practice some roots repeat and others are missed with implicit orthogonalization.

Lanczos Implicit Orthogonalization

$$\underline{b}_1 = \underline{C}^o \qquad S_1 = (\underline{b}_1, \underline{b}_1) \qquad \beta_o = 0$$

$$* \ \underline{d}_1 = \underline{H}\,\underline{b}_1 \qquad e_1 = (\underline{b}_1, \underline{d}_1) \qquad E_1 = e_1/S_1$$

$$\underline{b}_2 = \underline{d}_1 - E_1\underline{b}_1 \qquad S_2 = (\underline{b}_2, \underline{b}_2) \qquad \beta_1 = S_2/S_1$$

$$* \ \underline{d}_2 = \underline{H}\,\underline{b}_2 \qquad e_2 = (\underline{b}_2, \underline{d}_2) \qquad E_2 = e_2/S_2$$

$$\vdots$$

$$\underline{b}_K = \underline{d}_{K-1} - E_{K-1}\,\underline{b}_{K-1} - \beta_{K-2}\,\underline{b}_{K-2}$$

$$* \ \underline{d}_K = \underline{H}\,\underline{b}_K$$

$$S_K = (\underline{b}_K, \underline{b}_K)\ ; \qquad e_K = (\underline{b}_K, \underline{d}_K)\ ; \qquad \beta_{K-1} = \frac{S_K}{S_{K-1}}\ ; \qquad E_K = \frac{e_K}{S_K}$$

$$\vdots$$

$$s_{ij} = (\underline{b}_i, \underline{b}_j) = S_i\,\delta_{ij}$$

*Time consuming step, needed K times to form $K \times K$ $\underline{h}$ matrix.

$$h_{ij} = (\underline{b}_i, \underline{H}\,\underline{b}_j) = \begin{cases} e_i & i = j \\ S_{j+1} & i = j+1 \\ S_{i+1} & j = i+1 \\ 0 & |i-j| > 1 \end{cases}$$

Lanczos Explicit Orthogonalization

$$\underline{b}_1 = \underline{C}^o \qquad S_{11} = (\underline{b}_1, \underline{b}_1) \qquad \beta_o = 0$$

$$* \; \underline{d}_1 = \underline{H}\,\underline{b}_1 \qquad h_{11} = (\underline{b}_1, \underline{d}_1) \qquad E_1 = e_1/S_1$$

$$\vdots$$

$$\underline{f}_K = \underline{d}_{K-1} - E_{K-1}\,\underline{b}_{K-1} - \beta_{K-2}\,\underline{b}_{K-2}$$

$$\underline{b}_K = \underline{f}_K - \sum_{j=1}^{K-1} \underline{b}_j(\underline{b}_j, \underline{f}_K) \qquad S_{KK} = (\underline{b}_K, \underline{b}_K)$$

$$* \; \underline{d}_K = \underline{H}\,\underline{b}_K \qquad h_{jK} = (\underline{b}_j, \underline{d}_K) \qquad j = 1,\ldots K$$

$$E_K = \frac{h_{KK}}{S_{KK}}$$

$$s_{ij} = S_{ii}\,\delta_{ij}$$

$$h_{ij} = (\underline{b}_i, \underline{H}\,\underline{b}_j) \qquad \text{not tri-diagonal}$$

A. Advantageous only if E_p converges with $K \ll N$.

B. Does not repeat roots, but still may miss roots.

C. Requires all $\underline{b}_i$ each iteration.

Methods Based on Partitioning:

B_k approximation

Variation - perturbation expansion method

Z. Gershorn and I. Shavitt; Int. J. Quantum Chem. 2, 751 (1968)

L. E. Nitzsche and E. Davidson; J. Chem. Phys. 68, 3103 (1978)

G. A. Segal and R. W. Wetmore; Chem. Phys. Lett. 32, 556 (1974)

L. E. Nitzsche and E. Davidson; J. Am. Chem. Soc. (in press)

K. Freed, work in progress

McMurchie and Davidson, work in progress

*Time consuming step still needed only K times to form $K \times K$ matrix.

Partition space

P Important (small)

Q Less important (large)

Approximate: $H \cong \tilde{H}$

$$\begin{pmatrix} \underline{H}_{PP} & \underline{H}_{PQ} \\ \underline{H}_{QP} & \underline{D}_Q \end{pmatrix} \begin{pmatrix} \underline{C}_P \\ \underline{C}_Q \end{pmatrix} = \tilde{E} \begin{pmatrix} \underline{C}_P \\ \underline{C}_Q \end{pmatrix}$$

$\underline{D}_Q$ = diagonal of $\underline{H}_{QQ}$

$$\left[\underline{H}_{PP} - \underline{H}_{PQ}(\tilde{E} - \underline{D}_Q)^{-1} \underline{H}_{QP}\right] \underline{C}_P = \tilde{E}\, \underline{C}_P$$

$$\underline{C}_Q = (E - \underline{D}_Q)^{-1} \underline{H}_{QP} \underline{C}_P$$

$$\tilde{E} = E^o + \delta$$

$$(E - \underline{D}_Q)^{-1} \cong (E_o - \underline{D}_Q)^{-1} - \delta (E^o - \underline{D}_Q)^{-2}$$

$$\left[\underline{H}_{PP} - E_D + \underline{H}_{PQ}(E_o - \underline{D}_Q)^{-1}\right] \underline{C}_P = \delta \left[\mathbb{1} + \underline{H}_{PQ}(E_o - \underline{D}_Q)^{-2} \underline{H}_{QP}\right] \underline{C}_P$$

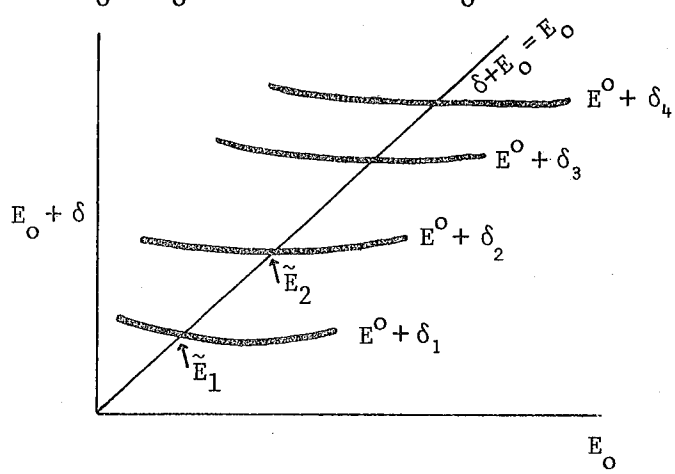
or

$$\underline{H}_{eff}\, \underline{C}_P = \delta\, \underline{S}_{eff}\, \underline{C}_P$$

$$\underline{H}_{eff} = \left(\mathbb{1}_{PP},\ \underline{H}_{PQ}(E_o - \underline{D}_Q)^{-1}\right) \underline{\tilde{H}} \begin{pmatrix} \mathbb{1}_{PP} \\ (E_o - \underline{D}_Q)^{-1} \underline{H}_{QP} \end{pmatrix}$$

$$\underline{S}_{eff} = \left(\mathbb{1}_{PP},\ \underline{H}_{PQ}(E_o - \underline{D}_Q)^{-1}\right) \begin{pmatrix} \mathbb{1}_{PP} \\ (E_o - \underline{D}_Q)^{-1} \underline{H}_{QP} \end{pmatrix}$$

so, for all E_o, $E_o + \delta \geqslant \tilde{E}$, and $E_o = \tilde{E}$ for $\delta = 0$.



Because of variation property $E_o + \delta$ varies by $\sim 10^{-8}$ when E_o is changed by 10^{-1}. For a grid in E_o over range of spectrum, compute $H_{eff}(E_o)$ and $S_{eff}(E_o)$ for all E_o simultaneously

$$S_{ij}^{eff} = \delta_{ij} + \sum_{k \in Q} \frac{H_{ki} H_{kj}}{(E_o - H_{kk})^2} \qquad i,j \in P$$

$$H_{ij}^{eff} = H_{ij} + \sum_{k \in Q} \frac{H_{ki} H_{kj}}{(E_o - H_{kk})} \qquad i,j \in P$$

weighted scalar product H_{Qi} "sparse"

Compute

$$C_Q = (\tilde{E} - D_Q)^{-1} H_{QP} C_P$$

if any C_Q are "large" ($> \varepsilon$). Change P and repeat.

Repeat for decreasing ε *until* C *and* E *converge.*

Little experience yet. Probable $\varepsilon \cong 10^{-2}$ for acceptable accuracy.

$$p \sim 10 - 10^2$$

independent of N for quantum C item.

$$\text{cost} \sim p^2 N \qquad \text{(CPU and I/O)}$$

Faster than N^2!

THE USE OF VARIATION-PERTURBATION TECHNIQUES IN CI-CALCULATIONS

B. Roos
Lund, Sweden

We want to solve the problem

$$\hat{H}|\psi\rangle = E|\psi\rangle \tag{1}$$

by expanding the eigenstate $|\psi\rangle$ in a set of known "basis" functions:

$$|\psi\rangle = \sum_{\mu=1}^{N} C_\mu |\mu\rangle, \tag{2}$$

where we normally assume $\langle\mu|\nu\rangle = \delta_{\mu\nu}$. In actual application N may range between 10^3 and 10^5.

Partition: $$\hat{H} = \hat{H}_o + \hat{H}_1 . \tag{3}$$

$\hat{H}_o$ is chosen to be diagonal in the basis space:

$$H_o = \sum_\mu |\mu\rangle \alpha_\mu \langle\mu| \tag{4}$$

where α_μ are arbitrary, and can be chosen to improve convergency of the perturbation expansion. We have $\hat{H}|\mu\rangle = \alpha_\mu|\mu\rangle$, and especially $\hat{H}|0\rangle = E_o|0\rangle$ $(\alpha_o = E_o)$ where $|0\rangle$ is the zeroth order approximation to $|\psi\rangle$ $(|0\rangle = |\psi^{(0)}\rangle)$. The kth order Rayleigh-Schrödinger perturbation theory gives the kth order contribution to $|\psi\rangle$ as:

$$(E_o - \hat{H}_o)|\psi^{(k)}\rangle = \hat{H}_1|\psi^{(k-1)}\rangle - \sum_{n=0}^{k-1} E_{k-n}|\psi^{(n)}\rangle , \tag{5}$$

where E_k is the kth order correction to E. Now expand $|\psi^k\rangle$ in the basis $|\mu\rangle$:

$$|\psi^k\rangle = \sum_\mu C_\mu^{(k)}|\mu\rangle \tag{6}$$

Inserting Eq. (6) into (5) gives,

$$(E_o - \alpha_\mu)C_\mu^{(k)} = \left[\sum_{\nu=1}^{N} \langle\mu|\hat{H}_1|\nu\rangle C_\nu^{(k-1)} - \sum_{n=0}^{k-1} E_{k-n} C_\mu^{(n)}\right] \tag{7}$$

Closed expressions for E_{2k-1} and E_{2k} can be used to update the energy. Introducing a vector

$$\underline{\sigma}^{(k)} = \underline{\underline{H}}\ \underline{c}^{(k-1)}$$

where

$$H_{\mu\nu} = \langle \mu | \hat{H} | \nu \rangle$$

We can write Eq. (7) as

$$(E_o - \alpha_\mu) C_\mu^{(k)} = \sigma_\mu^{(k)} - \alpha_\mu C_\mu^{(k-1)} - \sum_{n=0}^{k-1} E_{k-n} C_\mu^{(n)} \tag{8}$$

The crucial step is the calculation of the σ-vector.

After having obtained the kth order correction in Eq. (8), it is very easy to improve the results by making a variational calculation in the basis of perturbations. Introduce the new basis

$$|\psi^{(0)}\rangle, \quad |\psi^{(1)}\rangle \ldots, \quad |\psi^{(k-1)}\rangle \quad ,$$

and expand $|\psi\rangle$ in this basis:

$$|\psi\rangle = \sum_{i=0}^{k-1} a_i |\psi^{(i)}\rangle \tag{9}$$

This leads to the secular problem

$$\underline{\underline{H}}\ \underline{a} = \tilde{E}^{(k)} \underline{\underline{S}}\ \underline{a} \tag{10}$$

where now

$$H_{ij} = \langle \psi^{(i)} | \hat{H} | \psi^{(j)} \rangle = \underline{c}^{(i)^t} \cdot \sigma^{(j+1)} \quad ,$$

and

$$S_{ij} = \langle \psi^{(i)} | \psi^{(j)} \rangle = c^{(i)^t} c^{(j)} \quad .$$

These matrix elements are thus obtained as simple scalar products. According to McDonald's theorem this method is bound to converge even if the perturbation expansion happens to diverge. An example ($N = 971$):

k	E_{2k}	Linear VE*	Optimal VE†
2	-0.4132	-0.2428	-0.267611
4	-0.2392	-0.2793	-0.313136
6	-0.1374	-0.1333	-0.320340
8	-2.1054	-0.1804	-0.320674
9	-2.7582	-0.0561	-0.320685
10	+7.9544	0.0480	-0.320688

*Variational energy with all a_i in Eq. (9) equal to one.
†Variational energy according to Eq. (10) ($\tilde{E}^{(k)}$).

Some remarks on the method:

- The method converges even if the perturbation expansion diverges. Convergence in this expansion, however, always is enforced by a proper choice of α_μ (level shifting technique).

- The method has been extended to a multi-configuration reference function $|\psi^{(0)}\rangle$ [*Roos and Siegbahn, to be submitted*].

- Largest case studied has $N \sim 80000$ giving more than 5×10^8 non-zero matrix elements, which were actually never calculated, since they could all be expressed in terms of a much smaller number of integrals ($\sim 0.5 \times 10^6$) which were used directly to construct $\underline{\sigma}$.

- More than one root can be obtained by adding eigenvectors of previously obtained roots to the basis used in the variational step.

- An important feature of the method is the fact that the H-matrix elements can be used in random order.

Bibliography

E. Brändas and O. Goscinski, Phys. Rev. A1, 552 (1970).

B. Roos and P. Siegbahn, in Chemical and Biochemical Reactivity, The Jerusalem Symposia on Quantum Chemistry and Biochemistry, The Israel Academy of Sciences and Humanities (Jerusalem, 1974).

HISTORY OF RELAXATION ALGORITHMS AND THEORETICAL CHEMISTRY

Richard C. Raffenetti

Argonne National Laboratory
Argonne, Illinois

Early references:

Systems of equations: R. U. Southwell (1940)

Natural frequency eigensystems: J. L. B. Cooper (1948)

Theoretical chemistry:

Coordinate relaxation: S. F. Boys (1950)

Large systems, updating: R. K. Nesbet (1965)

Use of matrix symmetry: I. Shavitt (1970)

Optimal CR, interior solutions: I. Shavitt, C. F. Bender, A. Pipano, R. Hosteny (1973)

Simultaneous CR: I. Shavitt - R. Raffenetti

THE RESIDUAL VECTOR

The length $||r(\tilde{x})||$ of the residual vector

$$r(\tilde{x}) \equiv (H - \lambda S)\ \tilde{x}/||\tilde{x}||$$

is often used to measure the quality of the vector $\tilde{x}$ as an approximation to the eigenvector x. In an iterative process in which a sequence of eigenvector approximations $\{x^i: i = 0, 1, 2 \ldots\}$ approaches x, the sequence of residual vectors $\{r(x^i): i = 0, 1, 2 \ldots\}$ approaches the null vector.

For the general eigenvalue problem the length of a vector v is defined as

$$||v|| \equiv (v^T S v)^{1/2}$$

COORDINATE RELAXATION

Treat the secular equations as a linear system

$$(H - \rho S)x = \underline{0}$$

and iterate first order changes in a trial vector

$$x \leftarrow x + \delta x$$

to satisfy each of the equations

$$(h_k^T - \rho s_k^T)x = 0.$$

If $\delta x \equiv \alpha e_\ell [(e_\ell)_j = \delta_{e_j}]$, then

$$\alpha = -r_k(x)/(h_{k\ell} - \rho s_{k\ell})$$

with $r_k(x) = (h_k^T - \rho s_k^T)x$. The usual choice of coordinate ℓ is k so that

$$\alpha = -r_k(x)/(h_{kk} - \rho s_{kk}).$$

The eigenvalue is not known so that ρ is taken to be the Rayleigh quotient

$$\rho(x) = (x^T Hx)/(x^T Sx)$$

and ρ approaches the eigenvalue as x approaches the eigenvector.

The update quantities are

$$\Delta q = 2\alpha(s_k^T x + s_{kk}\alpha)$$

and

$$\Delta\rho = \alpha r_k(x)/(q + \Delta q)$$

with $\rho = (x^T Hx)/q$

and $q = x^T Sx$.

SHAVITT CR ALGORITHM

- Effective elimination of the zero matrix elements.
- Use of unique matrix elements only ($H = H^T$).

The matrix is stored out of core and is ordered by rows. Only the non-zero elements are stored along with their column indices. The row index is incremented or if rows are not ordered the row index may be stored also.

The use of symmetry is based on the matrix decomposition $\square = \triangle + \diagdown + \triangledown$ where the block $\triangle$ is the transpose of $\triangledown$ and $\diagdown$ is the diagonal. The result $\square x = x'$ is obtained from $x' = u' + v'$ where $u' = (\triangle + \diagdown)x$ and $v' = \triangledown x$.

The result of multiplying a row of $\square$ onto x can be obtained by accumulating the column sums v' for use in the ith iteration during the $(i-1)^{st}$ iteration. This device requires that all rows be processed in normal (descending) order.

OPTIMAL COORDINATE RELAXATION (OCR)

Shavitt, Bender, Pipano, Hosteny (1973)

Choice of α at each step is to *minimize* the Rayleigh quotient. One obtains

$$a\alpha^2 + b\alpha + c = 0$$

with

$$a = (h_{\ell\ell}x_\ell - h_k^T x)/q$$

$$b = h_{\ell\ell} - \rho s_{\ell\ell}$$

$$c = h_k^T x - \rho s_k^T x \equiv r_k(x) \ .$$

Choose α which decreases ρ .

Note that expanding the expression for α gives

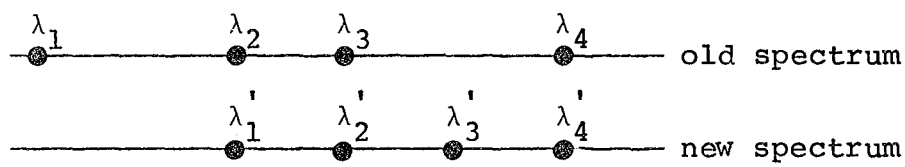
$$\alpha = -(c/b)[1 + (2ac/b^2) + \ldots]$$

and the first term is equivalent to that obtained in ordinary CR.

HIGHER EIGENSOLUTIONS

- Root-shifting: a variant of deflation.
- Orthogonality constraint.

Root-shifting: Implicity modify H so that eigenvalues corresponding to lower eigenvectors are shifted out of the way.



$$H^{(p)} = H + \sum_{k=1}^{p-1} \sigma_k S x_k x_k^T S \ .$$

Apply ordinary OCR to $H^{(p)}$ instead of H but without forming $H^{(p)}$ explicitly. (All prior devices for effective algorithms can still be used.)

Orthogonality constraint: Keep the vector being iterated explicitly orthogonal to lower eigenvectors. The computational work is greater than that for root shifting and the convergence is usually somewhat faster. The total work is about equal. Orthogonality is maintained in an implicit fashion.

THE BASIC CR ALGORITHM

$(Ax = \lambda x)$

Initialize: determine x; compute $x^T Ax$, $x^T x$, $\rho(x)$

Test for convergence: if satisfied, then stop, else

begin the row iteration: $i = 1, 2, \ldots n$

Obtain the ith row of A.

Determine α.

$x \leftarrow x + \alpha e_i$

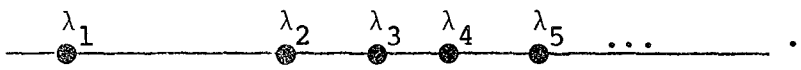
Update $x^T Ax$, $x^T x$, $\rho(x)$

End the row iteration

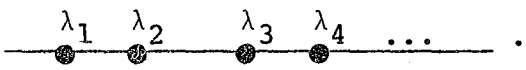
Return to the convergence test.

CR AND OCR CONVERGENCE

Convergence of an eigenvector and eigenvalue is slowed when higher eigenvalues are "close" to the one being sought. In CI problems the spectrum of eigenvalues for a molecule at its equilibrium geometry may schematically look like



Away from equilibrium the spectrum may be



The success of CR and OCR has been due to the fact that most of the earlier calculations have been carried out to obtain the lowest eigensolution in the former situation. We are now interested in higher eigensolutions and in molecules not close to their equilibrium geometries. In either case the eigenvalues are closer and convergence is slower.

SIMULTANEOUS COORDINATE RELAXATION (SCR)

- Relax p trial vectors simultaneously

$$Y_p = (y_1 y_2 \cdots y_p), \quad Y_p^T Y_p = Q \neq I$$

- Employ the root-shifting formalism without having accurate lower eigenvectors

$$H^{(k)} = H + \sigma_{k-1} Y_{k-1} Q^{-1} Y_{k-1}^T$$

with

$$Q^{-1} \approx I$$

or

$$Q^{-1} \approx 2I - Q \quad \text{where} \quad Q = I + \Delta \ .$$

- At the end of each major iteration solve the p x p problem

$$PC = QC\Lambda \qquad P = Y_p^T H Y_p$$

and transform Y_p

$$X_p \leftarrow Y_p C \ .$$

- The root-shifting formalism ensures that each higher eigenvalue is approached from above.

- Convergence of the lowest eigenvector is not slowed by the presence of higher vectors.

- If all members of a cluster of "close" eigenvalues are iterated simultaneously, the convergence rate is increased.

- The IO operations are diminished.

- Principal quantities may be updated.

- Sparsity is not destroyed.

THE SIMULTANEOUS CR ALGORITHM

$$(AX_p = X_p\Lambda)$$

Initialize: determine X_p; compute P, Q, ρ_j $(j = 1, 2, \ldots p)$

Test for convergence: if satisfied, then stop, else

begin the row iteration: $i = 1, 2, \ldots n$

Obtain the ith row of A

Compute $a_i^T x_j$ $\quad j = 1, 2, \ldots p$

Begin the column iteration: $j = 1, 2, \ldots p$

$x_j \leftarrow x_j + \alpha e_i$

Update P, Q, ρ_j

End the column iteration

End the row iteration

Solve: $PC = QC\Lambda$

$X_p \leftarrow X_p C$

Update $P \leftarrow \Lambda$, $Q \leftarrow I$, $\rho_j = \Lambda_{jj}$ $(j = 1, 2, \ldots p)$

Return to the convergence test.

ADVANTAGES AND DISADVANTAGES OF SCR

ADVANTAGES:

- CR methods are single vector (Gauss-Seidel) processes. Jacobi-type methods require two vectors ($x^{i+1} = Ax^{i}$).

- SCR has improved convergence (vs CR).

- The IO processing is decreased substantially.

DISADVANTAGES:

- The matrix must be ordered by rows.

TEST MATRICES

The "Nesbet" matrix used for the convergence tests shown on the following pages is defined as follows:

$$a_{ii} = 2i - 1 , \qquad i = 1,2,3, \ldots 50$$

$$a_{ij} = a_{ji} = 1 , \qquad i < j$$

The "modified" Nesbet matrix is the same matrix except that the diagonal elements 3, 5, 7, and 9 are replaced by 1.1, 1.2, 1.3, and 1.4 respectively.

The actual eigenvalues of these matrices are spaced essentially like the original diagonal elements.

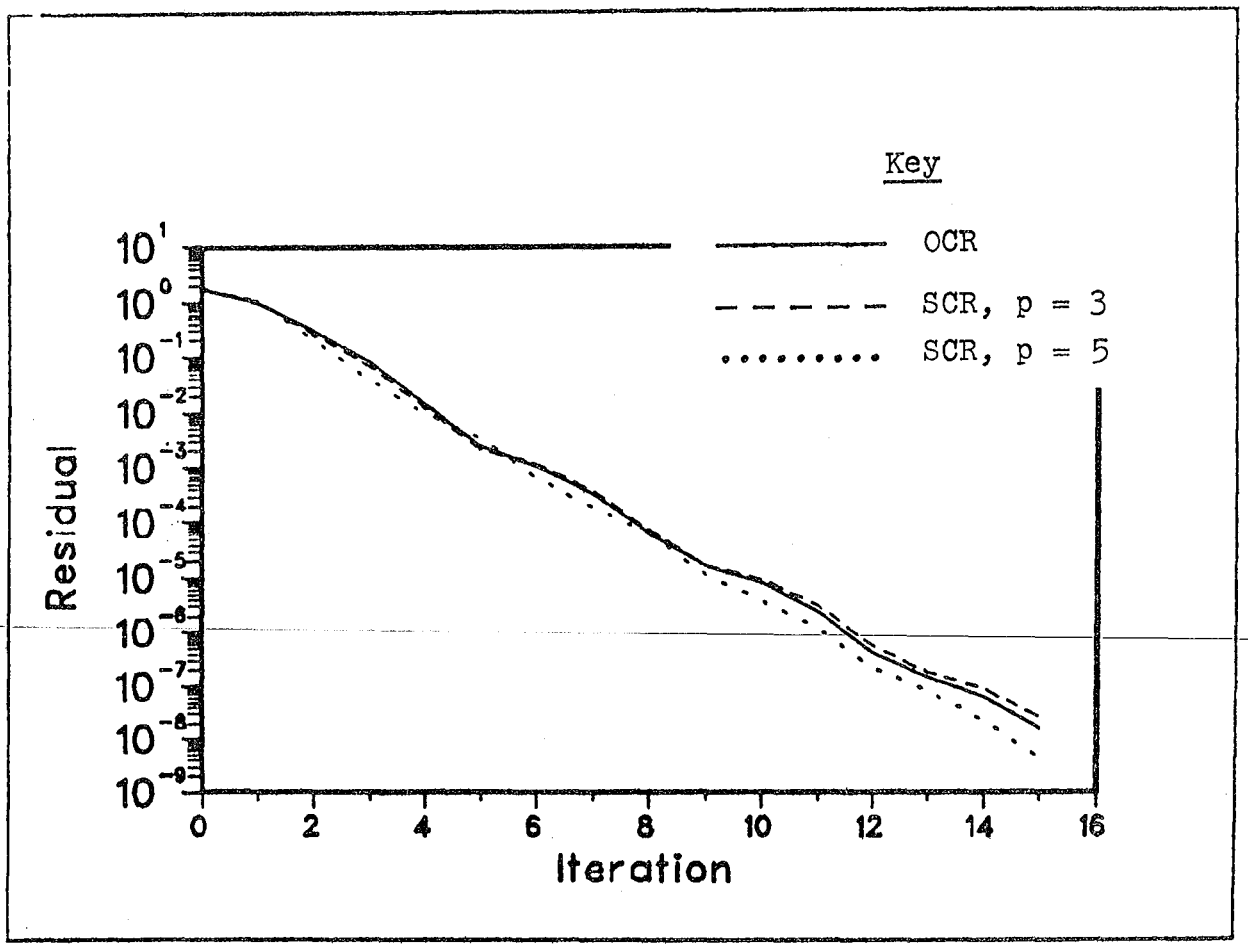
Note that this matrix is expressed in units far larger than the matrices encountered in chemical problems. A scaling of the entire matrix by ~1/50 would produce off-diagonal elements and eigenvalue spacings which better resemble Hamiltonian matrices. Solution of the eigenvalue problem is invariant to such a scaling.

0 0 0 0 5 1 0 2435 6 1

Optimal vs. Simultaneous Coordinate Relaxation:

SCR convergence of the 3rd eigenvector

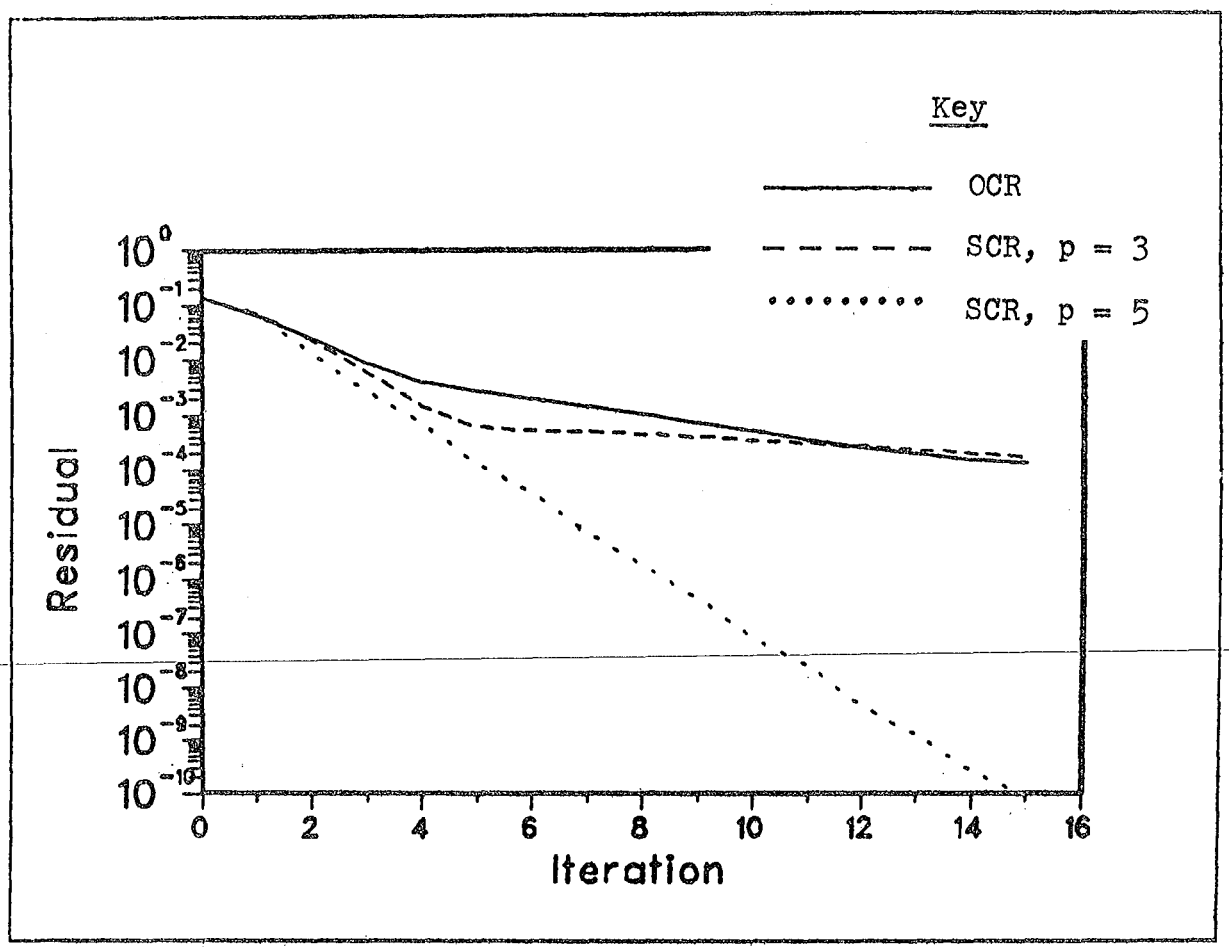
(Nesbet Matrix)



Optimal vs. Simultaneous Coordinate Relaxation:

SCR convergence of the 3rd eigenvector

(Modified Nesbet Matrix)

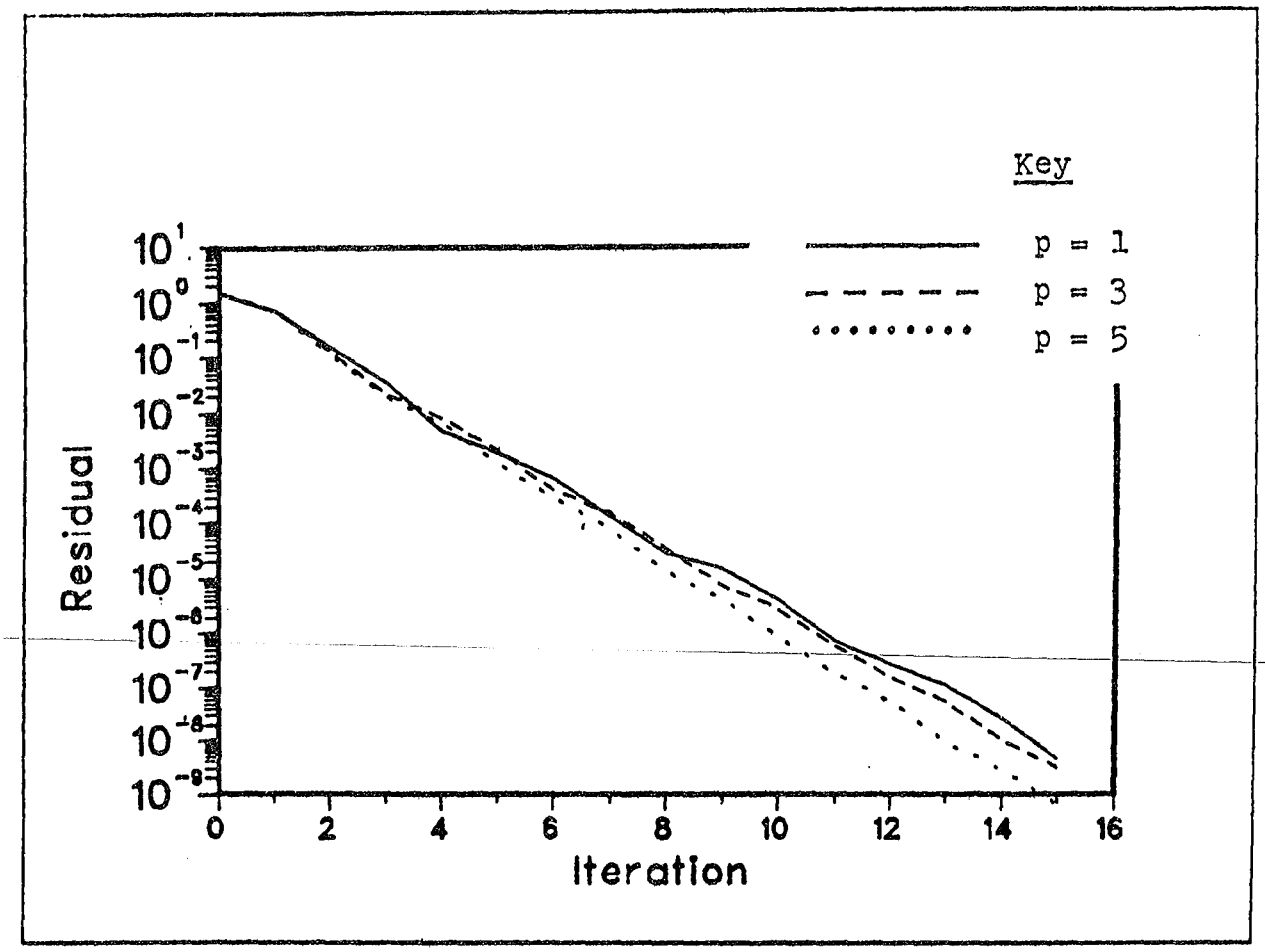


0 0 0 0 5 1 0 4 5 6 6 2

The Effect of the Size of the Subspace:

SCR convergence of the lowest eigenvector

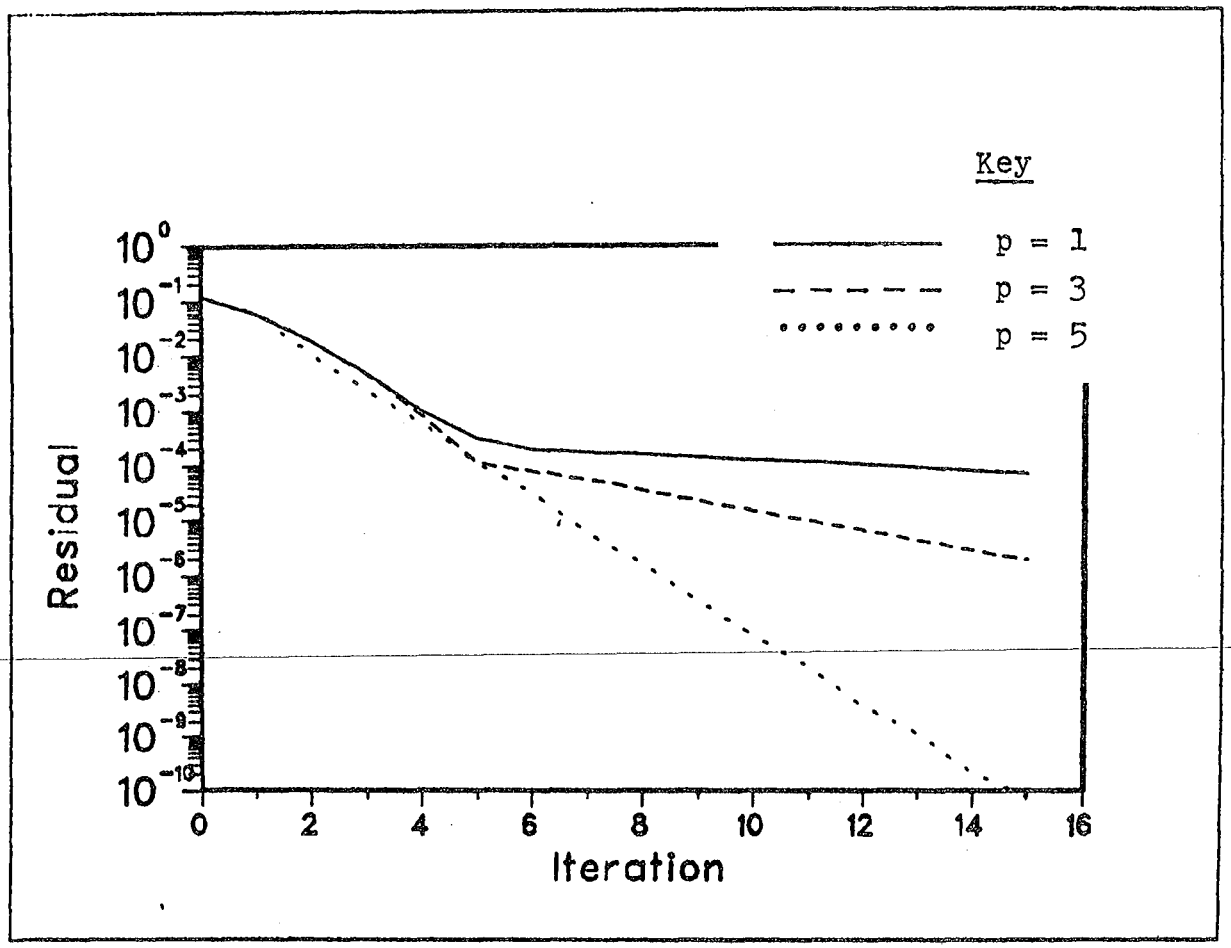
(Nesbet Matrix)



The Effect of the Size of the Subspace:

SCR convergence of the lowest eigenvector

(Modified Nesbet Matrix)

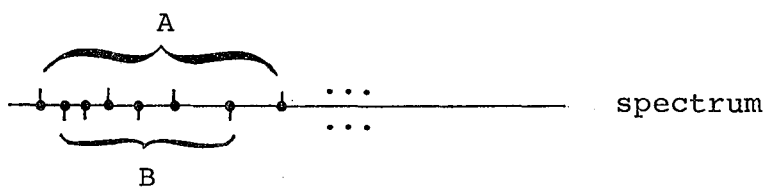


PROBLEMS

- Incipient "diagonal blocking" of the hamiltonian matrix at extremes of a molecular system.

- Starting vectors.

$$\begin{bmatrix} A & C \\ C^T & B \end{bmatrix} \qquad C \rightarrow \sim 0$$



Cases:

Molecular fragmentation

Rydberg-valence state separation

FUTURE DEVELOPMENT OF CR AND SCR

- Iteration of one of or a group of the interior eigenvectors and eigenvalues is possible.

The Rayleigh quotient has saddle points when the vector corresponds to an interior eigenvalue.

Therefore if a trial vector is close enough to an eigenvector the relaxation parameter may be chosen so as to cause the *minimum* change in the vector. The Rayleigh quotient may be raised or lowered. It will move to the nearest minimum, maximum, or saddle point, all of which satisfy

$$\frac{d}{d\alpha}\rho(x) = 0 \quad .$$

THE SIMULTANEOUS EXPANSION METHOD FOR THE ITERATIVE SOLUTION OF SEVERAL OF THE LOWEST EIGENVALUES AND CORRESPONDING EIGENVECTORS OF LARGE REAL-SYMMETRIC MATRICES

B. Liu

IBM Research Laboratory
San Jose, California 95193

Davidson's expansion method[1] for the iterative solution of matrix eigenvalue problems has been generalized to yield simultaneously several of the lowest eigenvalues, and corresponding eigenvectors of large real-symmetric matrices. The principal advantages of this simultaneous expansion method are a reduction of the number of times the matrix elements are read from peripheral storage, and a reduction of the number of iterations required for each solution. Assume that M solutions are being iterated upon simultaneously, and that each solution individually requires I iterations to reach convergence. The number of scans of the matrix elements would be M×I using the original Davidson method, and at most I using the simultaneous expansion method, which often would yield all M converged solutions in less than I iterations.

Consider the matrix eigenvalue problem

$$\underline{A}\,\underline{c}^k = \lambda^k \underline{c}^k, \qquad k = 1, 2, \ldots, M$$

where $\underline{A}$ is an N×N real-symmetric matrix. The main storage requirement of the simultaneous expansion method is 2M vectors of dimension N and a small amount of buffer space for reading the elements of $\underline{A}$ from peripheral storage. The processing time per iteration is essentially M times that required for one iteration of the original Davidson Method.

The basic steps of the simultaneous expansion method are as follows.

1. Select a set of orthonormal trial vectors

   $\{\underline{b}_i\}$ with $i = 1, 2, \ldots L$, where $L \geq M$.

2. Use a small matrix method to solve the LXL eigenvalue problem

   $$\underline{G}\ \underline{\alpha}^k = \lambda^k \underline{\alpha}^k, \qquad k = 1, 2, \ldots, M$$

   where the elements of $\underline{G}$ are given by

   $$G_{ij} = (\underline{b}_i, \underline{A}\ \underline{b}_j), \qquad 1 \leq i, j \leq L.$$

3. Form the correction vectors $\underline{f}^k$, $k = 1, 2, \ldots, M$, with components

   $$f_I^k = (\lambda^k - A_{II})^{-1}\, d_I^k\ , \qquad I = 1, 2, \ldots, N$$

   where

   $$\underline{d}^k = \sum_{i=1}^{L} {}^k_i(\underline{A} - \lambda^k)\underline{b}_i\ .$$

4. Normalize $\underline{f}^k$.

5. Schmidt orthonormalize and append $\underline{f}^1$ to $\{\underline{b}_i\}$. Schmidt orthogonalize $\underline{f}^2$ to $\{\underline{b}_i\}$, now with L+1 vectors. If the norm of the resulting vector is less than some threshold T, say 10^{-3}, then go to the next correction vector. Otherwise, normalize the resulting vector and append to the set $\{\underline{b}_i\}$. Repeat this process for all $\underline{f}^k$. At the end of this step the set $\{\underline{b}_i\}$ has L+m vectors with $1 \leq m \leq M$.

6. Schmidt orthonormalize $\{\underline{b}_i\}$. This step is necessary to insure true orthogonality of the expansion vector set, and avoid accumulation of numerical errors in the iterative process.

7. Increase L by m and go back to step 2.

The convergence of the kth solution may be checked[1] after step 2 by the sum of the squares of the last m components of $\underline{\alpha}^k$, or after step 3 by the size of $(\underline{d}^k, \underline{d}^k)^{1/2}$.

As example we consider the following test matrices.

$$A_{ij} = A_{ji} = 1 \qquad 1 \le i,j \le N$$

$$A_{ii} = 1 + 0.1 \times (i-1) \qquad 1 \le i \le 5$$

$$A_{ii} = 2i - 1 \qquad 5 < i \le N$$

The convergence of the 4 lowest eigenvalues are shown iteration by iteration in the following table for N=50 and 250, respectively. In each case the trial vectors were obtained by diagonalizing the leading 4×4 submatrix. Convergence for all 4 state to 1×10^{-12} in the eigenvalue were achieved in 4 iterations.

It is of interest to compare the simultaneous expansion method with the simultaneous coordinate relaxation method described earlier in this proceeding.[2] Both methods achieve the same reduction in scans of the matrix elements relative to the respective single-solution methods. However, the expansion method has the advantage of rapid convergence for nearly degenerate solutions.[1] The relaxation method often can only achieve rapid convergence when all nearly degenerate roots are treated simultaneously, thus increasing both main storage requirement and processing time. The simultaneous expansion method has the additional advantage of improved convergence for each solution due to a shared

expansion set consisting of correction vectors for all of the solutions treated. There is no comparable gain for the simultaneous relaxation method. Furthermore the expansion method does not require sequential access to the matrix elements and, therefore, can be used in direct CI calculations. On the other hand, the expansion method has a larger input/output requirement, needing to read 2L vectors of dimension N in each iteration. Also, in practice, the relaxation method often yields more rapid convergence for the lowest eigenvalue when only one solution is desired.[1] In conclusion, it is my opinion that simultaneous expansion method is to be preferred over the simultaneous relaxation method. However, it is most desirable to implement both methods in a general purpose configuration interaction program.

References

1. E.R.Davidson, J. Computational Phys. 17, 87 (1975).
2. R.Raffenetti, J. Computational Phys., to be published.

TABLE 1

N = 50, M = 4, L = 4

	1	2	3	4
*	.033608040442	.143251493711	.251974770602	.362342667413
0	.034653842412	.144531631596	.253303127752	.363512358083
1	.033608736525	.143252566719	.251976144864	.362344143590
2	.033608040833	.143251494463	.251974771778	.362342668936
3	.033608040442_	.143251493712	.251974770603	.362342667414
4	.033608040442_	.143251493711_	.251974770602_	.362342667413_

N = 250, M = 4, L = 4

0	.034653842412	.144531631596	.253303127752	.363512358083
1	.032927059063	.142406634660	.251084436398	.361544227781
2	.032925890029	.142404814222	.251082075853	.361541703060
3	.032925889256	.142404812722	.251082073478	.361541699938
4	.032925889255_	.142404812720_	.251082073476_	.361541699934_

*Householder-Givens method

POWER METHODS AND LANCZOS METHODS FOR THE EIGENVALUE PROBLEM $Ax = \lambda x$

C. Van Loan
Cornell University

Notation: $A = (a_{ij})$, $A^T = (a_{ji})$.

Power Iteration

$$y^{(0)} = \text{starting vector}$$

for

$$k = 1, 2, \ldots .$$

$$z^{(k)} = A\, y^{(k-1)}$$

$$\alpha^{(k)} = \| z^{(k)} \|$$

$$y^{(k)} = z^{(k)} / \alpha^{(k)}$$

Suppose

$$y^{(0)} = a_1 x_1 + \ldots + a_n x_n$$

where

$$Ax_i = \lambda_i x_i \qquad |\lambda_1| > |\lambda_2| \geqslant \ldots \geqslant |\lambda_n|$$

Then, $y^{(k)}$ is unit vector in direction of

$$a_1 x_1 + a_2 \left(\frac{\lambda_2}{\lambda_1}\right)^k x_2 + \ldots + a_n \left(\frac{\lambda_n}{\lambda_1}\right)^k x_n$$

and

$$|\alpha^{(k)}| = |\lambda_1| + O\left(\frac{|\lambda_2|^k}{|\lambda_1|^k}\right)$$

Inverse Iteration

$$\mu = \text{approximate eigenvalue}$$

$$y^{(0)} = \text{starting vector}$$

For $k = 1, 2, \ldots$, solve:

$$(A - \mu I)\, z^{(k)} = y^{(k-1)}$$

$$\alpha^{(k)} = \| z^{(k)} \|$$

$$y^{(k)} = z^{(k)} / \alpha^{(k)}$$

Suppose

$$y^{(0)} = a_1 x_1 + \ldots + a_n x_n$$

where

$$A x_i = \lambda_i x_i$$

Then $y^{(k)}$ is unit vector in direction of

$$\frac{a_1}{(\lambda_1 - \mu)^k} x_1 + \ldots + \frac{a_n}{(\lambda_n - \mu)^k} x_n$$

<u>Two questions</u>:

Given x, what λ minimizes $\|(A - \lambda I)x\|_2$?

Answer: $\lambda = \dfrac{x^T A x}{x^T x}$

Given λ, what unit vector x minimizes $\|(A - \lambda I)x\|_2$?

Answer: $(A - \lambda I)^T (A - \lambda I)x = \mu_{min}\, x$

<u>Rayleigh Quotient Iteration</u>

$$y^{(0)} = \text{initial unit starting vector}$$

for $k = 1, 2, \ldots$

$$\rho_{k-1} = y^{(k-1)^T} A\, y^{(k-1)}$$

solve:

$$(A - \rho_{k-1}\, 1)\, z^{(k)} = y^{(k-1)}$$

$$\alpha_K = \| z^{(k)} \|$$

$$y^{(k)} = z^{(k)} / \alpha_k$$

For symmetric A:

- cubic convergence locally
- essentially globally convergent
- $\|(A-\rho_k I)y^{(k)}\|_2 \downarrow 0$

Errors and Residuals

A symmetric:

$$Ax = \hat{\lambda}x + r$$

$$\Rightarrow \quad (A - rx^T)x = \hat{\lambda}x$$

$$\Rightarrow \quad \exists \lambda \in \lambda(A) \quad \text{so} \quad |\lambda - \hat{\lambda}| \leqslant \|r\|_2$$

A unsymmetric:

$$\underbrace{\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \cdot \\ \vdots & & \ddots & 1 \\ & & & \cdot \\ \varepsilon & \cdots & & 0 \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} 1 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{pmatrix}}_{x} = \underbrace{0}_{\hat{\lambda}} \underbrace{\begin{pmatrix} 1 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{pmatrix}}_{x} + \underbrace{\begin{pmatrix} 0 \\ \cdot \\ \cdot \\ 0 \\ \varepsilon \end{pmatrix}}_{r}$$

But $\lambda \in \lambda(A) \Rightarrow |\lambda| = \varepsilon^{1/n}$

Generalization: Invariant Subspaces

$$\begin{bmatrix} A \end{bmatrix} \begin{bmatrix} X & Y \end{bmatrix} = \begin{bmatrix} X & Y \end{bmatrix} \begin{bmatrix} S & T \\ O & R \end{bmatrix}$$

$$X = [x_1 | \ldots | x_k] \qquad x_k \in R^n$$

Span $\{x_1, \ldots, x_k\}$ is invariant for A: $AXy = XSy \in \text{span}\{x_1, \ldots, x_k\}$.

The eigenvalues of S are eigenvalues of A:

$$\lambda(S) \subset \lambda(A)$$

Orthogonal Iteration

$$Y^{(0)} = n\overset{r}{\square} = \text{starting matrix with orthonormal columns}$$

For $k = 1, 2, \ldots$

$$Z^{(k)} = A\,Y^{(k-1)} = \square$$

$$Z^{(k)} = Q^{(k)}\,R^{(k)} = \square\,\nwarrow \qquad \text{(modified Gram-Schmidt)}$$

$$Y^{(k)} = Q^{(k)}$$

The columns of $Y^{(k)} = \left[y_1^{(k)} \,|\, \ldots \,|\, y_r^{(k)}\right]$ form an orthonormal basis for the range of $A^k\,Y^{(0)}$.

Error in Orthogonal Iteration:

$$|\lambda_1| \geqslant \ldots. \geqslant |\lambda_r| > |\lambda_{r+1}| \geqslant \ldots. \geqslant |\lambda_n|$$

$$A\underbrace{[y_1 \,|\, \ldots \,|\, y_j]}_{\text{orthonormal}} = [y_1 \,|\, \ldots \,|\, y_j]\,T_j$$

$$\lambda(T_j) = \{\lambda_1, \ldots, \lambda_j\}$$

$$S_j = \text{span}\{y_1, \ldots, y_j\}$$

$$S_j^{(k)} = \text{span}\{y_1^{(k)}, \ldots, y_j^{(k)}\}$$

For $j = 1, \ldots, r$

$$S_j^{(k)} = S_j + O\left[\left|\frac{\lambda_{j+1}}{\lambda_j}\right|^k\right]$$

Question: Given $X = \underset{r}{\square}\,n$ with independent columns, what $r \times r$ S minimizes $\|AX - XS\|_F$? (where $\|\beta\|_F^2 = \sum_i \sum_j |b_{ij}|^2$)

Answer:

$$S = (X^T X)^{-1}\,(X^T AX) = X^T AX \qquad \text{if} \qquad X^T X = I$$

Ritz Acceleration

Suppose columns of X $(X^T X = I)$ approximate the dominant eigenspace.

If $\quad S = X^T AX$

and $\quad Q^T S Q = T = \triangledown$

$$|t_{11}| \geqslant \ldots \geqslant |t_{rr}|$$

then columns of XQ are better.

$$\min = \|AX - XS\|_F = \|A(XQ) - (XQ)T\|_F$$

Orthogonal Iteration with Ritz Acceleration

$Y^{(0)} = \square$ = starting matrix with orthonormal columns

For $\quad k = 1, 2, \ldots$

$$Z^{(k)} = A\, Y^{(k-1)}$$

$$Z^{(k)} = Q^{(k)} R^{(k)} = \square \triangledown$$

$$A^{(k)} = Q^{(k)^T} A\, Q^{(k)}$$

$$V^{(k)^T} A^{(k)} V^{(k)} = T^{(k)} = \triangledown \quad \text{(QR)}$$

$$Y^{(k)} = Q^{(k)} V^{(k)} = \square \square$$

Convergence is just as for orthogonal iteration, except we replace

$$\left| \frac{\lambda_{j+1}}{\lambda_j} \right|^k \quad \text{by} \quad \left| \frac{\lambda_{r+1}}{\lambda_j} \right|^k$$

Lanczos: Motivation

A symmetric

There exists an orthogonal Q such that $Q^T AQ = J$ is tridiagonal.

Proof:

$$\begin{matrix} x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \end{matrix} \longrightarrow \begin{matrix} x & x & 0 & 0 & 0 \\ x & x & x & x & x \\ 0 & x & x & x & x \\ 0 & x & x & x & x \\ 0 & x & x & x & x \end{matrix}$$

Lanczos Algorithm

$$A[q_1 | \dots | q_n] = [q_1 | \dots | q_n] \begin{pmatrix} \alpha_1 & \beta_1 & & 0 \\ \beta_1 & \alpha_2 & & \\ & & & \beta_{n-1} \\ 0 & & \beta_{n-1} & \alpha_n \end{pmatrix}$$

$$Aq_k = \beta_{k-1} q_{k-1} + \alpha_k q_k + \beta_k q_{k+1}$$

or

$$\beta_k q_{k+1} = Aq_k - \alpha_k q_k - \beta_{k-1} q_{k-1}$$

Know

$$q_1, \dots, q_k \qquad q_i^T q_j = \delta_{ij}$$

$$\alpha_1, \dots, \alpha_{k-1}$$

$$\beta_1, \dots, \beta_{k-1}$$

$$0 = q_k^T q_{k+1} = q_k^T A_{qk} - \alpha_k$$

$$1 = q_{k+1}^T q_{k+1} \Rightarrow |\beta_k| = \| A_{qk} - \alpha_k q_k - \beta_{k-1} q_{k-1} \|$$

Here it is: q_1 = unit starting vector

For $k = 1, 2, \dots$

$$\alpha_k = q_k^T A_{qk}$$

$$r_k = \begin{cases} Aq_k - \alpha_k q_k & (k = 1) \\ Aq_k - \alpha_k q_k - \beta_{k-1} q_{k-1} & (k > 1) \end{cases}$$

$$\beta_k = \| r_k \|_2$$

$$q_{k+1} = r_k / \beta_k$$

$\beta_i \neq 0$, $i = 1, \ldots, k \Rightarrow \{q_1, \ldots, q_{k+1}\}$ is orthonormal basis for span $\{q_1, Aq_1, \ldots, A^k q_1\}$

The Eigenvalues of T_k

After k-steps we have

$$T_k = \begin{pmatrix} \alpha_1 & \beta_1 & & & 0 \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \ddots & \ddots & \ddots & \\ & & & \beta_{k-1} & \\ 0 & & & \beta_{k-1} & \alpha_k \end{pmatrix}$$

$$Q_k = [q_1 | \ldots | q_k]$$

can show

$$AQ_k - Q_k T_k = \left[0|0| \ldots |0| \ \beta_k \ q_{k+1}\right]$$

If $T_k s_j = \theta_j s_j$, $\|s_j\|_2 = 1$, then $\exists \ \lambda \in \lambda(A)$, so $|\lambda - \theta_j| \leqslant |\beta_k| \ |s_{kj}|$.

Lanczos and Extreme Eigenvalues

Sample result:

$\lambda_1 \geqslant \lambda_2 \geqslant \ldots \geqslant \lambda_n$ e.v. of A

$\mu_1 \geqslant \ldots \geqslant \mu_k$ e.v. of T_k

$$|\lambda_n - \mu_k| \leqslant \frac{(\lambda_1 - \lambda_n)C}{p_k^2\left(\frac{1+v}{1-v}\right)}$$

$$v = \frac{\lambda_{n-1} - \lambda_n}{\lambda_1 - \lambda_n}$$

$p_k(z)$ is k^{th} Chebyshev polynomial.

Loss of Orthogonality: Why

Suppose q_1 is almost an eigenvector: $Aq_1 = \lambda q_1 + r$ $\quad (\|r\| = \varepsilon)$

Now q_2 is defined by $\beta_1 q_2 = Aq_1 - \alpha_1 q_1$, where $\alpha_1 = q_1^T A q_1$. Can show

$$\|Aq_1 - \alpha_1 q_1\| = \frac{\varepsilon}{\sqrt{1-\varepsilon^2}}$$

$\Rightarrow$ Cancellation .

Errors in Approximate Invariant Subspaces (symmetric case)

$$\boxed{A}\,\boxed{X} = \boxed{X}\,\boxed{S} + \boxed{E}$$

For every eigenvalue θ of S there is an eigenvalue λ of A, so

$$|\lambda - \theta| \leqslant \sqrt{2}\,\|E\| \;\; \|(X^T X)^{-\frac{1}{2}}\|_2$$

Coping with Orthogonality Loss

Re-orthogonalization:

- keep $q_1, \ldots, q_k$ around
- compute $\tilde{q}_{k+1}$
- set q_{k+1} = component of $\tilde{q}_{k+1}$ in span $q_1, \ldots, q_k$ $^{\perp}$

Selective re-orthogonalization:

[Parlett and Scott]

Iterative Lanczos

1. q_1 unit starting vector
2. generate $q_2, \ldots, q_k$
3. $T_K u = \lambda_{min} u \quad (\|u\| = 1)$
4. $q_1 \leftarrow [q_1 | \ldots | q_k]u$
5. Go to 2

Block Lanczos

$$Q_1 = \underset{p}{\square}^{\,n} \qquad \text{starting matrix:} \quad Q_1^T Q_1 = I$$

For $k = 1, 2, \ldots$

$$M_k = Q_k^T A\, Q_k$$

$$Z_{k+1} = \begin{cases} A\,Q_k - Q_k M_k & (k = 1) \\ A\,Q_k - Q_k M_k - Q_{k-1} R_k^T & (k > 1) \end{cases}$$

$$Z_{k+1} = Q_{k+1} R_{k+1} = \square\!\!\nwarrow \quad \text{(MGS)}$$

$$A[Q_1 | \ldots | Q_k] = [Q_1 | \ldots | Q_k]\hat{T}_k + [0 | \ldots | 0 | Z_{k+1}]$$

$$\hat{T}_k = \begin{pmatrix} M_1 & R_2^T & & \\ R_2 & M_2 & \ddots & \\ & \ddots & \ddots & R_k^T \\ & & R_k & M_k \end{pmatrix}$$

THE GENERALIZED EIGENVALUE PROBLEM IN QUANTUM CHEMISTRY

Nelson Beebe
University of Florida

Solution of a linear differential eigenvalue equation, $f\phi = \varepsilon\phi$, by the expansion method reduces to the problem of solving the generalized eigenvalue problem:

$$\mathbb{F}\,\mathbb{C} = \mathbb{S}\,\mathbb{C}\epsilon \tag{1}$$

where

$$F_{ij} = \langle \phi_i | f | \phi_j \rangle = F^*_{ji} \qquad \text{(Fock matrix)}$$

and

$$S_{ij} = \langle \phi_i | \phi_j \rangle = S^*_{ji} \qquad \text{(overlap matrix)}$$

and the overlap matrix *should* be a positive definite matrix. Löwdin has pointed out that if we know a matrix $\mathbb{B}$ such that

$$\mathbb{B}^+ \mathbb{S}\mathbb{B} = \mathbb{1} \qquad \text{(unit matrix)} \quad , \tag{2}$$

then Eq. (1) reduces to solving a standard eigenvalue problem

$$\tilde{\mathbb{F}}\,\tilde{\mathbb{C}} = \tilde{\mathbb{C}}\epsilon$$

where

$$\tilde{\mathbb{F}} = \mathbb{B}^+ \mathbb{F}\mathbb{B}$$

and $\mathbb{C}$ is obtained from

$$\mathbb{C} = \mathbb{B}\tilde{\mathbb{C}}$$

The optimum method for $\tilde{\mathbb{F}}\tilde{\mathbb{C}} = \tilde{\mathbb{C}}\epsilon$ is implemented in EISPACK routines TRED2 and TQL2, which require only $N^2 + N$ storage locations, rather than the $N^2 + \frac{1}{2}N(N+1) + N$ usually needed. Clearly $\mathbb{B}$ is not unique; it could be replaced by $\mathbb{B}\mathbb{U}$, where $\mathbb{U}$ is any unitary matrix. Several choices of $\mathbb{B}$ are commonly used:

1. Solve $\mathbb{S}\mathbb{U} = \mathbb{U}\mu$ and form $\mathbb{B} = \mathbb{U}\mu^{-\frac{1}{2}}\mathbb{U}^\dagger \equiv \mathbb{S}^{-\frac{1}{2}}$. This is known as *symmetric orthonormalization*, since it preserves symmetry properties in $\mathbb{F}$ [*Slater, Lowdin, ~1951*].

2. Solve $\mathbb{S}\mathbb{U} = \mathbb{U}\mu$ and form $\mathbb{B} = \mathbb{U}\mu^{-\frac{1}{2}}$. This is known as *canonical orthonormalization*, and it has been suggested that if any μ_k is small, that $\mu_k^{-\frac{1}{2}}$ be

replaced by 0 in forming $\mathbb{B}$; this introduces zero columns in $\mathbb{B}$ and effectively reduces the order of the problem.

3. Find an upper triangular matrix $\mathbb{B} = \mathbb{T} = (\mathbb{X})$ by Gram-Schmidt orthonormalization:

$$\mathbb{S} = \langle \phi | \phi \rangle$$

$$\mathbb{T}^{+}\mathbb{S}\mathbb{T} = \mathbb{1} = \mathbb{T}^{+}\langle \phi | \phi \rangle \mathbb{T} = \langle \phi\mathbb{T} | \phi\mathbb{T} \rangle = \langle \psi | \psi \rangle$$

The set ψ is simply the GS orthonormalized set obtained from ϕ. ψ is not produced explicitly; $\mathbb{T}$ may be computed so that it overwrites $\mathbb{S}$. Approximate linear dependencies which are found in the construction of $\mathbb{T}$ are treated by the introduction of a zero column into $\mathbb{T}$.

$\mathbb{T}$ is identical to the transposed inverse of the Cholesky matrix $\mathbb{L}$ in the decomposition $\mathbb{S} = \mathbb{L}\mathbb{L}^{+}$; however, $\mathbb{L}$ is not needed explicitly.

This last method is most economical since the transformation $\mathbb{T}^{+}\mathbb{F}\mathbb{T}$ can be done in $\frac{1}{2}N^3$ operations compared to $\frac{3}{2}N^3$ for $\mathbb{B}^{+}\mathbb{F}\mathbb{B}$ if $\mathbb{B}$ is a full matrix, and since the back-transformation $\mathbb{C} = \mathbb{T}\tilde{\mathbb{C}}$ can be arranged so that $\mathbb{C}$ overwrites $\tilde{\mathbb{C}}$. However, *NONE* of these methods, *including* canonical orthonormalization, deals reliably with the case where $\mathbb{S}$ is numerically nearly singular. The following example given by Fix and Heiberger illustrates this:

$$\mathbb{F} = \begin{bmatrix} 6 & & & & & & 1 & 0 \\ & 5 & & & & & & 1 \\ & & 4 & & & & & \\ & & & 3 & & & & \\ & & & & 2 & & & \\ & & & & & 1 & & \\ 1 & & & & & & 0 & \\ 0 & 1 & & & & & & 0 \end{bmatrix} \qquad \mathbb{S} = \mathrm{diag}\,(1,1,1,1,\delta,\delta,\delta,\delta)$$

The exact eigenvalues are: 3, 4, $2/\delta$, $1/\delta$, $1/2(-1 \pm \sqrt{36 + 4/\delta^2})$, $1/2(-1 \pm \sqrt{25 + 4/\delta^2}$. As $\delta \to 0$, only two of these remain finite. However, application of canonical orthonormalization or Gram-Schmidt orthonormalization with appropriate tolerances would predict eigenvalues 6, 5, 4, and 3 -- the wrong answer. This example is perhaps somewhat artificial, but the production of eigenvalues which are in fact meaningless can occur in practice. An example which I encountered was using a K atom basis given by Wachter on two different machines (IBM 370/165, 16 figures and CDC-6400, 13 figures) with Clementi and Veillard's

atomic SCF programs; one converged, the other failed due to production of nonsense eigenvalues.

The QZ algorithm developed by Moler and Stewart looked promising, since it solves Fix and Heiberger's example correctly, but tests comparing it with canonical orthonormalization on some simple atomic problems show that it does not perform significantly better.

RESPONSE TO BEEBE BY MOLER

The QZ algorithm is probably *not* the way to solve this problem because it destroys symmetry.

The Cholesky or Schmidt approach is risky because linear dependence may not be revealed by small diagonal elements in L and because introduction of zero columns in T may remove important information and raise the eigenvalues drastically.

The problem of finding a completely satisfactory algorithm for the symmetric generalized eigenvalue problem which preserves symmetry, which is efficient and numerically stable is still unsolved. One possible approach is described in a yet unpublished paper by Moler and Wilkinson. It is essentially a careful implementation of the symmetric orthonormalization method which employs modified Givens transformations and the QR algorithm. Further research is needed to investigate the effectiveness of this approach in problems of quantum chemistry.

FELER'S METHOD FOR FINDING EIGENVALUES AND EIGENVECTORS

Nelson Beebe
University of Florida

M. Guy Feler: *J. Comp. Phys.* 14, *341-349 (1974).*

IDEA: To solve large sparse eigenvalue problem,

$$\mathbb{H}c = E\mathbb{S}c$$

minimize the variance defined by

$$w(\lambda,\mathbb{v}) = \frac{\mathbb{v}^+(\mathbb{H}-\lambda\mathbb{S})^+(\mathbb{H}-\lambda\mathbb{S})\mathbb{v}}{\mathbb{v}^+\mathbb{S}\mathbb{v}}$$

with respect to $\mathbb{v}$, or with respect to both λ and $\mathbb{v}$. If λ is fixed, then applying the method of relaxation, we have for some arbitrary vector $\mathbb{r}$ and constant α

$$w(\lambda, \mathbb{v}+\alpha\mathbb{r}) = \frac{\rho_1(\alpha)}{\rho_2(\alpha)}$$

where $\rho_1(\alpha)$ and $\rho_2(\alpha)$ are polynomials of degree 2 in α, and their construction requires formation of the vectors $\mathbb{H}\mathbb{v}$, $\mathbb{H}\mathbb{r}$, $\mathbb{S}\mathbb{v}$, and $\mathbb{S}\mathbb{r}$, and scalar products between these. Iteration with fixed λ finds the eigenvalue E closest to λ, (i.e. $\min_E(\lambda-E)$). Possible application of this is:

1) Use Gershgorin disks to select a $\lambda < \min(E)$. Iteration will then converge to the *lowest* eigenvalue, which most other methods cannot guarantee.

2) Propagator, Green function, or equations-of-motion calculations often require determination of the poles of a matrix $\mathbb{G}(E)$ [or equivalently, the zeros of $\mathbb{G}^{-1}(E)$], where E is a variable parameter, and poles are required as a function of E. $\mathbb{G}^{-1}(E)$ is known, but it has a wide spectrum of eigenvalues. Setting $\lambda = 0$ would allow perhaps a rapid pole search as a function of E.

If λ is chosen as the Rayleigh quotient, $\mathbb{v}^+\mathbb{H}\mathbb{v}/\mathbb{v}^+\mathbb{S}\mathbb{v}$, then after some manipulations, the variance is obtained as

$$w\left(\lambda^{(\alpha)}, \mathbf{v} + \alpha \mathbf{r}\right) = \frac{\rho_6(\alpha)}{\rho_2^3(\alpha)}$$

where $\rho_6(\alpha)$ is obtained by analytic polynomial multiplication of quadratic polynomials.

Differentiation with respect to α gives

$$\frac{dw}{d\alpha} = 0 \Rightarrow \rho_7(\alpha) = \rho_2\rho_6' - 3\rho_6\rho_2^2\rho_2 = 0$$

and solution of the 7th degree polynomial equation gives seven α's, from which that giving a minimum in $w(\alpha)$ is chosen.

There are two disadvantages of this method:

1) Forming $\mathbb{H}\mathbf{r}$ and $\mathbb{S}\mathbf{r}$ requires full row of $\mathbb{H}$ and $\mathbb{S}$, so they must be stored as square matrices.

2) Preliminary experience indicates that convergence may be slow.

BIBLIOGRAPHY ON THE LARGE MATRIX EIGENVALUE PROBLEM IN QUANTUM CHEMISTRY AND IN RELATED FIELDS

A general review on the origin, characteristics, and structure of the large matrix is given in the chapter by *I.Shavitt*, "The Method of Configuration Interaction," in *MEST* (= Methods of Electronic Structure Theory), Volume 3 of the series *Modern Theoretical Chemistry*, *edited by H.F.Schaefer (Plenum Press, 1977), pp. 189-275.* Section 6 of this chapter reviews the matrix eigenvalue problem in quantum chemistry.

A primitive form of the coordinate relaxation method, based on setting one element of the residue vector to zero in each step, but with the actual modification of the trial vector deferred to the end of a complete iteraction (cycle of n steps), has been given by *S.F.Boys, Proc. Roy. Soc. A201, 125 (1950).*

A much improved form, based on applying each correction as soon as it is computed, similar to the method of *J.L.B.Cooper, Quart. Appl. Math. 6, 179 (1943)*, but with efficient continuous updating of all quantities, has been described by *R.K.Nesbet, J. Chem. Phys. 43, 311 (1965).*

A modification of Nesbet's method to allow the use of the rows of the lower triangle of the (symmetric) matrix in order has been described by *I.Shavitt, J. Comput. Phys. 6, 124 (1970).* The same approach is also applicable to some of the subsequent methods.

An extension of Nesbet's algorithm to real eigenvalues of nonsymmetric matrices has been given by *C.F.Bender and I.Shavitt, J. Comput. Phys. 6, 146 (1970).* An application of successive over-relaxation to this algorithm (also applicable to the original Nesbet algorithm) has been shown by *R.M.Nisbet, J. Comput. Phys. 10, 614 (1972).*

The coordinate relaxation scheme based on the minimization of the Rayleigh quotient, essentially as given by *D.K.Fadeev and V.N.Fadeeva, Computational Methods of Linear Algebra (Freeman & Co., 1963), Section G1*, has been described by

I.Shavitt, C.F.Bender, A.Pipano, and R.P.Hosteny, J. Comput. Phys. 11, *90 (1973).* This paper also shows how non-extremal eigenvalues can be obtained by "root shifting" [a modified form of the deflation method of *H.Hotelling, J. Educat. Psychol.* 24, *417 (1933)*] or by orthogonality constraints. Similar ground (and some additional aspects) was covered in the same year in the paper of *Z.Falk, Z. Angew. Math. Mech.* 53, *73 (1973)* where "group relaxations" are also discussed. A recent discussion of "group relaxations" (simultaneous relaxation of two or more components of the trial vector) has been given by *L.P.Cheung and D.M.Bishop, Comput. Phys. Commun.* 13, *247 (1978).*

The use of over-relaxation for these methods has been discussed by *H.R.Schwarz, Comput. Math. Appl. Mech. Eng.* 3, *11 (1974)*, and by *A.Ruhe, Math. Comp.* 28, *695 (1974)* (this last paper describes "convergent splitting" with over-relaxation).

A relaxation method based on variance minimization, capable of computing interior eigenvalues, has been described by *M.G.Feler, J. Comput. Phys.* 14, *341 (1974).*

The simultaneous coordinate relaxation method, developed independently by R.C.Raffenetti and I.Shavitt and discussed at this workshop by R.C.Raffenetti, has not been published yet.

The use of the Lanczos method in quantum chemistry and physics has been described by *H.Nissimov, Phys. Lett.* 46B, *1 (1973)*, and by *R.F.Hausman, C.F.Bender and S.D.Bloom, Chem. Phys. Letters* 32, *483 (1975).*

The method described at this workshop by Davidson, in which the Krylov sequence of the Lanczos method is replaced by a sequence derived from perturbation theory, is descirbed in *E.R.Davidson, J. Comput. Phys.* 17, *87 (1975).* Related perturbation theory approaches are described in the chapter by *B.O.Roos and P.E.M.Siegbahn in MEST* (see above), *page 277*, and in *R.Seeger, R.Krishnan, and J.A.Pople, J. Chem. Phys.* 68, *2519 (1978).* A root-homing version of Davidson algorithm (for obtaining an eigenvector "closest" to a particular trial vector) has been described by *W.Butscher and W.E.Kammer, J. Comput. Phys.* 20, *313 (1976).*

The simultaneous multi-root version of Davidson's algorithm, described at this workshop by B.Liu, has not yet been published.

Methods based on matrix partitioning (and perturbation theory) have been discussed by *P.O.Löwdin, J. Math. Phys.* 6, *1341 (1965); Z.Gershgorn and I.Shavitt,*

Int. J. Quantum Chem. 2, *751 (1968); S.Iwata and K.F.Freed, Chem. Phys.* 11, *433 (1975); G.A.Segal and R.W.Wetmore, Chem. Phys. Lett.* 32, *556 (1975);* and *L.E.Nitsche and E.R.Davidson, J. Chem. Phys.* 68, *3103 (1978).*

SOME REFERENCES FROM THE NUMERICAL ANALYSIS LITERATURE FOR COMPUTATIONAL CHEMISTS

Introductory texts which indicate the kind of analysis one does in the field of matrix computations:

G.E.Forsythe and C.B.Moler, Computer Solution of Linear Algebraic Equations (Prentice Hall, 1967).

G.W.Stewart, Introduction to Matrix Computations (Academic Press, 1973)

The standard treatise on $Ax = \lambda x$ with associated error analysis:

J.H.Wilkinson, The Algebraic Eigenvalue Problem (Oxford, 1965).

The most recent volume on sparse matrix computations:

J.Bunch and D.Rose, Sparse Matrix Computations (Academic Press, 1976).

Three papers in this book are of particular interest:

G.W.Stewart, "A bibliographical tour of the large sparse generalized eigenvalue problem"

W.Kahan and B.N.Parlett, "How far should you go with the Lanczos process?"

Cline, Golub, Platzman, "Calculation of normal modes of oceans using a Lanczos method".

A very extensive bibliography with Highland chit-chat:

Iain Duff, "A Survey of the Sparse Matrix Research," Proc. of the IEEE 65, *500 (1977).*

A paper which will tell you all about simultaneous iteration (i.e., $A^k Y_0$) is:

G.W.Stewart, "Simultaneous iteration for computing invariant subspaces of non-hermitian matrices," Numer. Math. 25, *123 (1976).*

The Lanczos algorithm and its several variants are discussed in the following papers:

B.N.Parlett and D.S.Scott, "The Lanczos Algorithm with Implicit Deflation," U.C.Berkeley, ERL Report UCB/ERL M77/70 (College of Engineering, 1977).

R.Underwood, "An Iterative Lanczos Method for the Solution of Large Sparse Symmetric Eigenproblems," Stanford CS Dept. Report CS-496 (1975).

J.Cullum and W.E.Donath, "A Block Lanczos Algorithm for Computing the q Algebraically Largest Eigenvalues and a Corresponding Eigenspace of Large, Sparse, Real, Symmetric Matrices," Proc. IEEE Conf. on Decision and Control, Phoenix, Arizona (1974).

J.Cullum, "The Simultaneous Computation of a few Algebraically Largest and Smallest Eigenvalues of a Large Sparse Symmetric Matrix," Report RC-6827 (1977), from IBM, Yorktown Heights 10598.

J.Cullum and W.E.Donath, "A Block Generalization of the Symmetric S-Step Lanczos Algorithm," IBM Watson Research Center Report RC-4845 (1974).

C.C.Paige, "Practical Use of the Symmetric Lanczos Process with Reorthogonalization," BIT 10, 183 (1970).

G.H.Golub, "Some Uses of the Lanczos Algorithm in Numerical Linear Algebra," in Topics in Numerical Analysis, J.Muller (editor) (Academic Press, 1974).

C.C.Paige, "Computational Variants of the Lanczos Method for the Eigenproblem," J. Inst. Math. & Applic. 10, 373 (1972).

BIBLIOGRAPHY ON THE GENERALIZED EIGENVALUE PROBLEM $\underline{\underline{A}}\underline{x} = \lambda\underline{\underline{B}}\underline{x}$

1. *J.H.Wilkinson and C.Reinsch, Handbook for Automatic Computation, Vol. II: Linear Algebra (Springer-Verlag, Berlin, 1971), pp. 196-197, 303-314. [prepublished in Num. Math. 11, 99 (1968)].*

2. *G.Peters and J.H.Wilkinson, "$\underline{\underline{A}}\underline{x} = \lambda\underline{\underline{B}}\underline{x}$ and the generalized eigenproblem," SIAM J. Numer. Anal. 7, 479 (1970).*

3. *G.W.Stewart, "On the sensitivity of the eigenvalue problem $Ax = \lambda Bx$," SIAM J. Numer. Anal. 9, 669 (1972).*

4. *G.Fix and R.Heiberger, "An algorithm for the ill-conditioned generalized eigenvalue problem," SIAM J. Numer. Anal. 9, 78 (1972).*

5. *C.B.Moler and G.W.Stewart, "An algorithm for generalized matrix eigenvalue problems," SIAM J. Numer. Anal. 10, 241 (1973)* (Describes the QZ method.)

6. *L.Kaufman, "The LZ-algorithm to solve the generalized eigenvalue problem," SIAM J. Numer. Anal. 11, 997 (1974).*

7. *B.Ford and G.Hall, "The generalized eigenvalue problem in quantum chemistry," Comp. Phys. Comm. 8, 337 (1974).*

8. *R.C.Ward, "The combination shift QZ algorithm," SIAM J. Numer. Anal. 12, 835 (1975).*

9. *L.Kaufman, "Algorithm 496 -- The LZ algorithm to solve the generalized eigenvalue problem for complex matrices (F2)," ACM Trans. Math. Software 1, 271 (1975).*

10. *C.R.Crawford, "A stable generalized eigenvalue problem," SIAM J. Numer. Anal. 13, 854 (1976),* (for banded symmetric matrices).

11. *C.B.Moler and G.W.Stewart, "An Algorithm for the Generalized Eigenvalue Problem," Stanford University Computer Science Dept. Report STAN-CS-232-71 (August 1971).* (Same as Ref. 5, but contains additionally a listing of the QZ FORTRAN program.)

12. *L.Kaufman, "A Generalization of the LR Algorithm to Solve* $\underline{\underline{A}}\underline{x} = \lambda \underline{\underline{B}}\underline{x}$*," Stanford Computer Science Report STAN-CS-72-276 (April 1972).* (Contains preliminary version of FORTRAN program in Ref. 9.)

13. *L.Kaufman, "The LZ Algorithm to Solve the Generalized Eigenvalue Problem," STAN-CS-73-363 (May 1973).* (More extended version of Ref. 6, and intermediate version of FORTRAN program in Ref. 9.)

14. *G.W.Stewart, "Gershgorin Theory for the Generalized Eigenvalue Problem* $Ax = \lambda Bx$*," Math. Comp. 29, 130 (1975).*

15. *G.H.Golub, R.Underwood and J.H.Wilkinson, "The Lanczos Algorithm for the Symmetric* $\underline{\underline{A}}\underline{x} = \lambda \underline{\underline{B}}\underline{x}$ *Problem," STAN-CS-72-270 (March 1972).*

16. International Mathematical and Statistical Library (IMSL) subroutines EIGZF, EQZQF, EQZTF, EQZVF, UERTST, VHSH2C, VHSH2R, VHSH3R (March 1975).

17. *B.T.Smith, J.M.Boyle, J.J.Dongarra, B.S.Garbow, Y.Ikebe, V.C.Klema, and C.B.Moler, Lecture Notes in Computer Science, Vol. 6: Matrix Eigensystem Routines - EISPACK Guide, second edition, edited by G.Goos and J.Hartmanis (Springer-Verlag, Berlin, 1976).* (FORTRAN versions of the ALGOL procedures in Ref. 1. Listings are given and magnetic tape copies may be ordered.)

18. *B.S.Garbow, J.J.Dongarra, C.B.Moler, and B.T.Smith, Lecture Notes in Computer Science, Vol. 51: Matrix Eigensystem Routines - EISPACK Guide Extension (Springer-Verlag, 1977)* (Supplement to Vol. 6, second edition, includes QZ and SVD.)

REFLECTIONS ON THE NRCC CONFERENCE

B. N. Parlett
University of California, Berkeley

My first impression was that chemists who have specialized in the eigenvalue problem and linear systems are fairly well acquainted with matrix computations and the numerical analysts did not have much to offer them. My second impression is that the workshop may produce some valuable insights in the near future when the numerical analysts have absorbed two rather surprising facts which emerged unstated from the discussions.

The first fact is that the typical matrix H which arises does not have a sparsity structure that can be readily exploited. (Most numerical analysts are brought up on two- and three-dimensional elliptic problems which have lots of nice properties.) The presence of 10^7 nonzero elements when $n = 10^4$ came as quite a shock.

The second fact was less apparent and more interesting. It appears that H always has a strong diagonal (thanks to Hartree-Fock), i.e. the matrix of normalized eigenvectors is diagonally dominant, and so several numerical methods which are, in general, very poor, appear to be quite satisfactory for the problems under discussion. This "strong diagonal" property makes H *very nice* and the challenge to the analysts is to exploit it to the hilt. I do not believe that the current methods are close to achieving that goal.

I wrote recently in *Progress in Numerical Analysis* that I expected numerical analysis to break up into several almost autonomous disciplines. Rather than solving more and more *general* problems, the development of the field will lie in more and more specialized applications. I see the NRCC workshop as confirming this prediction to a considerable extent. The chemists will not be well advised to borrow the "best" methods for the general large, sparse, symmetric problems, but will be well advised to consult with the experts on how to build their own codes. Proliferation lies ahead whether we want it or not.

RECOMMENDATIONS FOR WORK BY NRCC

1. It would be useful if NRCC acquires the most important programs for the types of problems discussed here, from both chemical and numerical analyst sources (such as EISPACK), and makes these accessible in a fairly compatible form (as far as feasible) to facilitate utilization, comparison, and testing.

2. It would be useful if NRCC acquires a set of representative matrices of various sizes (and somewhat varying characteristics, if feasible) for use in testing, comparison, and analysis. The documentation should include available answers and experience (such as convergence rates) with current methods.

3. NRCC should acquire, on a continuing basis, reprints, preprints, and unpublished reports on numerical methods from both chemical and numerical analyst sources, and make bibliographies of these available. Help in obtaining copies of unpublished material will be quite useful.

4. NRCC should solicit proposals from numerical analysts for work on developing and testing numerical methods useful in chemistry.

SUMMARY OF DISCUSSION ON THE SOLUTION OF LARGE LINEAR SYSTEMS

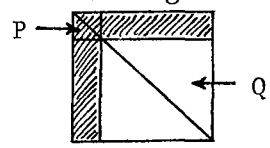
I. S. Duff
A.E.R.E., Harwell, England

Definition of Problem

1. It is not possible to store original matrix in core.

2. Order of system is 1000 - 10000.

3. Problems usually have the following structure



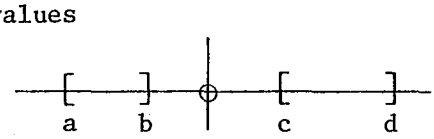
where border is fairly dense and Q is of order N^4 with typically N^6 non-zeros. P is typically of order 1 to 50. Although Q has large diagonal elements and is usually symmetric positive definite, this does not hold for the matrix as a whole. It is not clear that any advantage can be taken of structure in Q.

A few problems have block tri-diagonal structure.

4. Can be unsymmetric although a very main class is symmetric, but not positive definite.

5. The number of negative eigenvalues is small (commonly 10-20), and thus are associated with small leading submatrix P of number 3 above.

6. Distribution of eigenvalues



$d/c = 1000$

with eigenvalues uniformly distributed in [c,d].

7. Main storage is assumed large enough to hold at least 5 vectors of length n.

8. Often we wish to solve for about 5-20 vectors simultaneously.

Comments on Numerical Methods

1. Conjugate gradients only efficient and guaranteed for positive semi-definite symmetric matrix.

2. SYMMLQ available to problem at hand but in view of suggested spectrum in

number 6, it may not be very efficient. In fact, it can be $O(n^3)$ on each solution. For example, in number 6,

$$\left(\sqrt{ad/bc} - 1 \Big/ \sqrt{ad/bc} + 1\right)^{\#\text{ iterations}} < \varepsilon$$

$$\left(\sqrt{d/c} - 1 \Big/ \sqrt{d/c} + 1\right)^{\#\text{ iter.} - \#\text{neg. eig'values}} < \varepsilon$$

So we can estimate the number of iterations since an estimate of c can be obtained from Lanczos process.

The convergence is better if the eigenvalues are clustered rather than uniformly distributed. One way of doing this is to precondition the matrix (see below).

Since we are often solving for multiple RHS, the development and use of a block Lanczos scheme was suggested.

See presentation by Cline for description of SYMMLQ.

<u>Preconditioning</u>. If A_o^{-1} is an approximation to A^{-1}, then $A_o^{-1}A$ should have many eigenvalues clustered near 1. Thus, if we use C.G. or SYMMLQ on $A_o^{-1}A$, convergence should be rapid. To be a useful technique the inverse of A_o must be easy to calculate (see also presentation by Widlund).

For the present example, a possible preconditioning matrix might be

$$\begin{pmatrix} D^{-1} & 0 \\ -C^T D^{-1} & I \end{pmatrix}$$

where D is diagonal of matrix Q in number 3, and that matrix has been reordered

$$\begin{pmatrix} Q & C \\ C^T & P \end{pmatrix} .$$

It was pointed out that information on the eigenspectrum of the matrix could be obtained cheaply by using Davidson's second method on $\begin{pmatrix} I & C \\ C^T & P \end{pmatrix}$.

Direct

1. The danger of just using pivot size (or change of size in diagonal elements) as a numerical stability test when using Choleski on non-positive definite systems was noted.

2. The possibility of pivoting when using column access was pointed out and the use of 2×2 pivots if Choleski broke down was mentioned.

3. If we order the matrix to put the large positive principal minor first (Q), then two things are possible: (i) advantage may possibly be taken of sparseness in Q; (ii) we need not pivot on Q and can do necessary pivoting on the indefinite modified P in core with full numerical stability ensured for that part of the computation.

4. The possibility of using a direct method with drop tolerances coupled with iterative refinement or other iterative methods was discussed.

Recommendations for Work by NRCC

1. Examination of use of SYMMLQ on typical problems and research into preconditioning techniques to accelerate such methods.

2. Investigation into the efficient handling of large amounts of data out of core. Especially to see if I/O can be performed simultaneously with arithmetic to reduce expansion factor (elapsed time/CPU time) to near 1.

3. Further investigation of Choleski's method to see: (i) why little instability arises in practice on typical problems; (ii) if schemes based on processing the sparse part first (see 3 above) are beneficial.

BIBLIOGRAPHY

1. General survey of sparse matrix techniques (mainly direct methods) and extensive bibliography. *I.S.Duff, "A survey of sparse matrix research," Proc. IEEE 65 (1977) p.535.*

2. Conference proceedings. *J.Bunch and D.Rose, "Sparse matrix computations," (Academic Press, 1976).*

3. *I.S.Duff and G.W.Stewart, Proceedings of Conf. at Oak Ridge, November 1978. (SIAM, to be published).*

4. Tutorial conference on sparse matrix methods: *V.A.Barker, "Direct and iterative methods for eigenvalue problems (Springer Verlag, 1976).*

5. Orderings on symmetric matrices: *J.A.George and D.R.McIntyre, "On the application of the minimum degree algorithm to finite element systems,"*

(to appear in SIAM J. Numr. Anal., 1978).

6. *C.C.Paige and M.A.Saunders, "Solution of sparse indefinite systems of equations and least squares problems," SIAM J. Numer. Anal. (1975).*

7. LINPACK (routines for in-core linear equation solver): *J.J.Dongarra, J.R.Bunch, C.Moler, G.W.Stewart, LINPACK User's Guide, preliminary edition, Applied Math Division, Argonne National Laboratory. (Permanent edition: SIAM Publications, in preparation.)*

LINEAR EQUATION SYSTEMS IN BOUND STATE AND SCATTERING PROBLEMS*

R. K. Nesbet

IBM Research Laboratory
San Jose, California 95193

ABSTRACT:

The matrix expression $m^{+}(h-\varepsilon)^{-1}m$, where h is a large Hermitian matrix, occurs in linear expansion methods for bound state or scattering solutions of Schrödinger's equation. Applications include partitioning methods for the matrix eigenvalue problem, variational methods in electron scattering theory, and the Stieltjes imaging theory of oscillator strength distributions and photoionization. Examples of such applications will be given. Methods for evaluation of the indicated matrix expression that avoids direct inversion of $h-\varepsilon$ include direct diagonalization and a modified Cholesky factorization. These methods will be discussed.

I. AREAS OF APPLICATION

A. Large Matrix Eigenvalue Problem Partitioning

In a matrix representation of the Schrodinger equation

$$(H-E)\Psi = 0 \tag{1}$$

it is often convenient to separate the basis into two segments, such that the smaller segment consists of a few functions chosen to dominate the wave function of interest, while the residual segment contains a much larger number of functions whose individual influence is small. In molecular spectroscopy, such a basis separation, which corresponds to partitioning of the matrix of H, is sometimes referred to as a Van Vleck transformation [*J.H.Van Vleck, Phys. Rev.* 33, *467 (1929)* for example]. The formalism of partitioning has been extensively developed by Löwdin [*P.O.Löwdin, J. Chem. Phys.* 19, *1396 (1951); in Perturbation Theory and*

Applications, ed. Wilcox (Wiley, New York, 1966)] in terms of projection operators.

If the small segment of the basis space is defined by a projection operator P, the residual space corresponds to the orthogonal complement projection operator Q, where these operators have the properties

$$P^2 = P \quad , \qquad Q^2 = Q \ , \qquad P + Q = I \quad . \tag{2}$$

The resulting partitioned form of Eq. (1) is

$$\begin{cases} (H_{PP} - E)\Psi_P + H_{PQ}\Psi_Q = 0 \\ \\ H_{QP}\Psi_P + (H_{QQ} - E)\Psi_Q = 0 \end{cases} \tag{3}$$

The second line of Eq. (3) has the formal solution

$$\Psi_Q = -(H_{QQ} - E)^{-1}\Psi_P \quad . \tag{4}$$

When this is substituted into the first line of Eq. (3), the resulting equation is

$$\left\{(H_{PP} - E) - H_{PQ}(H_{QQ} - E)^{-1} H_{QP}\right\}\underline{\Psi}_P = 0 \tag{5}$$

This is of the form of an effective Hamiltonian operator acting in the P-space. The second term in Eq. (5) is of the general form of a Green's function, expressed in a finite basis representation.

The practical use of Eq. (5) is to introduce approximations that simplify the Green's function term. In this way, the Q-space (of very large dimension) is formally eliminated in favor of an effective operator in the P-space (of small dimension), in which an exact solution of the reduced problem can be carried out.

When used as a formal method for the large matrix eigenvalue problem, the idea is to estimate E, construct the Green's function term, then solve exactly in the P-space. The estimate of E is then updated, and the method is iterated to convergence. Approximations can be introduced in evaluating the Green's function term

$$\Delta H_{PP} = - H_{PQ}\,(H_{QQ} - E)^{-1}\,H_{QP} \quad , \tag{6}$$

such as assuming $E \cong E_o$ to avoid iteration, or neglect making a diagonal approximation to at least the remote portions of H_{QQ}. Such approximations can be systematized by use of various forms of perturbation theory.

Unless a diagonal approximation can be justified, partitioning requires an efficient method for evaluation of the contracted form indicated in Eq. (6). Here H_{QQ} is considered as a real symmetrix matrix of very large dimension (too large for core memory in a computer), H_{QP} is a rectangular matrix with one very large and one small dimension, while ΔH_{PP} is small.

B. Electron-Atom Scattering, Variational Theory

In electron scattering by an N-electron atom or molecule, an approximate solution of Eq. (1) is sought with Ψ of the form

$$\Psi = \sum_p \mathcal{A}\, \Theta_p \psi_p + \sum_\mu \Phi_\mu c_\mu \quad , \tag{7}$$

$$= \Psi_P + \Psi_Q \quad . \tag{8}$$

A quadratically integrable component Ψ_Q is partitional from a term Ψ_P that contains specific continuum orbital wave functions

$$\psi_p = f_p(r)\, Y_{\ell m}(\theta,\phi)\, U_{m_s} \quad , \tag{9}$$

such that for open scattering channels,

$$\tfrac{1}{2}k_p^2 = E - E_p > 0 \tag{10}$$

the asymptotic radial channel orbitals are of the form

$$f_p(r) \sim k_p^{-\frac{1}{2}}\, r^{-1} \left[\sin(k_p r - \tfrac{1}{2}\ell_p\pi)\alpha_{0p} + \cos(k_p r - \tfrac{1}{2}\ell_p\pi)\alpha_{1p} \right] \tag{11}$$

In Eq. (7), $\mathcal{A}$ is an antisymmetrizing operator, Θ_p is an N-electron target atom or molecule wave function, and $\{\Phi_\mu\}$ are a set of Slater determinants or configuration state functions for the N+1-electron system. In Eq. (10), $\hbar k_p$ is the momentum of the scattered electron, E is the total energy, and E_p is the energy corresponding to target state Θ_p. The asymptotic form indicated in Eq. (11) is appropriate to scattering by a neutral atom.

It must be modified for a Coulomb or static electric dipole potential.

The multichannel variational method of Kohn [*W.Kohn, Phys. Rev.* 24, *1763 (1948)*] can be applied in a form that makes use of the partitioning method, introduced in scattering theory by Feshbach [*H.Feshbach, Ann. Phys. (N.Y.)* 5, *357 (1958);* 19, *287 (1962)*]. Formal developments and applications have been reviewed recently [*R.K.Nesbet, Adv. At. Mol. Phys.* 13, *315 (1978)*].

The variational functional matrix is

$$\Xi_{st} = (\Psi_s | H - E | \Psi_t) = \sum_{ij} \sum_{pq} \alpha^*_{ips} \; m^{pq}_{ij} \; \alpha_{jqt} \tag{12}$$

The indices s and t refer to various independent degenerate solutions of Eq. (1) at given energy E. The number of such solutions is equal to the number of open scattering channels at E. In Eq. (12), the matrix m is

$$m^{pq}_{ij} = \left(\mathcal{A}\psi_{pi}\Theta_p \,|\, H'_{PP} - E \,|\, \mathcal{A}\Theta_q\psi_{qj} \right) \tag{13}$$

where the effective Hamiltonian acting on Ψ_P, obtained by partitioning as in Eqs. (3), is

$$H'_{PP} = H_{PP} - H_{PQ}(H - E)^{-1}_{QQ} H_{QP} \quad . \tag{14}$$

The matrix m^{pq}_{ij} is of dimension $2N_c \times 2N_c$, where N_c is the number of open channels. Indices i,j have values 0,1 only, referring respectively, to the sin and cos components of Eq. (11). The specific form of Eq. (13) is

$$m^{pq}_{ij} = M^{pq}_{ij} - \sum_{\mu} \sum_{\nu} M_{ip,\mu} (M^{-1})_{\mu\nu} M_{\nu,jq} \tag{15}$$

where M denotes H - E. The second term here is a generalized matrix optical potential, of the standard Green's function form indicated in Eq. (6). The bound-bound matrix, $M_{\mu\nu}$ is in general very large (symmetric), while m^{pq}_{ij} is small, and the bound-free matrix $M_{\nu,jq}$ is rectangular. In scattering theory, the free-free matrix M^{pq}_{ij} is not symmetric and its asymmetry carries over into m^{pq}_{ij}.

Because of the large size of $M_{\mu\nu}$, most of the work of a variational scattering calculation is involved in evaluating Eq. (15). The subsequent

variational calculation provides an approximate solution of the matrix equations

$$m\alpha \cong 0 \tag{16}$$

where m is the matrix m_{ij}^{pq}. A convenient matrix notation, retaining indices ij but suppressing channel indices is

$$\begin{bmatrix} m_{00} & m_{01} \\ m_{10} & m_{11} \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \end{bmatrix} \cong 0 \tag{17}$$

where

$$\begin{aligned} \alpha_0 &= \{\alpha_{0ps}\} \\ \alpha_1 &= \{\alpha_{1ps}\} \end{aligned} \tag{18}$$

are matrices containing the coefficients of the asymptotic wave function indicated in Eq. (11), with an additional index s to denote one of the N_c independent degenerate solutions at given energy E. The submatrices have the properties

$$m_{00}^{+} = m_{00} \ , \qquad m_{11}^{+} = m_{11} \ , \qquad m_{01} - m_{10}^{+} = \tfrac{1}{2} I \ , \tag{19}$$

where $(^{+})$ denotes a matrix transposed for real matrices. In this notation the variational functional is

$$\Xi = \alpha^{+} m \alpha \alpha \ . \tag{20}$$

Scattering cross sections are computed directly from elements of the scattering matrix

$$S = (I + iK)(I - iK)^{-1} \ , \tag{21}$$

defined in terms of the real symmetric open-channel reactance matrix

$$K = \alpha_1 \alpha_0^{-1} \ , \tag{22}$$

where the matrices α_i are defined by Eqs. (18).

In the Kohn variational method, α_0 is taken to be a unit matrix, so that

$$\alpha_0 = I \ , \qquad \alpha_1 = K \tag{23}$$

and α_1 is replaced by a matrix [K[stationary with respect to

$$\delta\alpha_0 = 0 \quad , \qquad \delta\alpha_1 = \delta K \quad . \tag{24}$$

The variation of Ξ is

$$\delta\Xi = \delta K^{+}(m_{10} + m_{11} K) + (m_{10} + m_{11} K)^{+} \delta K + \tfrac{1}{2}\, \delta K \tag{25}$$

The Kohn functional

$$[K] = K_t = 2\Xi(K_t) \tag{26}$$

is stationary as a result of Eq. (25) if

$$K_t = -m_{11}^{-1}\, m_{10} \quad , \tag{27}$$

and its value is

$$[K] = -2(m_{00} - m_{10}^{+}\, m_{11}^{-1}\, m_{10}) \quad . \tag{28}$$

Recent developments of this formalism have made it possible to avoid the spurious singularities arising from Eq. (28) when m_{11} has zero eigenvalues.

Quite refined calculations of electron-atom scattering cross sections have been carried out with the variational method. A recent example is the complicated resonance structure in the range 22.4 eV - 23.1 eV for e^--He scattering [*R.K.Nesbet, J. Phys. B11, L21 (1978)*]. The theoretical calculations reproduced details of energy-dependent structure seen in a recent high resolution experiment [*J.N.H.Brunt, G.C.King, and F.H.Read, J. Phys. B10, 433 (1977)*], but could analyze the structures into specific contributions of superimposed resonances due to short-lived states of He^-. Because only the total metastable excitation cross section was observed, such analysis could not be carried out unambiguously on the experimental data by itself.

C. Moment Theory, Stieltjes Imaging

Langhoff [*P.W.Langhoff, Chem. Phys. Lett. 22, 60 (1973); P.W.Langhoff, C.T.Corcoran, J.S.Sims, F.Weinhold, and R.M.Glover, Phys. Rev. A14, 1042 (1976)*] has used moment theory to develop a Stieltjes imaging method for computing the oscillator strength distribution function for an atom or

molecule. Although only quadratically integrable electronic wave functions are used, the method makes possible the computation of the continuous photoionization cross section. Practical difficulties with the original version of the method have been discussed and to a large extent resolved [*R.K.Nesbet, Phys. Rev. A14, 1065 (1976)*], and applications to molecular photoionization have been published [*T.N.Rescigno, C.F.Bender, B.V.McKoy, and P.W.Langhoff, J. Chem. Phys. 68, 970 (1978); J.Barsuhn and R.K.Nesbet, J. Chem. Phys. 68, 2783 (1978)*].

In a discrete basis representation of wave functions dipole-coupled to a reference state Ψ_0, the frequency-dependent polarizability is given by

$$\alpha(\omega) = \sum_{i=1}^{N} \frac{f_i}{\varepsilon_i^2 - \omega^2} \quad . \tag{29}$$

This expression has nonphysical poles for real values of $\hbar\omega$ above the ionization threshold, where $\alpha(\omega)$ should be a continuous complex function whose imaginary part is proportional to the total photoionization cross section. To avoid these poles, it is convenient to consider the inverse energy moments of the oscillator strength distribution, defined by

$$\mu_k = \sum_{i=1}^{N} f_i \, \varepsilon_i^{-k} \quad , \quad k << N_B , \tag{30}$$

where N_B is the number of basis functions. The computed oscillator strengths f_i and excitation energies ε_i vary irregularly with the choice of basis, but the moments are nonsingular physical quantities and have well-defined limiting values.

The polarizability is defined as a function of a complex frequency z by the Stieltjes integral

$$\alpha(z) = \int_0^\infty \frac{df(\varepsilon)}{\varepsilon^2 - z^2} \quad , \tag{31}$$

where $f(\varepsilon)$ represents both the discrete and continuous portions of the oscillator strength distribution function. In the Stieltjes imaging method, Eq. (31) is approximated by a generalized Gauss quadrature formula

$$\alpha(z) \cong \sum_{a=1}^{n} \frac{f_a}{\varepsilon_a^2 - z^2} \quad , \tag{32}$$

valid for points z not on the positive real axis. Here the quadrature points are ε_a and the weights are f_a, defined so that the first 2n moments

$$\mu_k = \int_0^\infty \varepsilon^{-k}\, df(\varepsilon) = \sum_{a=1}^{n} f_a\, \varepsilon_a^{-k} \quad , \qquad k = 0, \ldots, 2n-1 \tag{33}$$

are given exactly by the quadrature formula. The set of values $\{\varepsilon_a, f_a\}$ defines a <u>principal representation</u> of the oscillator strength distribution.

The cumulative oscillator strength function

$$F(\varepsilon) = \int_0 df \tag{34}$$

is defined so that the oscillator strength distribution is

$$df(\varepsilon) = \frac{dF}{d\varepsilon}\, d\varepsilon = g(\varepsilon) d\varepsilon \quad . \tag{35}$$

Above the ionization threshold, the total photoionization cross section is

$$\sigma_{PI}(\varepsilon) = 2\pi^2\, \alpha g(\varepsilon) \quad , \tag{36}$$

if all quantities are in atomic units. The principal representation approximation to $F(\varepsilon)$ is the histogram

$$f(\varepsilon) = \sum_{\varepsilon_a < \varepsilon} f_a \quad . \tag{37}$$

Given a principal representation, the histogram is fitted by a smooth curve (passing through the midpoint of each vertical rise), whose slope gives $g(\varepsilon)$ and hence $\sigma_{PI}(\varepsilon)$.

For complex atoms, the oscillator strength for transition from state $\Psi_o(L_o)$ to state $\Psi_1(L_1)$ is given by

$$f_i = C(L_0 L_1)\, d_i \varepsilon_i d_i \tag{38}$$

where

$$\varepsilon_i = (h - \varepsilon_o)_{ii} \quad , \tag{39}$$

d_i is a transition moment, and C is an angular momentum coupling coefficient. The corresponding expression for the moment μ_k is

$$\mu_k = \sum_{L_i} C(L_0 L_1) \sum_i \sum_j d_i \left[(h - \varepsilon_o)^{-k+1}\right]_{ij} d_j \tag{40}$$

expressed as a contracted form involving the inverse of a large matrix. This form requires computational methods similar to those needed for Eqs. (6) and (15).

II. METHODS FOR $m^+(h - \varepsilon)^{-1} m$

A. Diagonalization of h

This is the most straightforward method, but inefficient or impossible for large matrices, because all n eigenvalues are required. The matrix h is real and symmetric but not, in general, sparse. If results are needed for many ε values, diagonalization is advantageous because it has only to be carried out once.

B. Modified Cholesky Method [*R.K.Nesbet, J. Comput. Phys. 8, 483 (1971)*]

Although $h-\varepsilon$ is not positive definite, it can be represented in the form

$$h - \varepsilon = t \sigma t^+ \tag{41}$$

where t is a lower triangle and σ is a diagonal matrix whose elements are either +1 or -1. If m is a rectangular matrix and

$$b = t^{-1} m \tag{42}$$

then

$$m^+(h-\varepsilon)^{-1} m = b^+ \sigma b \quad . \tag{43}$$

Use of Eq. (43) eliminates the back-substitution step that would be required if $(h-\varepsilon)^{-1} m$ were evaluated directly. Algorithms including the data-handling necessary for large matrices have been described in the indicated publication and implemented for electron-atom scattering calculations. In practice the method is numerically stable.

C. Computation of Moments

The modified Cholesky method can also be used to compute moments, but back-substitution cannot be avoided. To evaluate

$$\mu_k = C\, d^+ (h - \varepsilon)^{-k}\, d \tag{44}$$

making use of Eq. (41), the steps are

$$\begin{aligned} b_0 &= d \\ b_1 &= t^{-1}\, b_0 \\ b_2 &= (T^+)^{-1}\, \sigma\, b_1 \quad . \end{aligned} \tag{45}$$

Then

$$\begin{aligned} d^+(h-\varepsilon)^{-k}\, d &= b_k^+\, \sigma\, b_k \qquad k \text{ odd} , \\ &= b_k^+\, b_k \qquad k \text{ even} . \end{aligned} \tag{46}$$

III. QUESTIONS AND PROBLEMS

Can sparse matrix methods be used for matrices that are not strictly sparse?

Can an n^3 process (such as the modified Cholesky factorization) be avoided?

The optimal method would be an n^2 iterative procedure, valid for $h-\varepsilon$ not positive definite, that would converge in some number of steps much less than n.

0 0 0 0 5 1 0 /-899 8 2

DIRECT METHODS FOR SOLUTION OF $A\underset{\sim}{x} = \underset{\sim}{b}$

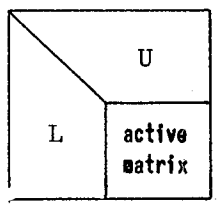
I. S. Duff
Harwell, England

A is general sparse matrix.

Gaussian elimination to find permutations P and Q, and triangular factors L and U

$$PAQ = LU$$

At intermediate stage we have



Basic operation:

$$a_{ij} \leftarrow a_{ij} - \underbrace{a_{ik}[a_{kk}]^{-1}}_{\text{multiplier}} a_{kj}$$

or by rows,

$$\text{row } i \leftarrow \text{row } i - \text{multiplier} \times \text{pivot row} .$$

x	x	x	x
x	o	x	o

<u>Fill-in</u>:

x	x	o	x	o
x	o	x	x	o
o	x	o	o	x
o	x	x	o	o
x	x	x	x	x

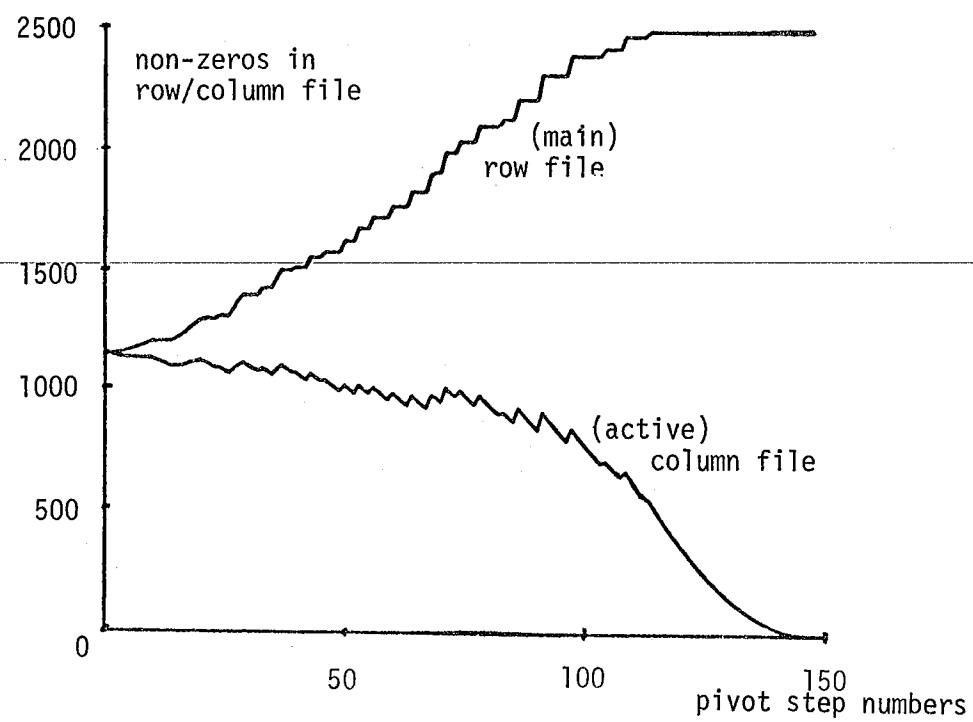
<u>x</u>	x	o	x	o
x	x	x	x	o
o	x	o	o	x
o	x	x	o	o
x	x	o	x	x

<u>x</u>	x	o	x	o
x	<u>x</u>	x	x	o
o	x	<u>x</u>	x	x
o	x	x	x	x
x	x	x	x	x

Extreme Example of Fill-in

```
* x x x x x x x x x x
x x
x   x
x     x
x       x
x         x
x           x
x             x
x               x
x                 x
x                   x
```

Use of * as pivot gives total fill-in and subsequent active matrices are full. Pivoting on black diagonal elements gives no change to the initial structure and operations and storage are both O(n).



This figure shows a fairly typical decay in the number of non-zeroes in the active matrix and a fairly steady growth in the number in the L/U factors, the final number being about twice the initial number of non-zeros. Note the plateau

at the end of the row file curve; after 118 pivots the active matrix is full, so no more fill-in can take place.

One major difference from full matrices is illustrated below:

1) Solve $A\underline{x} = \underline{b}$

2) Solve $A_1\underline{x} = \underline{b}$

where A_1 has the same sparsity structure as A.

3) Solve $A\underline{x} = \underline{b}_1$

---

(1) $PAQ \longrightarrow LU$ analyze

(2) $A_1 \xrightarrow[LU]{P,Q} L_1U_1$ factor

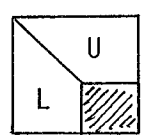
(3) $\underset{\sim}{b} \xrightarrow[LU/L_1U_1]{P,Q} \underset{\sim}{x}$ operate

Ratio of Times:	(1)	:	(2)	:	(3)		
Full case:	n	:	n	:	1	=	$0(n^2)$
Sparse case:	50	:	10	:	1	=	$0(n)$

Pivoting, For Numerical Accuracy

Total or complete pivoting

$$|a_{kk}^{(k)}| = \max_{i,j \geq k} |a_{ij}^{(k)}|$$



i.e., largest element in active matrix chosen at each stage.

1) Expensive, so not often used even in full case.

2) No attempt made to maintain initial sparsity.

Partial pivoting

$$a_{kk}^{(k)} = \max_{i \geq k} |a_{ki}^{(k)}| \quad \text{or} \quad \max_{i \geq k} |a_{ik}^{(k)}|$$

1) Can be unstable but used often in full case.

2) Still severely restricts maintenance of sparsity.

Threshold pivoting

$$|a_{kk}^{(k)}| \geqslant u. \max_{i \geqslant k} |a_{ki}^{(k)}|$$

where $u \in [0,1]$.

We can now maintain a balance between partial pivoting ($u = 1$) and ignore numerical stability entirely ($u = \varepsilon$).

Markowitz Criterion [*1957*]

$$\min_{\substack{i,j \\ a_{ij} \neq 0}} (r_i - 1) \quad (c_j - 1)$$

*	x	x	x
x	•	•	•
x	•	•	•

So we are: 1) minimizing the maximum possible fill-in, and 2) minimizing the number of $*,+$ operations.

Also proposed are:

$$\min_{\substack{i \\ a_{ij} \neq 0}} r_i \quad (\min_j c_j) \qquad \text{or vice versa}$$

restricted *MARKOWITZ* when we do not search for the absolute minimum.

Orderings to Preserve Sparsity

We find that if we a priori restrict our search to a specified row/column of the active matrix, then we do not maintain sparsity sufficiently on general systems. So we choose from among all the non-zeros in the active matrix.

Two of the commonest choices are:

1) Choose $a_{ij}^{(k)}$ (satisfying stability criterion) such that fill-in is minimized;
2) If r_i is the number of non-zeros in row i, and c_j is the number of non-zeros in column j, choose $a_{ij}^{(k)}$ (satisfying stability criterion) such that $(r_i - 1)\cdot(c_j - 1)$ is minimized.

Number 1) is rejected because extra cost in implementation is not matched by gains in sparsity.

Stability of Proposed Pivoting Method

If we say that the L,U factors produced are the exact factors of a perturbation to the original matrix, i.e.

$$A + E = L\,U$$

(we ignore the irrelevant permutations here), then we wish to find a bound on the elements of E.

The elements of E can be shown to depend on:

1) ε (MACHEPS ... dependent on machine)
2) Number of multipliciations/divisions on any one position... $\leqslant \min(i,j)$
3) Maximum element occurring in active matrix.

Only number 3) need cause us any concern.

We can control the size of the largest element in active matrices (here the original matrix is assumed to be well scaled) by adjusting the value of u. Indeed, at any one stage, growth is bounded by

$$(1 + 1/u) \quad ,$$

so that the maximum growth occurring in any one position is bounded by

$$(1 + 1/u)^{\#\text{ operations}} \quad .$$

This is, of course, a gross upper bound.

We can, of course, monitor the growth either by examining the size of each new non-zero created or by an a posteriori bound based on the factors L and U. We do the latter because of the overheads involved with the former, particularly in the factor entry.

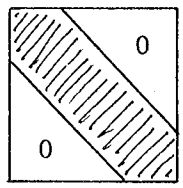
Amount of searching to find pivot.

Order of Matrix	147	57	292
Number of non-zeroes	2449	281	2208
Average length of search	8.2	2.8	3.1

Pre-Ordering Techniques

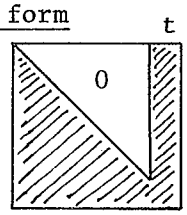
The basic idea with these techniques is to find an ordering of the rows and columns of the matrix so that the non-zero elements are confined to certain well defined regions of the matrix. It is then normal to arrange that the solution process preserves this partitioning of the matrix. Tewarson [*1973*] lists "desirable forms". We examine some of these in this lecture.

1) Band matrix



If there are some zeros within the band including its edges, we have a variable band structure. Some basic techniques for obtaining this form are discussed by Cuthill in *Rose and Willoughby (1972)*.

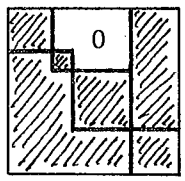
2) Bordered triangular form



Algorithms for obtaining this form are based on the notion of a minimum essential set. An essential set is a set of vertices whose removal (plus their incident edges) which leaves the diagram acyclic. In the figure above the removal of the t vertices, corresponding to the last t rows and columns of the matrix, leaves the remaining graph acyclic.

Algorithms so far proposed for this tend to choose too large a t for this form to be useful in practice. Cheung and Kuh, and Sangiovanni-Vincentelli at UC Berkeley have done some work on this, but I don't recommend the paper in Bunch and Rose (S-V).

3) Bordered block triangular form



Even less work has been done on this form and no really good algorithms exist for obtaining it. Chemical engineers Kevorkian and Snoek [*Himmelblau Book*], Sargent [*papers in Chem. Eng*], and Westerberg [*papers in Chem. Eng.*] have proposed heuristic techniques.

Certainly in view of the cheapness of block triangularization algorithms, we could afford two or three selections of t columns.

Tearing

Utilizes formula relating inverse of a matrix to the inverse of a matrix different from it by a small rank change. Sherman-Morrison Householder (Woodbury) formula:

$$(A + USV^T)^{-1} = A^{-1} - A^{-1} U(S^{-1} + V^T A^{-1} U)^{-1} V^T A^{-1}$$

where, normally, U,S,V are of small rank compared with A (and $A + USV^T$). With reference to the previous two forms, U,S,V will have rank t.

Some people who have worked in this area are: *Gabriel Kron (1946-1956), Donald Steward (1962-1969), Barkley and Motard (1972), Hellerman and Rarick (1971), and Lin and Mah (1975)* -- but I wouldn't recommend reading any of these!

A major problem with forms 1) and 3) [band matrix and bordered block triangular form] is that the problem of obtaining the optimal form (e.g., minimize t in 2) and 3)) for an arbitrary given matrix has been shown to be NP-complete. Therefore heuristics have been proposed to find sub- (but hopefully near-) optimal orderings. Unfortunately, apart from form 1), they have not been of much use. However, form 3) with the shaded area zero is called a "block triangular form."

Good algorithms for obtaining this form exist and there are several areas (most noticeably chemical engineering and linear programming) which give rise to matrices which can be decomposed into this form. We now examine this form.

A matrix is bireducible ($\exists$, P, Q $\ni$ PAQ is in BLTF) if and only if any permutation of it with a zero-free diagonal is reducible ($\exists$, P $\ni$ PAP^T is in BLTF). Furthermore, the block triangular form is essentially independent of this initial permutation (e.g. *Duff, 1977*).

Thus it is possible to perform the BLTF algorithm in two steps:

1) $A \rightarrow AQ = A_1$ where A_1 has a zero-free diagonal.
Most algorithms $O(nt)$!! perform well in practice.
Hopcraft and Karp's [*1974*] algorithm is $O(n^{\frac{1}{2}}t)$, although no good implementation yet exists.

2) $PA_1P^T = A_2$ (BLTF), corresponds to finding strong components of a diagraph. Tarjan algorithm [*1972*] is $O(n) + O(t)$ and has been implemented by Duff and Reid [*1976-78*].

Wiberg [*1977*] has devised an algorithm which combines steps 1) and 2).

Relative Times for Two Phases of Block Triangularization

Matrix N	Matrix NZ	Transversal selection	Tarjan algorithm
50	100	41	15
50	300	22	30
100	200	78	32

times are in milliseconds on an IBM-360/67.

So: 1) no great evidence of O(nT) behavior of transversal selection (i.e. obtaining a zero-free diagonal) on typical examples, and

2) transversal selection not always that much slower than symmetric permutation and may even be faster.

Overheads of Block

Order of Matrix	147	199	822
Number of N-Z	2449	701	4841
Block	70	30	250
PAQ → LU	1470	210	480
$A \xrightarrow[Q]{P} LU$	280	50	180
LU + rhs → ℓhs	20	10	40

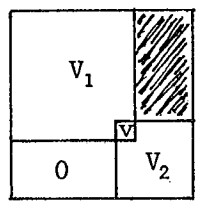
times are in milliseconds on an IBM-370/168.

199:701	240	without BLOCK
822:4841	1520	without BLOCK

Use of Dominators to Decompose Graph/Matrix

Tarjan has attempted to extend the partitioning discussed earlier to a decomposition of irreducible blocks by means of dominators, defined earlier. We find a vertex V such that there exists vertices U and W, for which V dominates W with respect to U. (If no such triple exists then the graph is said to be strongly biconnected.) We then partition the vertex set V into three sets: {V}, V_1, and V_2, where V_1 contains all vertices reachable from U by a path which avoids V.

If we order the rows and columns of the matrix according to an ordering on the vertices, numbering those in V_2 before V before those in V_1 we get the partitioned matrix



and we can both preserve the zero block in this matrix and confine our pivot selection to the "diagonal" blocks.

Tarjan has indicated a scheme to perform a "canonical" decomposition in nearly $O(n)+O(T)$ time, but to my knowledge it hasn't yet been implemented or used.

1) Recent improvements in S.M. codes

n (order)	199	822	900
nz (non-zeros)	701	4841	4380
MA18 [*1971*]	410	3090	88800
MA28 [*1977*]	240	1760	11400

2) Analytic : Factor : Operate ratios (MA28).

n	199	822	900
nz	701	4841	4380
Analytic	240	1760	11400
Factor	50	260	1520
Operate	10	40	90

times in milliseconds on an IBM-370/168.
codes MA28A and MA18A are from Harwell subroutine library.

3) Use of Block

n	199	822	900
nz	701	4841	4380
MA28	240	1760	11400
MA28 + block	240	940	11700

Although the following refers to a technique applicable to problems arising in finite element calculations (in, for example, structural mechanics), we hope to generalize it so as to handle symmetric indefinite and asymmetric matrices. As you can see, it is well organized to handle matrices out of core.

Finite Element Matrices

The region over which our differential equation (for example) is to be solved is first divided into small subregions or elements (for example, if our region is in R^2 then a triangularization of it would yield triangular elements).

Equations are set up for each element. The matrix, $B^{(k)}$ say, associated with element k will have a non-zero in position (i,j) only if variables x_i and x_j belong to element k. It is common to store $B^{(k)}$ as a small full matrix of order of the number of variables in element k and hold an index vector indicating the position of the variables in the main array.

Then our equations $A\underline{x} = \underline{b}$ can be considered to have the form

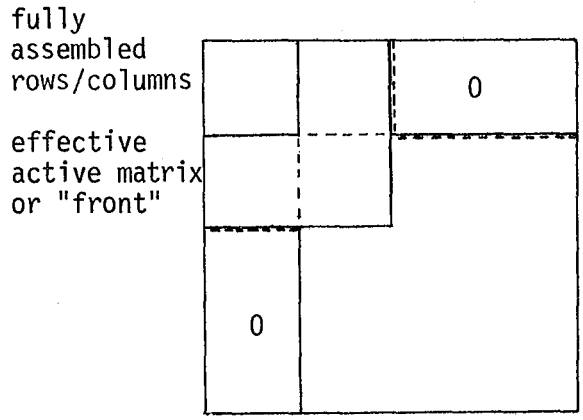
$$A = \sum_k B^{(k)} \qquad \underline{b} = \sum_k \underline{c}^{(k)}$$

where $\underline{c}^{(k)}$ is the contribution of element k to the right-hand side.

We need to:

1) Calculate the $B^{(k)}$ and $\underline{c}^{(k)}$.
2) Assemble the matrices to get A and $\underline{b}$.
3) Solve the assembled system.

The techniques we now discuss take advantage of the fact that we can perform some of the eliminations in a Gaussian elimination decomposition of A before the assembly of A is complete.



Notes on this Partially Assembled Matrix

- The rows/columns in the upper left submatrix are fully summed and the rectangular matrices contain only zeroes.
- If a pivot is chosen from anywhere within the fully summed block, fill-in and arithmetic operations are confined to the submatrix in the upper left. Hence we call this the effective active matrix, since this is all of the matrix we require during elimination of these fully summed rows/columns.
- We can easily organize the computation to make use of backing store, since elements need not be assembled into effective active matrix until fully assembled part is exhausted and rows/columns which have been pivotal can be sent off to backing store.
- Such frontal techniques can be easily generalized to asymmetric or indefinite systems by choosing pivots from fully assembled matrix according to some pivoting criterion.

DIRECT METHODS FOR SOLVING SPARSE SYSTEMS OF EQUATIONS

S. Eisenstat
New Haven, Connecticut

Problem :

To solve system of linear equations

$$A\underset{\sim}{x} = \underset{\sim}{b}$$

where

i) A is large ($\geqslant$ 1000 unknowns)

ii) A is sparse, i.e., most $a_{ij} = 0$.

iii) A is nonsymmetric, but ...

iv) ... pivoting is not required to maintain numerical stability.

Model Problem :

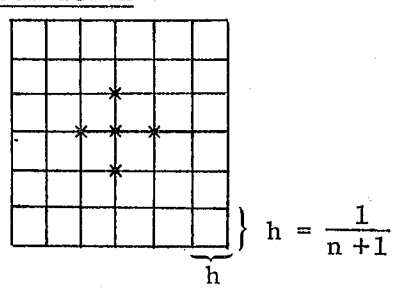
$$a \frac{\partial^2 u}{\partial x^2} + b \frac{\partial^2 u}{\partial x \partial y} + c \frac{\partial^2 u}{\partial y^2} + d \frac{\partial u}{\partial x} + e \frac{\partial u}{\partial y} + fu = g \qquad \text{in } D$$

$$u = \phi \qquad \text{on } \partial D$$

where D = unit square

Grid Discretization :

D :



$$u_{ij} \approx u(x_i, y_j)$$

Five-point difference equation :

$$A_{ij}u_{ij} + B_{ij}u_{i-1,j} + C_{ij}u_{i+1,j} + D_{ij}u_{i,j-1} + E_{ij}u_{i,j+1} = F_{ij}$$

$$u_{ij} = \phi(x_i,y_j) \quad , \qquad (x_i,y_j) \in \partial D$$

$\Rightarrow$ System of $N = n^2$ equations.

<u>(Dense) Gaussian elimination</u> :

(1) For $k = 1, \ldots, N-1$, use k^{th} equation to eliminate k^{th} variable from remaining N-k equations. Back-solve resulting upper triangular system for $\underset{\sim}{x}$.

(2) Form LU-factorization of A where L is lower triangular and U is upper triangular with unit diagonal. Successively solve triangular systems

$$L\underset{\sim}{y} = \underset{\sim}{b}$$

$$U\underset{\sim}{x} = \underset{\sim}{y}$$

Cost of dense Gaussian elimination work

$$\sim \sum_{i=1}^{N-1} (N-i)(N-i+1) \sim \frac{1}{3} N^3 \qquad \text{multiply-adds}$$

Storage $\sim N^2$ words

$\therefore$ for $n \times n$ model problem, work $\sim 1/3\ n^6$, storage $\sim n^4$ which is unacceptable for n large.

<u>Goals of Sparse Direct Methods</u> :

(i) Avoid storing zero entries of A, L, and U

(ii) Avoid operating on entries which are known to be zero.

potential savings $\longrightarrow$

dense band profile sparse

$\longrightarrow$ complexity/overhead

Example :

$$\begin{bmatrix} x & x & x & x & x \\ x & x & & & \\ & x & x & & \\ & & x & x & \\ x & & & x & x \end{bmatrix}$$

Zero structure of A

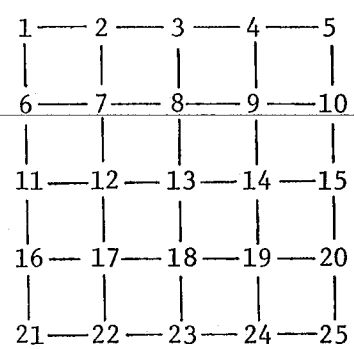
$$\begin{bmatrix} x & x & x & x & x \\ o & x & \underline{x} & \underline{x} & \underline{x} \\ & x & x & & \\ & & x & x & \\ o & \underline{x} & \underline{x} & x & x \end{bmatrix}$$

$\underline{x}$ denotes elements of A which have *filled in* during elimination

$$\begin{bmatrix} x & x & x & x & x \\ o & x & \underline{x} & \underline{x} & \underline{x} \\ & o & x & \underline{x} & \underline{x} \\ & & x & x & \\ o & o & \underline{x} & x & x \end{bmatrix}$$

$$\begin{bmatrix} x & x & x & x & x \\ o & x & \underline{x} & \underline{x} & \underline{x} \\ & o & x & \underline{x} & \underline{x} \\ & & o & x & \underline{x} \\ o & o & o & x & x \end{bmatrix}$$

Natural ordering :



fill-in

$$\nu = 105 + (128) \rightarrow 233$$

$$
\begin{bmatrix}
x & x & & & & x & & & & & & & & & & & & & & & & \\
x & x & x & & & o & x & & & & & & & & & & & & & & & \\
 & x & x & x & & o & o & x & & & & & & & & & & & & & & \\
 & & x & x & x & o & o & o & x & & & & & & & & & & & & & \\
 & & & x & x & o & o & o & o & x & & & & & & & & & & & & \\
x & o & o & o & o & x & x & o & o & o & x & & & & & & & & & & & \\
 & x & o & o & o & x & x & x & o & o & o & x & & & & & & & & & & \\
 & & x & o & o & o & x & x & x & o & o & o & x & & & & & & & & & \\
 & & & x & o & o & o & x & x & x & o & o & o & x & & & & & & & & \\
 & & & & x & o & o & o & x & x & o & o & o & o & x & & & & & & & \\
 & & & & & x & o & o & o & o & x & x & o & o & o & o & x & & & & & \\
 & & & & & & x & o & o & o & x & x & x & o & o & o & o & x & & & & \\
 & & & & & & & x & o & o & o & x & x & x & o & o & o & o & x & & & \\
 & & & & & & & & x & o & o & o & x & x & x & o & o & o & o & x & & \\
 & & & & & & & & & x & o & o & o & x & x & x & o & o & o & o & x & \\
 & & & & & & & & & & x & o & o & o & x & x & x & o & o & o & o & x \\
 & & & & & & & & & & & x & o & o & o & x & x & x & o & o & o & o \\
 & & & & & & & & & & & & x & o & o & o & x & x & x & o & o & o \\
 & & & & & & & & & & & & & x & o & o & o & x & x & x & o & o \\
 & & & & & & & & & & & & & & x & o & o & o & x & x & x & o \\
 & & & & & & & & & & & & & & & x & o & o & o & x & x & x \\
 & & & & & & & & & & & & & & & & x & o & o & o & x & x
\end{bmatrix}
$$

x = nonzero

o = fill-in

<u>Band Elimination</u> :

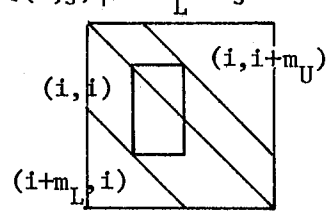
m_U = upper bandwidth

$= \max\{j-i \mid i \leq j,\ a_{ij} \neq 0\}$

m_L = lower bandwidth

$= \max\{j-i \mid i \geq j,\ a_{ij} \neq 0\}$

Band (A) $= \{(i,j) \mid i - m_L \leq j \leq i + m_U\}$



Perform Gaussian elimination under assumption that all elements within Band (A) are nonzero and all other elements are zero.

Note: No fill-in can occur outside Band (A).

Cost :

Work $\sim N m_L m_U$, multiply-adds

Storage $\sim N(m_L + m_U)$, words

$\therefore$ cost for $n \times n$ model problem is work $\sim n^4$, storage $\sim 2n^3$, which is significantly better than dense Gaussian elimination.

Reordering to Minimize Bandwidth :

Problem formulation generally induces "natural" ordering of variables and equations, but -- could also solve permuted system

$$PAP^T \underset{\sim}{y} = \underset{\sim}{b} , \qquad P\underset{\sim}{x} = \underset{\sim}{y}$$

which might have smaller bandwidth (and therefore require less work/storage). [cf. *Cuthill-McKee, Gibbs-Poole-Stockmeyer*].

Diagonal Ordering :

$\nu = 105 + (\overset{\text{fill-in}}{100}) \rightarrow 205$

```
1 — 2 — 4 — 7 — 11
|   |   |   |   |
3 — 5 — 8 — 12 — 16
|   |   |   |   |
6 — 9 — 13 — 17 — 20
|   |   |   |   |
10 — 14 — 18 — 21 — 23
|   |   |   |   |
15 — 19 — 22 — 24 — 25
```

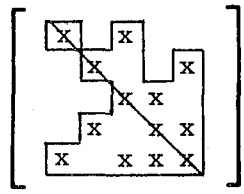
```
x x x
x x o x x
x o x o x x
  x o x o o x x
  x x o x o o x x
    x o o x o o x x
      x o o x o o o x x
      x x o o x o o o x x
        x x o o x o o o x x
          x o o o x o o o x x
            x o o o x o o o x x
              x x o o o x o o o x x
                x x o o o x o o o x x
                  x x o o o x o o o x
                    x x o o o x o o o x
                      x x o o o x o o o x x
                        x x o o o x o o o x x
                          x x o o o x o o o x
                              x x o o x o o x
                                x x o o x o x x
                                  x x o o x o x
                                      x x o x o x
                                        x x o x x
                                            x x x
```

x = nonzero

o = fill-in

Profile Elimination :

(AKA envelope, frontal, variable bandwidth elimination.)



f_i^L = first nonzero column in i^{th} row of A

f_i^U = first nonzero row in i^{th} column of A

$$ENV(A) = \{(i,j) \mid f_i^L \leq j \leq i \quad \text{or} \quad f_j^U \leq i \leq j\}$$

Perform Gaussian elimination assuming all elements outside ENV(A) are zero.

Cost for model problem on $n \times n$ grid is

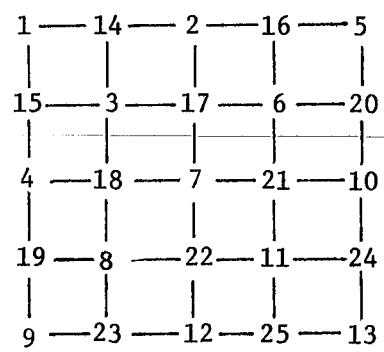
$$\text{work} \sim \frac{1}{2} n^4 \quad , \qquad \text{storage} \sim \frac{4}{3} n^3$$

versus n^4 and $2n^3$ for band elimination.

Reordering Algorithms :

King; Levy; reverse Cuthill-McKee

Alternate diagonal ordering



$\nu = 105 + (82) \rightarrow 187$

```
[ x                         x x                 ]
  x                         x   x x             
    x                       x x   x x           
      x                       x     x x         
        x                       x       x       
          x                     x x     x x     
            x                     x x     x x   
              x                     x x     x x 
                x                     x       x 
                  x                     x x     x
                    x                     x x   x x
                      x                     x x   x
                        x                       x x
x x x                       x o o o o           
x   x x                     o x o o o o         
  x     x x                 o o x o o o o o     
  x x     x x               o o o x o o o o o   
    x x     x x             o o o o x o o o o o 
      x       x x             o o o o x o o o o 
        x x       x             o o o o x o o o o
          x x     x x           o o o o o x o o o o
            x x     x x           o o o o o x o o o
              x x     x             o o o o o x o o
                x x   x               o o o o x o
                  x x x                 o o o o x
```

Sparse Elimination :

Perform Gaussian elimination but do not store or operate on any zero entries.

Cost :

$$\text{Work} \sim \sum_{i=1}^{N} r_i \, C_i$$

$$\text{Storage} \sim \sum_{i=1}^{N} (r_i + C_i)$$

where

r_i = number nonzeroes in the ith row of U

C_i = number nonzeroes in i^{th} column of L

$\therefore$ cost for n n model problem with alternate diagonal ordering is

$$\text{Work} \sim \frac{1}{4} n^4 \text{ multiply-adds ,}$$

$$\text{Storage} \sim \frac{2}{3} n^3 \text{ words}$$

versus $\frac{1}{2} n^4$ and $\frac{4}{3} n^3$ for profile elimination.

<u>Reordering Algorithms</u> :

Minimum degree (AKA Markowitz)

Nested dissection (cf. George)

Nested dissection ordering :

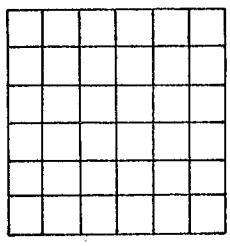
```
1 — 2 — 21 — 6 — 5
|   |   |    |   |
3 — 4 — 22 — 8 — 7
|   |   |    |   |
17 — 18 — 23 — 20 — 19
|   |   |    |   |
15 — 16 — 24 — 12 — 11
|   |   |    |   |
13 — 14 — 25 — 10 — 9
```

$\nu = 105 + (76) \rightarrow 181$

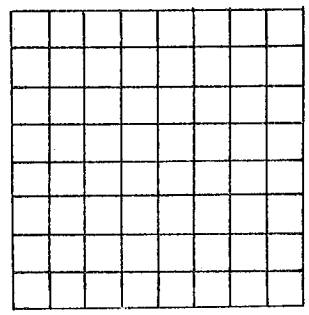
```
x x x
x x o x                         x
x o x x                     x   o
  x x x                     o x   o x
        x x x
        x x o x                 x
        x o x x               x o
          x x x               o x o x
                x x x
                x x o x                     x
                x o x x           x         o
                  x x x           o x     x o
                        x x x
                        x x o x             x
                        x o x x x           o
                          x x x o x       x o
    x o                     x o x x   o o   o o
      x                       x x x   o o x o o
            x o     x o           x x o o   o o
              x       x           x x o o x o o
  x o o   x o o                 o o o o x x o o o
      x       x                 o o o o x x x o o
                                  x   x o x x x o
                      x       x o o o o o o x x x
                  x o o   x o o o o o o o o o x x
```

Nested dissection :

Eliminate variables on separating cross last



9-point



5-point

Cost for $n \times n$ model problem :

	9-point	*5-point*
Work	$\sim 20\ n^3$	$8n^2 \log_2 n$
Storage	$\sim 10\ n^3$	$4n^2 \log_2 n$

versus $\frac{1}{4} n^4$ work and $\frac{2}{3} n^3$ storage for sparse elimination and using alternate diagonal ordering.

32 ×32 Model Problem :

	*Work** *(Multiply-adds)*	*Storage†* *(Words)*
Band (natural)	1.06×10^6	6.55×10^4
Envelope (diagonal)	5.67×10^6	4.57×10^4
Sparse (alternate diagonal)	3.04×10^5	2.77×10^4
Sparse (nested dissection)‡	2.40×10^5	2.30×10^4

*To factor A

†Number of elements in Band (A) or ENV(A) or number of nonzeroes in L+U

‡Five-point nested dissection.

TRUE COST = Cost + Overhead

- Additional storage required to specify data structures.
- Additional "bookkeeping" operations required to access and manipulate data structures.
- Loop control operations.

Fact : The efficiency of a program is generally determined by the efficiencies of its innermost loops.

90-10 Rule : 90% of run-time is spent executing 10% of the code.

Implementation of Sparse Elimination : [*Chang*]

Goal : Factor A into LU without storing or operating on zero entries of A or L+U. To be efficient, one must avoid testing for zero entries and must also avoid comparing column or row entries -- or else arithmetic operations will not comprise the bulk of computation.

Storage Scheme :

Store nonzero entries of A, L, U by rows, together with column index for each entry:

JA :	. . .	1	3	4	. . .	column index
A :	. . .	a_{K1}	a_{K3}	a_{K4}	. . .	value

IA(K) = pointer to start of row K

An alternate form of Gaussian elimination for K = 1, ... ,N-1 or for I = 1, ... ,K-1 is to use the Ith equation to eliminate the Ith variable from K^{th} equation. NOTE : I^{th} variable only needs to be eliminated from K^{th} equation if $L_{KI} \neq 0$.

Assume positions of all nonzero entries in L and U were known (assuming exact cancellation never occurs, this can be done symbolically). Then proceed as follows at K^{th} step:

1. Expand K^{th} row of A into vector of length W, inserting zeroes where fill-in in L+U will occur

D :	a_{K1}		0	0		a_{K6}		0	a_{K9}	0	
				L					U		

2. For I = 1, ... K-1, where $L_{KI} \neq 0$, subtract multiple of Ith row of U from D.

   For $J = J_{min}, J_{max}$:

   $$D(JU(J)) = D(JU(J)) - L_{KI}*U(J)$$

3. Store nonzero elements of D as K^{th} rows of L and U.

Remarks : Symbolic factorization (SYMFAC) need only be done once for a given zero-nonzero structure. SYMFAC takes significantly less time than numerical factorization (NUMFAC).

Innermost Loops :

(1) Inner product :

```
    DO 1 J = JMIN,JMAX
1   SUM = SUM + A(J)*B(J)
```

(2) Band elimination (outer product) :

```
    DO 1 J = JMIN,JMAX
1   A(MI+J) = A(MI+J) + UIK*A(MK+J)
```

(3) Profile elimination, band elimination (inner product) :

```
    DO 1 J = JMIN,JMAX
1   SUM = SUM + A(MI+J)*A(MK+J)
```

(4) Sparse elimination :

```
    DO 1 J = JMIN,JMAX
1   D(JU(J)) = D(JU(J)) + LIK*U(J)
```

32 × 32 Model Problem :

	*Factorization time**	*Total storage*†
Band (natural (natural)	7.42 (7.0)	6.55×10^4 (1.00)
Profile (diagonal)	3.97 (7.0)	4.77×10^4 (1.04)
Sparse (5-point nested dissection)	2.28‡ (9.45)	4.71×10^4 (2.04)
	↑ μsec/multiply	↑ words/nonzero

*In seconds on IBM 370/158 using FORTRAN H extended (OPT=2) compiler.
†In words (1 word = 4 bytes).
‡Symbolic factorization required an additional 0.75 seconds.

CONCLUSIONS :

1. Profile elimination is superior to band elimination
   - Fewer nonzeroes in ENV(A) than in Band (A), whence less work/storage.
   - Equally efficient.
   - Added storage to specify ENV(A) is not significant
2. Sparse elimination may be superior to profile elimination
   - Fewer nonzeroes in sparse factorization than in ENV(A), whence less work/storage.
   - Sparse elimination is slightly less efficient.
   - Added storage to specify data structure is significant.
   - For high order discretizations on coarse meshes, savings in the number of nonzeroes is less significant.

Direct vs. Iterative :

Direct Methods have the advantages of exact solution, fixed cost, and a solution for additional right-hand sides is relatively cheap. The disadvantages are that there is excessive (?) storage and excessive (?) work.

Iterative Methods have the advantages of very low storage requirements; rapid convergence whence less work; and can take advantage of good starting guess (e.g., nonlinear or time-dependent problems). Disadvantages are: choice (existence?) of good parameters (i.e., "nobody knows how to tune SIP"); total work highly dependent on choice of initial guess; and stopping criterion is delicate balance between speed and accuracy.

Recommendations :

For one-dimensional problems, use profile elimination.

For two-dimensional problems, use direct methods (sparse/profile) unless storage is a limiting factor or iterative methods are demonstratively superior (e.g., iterative runs faster in practice or problem is time-dependent with time-varying coefficients).

For three-dimensional problems, avoid direct methods unless iterative methods don't work.

Example : 7-point discretization of Poisson equation on unit cube.

Direct	*Iterative*
$W \sim O(n^6)$	$W \leqslant O(n^4 \log n)$
$S \sim O(n^4)$	$S \sim O(n^3)$

REMARKS ON ITERATIVE METHODS FOR THE SOLUTION OF LARGE SYSTEMS OF LINEAR ALGEBRAIC EQUATIONS*

Olof Widlund

Courant Institute of Mathematical Sciences
New York University
251 Mercer Street, New York, N.Y. 10012

1. Introduction. This is a summary of a talk given at the NRCC Meeting in Santa Cruz, California, August 9-11, 1978. First, a brief survey is given of classical, stationary iterative methods with an emphasis placed on their limitations. The rest of the paper is devoted to conjugate gradient type methods, which have been studied intensely in recent years. Attention is focused on the use of preconditioning and on methods which provide alternatives to the standard conjugate gradient methods.

We shall restrict ourselves to problems of the form

$$Ax = b ,$$

where A and b are real and A is a square matrix. We shall also always assume the existence of a solution.

*This work was supported by the U.S. Dept. of Energy under contract No. EY-76-C-02-3077.

2. Basic Iterative Methods. The very simple iterative method

$$Dx^{(k+1)} = Dx^k - (Ax^k - b)$$

or

$$x^{(k+1)} = (I - D^{-1}A)x^k + D^{-1}b$$

is known as Jacobi's method. Here D is the matrix of diagonal elements of A. Let L and U be the strictly lower and upper triangular matrices given by

$$I - D^{-1}A = L + U .$$

The Gauss-Seidel method is defined by

$$(I - L)x^{(k+1)} = Ux^k + D^{-1}b$$

and can be considered as a special case of the successive overrelaxation method (SOR). This method which enjoys a well deserved popularity, is defined by

$$(I - \omega L)x^{(k+1)} = \left(\omega U + (1 - \omega)I\right)x^k + \omega D^{-1}b$$

where ω is a real parameter, independent of k, chosen in the interval (0,2) in order to enhance the convergence. The SOR method can also be considered as resulting from an acceleration of the Gauss-Seidel method. The new iterate is thus chosen along the half line originating at the previous iterate and passing through the point $x^{(k+1)}$ defined by the Gauss-Seidel method. The use of a method in this family requires the solution of a linear system of equations with a

lower triangular matrix, a fact which in no way complicates the calculations.

The Jacobi method can similarly be accelerated and its rate of convergence is often used as a standard of comparison when evaluating the performance of iterative methods.

The study of the rate of convergence of an algorithm of this type begins with the substitution $x^k = x + e^k$, where x is the exact solution and e^k the error after k steps. For the SOR method, we then obtain

$$e^{(k+1)} = \mathcal{L}_\omega e^{(k)}$$

where

$$\mathcal{L}_\omega = (I - \omega L)^{-1} \left(\omega U + (1 - \omega) I\right) .$$

The rate of convergence is measured by the spectral radius of the matrix $\mathcal{L}_\omega$.

For the optimally extrapolated Jacobi method, with A positive definite symmetric, the correspoinding spectral radius is given by

$$\left(\kappa(D^{-1/2} A\, D^{-1/2}) - 1\right) / \left(\kappa(D^{-1/2} A\, D^{-1/2}) + 1\right)$$

where the condition number $\kappa(B)$ of a matrix B is the ratio of its largest and smallest eigenvalues. Choosing the optimal acceleration parameters for the extrapolated Jacobi and SOR methods requires knowledge of the spectrum of the matrices involved. Exact information of this kind is normally not available and the development of an appropriate procedure of

estimating and improving this parameter and a good stopping criterion are often the most time consuming parts of the preparation of a computer code of this kind. Similar problems arise when semiiterative methods of Chebyshev type are used. The three standard monographs in this area, see Varga [14], Wachspress [15] and Young [18], contain discussions of this problem. Professor David Young and his associates at the University of Texas at Austin have incorporated recent ideas into computer codes developed for the ITPACK project. ITPACK, which will provide standard FORTRAN programs for a number of important iterative methods, is now closely tied to the ELLPACK effort. ELLPACK is a software project for solving elliptic problems coordinated by Professor John Rice of Purdue. While the development and study of iterative methods and algorithms for elliptic finite difference schemes have been connected intimately, these methods can of course be used with success for many other problems.

We shall now briefly discuss the limitations of these methods. The theory on the convergence of iterative methods of the type discussed in this section has reached a mature state, see in particular the monograph by Young [18]. Young [19] is recommended for a shorter survey. The positive definite symmetric case is considered almost exclusively and this restriction is not accidental. There are examples of positive definite symmetric matrices for which Jacobi's method diverges but the optimally accelerated Jacobi method is always

convergent for such matrices. For symmetric matrices the SOR method converges if and only if A is positive definite and $\omega \in (0,2)$. Cases are known for which SOR is not appreciably more efficient than a Jacobi method but for certain families of matrices an order of magnitude improvement can be realized. This is true if the matrix has property A, which means that a permutation P exists such that

$$P^{-1}A\,P = \begin{pmatrix} D_1 & B \\ B^T & D_2 \end{pmatrix}, \qquad D_1\,,\ D_2 \ \text{diagonal.}$$

For this family the role of κ, in the estimate given above for the rate of convergence of the Jacobi method will be played by constant $\times\ \sqrt{\kappa}$. Similar gains can be made when all off diagonal elements are nonpositive. It should also be noted that the SOR method is known to be convergent for certain nonsymmetric problems and that extensions exist to nonlinear systems.

We remark that block versions of these methods have often proven quite successful. In particular a block tridiagonal structure of A with tridiagonal diagonal blocks can be exploited. We also note that property A and block property A can be used to develop compressed versions of the algorithm with reduced storage requirements. A similar device is described by Reid [13] for the conjugate gradient method.

Finally, we define still another iterative method, the symmetric SOR method. In every other step the order of the unknowns is reversed in the SOR method, resulting in an error

equation, corresponding to a double sweep, which is of the form

$$e^{(k+1)} = \mathcal{L}_\omega^T \mathcal{L}_\omega e^k , \quad \mathcal{L}_\omega^T \text{ the transpose of } \mathcal{L}_\omega .$$

As it stands, this method offers no real benefits in comparison to the methods previously introduced. It has however been shown for certain elliptic problems, that Chebyshev and conjugate gradient acceleration of this method results in a very considerable improvement of the convergence rate, see Young [18] and Axelsson [1,2].

3. Conjugate Gradient Type Methods. These methods can be considered as members of the family of nonstationary iterative methods. Other well known members of that family, the Richardson and Chebyshev methods are considered in detail in the monographs mentioned above. The conjugate gradient method often requires only marginally more work per step and it is becoming an increasingly popular choice. It requires no a priori information on the spectrum of the operator and is optimal in a sense which shall be specified below. For an introduction to standard material on the conjugate gradient method, see Hestenes and Stiefel [7], Luenberger [8] and Reid [12,13].

The standard conjugate gradient method can be characterized mathematically in the following way: Let A be positive definite symmetric and let x_0 be the initial guess. The Krylov sequence with respect to the initial residual $r_0 = b - Ax_0$

is given by

$$r_0 \ , \ Ar_0 \ , \ \ldots , \ A^{k-1}r_0 \ , \ \ldots \ .$$

Denote by S^k the subspace spanned by the k first elements of this sequence. The kth approximation x^k then satisfies

$$(x^k - x)^T A(x^k - x) \leq (y - x)^T A(y - x)$$

for all $y = x_0 + z$, $z \in S^k$, i.e. x^k is the element with the smallest error with respect to the A-norm $\sqrt{x^T Ax}$. When properly implemented only a few vectors of storage are needed and a three term recursion relationship can be used to define x^k. Extensions to more general operators are discussed below.

Any of the iterative methods of section 2 can be studied in terms of a splitting of the operator A,

$$A = A_0 - R \ .$$

A simple iterative method is then applied to the transformed, preconditioned equation $A_0^{-1}Ax = A_0^{-1}b$. It is therefore natural to consider the Krylov sequence,

$$A_0^{-1}r_0 \ , \ (A_0^{-1}A)A_0^{-1}r_0 \ , \ \ldots , \ (A_0^{-1}A)^{k-1}A_0^{-1}r_0 \ , \ \ldots \ .$$

It must of course be economically feasible to compute the solution of a system of the form $A_0 y = c$ in each step. If both A and A_0 are positive definite, symmetric, the conjugate gradient algorithm easily generalizes. In the algorithm the operator A is replaced by $A_0^{-1}A$ and the inner product $x^T y$, used in the computation of certain parameters, is replaced

by $x^T A_0 y$. These ideas go back, at least, to Hestenes [6]. For elliptic problems this device has proven very useful when A_0 is chosen to correspond to a fast Poisson solver, a symmetric SOR operator or an incomplete Cholesky factorization of A, see Axelsson [2], Concus, Golub and O'Leary [4] and Meijerink and van der Vorst [10].

The choice of A_0 for a given A is definitely an art. Ideally $A_0^{-1}A$ should have the form $\alpha I + B + C$ where α is a scalar, B an operator of low rank and C an operator with small norm. The conjugate gradient algorithm converges very quickly in such cases. Another important consideration is of course the cost of obtaining the solution for the simplified model $A_0 y = c$.

The rate of convergence of the conjugate gradient algorithm can be estimated as follows. The decrease of the square of the A-norm of the error,

$$(x^k - x)^T A(x^k - x) \Big/ (x^0 - x)^T A(x^0 - x) ,$$

is bounded from above by

$$\min_{P_{k-1}} \max_{\lambda \in \sigma(A_0^{-1}A)} \left(1 + \lambda P_{k-1}(\lambda)\right)^2 ,$$

see Luenberger [8]. P_{k-1} is any polynomial of degree k-1 and $\sigma(A_0^{-1}A)$ the spectrum of $A_0^{-1}A$ i.e. the eigenvalues of the generalized eigenvalue problem $A\phi = \lambda A_0 \phi$. This expression is bounded from above by

$$4\left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^{2k} ,$$

where κ is the condition number of $A_0^{-1}A$. This last bound is known to be a gross overestimate in certain cases.

We conclude by surveying some work on methods for operators which are not positive definite, symmetric.

If A is symmetric but indefinite numerically stable algorithms, SYMMLQ and MINRES, due to Paige and Saunders [11] can be used. Preconditioning by a positive definite symmetric A_0 can sometimes be very helpful. A revised version of SYMMLQ, which incorporates preconditioning, is available from the author upon request. Professor S. Eisenstat of Yale and his associates have worked out alternative stable algorithms. Apparently, it is not widely recognized, that error bounds similar to those given above in the positive definite case can be obtained for the SYMMLQ algorithm, see Widlund [17].

The case when a nonsymmetric operator A can be split into $A = A_0 - R$ where A_0 is positive definite, symmetric and R is antisymmetric has been considered by Concus and Golub [3], Hageman, Luk and Young [5] and Widlund [16,17].

A Chebyshev method, which requires estimates of the eigenvalues of $A_0^{-1}A$, has been developed by Manteuffel [9]. It converges if the eigenvalues of $A_0^{-1}A$ lie in an ellipse not containing the origin.

Any system of equations, with an invertible matrix A, can be transformed into a positive definite symmetric problem by the Gauss transforms

$$A^TAx = A^Tb$$

and

$$A\,A^Ty = b \quad \text{where} \quad x = A^Ty\ .$$

Preconditioning by an arbitrary invertible A_0 can precede these transformations. In certain cases a good choice of A_0 makes these methods attractive in spite of the extra work involved in each step. We remark that since the conjugate gradient algorithms require the operator only in terms of an operator-vector multiplication, it is normally advisable to retain the operators A^TA, $A_0^{-1}A$, etc., in factored form.

References

1. Axelsson, O., BIT, v. 13, 1972, pp. 443-467.
2. Axelsson, O., in Lecture Notes in Mathematics, v. 572, (Barker, V. A., ed.), Springer, 1977.
3. Concus, P. , and Golub, G. H. in Lecture Notes in Economics and Mathematical Systems, v. 134, (Glowinski, R. and Lions, J. L., ed.), Springer, 1976.
4. Concus, P., Golub, G. H. and O'Leary, D. P., in "Sparse Matrix Computations," (Bunch, J. R. and Rose, D. J., ed.), Academic Press, 1976, pp. 309-332.
5. Hageman, L. A., Luk, F. and Young, D. M., report CNA-129, Center for Numerical Analysis, University of Texas, 1977
6. Hestenes, M. R. in Proceedings of Symposia in Applied Math., VI, McGraw-Hill, 1956, pp. 83-102.
7. Hestenes, M. R., and Stiefel, E., J., Res. Nat. Bur. Standards, v. 49, 1952, pp. 409-436.
8. Luenberger, D. G., "Introduction to Linear and Nonlinear Programming," Addison-Wesley, 1973.
9. Manteuffel, T., Numer. Math., v. 28, 1977, pp. 307-327.
10. Meijerink, J. A. and van der Vorst, H. A., Math. Comp., v. 31, 1977, pp. 148-162.
11. Paige, C. C. and Saunders, M. A., SIAM J. Numer. Anal., v. 12, 1975, pp. 617-629.

12. Reid, J. K., in "Large Sparse Sets of Linear Equations," (Reid, J. K., ed.), Academic Press, 1971.

13. Reid, J. K., SIAM J. Numer. Anal., v. 9, 1972, pp. 325-332.

14. Varga, R. S., "Matrix Iterative Analysis," Prentice Hall, 1962.

15. Wachspress, E. L., "Iterative Solution of Elliptic systems and Applications to the Neutron Diffusion Equations of Reactor Physics," Prentice Hall, 1966.

16. Widlund, O., SIAM J. Numer. Anal., v. 15, 1978, pp. 801-812.

17. Widlund, O., to appear.

18. Young, D. M., in Proceedings of an NSF-CBMS regional conference in Pittsburgh, Pa., 1972. Academic Press, 1973, pp. 101-156.

MODIFIED GRAM-SCHMIDT

Cleve Moler
University of New Mexico

The Modified Gram-Schmidt algorithm (MGS) is a rearrangement in the Conventional Gram-Schmidt algorithm (CGS) which, in inexact arithmetic, produces vectors that are usually "closer to orthogonal" and better suited for such applications as solution of least squares problems.

Let S be an arbitrary m by n real matrix with columns s_j where $j = 1, \ldots, n$. The objective is to produce a matrix Q whose columns q_j are orthonormal and where each s_j is a linear combination of the q_k for $k \leqslant j$, that is,

$$s_j = \sum_{k \leqslant j} r_{kj} q_k \quad .$$

If R is the upper triangular matrix with elements r_{kj} (and zeros below the diagonal), this can be written

$$S = QR \quad .$$

The CGS algorithm produces R a row at a time. The inner products are taken between the emerging columns of Q and the columns of the *original* S. The algorithm is:

```
for  j = 1, ... , n
     q_j  =  s_j
     for  k = 1, ... j-1
          r_kj  =  (q_k, s_j)
          q_j   =  q_j  -  r_kj q_k

     r_jj  =  ||q_j||
     q_j   =  q_j / r_jj
```

The MGS algorithm produces R a column at a time. The inner products are taken between the emerging columns of Q and the columns that have been

produced by previous steps. The q_j can overwrite the s_j. The algorithm is:

```
for   j = 1, ...,n
      q_j = s_j
for   k = 1, ...,n
      r_kk = ||q_k||
      q_k = q_k/r_kk
      for   j = k+1, ...,n
            r_kj = (q_k,q_j)
            q_j = q_j - r_kj q_k
```

As an example, let S be the 4×3 matrix

$$S = \begin{pmatrix} 1 & 1 & 1 \\ \varepsilon & 0 & 0 \\ 0 & \varepsilon & 0 \\ 0 & 0 & \varepsilon \end{pmatrix}$$

where ε is not small enough to be neglected when compared to 1, but whose square is, for example $\varepsilon = 10^{-5}$ on a computer with eight significant digits. Then CQS produces

$$Q = \begin{pmatrix} 1 & 0 & 0 \\ \varepsilon & -\sigma & -\sigma \\ 0 & \sigma & 0 \\ 0 & 0 & \sigma \end{pmatrix}$$

where $\sigma = 1/\sqrt{2}$. Note that the second and third columns of Q are nowhere near being orthogonal to each other. In contrast, MGS produces

$$Q = \begin{pmatrix} 1 & 0 & 0 \\ \varepsilon & -\sigma & -\tau \\ 0 & \sigma & -\tau \\ 0 & 0 & 4\tau \end{pmatrix}$$

where $\tau = 1/\sqrt{6}$. Now the second and third columns are exactly orthogonal to each other. The only difficulty is that they are "not quite" orthogonal to the first, but this is the best that can be done with the arithmetic we have assumed.

THE SYMMLQ ALGORITHM

A. K. Cline
University of Texas

SYMMLQ is an algorithm for solving symmetric indefinite systems developed by Paige and Saunders. It is related to the Lanczos process.

$$AV_K = V_K T_K + \beta_{K+1} v_{K+1} e_K^T$$

where

$$V_K = [v_1 | v_2 | \ldots | v_K]$$

$$T_K = \begin{bmatrix} \alpha_1 & \beta_2 & & 0 \\ \beta_2 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \beta_K \\ 0 & & \beta_K & \alpha_K \end{bmatrix}$$

We seek to solve $Ax = b$. Let

$$v_1 = \frac{1}{\|b\|} b$$

If $x = V_K y_K$

$$AV_K y_K = \|b\| v_1$$

$$\left(V_K T_K + \beta_{K+1} v_{K+1} e_K^T\right) y_K = \|b\| v_1$$

$$T_K y_K = \|b\| e_1$$

Alternatively, use SYMMLQ. Let $T_K Q_K = L_K$, L_K is lower triangular.

Thus

$$L_K Q_K^T y_K = \|b\| e_1$$

$$L_K z_K = \|b\| e_1$$

where $z_K = Q_K^T y_K$ and $x = V_K y_K = V_K Q_K z_K = W_K z_K$.

Note: 1. Lower triangular system is solved in one direction for z_K.

2. x is computed as linear combination of ω_K's.

It turns out:

1. L_K is easily determined from L_{K-1} and α_K, β_K.
2. ω_K is easily determined from ω_{K-1} and v_K.
3. z_K is easily determined from z_{K-1}.

Algorithm

1. $\beta_1 = \|b\|$, $v_1 = 1/\beta_1\, b$, $\alpha_1 = v_1^T A\, v_1$, $\bar{a}_1 = \alpha_1$

$\bar{\omega}_1 = v_1$, $x_o = 0$.

for $K = 1, 2, \ldots.$

2. Lanczos:

$$\beta_{K+1}\, v_{K+1} = A\, v_K - \alpha_K\, v_K - \beta_K\, v_{K-1}$$

$$\beta_{K+1} = \|\beta_{K+1}\, v_{K+1}\|$$

$$v_{K+1} = 1/\beta_{K+1} \cdot \beta_{K+1}\, v_{K+1}$$

$$\alpha_{K+1} = v_{K+1}^T A\, v_{K+1}$$

3. If $K = 1$, $\bar{e}_2 = \beta_2$; otherwise $f_{K+1} = s_{K-1}\, \beta_{K+1}$

$$\bar{e}_{K+1} = c_{K-1}\, \beta_{K+1}$$

4. $c_K = \bar{a}_K / \sqrt{\bar{a}_K^2 + \beta_{K+1}^2}$ $\qquad s_K = \beta_{K+1} / \sqrt{\bar{a}_K^2 + \beta_{K+1}^2}$

$$a_K = \sqrt{\bar{a}_K^2 + \beta_{K+1}^2}$$

$$e_{K+1} = c_K\, \bar{e}_{K+1} - s_K\, \alpha_{K+1}$$

$$\bar{a}_{K+1} = s_K \bar{e}_{K+1} + c_K \alpha_{K+1}$$

$$\omega_K = c_K \bar{\omega}_K - s_K V_{K+1}$$

$$\bar{\omega}_{K+1} = s_K \bar{\omega}_K + c_K V_{K+1}$$

5. $\gamma_K = -(f_K \gamma_{K-2} + e_K \gamma_{K-1})/a_K$, if $K \geqslant 3$

$$\gamma_1 = \beta_1/a_1 , \qquad \gamma_2 = -(e_2\gamma_1)/a_2$$

6. $x = x_{K-1} + \gamma_K \omega_K$

Per step:
- 5 vector adds
- 7 scalar times vector
- 2 inner product
- 1 A times vector

Convergence: Depends on size of polynomial of degree K which is smallest on spectrum but having value of 1 at origin. Can be improved by pre-conditioning. Instead of forming A_V at each step, we need to solve

$$\bar{A}\bar{v} = v$$

then form

$$A \bar{v} .$$

Other generalizations:

1. Minimum resident variant
2. Non symmetric
3. Least squares
4. Singular values

Reference : *C.C.Paige and M.A.Saunders, "Solution of sparse indefinite systems of linear equations," SIAM Jour. Numer. Anal. 12, 617-629 (1975).*

FOUR INDEX INTEGRAL TRANSFORMATION: AN $n^{4.?}$ PROBLEM?

Steve Elbert
Iowa State University

Introduction

To solve the time-independent Schroedinger equation for a system of N electrons in the field of K (fixed) nuclei

$$H_{op}\,\Psi(1\ldots N) = E\Psi(1\ldots N)$$

$$H_{op} = -\tfrac{1}{2}\sum_{i=1}^{N}\nabla_i^2 - \sum_{i=1}^{N}\sum_{\alpha=1}^{K}\frac{Z_\alpha}{r_{\alpha i}} + \sum_{i<j}^{N}\frac{1}{r_{ij}} + \sum_{\alpha<\beta}^{K}\frac{Z_\alpha Z_\beta}{r_{\alpha\beta}}$$

choose an expansion set of n (< 200) "atomic orbitals" (AO's), $\phi_a(q)$, $(i = 1,2,\ldots,n)$ of either Gaussian or exponential type and evaluate the electron-electron interaction integrals

$$G_a(p,q,r,s) = \int d^3\vec{r}_1 \int d^3\vec{r}_2\ \phi_a(p,\vec{r}_1)\,\phi_a(q,\vec{r}_1)\,r_{12}^{-1}\,\phi_a(r,\vec{r}_2)\,\phi_a(s,\vec{r}_2)$$

for all p,q,r,s. Methods that go beyond Hartree-Fock require that we further transform integrals over AO's to integrals over "molecular orbitals" (MO's), ϕ_m, where

$$\phi_m(i) = \sum_{p=1}^{n} T_{ip}\,\phi_a(p) \qquad i = 1,2,\ldots,M\leq n$$

Interaction integrals G_m over MO's are obtained by the 4-index transformation

$$G_m(i,j,k,\ell) = \sum_p\sum_q\sum_r\sum_s A(i,p)\,B(j,q)\,C(k,r)\,D(\ell,s)\,G_a(p,q,r,s)$$

where $\underset{\sim}{A}$, $\underset{\sim}{B}$, $\underset{\sim}{C}$, $\underset{\sim}{D}$ are subspaces of $\underset{\sim}{T}$ used to form $\underset{\sim}{\phi}_m$. In practice, $\underset{\sim}{G}_a$ ranges from dense to as high as 80-90% sparse (for large systems) while the coefficient matrices $\underset{\sim}{A}$-$\underset{\sim}{D}$ may range from dense to very sparse depending on the amount of symmetry present.

For large n, this 4-index transformation becomes quite time consuming and it is absolutely essential that one have efficient codes that take advantage of sparsity and any symmetry that may exist.

Computational Complexity

An $O(n^5)$ Method

		Maximum number multiplications
$(i\|qrs) = \sum_p A(i,p)\, G_a(p,q,r,s)$	all i,q,r,s	n^5
$(ij\|rs) = \sum_q B(j,q)\, (i\|qrs)$	all i,j,r,s	n^5
$(ijk\|s) = \sum_r C(k,r)\, (ij\|rs)$	all i,j,k,s	n^5
$G_m(i,j,k,\ell) = \sum_s D(\ell,s)\, (ijk\|s)$	all i,j,k,ℓ	n^5
	Total	$4n^5$

Alternatively, this can be regarded as two back-to-back *two-index* transformations:

$\underset{\sim}{G}^{pq}_{k\ell} = \underset{\sim}{C}^{+}\, \underset{\sim}{G}_a(p,q,r,s)\underset{\sim}{D}$	$n^2(2n^3)$
$G_m(i\,j\,k\,\ell) = \underset{\sim}{A}^{+}\, \underset{\sim}{G}^{pq}_{k}\, \underset{\sim}{B}$	$n^2(2n^3)$
Total	$4n^5$

In general, the elements $G^{pq}_{k\ell}$ are not generated in the same order as they are needed so a partial re-ordering (matrix transposition) of the $G^{pq}_{k\ell}$'s will be required before the second transformation can be carried out. Using the direct access "bin sorting" techniques of Yoshimine, however, this re-ordering is at most an $O(n^4)$ step.

An $O(n^6)$ Method

Let $[AB]^{ij}_{pq} = A(i,p)\, B(j,q)$ and $[CD]^{k\ell}_{rs} = C(k,r)\, D(\ell,s)$. Then,

$$G_m(i\,j\,k\,\ell) = \sum_{p,q} \sum_{r,s} [AB]^{ij}_{pq}\; G_a(p\,q\,r\,s)\; [CD]^{k\ell}_{rs}$$

is an obvious $O(n^6)$ method.

For dense matrices it is possible to improve on the n^3 dependence for ordinary matrix multiplication as shown in Table 1. The Winograd and Strassen's original algorithm are given below.

<u>Winograd Algorithm for</u> $\underset{\sim}{A}\,\underset{\sim}{B} = \underset{\sim}{C}$

$$C_{ij} = \sum_{K=1}^{\overline{N/2}} (A_{i,2K-1} + B_{2K,j})(A_{i,2K} + B_{2K-1,j}) - (\xi_i + \eta_j) + A_{i,N}\,B_{N,j}$$

if N is odd

$$\xi_i = \sum_{K=1}^{\overline{N/2}} A_{i,2K-1} \cdot A_{i,2K}$$

$$\eta_j = \sum_{K=1}^{\overline{N/2}} B_{2K-1,j} \cdot B_{2K,j}$$

unstable when $\|A_{i,2K-1}\| / \|B_{2K,j}\| \gg 1$ or $\ll 1$.

<u>Strassen Algorithm for</u> $\underset{\sim}{A}\,\underset{\sim}{B} = \underset{\sim}{C}$

$$M_1 = (A_{11} + A_{22})(B_{11} + B_{22})$$

$$M_2 = (A_{21} + A_{22})B_{11}$$

$$M_3 = A_{11}(B_{12} - B_{22})$$

$$M_4 = A_{22}(-B_{11} + B_{21})$$

$$M_5 = (A_{11} + A_{12})B_{22}$$

$$M_6 = (-A_{11} + A_{21})(B_{11} + B_{12})$$

$$M_7 = (A_{12} - A_{22})(B_{21} + B_{22})$$

$$C_{11} = M_1 + M_4 - M_5 + M_7$$

$$C_{21} = M_2 + M_4$$

$$C_{12} = M_3 + M_5$$

$$C_{22} = M_1 + M_3 + M_6 - M_2$$

TABLE 1. Matrix multiplication

Method	Scalar multiplications	Scalar additions	Scalar arithmetic
Standard	n^3	$n^3 - n^2$	$2n^3 - n^2$
Winograd (exploits commutivity of real multiplication)	$1/2\ n^3 + n^2$	$3/2\ n^3 - n^2$ $2n^2$ scaling operations	$2n^3$
Strassen (exploits 7 mult. instead of 8 in product of two 2×2's)	$O(n^{\log_2 7})$ =	$O(n^{2.81})$	$\leqslant 4.7\ n^{2.81}$
Limit	$3n^2 - 3n+1$?		$O(n^2)$??

There are six possible kinds of index symmetry for the 4-index arrays discussed here. The operator generating the array (symmetric or antisymmetric) may be restricted to certain types of index symmetry

Index Symmetry	Pattern	Operators
$ij \neq ji;\ k\ell \neq \ell k;\ (ij) \neq (k\ell)$	□ □ □	A, S
$ij = ji;\ k\ell \neq \ell k;\ (ij) \neq (k\ell)$	◺ □ □	S
$ij \neq ji;\ k\ell = \ell k;\ (ij) \neq (k\ell)$	□ ◺ □	S
$ij \neq ji;\ k\ell \neq \ell k;\ (ij) = (k\ell)$	□ □ ◺	A
$ij = ji;\ k\ell = \ell k;\ (ij) \neq (k\ell)$	◺ ◺ □	A
$ij = ji;\ k\ell \neq \ell k;\ (ij) = (k\ell)$	◺ □ ◺	does not occur
$ij \neq ji;\ k\ell = \ell k;\ (ij) = (k\ell)$	□ ◺ ◺	does not occur
$ij = ji;\ k\ell = \ell k;\ (ij) = (k\ell)$	◺ ◺ ◺	A

$1/r_{ee}$ is antisymmetric

The most difficult case to treat (i.e. exploit all the index symmetry) is ◺ ◺ ◺ . Use of symmetry reduces external storage requirements and I/O but complicates the implementation at every stage.

Table 2 shows the number of multiplications needed for each of 8 different algorithms for the 4-index transformation that have been published since 1970. Elbert's 25/24 n^5 method is the only significant improvement in recent years. It should also be pointed out that the n^5 method was actually in use prior to 1970 [*Tang and Edmiston*]. A 1961 [*unpublished*] formulation by E. R. Davidson is known.

Figures 2 and 3 give the algorithm loop structure for Elbert's [*1978*] new algorithm and the older method of Bender and Shavitt [*1972*].

Resource Management

Table 3 summarizes the I/O requirements for the various methods.

Internal Storage Requirements

1.	Fast	n^2	(+bin buffers)	[*Yoshimine, Elbert and Diercksen*]
		$(n^3+n^2)/2$		[*Bender, Pendergast and Fink*]
2.	Slow	$[n]^4$		[*Shavitt*]
3.	Virtual	n^3	(threshing?)	[*Pounder*]

External Storage and I/O (assumes no zero integrals)

$\triangle$ = canonical integral list

$[n]^2\,\overset{[n]^2}{\square}$ = "square" canonical integral list

Bender-Shavitt (single pass in, n passes out)

$$\underset{\text{in}}{\triangle} + n\,(\overset{\text{DA}}{\underset{\text{in-out}}{\triangle + \triangle}})$$

1/4 n^5 words transferred

$\rightarrow$ 1/8 n^4 for small m/n

Pounder (n passes in, single pass out)

$$n(\underset{\text{in}}{\triangle}) + \underset{\text{out}}{\triangle} + \text{paging (especially for } n^3) \quad > 1/8\ n^5 \text{ words transferred}$$

Pendergast and Fink (save all partial sums)(n passes in, single pass out)

$$n(\underset{\text{in}}{\triangle}) + [n]^2\,\overset{nm}{\underset{\text{in-out}}{\square}} + \underset{\text{in-out}}{\square} + \underset{\text{in-out}}{n[m]^3} + \underset{\text{out}}{\blacksquare} \quad > 1/8\ n^5 \text{ words transferred}$$

Yoshimine, Elbert (single extended pass in, single pass out; transpose intermediate results)

		D.A.				
□	+	□	+	◺		space for □ needed, but □+□ reduces read-write conflicts 7/8 n^4 words transferred
		inout		out		

Diercksen (same as above plus sequential write/read around d.a.)

				D.A.						
□	+	□	+	□	+	□	+	◺		11/8 n^4 words
		out		inout		in		out		

This procedure provides better restart security and is less memory-intensive during the cpu bound phase.

The transposition of the half-transformed integrals is done with a bin sort developed by Yoshimine. To be effective, a reasonable size buffer (one track will avoid rotational delay) is needed for *each* bin. This may require large amounts of fast/slow memory. If the memory requirements are too large, a P-ary sort-merge may be required. How is this best carried out?

> "No clear cut strategy for optimum disk sorting has been worked out; the number of available options greatly exceeds the number of strategies that have been theoretically analyzed ... a good deal of experimentation still needs to be done."
>
> — *Knuth*, Vol. 3 *(1973)*

BIBLIOGRAPHY

I. Full four-index transformations requiring $O(n^5)$ multiplications:

K.C.Tang and C.Edmiston (1970), "More efficient method for the basis transformation of electron interaction integrals," J. Chem. Phys. 52, 997-998. Does not exploit index symmetry very well; does not describe storage or I/O • 36/24 n^5 multiplications.

M.Yoshimine (1971), in Proceedings of the Conference on Potential Energy Surfaces in Chemistry, W.A.Lester, editor, Report RA-18, IBM Research Laboratory, San Jose, CA, p.87.

S.T.Elbert (1973), Ab initio Calculations on Urea, Ph.D. Thesis, University of Washington. Analysis of the number of multiplications for all four types of symmetry blocks for anti-symmetric operators. n^2 words of storage, bin sort, 29/24 n^5 multiplications.

C.F.Bender (1972), "Integral transformations: a bottleneck in molecular quantum mechanical calculations," J. Comp. Phys. 9, 547-554. This is an analysis of the best way to carry out a transformation if the transformed basis is shorter than the initial basis set. See *Shavitt (1977)* for fuller details of the algorithm. n^3 word storage, multipass sequential I/O, 29/24 n^5 multiplications.

M.Yoshimine (1973), in Energy, Structure and Reactivity, P.W.Smith and W.B.McRae, editors, (Wiley, New York), p.143. An explanation of bin sort transposition of half transformed integrals.

G.H.F.Diercksen (1974), "Optimized transformation of four center integrals," Theor. Chim. Acta 33, 1-6. There are some errors in this analysis. n^2 words of storage, bin sort, 29/24 n^5 multiplication.

P.Pendergast and W.H.Fink (1974), "A thorough analysis and exposition of the four-index transformation," J. Comp. Phys. 14, 286-300. Each partial sum stored on tape. n^3 words of storage, multipass I/O, 29/24 n^5 multip.

C.N.M.Pounder (1975), "The two-electron integral transformation and two-body density matrix transformation," Theor. Chim. Acta 39, 247-253. Implemented using virtual storage, n^3 words of storage, multipass I/O, 29/24 n^5 multiplications.

I.Shavitt (1977), "The Method of Configuration Interaction," in Methods in Electronic Structure Theory, H.F.Schaefer, editor (Plenum Press, N.Y.), p.205-208. Brief review, outline of *Bender's (1972)* program. n^3 words of storage, multipass I/O, 29/24 n^5 multiplications.

S.T.Elbert (1978), "An improved fully symmetric four-index transformation," to be published. Review of methods. n^5 words of storage, bin sort, 25/24 n^2 multiplication.

II. Partial four-index transformations:

J.A.Pople, R.Seeger and R.Krishnan (1977), "Variational configuration interaction methods and comparison with perturbation theory," Section 3, Avoidance of Full Integral Transformation, in Inter. J. Quantum. Cehm. S11, 149-163. Useful for Moller Plesset perturbation theory.

L.M.Cheung, S.T.Elbert and K.Ruedernberg (1978), "MCSCF optimization through the combined use of natural orbitals and the Brillouin-Levy-Berthier theorem. Section VI: Aspects of computational implementation." Inter. J. of Quantum Chem. (to be published). Useful when only single excitations from a reference list are present.

P.Pendergast and E.F.Hayes (1978), "A partial transformation for application to perturbation theory configuration interaction," J. Comp. Phys. 26, 236-242. Useful for Moller Plesset perturbation theory.

III. Matrix multiplications:

S.Winograd (1968), "A new algorithm for inner product," IEEE Trans. Comp. 17, 693-694. 1/2 n^3+n^2 multiplications and 3/2 n^3-n^2 additions.

V.Strassen (1969), "Gaussian elimination is not optimal," Numer. Math. 13, 354-356. Two $n \times n$ matrices can be multiplied together in $\leqslant 4.7\ n^{\log_2 7} \approx 4.7\ n^{2.81}$ total arithmetic operations.

R.P.Brent (1970), "Error analysis of algorithms for matrix multiplication and triangular decomposition using Winograd's identity," Numer. Math. 16, 145-156. Winograd's algorithm may be unstable without scaling.

A.Borodin (1973), "Computational Complexity: Theory and Practice," in Currents in the Theory of Computing, A.U.Aho, editor, (Prentice-Hall, Englewood Cliffs, N.J.), pp.35-89. Review of useful theorems for matrix multiplication.

R.W.Brockett and D.Dobkin (1973), "On the Optimal Evaluation of a Set of Bilinear Forms," Conference Record of Fifth Annual ACM Symposium of Theory of Computing, pp.88-95. $(\nu \times \mu)\times(\mu \times \omega)$ requires $\geqslant (\mu\nu + \mu\omega + \nu\omega) - (\mu + \nu + \omega) + 1$ multiplications.

S.Winograd (1971), "On multiplication of 2×2 matrices," Linear Algebra & Appl. 4, 381-388. Three additions removed from Strassen's recursion formula [for $(2\times2)\times(2\times2)$, 7 multiplications and 15 additions required].

IV. Sorting methods:

M.Yoshimine (1969), "The Use of Direct Access Devices in Problems Requiring The Reordering of Long Data Lists," Report RJ-555, IBM Research Laboratory, San Jose, CA.

D.Knuth (1975), The Art of Computer Programming, Vol. 3: Sorting and Searching. 2nd edition (Addison-Wesley, Reading, Mass.) Comprehensive survey of methods.

TABLE 2. Number of multiplications for various algorithms.

Method	Sum 1	Sum 2	Sum 3	Sum 4	Total for $n = N = M$
Tang & Edmiston (1970)	$N^2[N]^2M$	$N[N]^2[M]^2$	$N^2[N]^2M$	$N[N]^2[M]^2$	$(36n^5+24n^4+12n^3)/24$
Bender (1972) Shavitt (1977)	$[N]^4M1$	$2[N]^3[M]^2$	$[N]^2M[M]^2$	$4N[M]^4$	$(29n^5+65n^4+73n^3+43n^2+6n)/24$
Yoshimine (1970,1973)	$N^2[N]^2M$	$N[N]^2[M]^2$	$N^2M[M]^2$	$N[M]^4$	$(33n^5+42n^4+15n^3+6n^2)/24$
Elbert (1973)	$N^2[N]^2M$	$N[N]^2[M]^2$	$N^2[M]^3$	$N[M]^4$	$(29n^5+42n^4+19n^3+6n^2)/24$
Diercksen (1974)	$N^2[N]^2M$	$N[N]^2[M]^2$	$N^2[M]^3$	$N[M]^4$	$(29n^5+42n^4+19n^3+6n^2)/24$
Pendergast & Fink (1974)	$N^2[N]^2M$	$N[N]^2[M]^2$	$N^2[M]^3$	$N[M]^4$	$(29n^5+42n^4+19n^3+6n^2)/24$
Pounder (1975)	$N^2[N]^2M$	$N[N]^2[M]^2$	$N^2[M]^3$	$N[M]^4$	$(29n^5+42n^4+19n^3+6n^2)/24$
Elbert (1978)	$N^2[N]^2M$	$N[N]^2[M]^2$	$N^2([M]^2M1 -[M]^3)$	$N[M]^4$	$(25n^5+42n^4+23n^3+6n^2)/24$

$$M1 = M+1$$

$$[x]^2 = x^2/2 + x/2 = \sum_i^n i$$

$$[x]^3 = x^3/3 + x^2/2 + x/6 = \sum_i^n i^2$$

$$[x]^4 = x^4/8 + x^3/4 + 3x^2/8 + x/4 = [[x]^2]^2$$

TABLE 3. Elements transferred in and out of primary main storage.

Method	Expression for elements transferred	Total for $n = N = M$
Tang & Edmiston (1970)	Unknown, probably must be carried out entirely in fast storage.	
Bender(1972) Shavitt (1977)	$[N]^4 + [M]^4(2N+1)$	$(6n^5+18n^4+30n^3+30n^2+12n)/24$
Yoshimine (1970,1973)	$[N]^2[N]^2 + 2[N]^2[M]^2 + [M]^4$	$(21n^4+42n^3+27n^2+6n)/24$
Elbert (1973)	$[N]^2[N]^2 + 2[N]^2[M]^2 + [M]^4$	$(21n^4+42n^3+27n^2+6n)/24$
Diercksen (1974)	$[N]^2[N]^2 + 4[N]^2[M]^2 + [M]^4$	$(33n^4+66n^3+39n^2+6n)/24$
Pendergast & Fink (1974)	$\geqslant M[N]^4 + 2MN[N]^2 + 2[N]^2[M]^2 + 2N[M]^3 + [M]^4$	$\geqslant (3n^5+61n^4+87n^3+35n^2+6n)/24$
Pounder (1975)	$> M[N]^4 + [M]^4$	$> (3n^5+9n^4+15n^3+15n^2+6n)/24$
Elbert (1978)	$[N]^2[N]^2 + 2[N]^2[M]^2 + [M]^4$	$(21n^4+42n^3+27n^2+5n)/24$

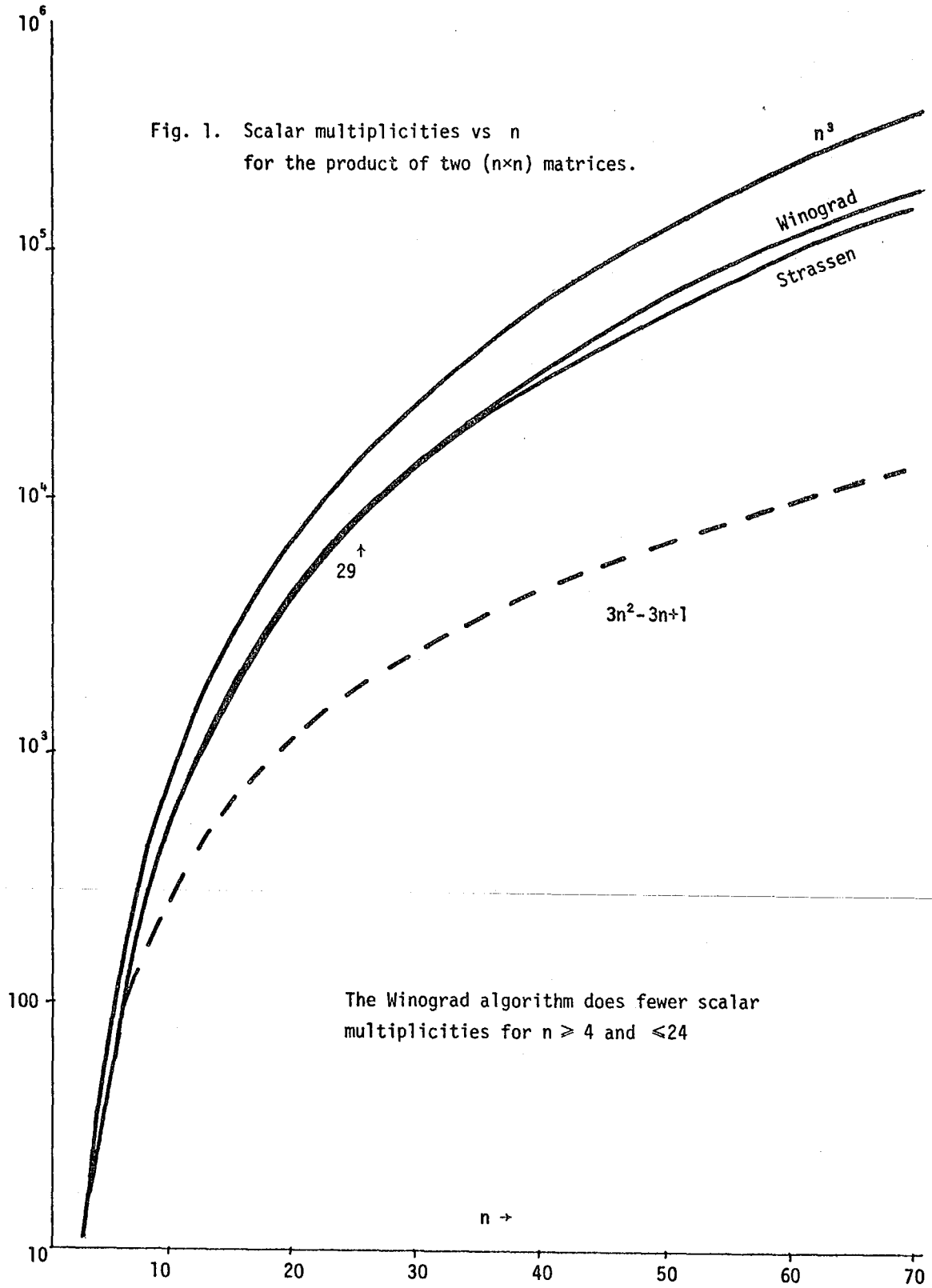


Fig. 1. Scalar multiplicities vs n for the product of two (n×n) matrices.

The Winograd algorithm does fewer scalar multiplicities for $n \geq 4$ and ≤ 24

Fig 2. Algorithm loop structure [*Elbert, 1978*]

```
r = 1 to N
    s = 1 to r
        p = 1 to N
            GPP = G(p,p,r,s)
            i = 1 to M
                S1(p,i) = T(p,i) * GPP
            q = 1 to p - 1
                GPQ = G(p,q,r,s)
                i = 1 to M
                    S1(q,i) = S1(q,i) + T(p,i) * GPQ
                    S1(p,i) = S1(p,i) + T(q,i) * GPQ
        S2(r,s,-,-) = 0
        i = 1 to M
            q = 1 to N
                SQI = S1(q,i)
                j = 1 to i
                    S2(r,s,i,j) = S2(r,s,i,j) + T(p,j) * SQI
```

```
i = 1 to M
    j = 1 to i
        r = 1 to N
            SRR = S2(r,r,i,j)
            k = RANGEK
                S3(r,k) = T(r,k) * SRR
            s = 1 to r - 1
                SRS = S2(r,s,i,j)
                k = RANGEK
                    S3(s,k) = S3(s,k) + T(r,k) * SRS
                    S3(r,k) = S3(r,k) + T(s,k) * SRS
        G'(i,j,-,-) = 0
        k = RANGEK
            s = 1 to N
                SSK = S3(s,k)
                if i = k then l = RANGEL else l = 1 to k
                    G'(i,j,k,l) = G'(i,j,k,l) + T(s,l) * SSK
```

Method	RANGEK	RANGEL	Multiplications when n = N = M
old	1 to i	1 to j	$(29n^5+42n^4+19n^3+6n^2)/24$
new	i to M	j to k	$(25n^5+42n^4+23n^3+6n^2)/24$

Fig. 3. Bender-Shavitt method without multipass I/O and without n^3 storage region.

```
mult.
          p = 1,N
            e' = 1
            q = 1,P
                SZ(p,q,-,-) = 0
                If p = q then e'=2
                r = 1,P
                    S1(-) = 0
                    S = 1,r for r ≠ p;  S = 1,9 for r = p
[N]^4                   If S = r then e = 2e
                        If rs = pq then e = 2e          elements required
                        GS = G(p,q,r,s)/e                  sequentially
[N]^4M                  i = 1,M
                            S1(i) = S1(i) + T(S,i)*GS
                    i = 1,M
                        SI = S1(i)
                        TRI = T(r,i)
                        j = 1,i
2[N]^3[M]^2                 S2(p,q,i,j) = S2(p,q,i,j) + TRI*S1(j) + T(r,j)*SI
```

(S2 stored on direct access device)

```
          i = 1,M
            j = 1,i
                p = 1,N
                    S3(-) = 0
                    q = 1,P
                        SPQ = S2(p,q,i,j)
                        K = 1,M
[N]^2[M]^2M                 S3(K) = S3(K) + T(q,K) + SPQ
                    S4A(i,j,-,-) = 0
                    K = 1,i
                        TPK = T(P,K)
                        SK = S3(K)
                        ℓ = 1,K for K ≠ i;  ℓ = i,j for k = i
2N[M]^4                     S4A(i,j,k,ℓ) = S4A(i,j,k,ℓ) + TPK*S3(ℓ) + T(P,ℓ)*SK
                    S4B(-,-,i,j) = 0
                    i^-1 = 1,M
                        TPI = T(P,i')
                        S1 = S3(p,i)
                        j' = 1,i' for i ≠ i, j' = j,i for i' = i
2M[M]^4                     S4B(i,j,i,j) = S4B(i,j,i,j) + TPI*S3(j') + T(Bj')*SI
```

(S4A and S4B stored on direct access device)

```
i = 1,M
    j = 1,i
        k = 1,i
            ℓ = 1,K for K ≠ i, ℓ = i,j for K = i
                G(i,j,k,ℓ) = S4A(i,j,k,ℓ) + S4B(i,j,k,ℓ)
```

OUTLINE OF YOSHIMINE'S TWO-PASS SORTING SCHEME

M. Yoshimine

IBM Research Laboratory
San Jose, California 95193

This algorithm requires that it should be possible to calculate the position of each item in the sorted list from information stored with the item.

Definitions ($\{n\}$ = smallest integer $\geqslant n$; $[n]$ = largest integer $\leqslant n$)

N = Number of items to be sorted.

K = Number of items which can be accommodated simultaneously in the central memory working array. This is the same as the capacity of a "segment" (or "core load").

L = $\{N/K\}$ = Number of segments into which the list is to be partitioned. This is also the number of "buckets" into which the central memory working array is to be divided during the first pass.

M = $[K/L] \approx K^2/N$ = Capacity of each bucket (i.e., the number of items it can hold). Each bucket belongs to one of the L segments (numbered 1,2, ...,L). When filled, each bucket is dumped as one record on a direct access file. The total number of records to be stored on the file during the first pass is approximately given by $\{N/M\} \approx (N/K)^2 \approx L^2$.

J = The position index of the current item in the final sorted list.

Procedure

Pass I : Read the items in the original list in order. For each item compute J (its position index in the final sorted list), then store the item in the next available position in bucket number $P = \{J/K\}$. Whenever a bucket is filled, write it out as a record on the direct access

file. Maintain a list of the locations of all the records belonging to each segment.

Pass II : Deal with the different segments in order, P = 1,2, ...,L. For the Pth segment, read the records belonging to it, one at a time, from the direct access file. For each item in these records, compute its final position index J, then store it in position J - (P-1)*K in the central memory working array. When all the records belonging to segment P have been processed, write out the central memory working array onto the final sorted file.

(Note that the central memory working array for K items is in addition to storage for various pointer arrays, buffers, etc.)

References

M.Yoshimine, J. Comput. Phys. 11, 449 (1973).

A.D.McLean, in Proceedings of the Conf. on Potential Energy Surfaces in Chemistry (W.A.Lester, editor), p.87 (Report RA-18, IBM Research Laboratory, San Jose, CA, January 1971).

P.S.Bagus, B.Liu, A.D.McLean, and M.Yoshimine, in Energy Structure and Reactivity (D.W.Smith and W.B.McRae, editors), p.130 (Wiley, New York, 1973).

STORAGE UTILIZATION AND SORTING

M. Yoshimine

IBM Research Laboratory
San Jose, California 95193

Introduction

Many problems in large-scale computation involve the manipulation of long data lists, typically containing millions of elements, that extend far beyond core storage capacity. For example, a list of two electron integrals produced in one order, dictated by the algorithm used for the most efficient computation of the integrals, may be needed in a quite different order for efficient processing at a later stage. Or, a large matrix with elements written onto a peripheral device by row may be needed later by column, thus requiring a reordering.

Reordering long data lists necessarily involves peripheral devices, and it is important to minimize the I/O time involved in the reordering process. This is the time for data transfer between the core and peripheral memory devices, including the access times to information on these devices. Non-optimal I/O procedures may result in substantially increased I/O times, and therefore elapsed time, before completion of a job. When the increase is from 4 hours to 160 hours, as is the case in the examples we discuss below, the substantial difference is an important consideration.

In the first section of this paper, we present a two-step ordering technique[1] using direct access devices in a way that minimizes I/O time. First, we outline the procedure and describe in detail the algorithm employed in a scaled-down example. We then present two more realistic examples estimating and discussing I/O times.

Outline of the Procedure

Suppose an input array $A'(a_i, i=1, N')$ is to be reordered to produce an output array $A(a_j, j=1,N)$, where i and j are position indices, and $N' \leqslant N$. It is assumed that the order in A is predetermined, that is,

a_i contains information on j's position in A. Further suppose that K is the maximum number of elements of the reordered list A that can be held simultaneously in core. The reordering is efficiently accomplished by the following two-step procedure.

Step 1. Sort A' into subarrays $A_1, A_2, \ldots, A_M$ with the number of elements in subarray A_i being $\leqslant K$. This can be done in one or more passes on the input array A' according to the formula, developed below, involving timing parameters of the appropriate I/O devices. It is assumed that elements of any subarray A_i can be efficiently reordered or processed in the core area reserved for K elements. An essential point is that elements of the subarray A_i be stored, in the manner described below, on a direct access device. The elements of the original array are used sequentially and can, therefore, be stored on magnetic tape.

Step 2. The elements of subarray A_i are retrieved from the direct access device and reordered, or otherwise processed, one subarray at a time. As previously indicated, the number of elements in a subarray is $\leqslant K$, and therefore all elements of the subarray can be finally processed without expensive intermediate I/O.

Detailed Description of the Procedure

The procedure will be described in detail, using a scaled-down example. Suppose that an input array A' has 20 elements ($N' = N = 20$) which are integers from 1 to 20 but randomly ordered (see Fig. 1). Suppose further that the available core space is 5 ($K = 5$), and the reordering will be done in one pass on the input array ($P = 1$). Then, the number of subarray A_i is 4 ($M = 1$) and the number of elements in A_i is 5. It is assumed that there is enough disk space available (direct access device).

Figure 1 shows the initial state of the input array A' (assumed to be a sequential data set), core and disk. The core contains four buffers with associated chain indices, and a disk record counter. The chain indices are set to zero initially.

The elements in A' are processed one at a time in a sequential order. The first element is 1, which belongs to A_1; it is therefore placed in buffer 1. The second is 17, which belongs to A_4, and is placed in buffer 4.

Figure 2 shows the core after the fourth element has been processed. Note that the disk record counter is still one, indicating that no disk record has been written.

Processing of the fifth element is slightly different. The element is 3, and thus belongs to A_1, but buffer one is full. One must write buffer 1 with its chain index onto the first record of the disk, replace the chain index for buffer 1 by the value in the disk record counter, increase the record counter by 1, and finally place 3 in buffer 1. Figure 3 shows the state of the core and disk after processing the fifth element. Note that the number on top in the disk record is the chain index.

After all input elements are processed (see Fig. 4), elements contained in the core buffers must be emptied out onto the disk, in the manner described above. Figure 5 shows the state of the core and the disk at the end of Step 1, which is also the initial state for Step 2. Note that the chain indices in the core are starting indices for retrieving the subarrays in Step 2.

Reorder the subarrays in Step 2 in the following manner: The chain index for the subarray A_1 is 17. Retrieve the 17th disk record and put its element in the appropriate place in the core (the 5th core storage in this particular example). Its chain index indicates that the next record to be retrieved is the 11th record. Process the 11th record in the same manner, and repeat this processing of the disk records until the chain index is zero, which signals that the subarray A_1 has been ordered. The ordered A_1 will then be outputted (see Fig. 6). Repeat this process for the remaining subarrays. When the last subarrays (A_4 in this example) have been ordered, the output array contains the desired ordered list (see Fig. 7).

This admittedly simple example illustrates the main points of the procedure. In real cases, millions of elements are processed and the size of the core is in the thousands. Thus, each buffer residing in the core and records on the disk will be large enough to contain several hundred elements. The main point is that Step 1 allows the subarray to be retrieved in its entirety with the help of the chain indices and (by the nature of the direct access device) without excessive intermediate I/O.

Following are two more realistic samples in which we use the two-step ordering technique described above.

Example 1. Two-electron integral list

Consider a list of integrals

$$A' \equiv \{(E,I)_i,\ i=1,2,\ldots,N\} \quad ,$$

where each element of n_i bytes consists of a floating point number E and an associated integer I. I is unique in the range $1 \leqslant I \leqslant N$, and the sequence of I values in the input array is arbitrary. A floating point number, E, is required to produce a reordered list of N elements The output list will be

$$A \equiv \{(E)_I,\ I=1,2,\ldots,N\} \ .$$

Suppose that $n_o K$ bytes of core storage are available for reordering the subarrays in Step 2. Then the definition of the μth subarray A_μ is

$$A_\mu \equiv \{(E,I)_i,\ (\mu-1)K < I \leqslant K\} \quad ,$$

and the number of such subarray is

$$M = [(N-1)/K] + 1 \quad . \tag{1}$$

(In these equations a square bracket denotes the integer part of the bracketed expression.)

Further suppose that the subarrays A are to be produced in P passes over the input array. Consideration of the optimum passes will be deferred to a later section which discusses the timing analysis. The number of subarrays produced in each pass is

$$m = [(M-1)/P] + 1 \quad , \tag{2}$$

and the subarrays produced in the pth pass are

$$A_\mu \ :\ (p-1)m < \mu < p_m \quad . \tag{3}$$

To produce m subarrays, we need m buffers in the core, and the length of each buffer is given by

$$L_c = [n_o K/n_i m] \quad . \tag{4}$$

The procedure for the pth pass is as follows:

Step 1. $(E,I)_i$ goes to the ℓth buffer if $(p-1)mK < I_i < pmK$, and

$$\ell = \left[\frac{((I_i-1) - (p-1)mK)}{K} \right] + 1 \quad . \tag{5}$$

When a buffer is full, write out onto the direct access device with the chain index, as described in the previous section.

Step 2. Process subarrays of A_μ as described in the previous section.

Example 2. Transposition of a large matrix

Suppose the elements a_{ij} of a matrix are stored by column

$$A' \equiv \{a_{11}, a_{21}, \ldots, a_{n_r 1}, a_{12}, \ldots, a_{n_r 2}, \ldots, a_{n_r n_c}\} ,$$

where n_r and n_c are, respectively, the number of rows and columns in the matrix. These elements must be reordered so that the transposed matrix is stored by column, that is, the final ordering must be

$$A \equiv \{a_{11}, a_{12}, \ldots, a_{1n_e}, a_{21}, a_{22}, \ldots, a_{2n_e}, \ldots, a_{n_r n_c}\} .$$

The meaning of the parameters K, M, m, L_c, P, n_o, n_i is as in Example 1, except that:

- K is chosen to accommodate an integral number of columns of the transposed matrix in Step 2; that is,

$$K = nn_c \quad , \tag{6}$$

  which leads to a simple algorithm for the Step 2 reordering.
- There is a definite relation between the initial and final ordering of the element a_{ij}. Thus, indices do not have to be stored with the matrix elements and we can set the ratio $n_i/n_o = 1$.

Subarrays of A_μ are

$$A_\mu \equiv \{a_{ij} \text{ with } (\mu-1)n < i < \mu n \text{ for all } j\} .$$

Subarrays produced in the pth pass are

$$A_\mu \; ; \quad (p-1)m < \mu < pm \quad ,$$

and the procedure on the pth pass is:

Step 1. a_{ij} goes to the ℓth buffer if $(p-1)m < i < pnm$, and

$$\ell = [(i-1)/n] + 1 - (p-1)m \quad . \tag{7}$$

Step 2. The following algorithm is used in reordering: the Rth element of any subarray is put into position S of the output block which receives the reordered elements where

$$S = n_c R_1 + R_2 + 1 \quad , \tag{8}$$

with

$$R_1 = (R-1)_{\text{mod } n} \quad ,$$

$$R_2 = [(R-1)/n] \quad .$$

We again note that Eq. (8) yields the correct results only when K is a multiple of n_e.

Timing Considerations

In this section we give (a) upper limits on the I/O time required by the process described for the examples, and (b) a formula for determining the optimum number of passes on the input array. We neglect CPU time because it is small compared to the I/O time required for the examples given above. It should be noted, however, that in determining the optimum number of passes for some complicated reordering procedure, the CPU time must be taken into account.

Maximum times required for Examples 1 and 2 are estimated for various values of parameters N, K, and P, and are compared with the times required by the non-optimum method described below.

I/O Time Required for the Two Examples

The total I/O time, T, can be divided into four parts, namely

$$T = TR + DW + DR + TW \quad , \tag{9}$$

where TR is the time required for the P passes over the input array; TW is the time for writing the output array. These two data sets are assumed to be sequential. DW and DR are times required for writing and reading subarrays on the direct access device, respectively. These four terms are estimated by

$$\left.\begin{aligned} TR &= TR_{eff} \times n_i \times N \times P \quad , \\ DW &= DW_{SAC}(KM/L_c) + DW_{eff} \times n_i \times N \quad , \\ DR &= DR_{RAC}(KM/L_c) + DR_{eff} \times n_i \times N \quad , \\ TW &= TW_{eff} \times n_o \times N \quad , \end{aligned}\right\} \qquad (10)$$

where TR_{eff}, DW_{eff}, DR_{eff}, and TW_{eff} are the effective transmission rates per byte. DW_{SAC} and DR_{RAC} are the sequential and random access times for the direct access device, and (KM/L_c) is the approximate number of records written. Use of Eqs. (1-3) to derive $KM/L_c = n_i M^2/n_o P$ and determination of the optimum value of $P = P_{opt}$ by the condition $dT/dP = 0$ gives

$$P_{opt} = \left(\frac{1}{K}\right)\left[\frac{(DW_{SAC} + DR_{RAC}) \times N}{n_o \times TR_{eff}}\right] \quad . \qquad (11)$$

Note that Eq. (11) assumes there is enough direct access device space available. If not, a higher P value dictated by available space must be chosen.

Non-Optimum Method

The non-optimum I/O method considered here reorders the core-load at one time without use of temporary data storage, and passes over the input array as many times as necessary. The I/O time, T', required for this method is given by

$$T' = TR_{eff} \times n_i \times N \times M + TW_{eff} \times n_o \times N \quad , \qquad (12)$$

where the first term is the time required to read the input array M times $(M = (N-1)/K+1)$ and the second is the time to write the output array.

Comparison of Times

Table 1, containing estimated I/O time for Example 1 for various P values with fixed N and K, clearly demonstrates the validity of Eq. (12). Tables 2 and 3 give estimated I/O time for Examples 1 and 2, respectively, for a number of N,K combinations with $P = P_{opt}$. These tables have been

prepared assuming that input and output arrays are stored on magnetic tape (IBM 3420-8), and the subarrays on disk (IBM 3330-II). The values of device-dependent parameters are

$$TR_{eff} = TW_{eff} = 0.9573 \ \mu s/byte \quad (6250 \ bpi) ,$$

$$DR_{eff} = DW_{eff} = 1.241 \ \mu s/byte ,$$

$$DW_{SAC} = 8.4 \ ms/record ,$$

$$DR_{RAC} = 38.4 \ ms/record .$$

Table 1 clearly shows that reading back of subarrays in Step 2 processing is the most time-consuming part of the procedure. However, the reading of the subarrays is not entirely random, and the value used for DR_{RAC} is almost certainly overestimated. Tables 2 and 3 clearly indicate the importance of the optimum I/O procedure. For instance the two-step procedure needs only 3.5 hours of I/O time while the other method takes 160 hours for reordering the integral list of $N = 10^8$ elements.

Concluding Remarks

The examples discussed above have been chosen because of their simplicity, so that the general procedure can be most clearly demonstrated. Modifications appropriate to other problems will not change the nature of the I/O processing.

These procedures have crucial applications in theoretical chemistry. For example, consider the so-called four index transformation

$$(ij|k\ell) = \sum_p \sum_q \sum_r \sum_s c_p^i \, c_q^j \, c_r^k \, c_s^\ell \, (pq|rs) .$$

which takes an input array of matrix element $(pq|rs)$, combines it with coefficients, and produces an output array $(ij|k\ell)$. Both input and output arrays can contain millions of elements. The reordering techniques discussed in this paper not only minimize the amount of I/O but also allow us to implement an algorithm, for combining $(pq|rs)$ with coefficients, in which the amount of computation is of the order n where n is the range of p,q,r,s. This minimizes the amount of computation, which is absolutely essential.

Another application can be found in constructing the CI Hamiltonian matrix, the element of which is given by

$$H_{IJ} = \sum_{pqrs} C^{IJ}_{pqrs} \, (pq|rs) \quad .$$

where C^{IJ}_{pqrs} are coefficients multiplying integral $(pq|rs)$. C^{IJ}_{pqrs} are in general generated in the order of the IJ index, and $(ps|rs)$ needed for a particular h_{IJ} may be scattered through the entire integral list. Thus, when the number of integrals is in the millions, the list of C^{IJ}_{pqrs}, which can also be in the millions, must be reordered so that h_{IJ} can be evaluated efficiently and with minimum I/O.[6]

References

1. *M.Yoshimine, "The use of direct access devices in problems requiring the reordering of long data lists," Report RJ-555, IBM Research Laboratory, San Jose, CA (1969).*

2. *A.D.McLean, "Potential energy surfaces from ab initio computation: current and projected capabilities of the ALCHEMY computer program," in Proceedings of the Conf. on Potential Energy Surfaces in Chemistry. Publication RA18, IBM Research Library, San Jose, CA (1971).*

3. *C.F.Bender, J. Comput. Phys. 9, 547 (1972).*

4. *G.H.F.Diercksen, Theor. Chim. Acta 33, 1 (1974).*

5. *S.T.Elbert (to be published).*

6. *M.Yoshimine, J. Comput. Phys. 11, 449 (1973).*

TABLE 1. Estimated I/O times for Example 1 for various numbers of pass with n_i = 12 bytes, n_o = 8 bytes, $N = 10^7$ and $K = 50 \times 10^3$.

P	1	2	3	4	5*	6
m	200	100	67	50	40	34
L_c	166	333	497	666	833	980
TR (sec)	115	230	345	460	574	689
DW (sec)	655	401	318	275	250	235
DR (sec)	2462	1302	922	725	610	541
TW (sec)	77	77	77	77	77	77
Total (sec)	3309	2010	1661	1537	1511	1541

TABLE 2. Estimated I/O times for Example 1, with n_i = 12 bytes and n_o = 8 bytes, for various N, K, and M values.

N	K	M	P_{opt}	T	T'
10^6	25	40	3	111 sec	467 sec
10^6	50	20	2	74 sec	237 sec
10^6	100	10	1	56 sec	123 sec
10^7	25	400	10	44.1 min	767 min
10^7	50	200	5	25.2 min	384 min
10^7	100	100	3	15.9 min	193 min
10^8	25	4000	32	21.1 hrs	1277 hrs
10^8	50	2000	16	11.0 hrs	538 hrs
10^8	100	1000	8	6.1 hrs	319 hrs
10^8	200	500	4	3.5 hrs	160 hrs

Note: Only times for P_{opt} are listed and T' are corresponding estimated times for the non-optimal method.

TABLE 3. Estimated I/O times for Example 2 for various N, K, and M values, with $n_i = n_o = 8$ bytes.

N	K 10	M	P_{opt}	T	T'
10^6	25	40	3	77 sec	313 sec
10^6	50	20	2	52 sec	161 sec
10^6	100	10	1	40 sec	84 sec
10^7	25	400	10	29.8 min	512 min
10^7	50	200	5	17.2 min	257 min
10^7	100	100	2	11.0 min	129 min
10^8	25	4000	32	14.1 hrs	851 hrs
10^8	50	2000	16	7.4 hrs	425 hrs
10^8	100	1000	8	4.1 hrs	213 hrs
10^8	200	500	4	2.4 hrs	107 hrs

Note: Only times for P_{opt} are listed and T' are corresponding estimated times for the non-optimum I/O method.

Fig. 1. The initial state of the input array A', core, and disk for Step 1.

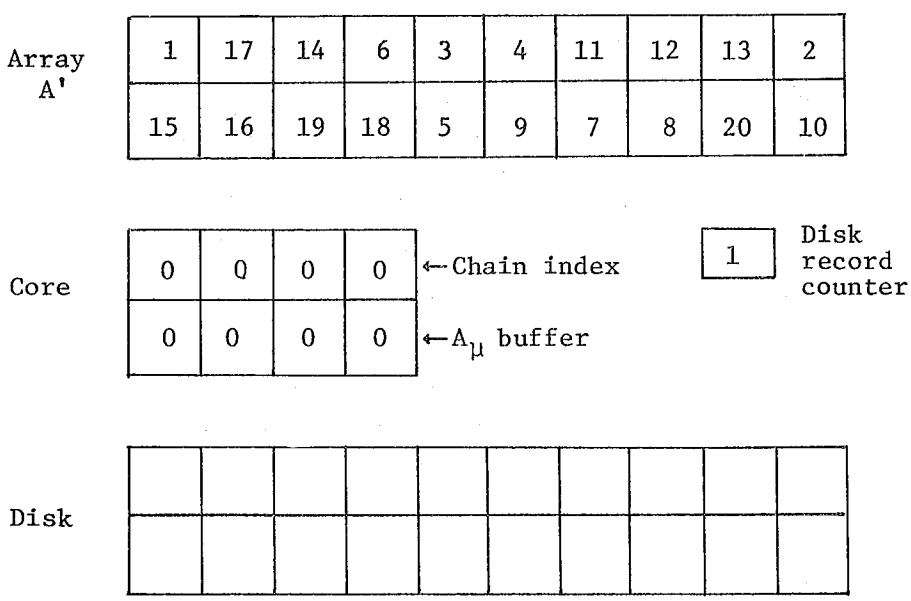


Fig. 2. State of the array A', the core, and disk after the 4th element has been processed.

A'

				3	4	11	12	13	2
15	16	19	18	5	9	7	8	20	10

Core

0	0	0	0
1	6	14	17

1

Disk

Fig. 3. State of the array A', core, and disk after the 5th element has been processed.

A'

					4	11	12	13	2
15	16	19	18	5	9	7	8	20	10

Core

1	0	0	0
3	6	14	17

2

Disk

0 1									

Fig. 4. State of the array A', core, and disk after all the elements have been processed.

A'

Core

11	16	7	10
5	10	15	18

17

0 1	1 3	0 14	3 11	4 12	2 4	5 13	0 17	8 16	9 19
6 2	0 6	12 19	13 7	10 18	14 8				

Fig. 5. The initial state of the disk, core, and output array A for Step 2.

Disk

0 ~~1~~	1 ~~3~~	0 ~~14~~	3 ~~11~~	4 ~~12~~	2 ~~4~~	5 ~~13~~	0 ~~17~~	8 ~~16~~	9 ~~19~~
6 2	0 6	12 9	13 7	10 18	14 8	11 5	16 10	7 15	15 20

Core

17	18	19	20	

Array A

Fig. 6. State of the disk, core, and output array A after the subarray A_1 has been processed.

		0 14	3 11	4 12		5 13	0 17	8 16	9 19
	0 6	12 9	13 7	10 18	14 8		16 10	7 15	15 20

	18	19	20	
1	2	3	4	5

1	2	3	4	5					

Fig. 7.

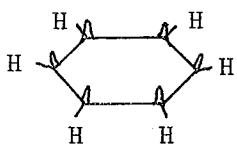
16	17	18	19	20

Array

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20

A DIFFERENT APPROACH TO INTEGRALS AND INTEGRAL TRANSFORMATIONS OR, DO WE REALLY NEED ALL THE INTEGRALS?

N.H.F.Beebe and Jan Linderberg



Molecule with basis functions $\phi_i(\vec{r})$ on each center ($\underline{N}$ in all). Compute integrals over 1-electron operators like ∇^2, $1/r$, $1 \rightsquigarrow N^2$ one-electron integrals plus integrals over $1/r_{12}$:

$$(ij|k\ell) = \iint \phi_i^*(1)\ \phi_j(1)\ \frac{1}{r_{12}}\ \phi_k^*(2)\ \phi_\ell(2)\ d1\,d2$$

$\rightsquigarrow \boxed{N^4}$ 2-electron integrals $\qquad \boxed{N \sim 20 \rightarrow 200}$

Hartree-Fock calculation:

Solve $\quad \mathbb{F}\mathbb{C} = \mathbb{S}\mathbb{C}\epsilon$

where $\quad f_{ij} = h_{ij} + \sum_{k\ell} \{2(ij|k\ell) - (i\ell|kj)\}\ P_{k\ell}$

and $\quad \mathbb{P} = \mathbb{C}\mathbb{C}^+$

from last interaction

and

$$s_{ij} = \langle \phi_i | \phi_j \rangle \equiv (\phi_i, \phi_j)$$

$$h_{ij} \sim \langle \phi_i | \tfrac{1}{2}\nabla^2 + \frac{Z}{r} | \phi_j \rangle$$

i.e., principle problem is data handling of N^4 $(ij|k\ell)$'s. After successful solution of Hartree-Fock problem, we want to form

$$\Psi = \Phi\mathbb{C} \qquad \text{[linear combination of } \phi\text{'s]}$$

and construct 2-electron integrals over ψ's by

$$(\alpha\beta|\gamma\delta) = \sum_i^N C_{i\alpha} \sum_j^N C_{j\beta} \sum_k^N C_{k\gamma} \sum_\ell^N (ij|k\ell) C_{\ell\delta}$$

These are used to construct matrix elements for the large sparse problem

$$\mathbb{H}\mathbb{C}' = \mathbb{S}'\mathbb{C}'\mathbb{E}$$

where each $\mathbb{H}_{ij}$ may involved up to M^2 2-electron integrals (M = number of electrons $\approx 10 \rightarrow 100$).

<u>Problems</u>:

1) Summation is N^5 problem.
2) If $\mathbb{S}$ in $\mathbb{F}\mathbb{C} = \mathbb{S}\mathbb{C}\epsilon$ is numerically almost singular, some C's are large and have both plus and minus signs $\Rightarrow$ error propagation into *every* $(\alpha\beta|\gamma\delta)$; e.g., must have $(\alpha\alpha|\beta\beta) > 0$ and $(\alpha\beta|\beta\alpha) > 0$.
3) Often bottleneck of computational quantum chemistry.

Obs: $(ij|k\ell) = V_{ij,k\ell}$ is matrix element of a *positive definite* operator $1/r_{12}$ $\Rightarrow$ $|\mathbb{V}| > 0$.

Idea: [*Jan Linderberg, September 1976*]

If $\mathbb{V}\mathbb{U} = \mathbb{U}\lambda$

$\uparrow$ eigenvalues

then $\mathbb{V} = \mathbb{U}\lambda\mathbb{U}^+$

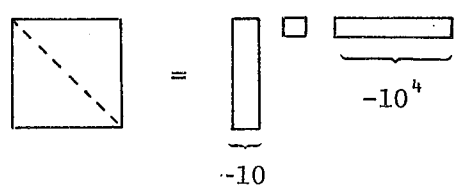
or $$V_{ij,k\ell} = \sum_{M}^{(N^2)} U_{ij,M}\,\lambda_M\,U^*_{k\ell,M}$$

Note: $\mathbb{V}$ contains N^4 numbers constructed from N functions.

If rank $(\mathbb{V})$ is small, then

$$\mathbb{V}_{ij,k\ell} \approx \sum_{M=1}^{10} U_{ij,M}\,\lambda_M\,U^*_{k\ell,M}$$

or



, i.e., data reduction!

NHFB: 1) What if 500, not 10, is upper limit $\Rightarrow$ currently impossible.
2) It's a lot of work for me.

Idea: [*JL, October 1976*]

$$\mathbb{U} > 0 \Rightarrow \mathbb{V} = \mathbb{L}\,\mathbb{L}^{+} \qquad \text{[Cholesky]}$$

$$\mathbb{L} \quad \mathbb{V}$$

Suppose

$$V_{ij,k} \approx \sum_{M=1}^{10} L_{ij,M}\, L^{*}_{k\,,M}$$

What if?

$$\left.\begin{aligned} L_{JJ} &= \left(V_{JJ} - \sum_{K=1}^{J-1} L_{JK}^{2}\right)^{\frac{1}{2}} \\ L_{IJ} &= \frac{\left(V_{IJ} - \sum_{K=1}^{J-1} L_{IK} L_{JK}\right)}{L_{JJ}} \end{aligned}\right\} \begin{aligned} &J = 1 \to M \\ &(I = J+1 \to M) \end{aligned}$$

<u>Algorithm</u>

1) Get all V_{JJ} and arrange in non-increasing order, remembering original order.
2) Select largest (i.e., V_{11}). Form $L_{11} \leftarrow V_{11}^{\frac{1}{2}}$.
3) Get entire column ($\equiv$ row) of U: V_{I1}, $(I = 1,\ldots M)$.
4) Set $L_{I1} \leftarrow V_{I1}/L_{I1}$ $(I = 2 \to M)$
5) Update $V_{II} \leftarrow V_{II} - L_{I1}L_{I1}$ $(I = 2 \to M)$
6) Cycle where at step J, V_{JJ} is largest remaining diagonal element:
   a) $L_{JJ} \leftarrow V_{JJ}^{\frac{1}{2}}$
   b) Get column V_{IJ} $(I = J+1 \ldots M)$
   c) $L_{IJ} = \left(V_{IJ} - \sum_{K=1}^{J-1} L_{IK} L_{JK}\right) / L_{JJ}$ $(I = J+1 \ldots M)$
   d) Update $V_{II} \leftarrow V_{II} - |L_{IJ}|^{2}$

STOP! when largest remaining element $V_{II} < \delta$ (δ is user-specified).
Call ν the number of columns of $\mathbb{L}$ computed ($\approx$ effective rank).
Advantages:

1) Only need ν (*not* N^2) rows of V.
2) Recompute whole $\mathbb{V}$ from

$$V_{IJ} = \sum_{K=1}^{\min(I,J,\nu)} L_{IK} L_{JK}$$

3) $\langle 1/r_{12} \rangle$ is rigorous lower bound if generated from truncated $L L^{+}$ (would actually prefer upper bound, but no one has any bound yet, and δ is small).

4) 2-electron integral transformation becomes

$$\mathbb{L}'_{*,K} = \mathbb{C}^{+} \mathbb{L}_{*,K} \mathbb{C}$$

i.e., ν 1-electron transformations, or νN^3 work. Form

$$(\alpha\beta|\gamma\delta) = \sum_M L'_{\alpha\beta,M} L'_{\gamma\delta,M}$$

in νN^4 work (one dot product/integral!).

5) Less propagation of errors for large $\mathbb{C}$'s.

6) Adjustable tolerance: some $(\alpha\beta|\gamma\delta)$ can be computed less accurately than others.

7) Possible to construct $\mathbb{F}$ from $\mathbb{L}$ directly, so reading νN^4 numbers each iteration, instead of N^4.

8) Linear dependencies *beneficial*! (small ν).

9) Symmetry properties conveniently included.

10) Program simpler.

11) Lots of *DOT PRODUCTS*

12) Can estimate $(\alpha\beta|\gamma\delta) = \sum_M L_{\alpha\beta,M} L_{\gamma\delta,M}$ from few terms (or one).

Disadvantages: only one or two,

1) What if $\nu \approx N^2$; then work $\sim N^5$ to form $\mathbb{L}'$, and N^6 to form $(\alpha\beta|\gamma\delta)$, [plus I/O!].

2) If ν is large, each new column of $\mathbb{L}$ requires disk access to all previous columns.

Other ideas:

1) Apply to linear equations and least-squares.

2) Use Winograd's idea:

$$A = \mathbb{B}\mathbb{C}$$

$$\Rightarrow A_{ij} = \sum_{k=1}^{N/2} (B_{i,2k-1} + C_{2k,j})(B_{i,2k} + C_{2k-1,j}) - (\xi_i + \eta_j)$$

Reference: *N.H.F.Beebe and J.Linderberg, Int. J. Quantum Chem. 12, 683 (1977).*

$$\xi_i = \sum_{k=1}^{N/2} B_{i,2k-1} B_{i,2k}$$

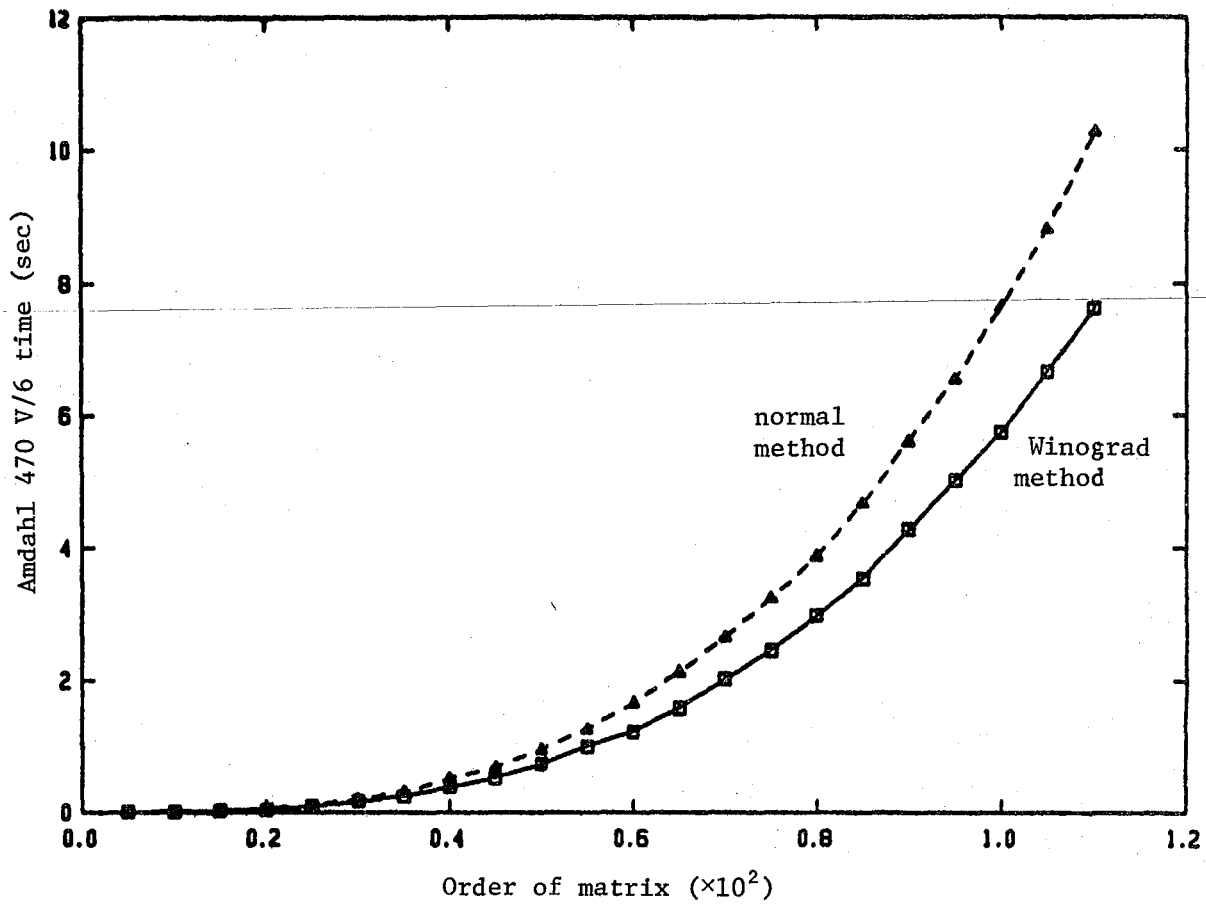
$$\eta_j = \sum_{k=1}^{N/2} C_{2k-1,j} C_{2k,j}$$

Normal method: N^3 multiplications + N^3-N^2 additions

Winograd: $1/2\ N^3 + N^2$ multiplications + $3/2\ N^3 + 2N(N-1)$ additions

Brent: 1) Scale to prevent B + C accuracy loss ($\sim N^2$).
2) Apply to Gaussian elimination, Cholesky et al.

Matrix multiplication (double precision with extended precision accumulation)



☆ US GOVERNMENT PRINTING OFFICE: 1979-690-867/187

This report was done with support from the Department of Energy. Any conclusions or opinions expressed in this report represent solely those of the author(s) and not necessarily those of The Regents of the University of California, the Lawrence Berkeley Laboratory or the Department of Energy.

Reference to a company or product name does not imply approval or recommendation of the product by the University of California or the U.S. Department of Energy to the exclusion of others that may be suitable.

TECHNICAL INFORMATION DEPARTMENT
LAWRENCE BERKELEY LABORATORY
UNIVERSITY OF CALIFORNIA
BERKELEY, CALIFORNIA 94720