ADAPTING ITERATIVE ALGORITHMS FOR SOLVING LARGE SPARSE LINEAR SYSTEMS FOR EFFICIENT USE ON THE CDC CYBER 205

DAVID R. KINCAID
AND
DAVID M. YOUNG

CENTER FOR NUMERICAL ANALYSIS
UNIVERSITY OF TEXAS AT AUSTIN

AUSTIN, TEXAS

ADAPTING ITERATIVE ALGORITHMS FOR SOLVING LARGE SPARSE LINEAR SYSTEMS FOR EFFICIENT USE ON THE CDC CYBER 205

David R. Kincaid
David M. Young

Center for Numerical Analysis
University of Texas at Austin
Austin, TX 78712

* Adapting and designing mathematical software to achieve optimum performance on the CYBER 205 will be discussed

* Comments and observations are made in light of recent work done at the Center for Numerical Analysis on

- modifying the ITPACK software package
- writing new software for vector supercomputers

Research goal - develop very efficient vector algorithms and software for solving large sparse linear systems using iterative methods

(older) SCALAR APPROACH - develop algorithms that minimize either number of iterations or arithmetic operations

* Not necessarily the correct approach for vector computers *

(newer) VECTOR APPROACH - avoid operations such as table lookups, indirect addressing, etc. that are inefficient on a vector computer, i.e., non-vectorizable

* Fully vectorizable code may involve more arithmetic operations but can be executed at a very high rate of speed *

* Advances in high performance computers and in computer architecture necessitates additional research in mathematical software to find suitable algorithms for the supercomputers of today and of the future *

THE VECTORIZATION OF THE ITPACK SOFTWARE PACKAGE

Scalar ITPACK:

package for solving large sparse linear systems
7 iterative algorithms available
sparse storage format used
Kincaid, Respess, Young, & Grimes [1982]
ITPACK 2C (ALGORITHM 586) in T.O.M.S.
"Transactions on Mathematical Software"

VECTORIZATION:

- First step: look for obvious vectorization changes since this was a large package of over 11,000 lines of code and we did not want to completely rewrite it

- Vector ITPACK (standard Fortran version): used a minimum of vector syntax available in CYBER 200 Fortran for a portable version of Vector ITPACK 2C

- Vector ITPACK (CYBER 205 version): a modified version of Vector ITPACK written using CYBER 200 Fortran vector syntax where possible

- DO loops which had been unrolled for scalar optimization were not recognized as vectorizable by optimizing vector compilers

- These loops were rewritten as simple tight DO loops so that they would be executed in vector mode

- The sparse storage scheme used for the matrix in Scalar ITPACK was row-oriented and inhibited vectorization (The IA-JA-A data structure as in Yale software YSMP used.)

- A column-oriented data structure was used in Vector ITPACK to increase vectorization (The COEF-JCOEF data structure as in Purdue software ELLPACK used.)

- The version of Vector ITPACK specifically for the CYBER 205 was tested on the CYBER 205 at Colorado State University (CSU) and has been added to their Program Library

- The improvements in time of the vector syntax version over the one in standard Fortran were not as significant as we had anticipated

- The automatic vectorization available in the CYBER 205 Fortran compiler did a very good job of optimization and vectorization

Moral: vector syntax best when used in designing and writing new code

PROBLEM:

$$\begin{cases} u_{xx} + 2\,u_{yy} = 0 & \text{on } S=(0,1)\times(0,1) \\ u = 1 + xy & \text{on boundary of } S \end{cases}$$

Discretization: standard 5-point finite difference formula

Stopping Criterion: 5.0×10^{-6}

Mesh Sizes: 1/16; 1/32; 1/64; 1/128; 1/256

Number of Unknowns: 225; 961; 3969; 16,129; 65,025

Computer: CSU CYBER 205

CYBER 200 Fortran: Large pages, unsafe vectorization

Scalar ITPACK (unrolled DO-loops & YALE storage used; T.O.M.S. version)

Modified Scalar ITPACK (rolled DO-loops & minor changes: Q8SDOT used)

Vector ITPACK (rolled DO-loops, ELLPACK storage, & CYBER 200 Fortran vector syntax used)

TABLE I: CHANGING SPARSE STORAGE

(Iteration Times in Seconds with H = 1/64)

Method	Iterations	Scalar ITPACK	Modified Scalar ITPACK	Vector ITPACK
(Natural Ordering)				
JACOBI CG	178	2.509	2.184	.262
JACOBI SI	362	5.214	4.480	.580
SOR	216	4.700	4.597	2.453
SSOR CG	34	1.976	1.788	.831
SSOR SI	43	1.791	1.682	.970
(Red-Black Ordering)				
JACOBI CG	178	2.402	2.056	.268
JACOBI SI	362	4.987	4.209	.590
SOR	196	4.110	4.017	.523
SSOR CG	341	20.327	18.472	2.177
SSOR SI	196	7.734	6.690	.701
RS CG	90	1.445	1.358	.118
RS SI	182	2.980	2.779	.223

TABLE II: CHANGING PROBLEM SIZE

(Number of Iterations)

Method	H = 1/16	1/32	1/64	1/128	1/256
(Natural Ordering)					
JACOBI CG	49	94	178	330	629
JACOBI SI	56	179	362	772	1372
SOR	50	104	216	422	872
SSOR CG	16	22	34	51	73
SSOR SI	19	29	43	61	88
(Red-Black Ordering)					
JACOBI CG	49	94	178	330	629
JACOBI SI	56	179	362	772	1372
SOR	52	101	196	396	839
SSOR CG	34	62	341	1058	3061
SSOR SI	51	107	196	373	752
RS CG	25	48	90	167	321
RS SI	42	88	182	375	704

TABLE III: CHANGING PROBLEM SIZE

(Iteration Time in Seconds)

Method	H = 1/16	1/32	1/64	1/128	1/256
(Natural Ordering)					
JACOBI CG	.010	.040	.251	1.800	14.115
JACOBI SI	.014	.091	.560	4.196	28.741
SOR	.035	.292	2.446	19.828	164.940
SSOR CG	.027	.133	.828	4.953	28.157
SSOR SI	.029	.163	.967	5.583	32.249
(Red-Black Ordering)					
JACOBI CG	.010	.041	.257	1.847	14.511
JACOBI SI	.013	.091	.571	4.277	29.394
SOR	.011	.066	.475	4.028	34.939
SSOR CG	.018	.075	2.105	25.779	302.712
SSOR SI	.021	.113	.663	4.452	36.053
RS CG	.006	.019	.109	.757	5.981
RS SI	.008	.033	.207	1.557	11.881

COMMENTS ON TABLE I

- Two versions of Scalar ITPACK were compared with the CYBER 205 version of Vector ITPACK

- Mesh size H = 1/64 used for all runs

- Scalar ITPACK: unrolled DO-loops used in basic vector operations for increased optimization on scalar computers

- Modified Scalar ITPACK: standard tight DO-loops used

- Vector Fortran compiler recognizes tight loops as vectorizable but not unrolled loops

- A slight increase in speed from Scalar to Modified Scalar version

- Vector ITPACK uses tight loops, Fortran vector syntax, and a column-oriented sparse storage scheme

- This data structure allows the matrix-vector product operation to vectorize to a great extent

* Considerable improvement in performance from scalar to vector version of ITPACK *

COMMENTS ON TABLE II & III

- These tables are results of using Vector ITPACK on the same problem with varying mesh sizes

- The number of iterations increase as the problem size increase

- Comparisons based on number of iterations misleading as to the best method!

- On scalar computers, SOR with natural ordering is widely used while JACOBI is not but on vector computers ...

- Most efficient method on the CYBER 205:

JACOBI CG method when natural ordering is used

RS CG when red-black ordering is used

SCALAR ITPACK vs. VECTOR ITPACK

- Total time for each method is not significantly greater than the iteration time in the vector version (this was not the case in the scalar version)

- Only N additional workspace locations required for the vector version over the scalar version

- Faster scaling and permuting of the system with the column-oriented sparse storage scheme

- Improved performance of the SSOR methods with the red-black ordering in the vector version in spite of the greater number of iterations

A PRE-CONDITIONED CONJUGATE GRADIENT PACKAGE

Thomas C. Oppe, a graduate student at UT Austin, is working on a package which allows flexibility in the choice of basic methods and acceleration schemes.

The package has been designed to make the addition of further preconditionings and acceleration schemes easy.

Particular attention has been paid to the choice of matrix storage schemes with a view to maximizing vectorizability.

Features of Package:

- Conjugate Gradient Acceleration

- Pre-conditioning matrix Q (Jacobi, Symmetric Successive Overrelaxation, Reduced System, Incomplete Cholesky, Modified Incomplete Cholesky, Neumann Polynomial, Parameterized Polynomials, Other pre conditionings planned such as Incomplete Block Cyclic Reduction)

- Realistic Stopping Tests

- Automatic estimation of iteration parameters with adaptive procedures

- Two possible data structures allowed

DATA STRUCTURES

Data structures which allow vectorization to varying degree:

EXAMPLE:

$$A = \begin{vmatrix} 4 & -1 & -2 & 0 \\ -1 & 4 & 0 & -2 \\ -2 & 0 & 4 & -1 \\ 0 & -2 & -1 & 4 \end{vmatrix}$$

ELLPACK Data Structure:

$$COEF = \begin{vmatrix} 4 & -1 & -2 \\ 4 & -2 & -1 \\ 4 & -1 & -2 \\ 4 & -2 & -1 \end{vmatrix} \qquad JCOEF = \begin{vmatrix} 1 & 2 & 3 \\ 2 & 4 & 1 \\ 3 & 4 & 1 \\ 4 & 2 & 3 \end{vmatrix}$$

- matrix-vector product vectorizes with the use of gathering routines

- operations such as forward (back) substitutions using lower (upper) triangular matrices do not vectorize

DIAGONAL Data Structure:

$$COEF = \begin{vmatrix} 4 & -1 & -2 \\ 4 & 0 & -2 \\ 4 & -1 & * \\ 4 & * & * \end{vmatrix} \qquad JCOEF = (0,\ 1,\ 2)$$

- the matrix-vector product operation vectorizes without the use of gathering routines

- operations such as forward (back) substitution and factorizations vectorize to some extent

REFERENCES

David R. Kincaid, John R. Respess, David M. Young, and Roger G. Grimes, "ALGORITHM 586 ITPACK 2C: A FORTRAN Package for Solving Large Sparse Linear Systems by Adaptive Accelerated Iterative Methods", ACM Transactions on Mathematical Software, Vol. 8, No. 3, September 1982.

David R. Kincaid, Tom Oppe, and David M. Young, "Adapting ITPACK Routines for Use on Vector Computers," Report CNA-177, Center for Numerical Analysis, University of Texas at Austin, TX, August 1982. (In the Proceedings of the 1982 Symposium on CYBER 205 Applications, Institute for Computational Studies at Colorado State University, Fort Collins, CO.)

David R. Kincaid and Thomas C. Oppe, "ITPACK on Supercomputers", Report CNA-178, Center for Numerical Analysis, University of Texas at Austin, TX, September 1982. (To appear in the Proceedings of the InterAmerican Workshop on Numerical Methods, Springer-Verlag, NY.)

David R. Kincaid and David M. Young, Jr., "The ITPACK Project: Past, Present, and Future", Report CNA-180, Center for Numerical Analysis, University of Texas at Austin, TX, March 1983. (To appear in ELLIPTIC PROBLEM SOLVERS II, Academic Press, NY.)