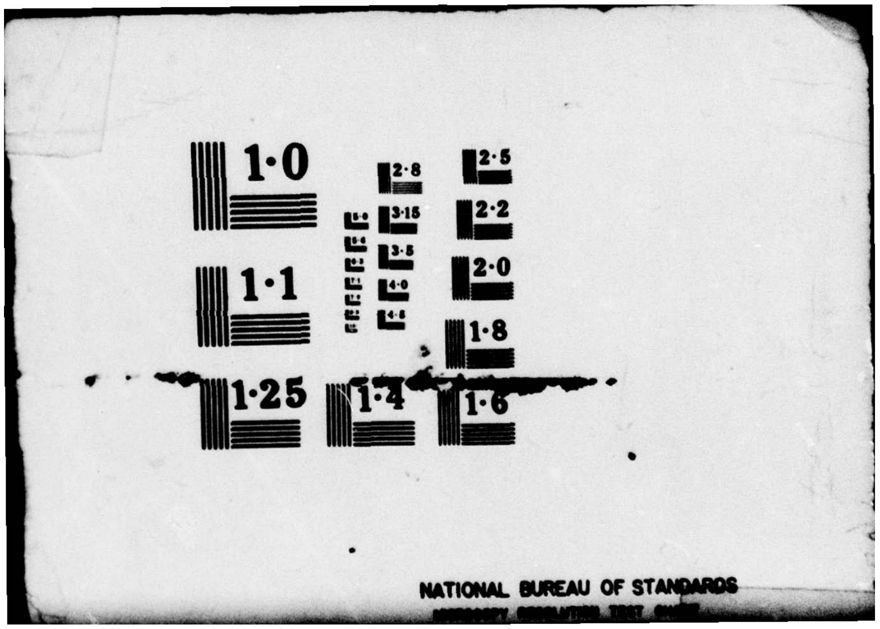
AD-A073 033 MICHIGAN UNIV ANN ARBOR SYSTEMS ENGINEERING LAB F/G 9/2
A COLLECTION OF EQUATION-SOLVING CODES FOR THE CRAY-1.(U)
AUG 79 D A CALAHAN, W G AMES, E J SESEK AFOSR-75-2812
UNCLASSIFIED SEL-133 AFOSR-TR-79-0932 NL

1 OF 1
AD
A073033

END
DATE
FILMED
9 79
DDC


1·0
2·8
2·5
3·15
2·2
3·5
2·0
4·0
4·5
1·1
1·8
1·25
1·4
1·6

SEL Report No. 133

A Collection of Equation-Solving Codes for the CRAY-1

D. A. Calahan
W. G. Ames
E. J. Sesek

August 1, 1979

Sponsored Jointly by
Directorate of Mathematical and Information Sciences,
Air Force Office of Scientific Research, and
Air Force Flight Dynamics Laboratory,
Wright-Patterson Air Force Base,
under Grant 75-2812

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
SYSTEMS ENGINEERING LABORATORY
THE UNIVERSITY OF MICHIGAN, ANN ARBOR

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(19) REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER (18) AFOSR-TR-79-0932	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) (6) A COLLECTION OF EQUATION-SOLVING CODES FOR THE CRAY-1.		5. TYPE OF REPORT & PERIOD COVERED (9) Interim rept.
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) (10) D.A. Calahan, W.G. Ames, E.J. Sesek		8. CONTRACT OR GRANT NUMBER(s) (15) AFOSR-75-2812
9. PERFORMING ORGANIZATION NAME AND ADDRESS The University of Michigan Electrical and Computer Engineering Department Ann Arbor, Michigan 48109		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61102F (16) 2304 (17) A3
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Office of Scientific Research/NM Bolling AFB, Washington, D.C. 20332		12. REPORT DATE (11) 1 ~~August 1~~ Aug 1979
		13. NUMBER OF PAGES 18
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) (14) SEL-133 (12) 23p.		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Vector Processing, Linear algebra

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report contains user documentation and timing results for a collection of assembly language equation-solving codes for the CRAY-1.

400 704 LB

DD FORM 1 JAN 73 1473

UNCLASSIFIED

A Collection of Equation-Solving Codes for the CRAY-1

D. A. Calahan

W. G. Ames

E. J. Sesek

Systems Engineering Laboratory
University of Michigan
Ann Arbor, Michigan 48109

August 1, 1979

SEL Report #133

Sponsored Jointly by
Directorate of Mathematical and Information Sciences
Air Force Office of Scientific Research, and
Air Force Flight Dynamics Laboratory
Under Grant 75-2812

Abstract

This report contains user documentation and timing results for a collection of assembly language equation-solving codes for the CRAY-1.

Acknowledgement

The development of these codes was made possible by the development of a CRAY-1 simulator by D. A. Orbits. The programming assistance of S. Arya is also acknowledged.

TABLE OF CONTENTS

I. Introduction

A. Implicitness

The most common application envisioned in the design of vector processors has been the solution of partial differential equations. It is now clear that many application-oriented researchers are planning to use this vastly increased computational capability to make solution algorithms more implicit [1]. For example, an algorithm implicit by line (i.e., in one dimension) would become implicit by strips ($1+\eta$ dimensions); or, more variables may be coupled in a multi-variable problem. It is usually found that the larger the problem and/or the more implicit the algorithm, the greater fraction of total formulation and solution time is devoted to the latter. The coding of the equation-solver then becomes a critical issue.

B. Vector Length

The vector length is the most obvious and general concern in vector processing. The length results either from simultaneous operations on a number of similar systems or from the density of structure (coupling of variables and grid nodes) within a single system.

In the solution of partial differential equations, the frequency of variable updating determines the number of grid node equations that can be formulated simultaneously. A point-Jacobi iteration would allow simultaneous formulation and updating of all variables. An ADI method would allow simultaneous updating of variables along a line. The coupling between grid points and between lines of grid points assumed in the equation solution determines the number of systems of equations which can be solved simultaneously. Thus, if grid points are assumed coupled in only one of two dimensions, then one can expect a system of tridiagonal or block tridiagonal matrices which can be solved simultaneously. This in turn yields a vector length equal to the number of uncoupled grid lines. On the other hand, an ADI method necessitates a solution of a single tridiagonal system.

The most obviously vectorizable algorithm would therefore be one in which variables are simultaneously updated and minimal coupling is assumed between grid nodes and/or variables at a grid point.

Iterative methods based on such schemes have notoriously poor convergence. However, if simultaneity--and hence vector length--is reduced to increase convergence rate, one must exploit single system density to achieve long vectors. This density is usually manifested by relatively small block sizes, bandwidths, or profiles in the matrix structure.

C. Data Flow

With attention only to the vector length and with the use of a high level language such as Fortran, it is not uncommon to obtain only 20%-50% of the optimum CRAY-1 performance in the equation solution. The data flow must also be considered to achieve high performance, as illustrated by the following instances associated with equation-solving codes.

(a) With short vector chained operations, the CRAY-1 protocol results in large bubbles in the arithmetic pipelines [2]. This occurs in the processing of small dense blocks or narrow bandwidths.

(b) Vectors of any length on which little computation is performed can create excessive data flow between memory hierarchies. This situation prevails in the above-mentioned simultaneous solution of equations, and results from the inherent decoupling of such systems.

(c) General equation-solving codes--ones which can accomodate arbitrary problem size parameters--may suffer from excessive data flow visa vis a special code written for small problem sizes which can maintain critical data in the cache memory. For example, a simultaneous block tridiagonal solver specialized to a fixed small block size can yield much higher execution rates than a general block tridiagonal solver (to be demonstrated).

The goal of high performance in spite of short vectors and apparent data flow bottlenecks appears to suggest the need for a plethora of specialized and highly-tuned codes to fill the same functions as a single code executing on a scalar processor. However, a mitigating effect is the computational dominance of the equation formulation over the equation solution as the coupling shrinks. This observation is based on (a) the number of entries in a matrix being of complexity $O(n^r)$, and (b) the triangular factorization operation count being $O(n^{r+\epsilon})$, where $\epsilon>0$ and n represents the number of variables coupled to each other. Thus, more inefficiency can be tolerated in the equation solver with small blocks, bands, and profiles.

D. Report Summary

This report provides user documentation for three classes of codes either expected to be of general utility or else resulting from on-going specialized algorithm research for the CRAY-1. All were developed with aid of a CRAY-1 timing simulator [3]. Many of the accumulation kernels on which the high performance of the codes depend are described in [4].

(a) Small dense systems. Highly-tuned accumulation kernels yield codes which achieve high execution rates with small full and banded systems.

(b) Simultaneous systems. Simultaneous full, banded, and block tridiagonal equation solvers have been developed around kernels which reduce the previously-mentioned memory traffic. A variety of block tridiagonal solvers are included representing the utility of such codes.

(c) Single system, single variable, odd-even tridiagonal solver. The use of a simulator was deemed essential to develop this challenging code which involves high memory traffic.

A fourth code, which can solve general single system sparse problems ranging from block tridiagonal systems, general full and banded matrices (which must be partitioned into 64 x 64 blocks for the CRAY-1), and arbitrary-sparsity finite element problems, is also being prepared [5].

II. Codes for the solution of small, dense systems

A. Solution of a single full unsymmetric system of equations (Calahan)

Description.

A system of equations $\underline{A}\ \underline{X} = \underline{B}$ is solved for $\underline{X}$, where $\underline{A}$ is an n x n real matrix and $\underline{X}$ and $\underline{B}$ are n x m real matrices. Double column accumulation [4] is used to achieve full cache utilization during both the triangular factorization of $\underline{A}$ and the forward and back substitution of $\underline{B}$. When m = 1, an alternate substitution routine is provided.

Subroutine call to triangularly factor A.

```
  CALL FULFAC (N, A, NDIMA, IERR)
```

where N is the dimension of $\underline{A}$

A is the array representing $\underline{A}$

NDIMA is the row dimension of array A

IERR contains a return code:

IERR = 0 implies N = 0

IERR > 0 is normal exit

IERR < 0 implies zero-valued pivot; position given by |IERR|.

Subroutine call to forward and back substitute

```
  CALL FULSOL (N, A, NDIMA, B, M, NDIMB)
```

where A contains the factored matrix

B is the array representing $\underline{B}$

M is the number of columns of $\underline{B}$

NDIMB is the row dimension of array B

Restrictions.

(a) $N \leq 64$.

(b) no pivoting

(c) Fetches (but not stores) from main memory will occur in FULFAC and FULSOL from the (n+1)st and (m+1)st columns of A and B; this space should contain data, not instructions.

When M = 1, a special call*

```
     CALL FULSOL1 (N, A, NDIMA, D, M, DNIMB)
```

should be used for best efficiency. The A array is not altered in either FULSOL OR FULSOL1.

Performance (simulated)

Matrix size	Factorization	Substitution	
4	6.5/.47	4.5/.49	
8	23/1.1	12/.77	
16	58/3.6	27/1.5	(52/1.5)
32	95/18	44/3.7	
64	122/113	60/11	

[Execution rate (MFLOPS)]/[time (kilo clocks)] for solution of a full system of equations. Result in parentheses for substitution of two columns of B (M = 2).

*FULSOL1 will solve systems where M > 1 and may be useful when M is odd and the extra column fetch of FULSOL is undesirable.

B. Solution of a single banded unsymmetric system of equations (Calahan)*

Description

A system of equations $\underline{A}\ \underline{x} = \underline{b}$ is solved for $\underline{x}$, where $\underline{A}$ is a banded real matrix and $\underline{x}$ and $\underline{b}$ are n x 1 real vectors. The matrix $\underline{A}$ is stored in compressed form.

Subroutine call to triangularly factor $\underline{A}$.

```
CALL BANFAC (N, NB, A, NDIM)
```

where N is the dimension of $\underline{A}$
NB is the half bandwidth (i.e., 2*NB+1 is the full bandwidth.)
A is an array containing the elements of $\underline{A}$
NDIM is the row dimension of A.

Subroutine call to forward and back substitute.

```
CALL BANSOL (N, NB, A, NDIM, B)
```

where B contains the elements of $\underline{b}$ on entry and $\underline{x}$ on exit.

Comments

$\underline{A}$ is stored in packed form so that the (i,j) position of $\underline{A}$ is stored in the (i - j + NB) + (j - 1)*NB address of A.

*The factorization algorithm is a recoded version of a band solver written by T. Jordan of LASL [6][7]. The substitution code is identical to Jordan's.

Half Bandwidth	Factorization	Substitution
2	3.8/14	6.2/8.1
4	9.6/19	11/8.1
8	23/28	20/8.1
16	52/45	36/8.1
32	88/210	65/18
64	117/1260	93/49

[Execution rate (MFLOPS)]/[time (kiloclocks)] for solution of a banded system of equations. Sixty four equations were solved except for half bandwidths of 32 and 64, where 128 and 256 equations were solved, respectively.

III. Codes for the Solution of Simultaneous Systems

A. Introduction

Let $\hat{\underline{A}}\ \hat{\underline{x}} = \hat{\underline{b}}$ represent the simultaneous system of equations described by the block-diagonal matrix

$$\begin{bmatrix} \underline{A}^{1} & & & & \\ & \underline{A}^{(2)} & & 0 & \\ & & \underline{A}^{(3)} & & \\ & 0 & & \ddots & \\ & & & & \underline{A}^{(m)} \end{bmatrix} \begin{bmatrix} \underline{x}^{(1)} \\ \underline{x}^{(2)} \\ \underline{x}^{(3)} \\ \vdots \\ \underline{x}^{(m)} \end{bmatrix} = \begin{bmatrix} \underline{b}^{(1)} \\ \underline{b}^{(1)} \\ \underline{b}^{(3)} \\ \vdots \\ \underline{b}^{(m)} \end{bmatrix} \qquad (1)$$

where the $\underline{A}^{(k)}$ are identically-structured n x n real unsymmetric matrices containing $a_{ij}^{(k)}$ and the $\underline{x}^{(k)}$ and $\underline{b}^{(k)}$ are n x 1 real matrices containing $x_i^{(k)}$ and $b_i^{(k)}$, respectively.

Solutions of such systems on a vector machine have a number of common characteristics.

1. Vectors are defined across the systems; e.g., the (i,j) positions of all $\underline{A}^{(k)}$, k = 1, 2...m, constitute a single vector.

2. Two storage array maps are common.

Map I. $\hat{\underline{A}}$, $\hat{\underline{x}}$, and $\hat{\underline{b}}$ stored by column, then by row, then by system, i.e.,

$a_{ij}^{(k)}$: i = 1,2,..n; j = 1, 2,..n; k = 1, 2,..m

$x_i^{(k)}$: i = 1,2,..n; k = 1,2,..m

$b_i^{(k)}$: i = 1,2,..n; k = 1,2,..m

Map II. $\hat{\underline{A}}$, $\hat{\underline{x}}$, and $\hat{\underline{b}}$ stored by system, then by column, then by row.

$a_{ij}^{(k)}$: k = 1, 2,..m; i = 1,2,..n; j = 1,2,..n

$x_i^{(k)}$: k = 1,2,..m; i = 1,2,..n

$b_i^{(k)}$: k = 1,2,..m; i = 1,2,..n

Map I may suffer from bank conflicts and loss of critical chaining when the systems are a multiple of 8 address locations apart.

A number of simultaneous system solvers are described in the report.

1. Simultaneous full systems.
2. Simultaneous banded systems.
3. Simultaneous 3 x 3 block tridiagonal systems.
4. Simultaneous 5 x 5 block tridiagonal systems.

B. Solution of full unsymmetric simultaneous systems of equations (Ames/Calahan)

Description

m simultaneous systems of the form of (1) are solved when the $\underline{A}^{(k)}$ are full matrices.

Subroutine call to triangularly factor $\hat{A}$.

CALL FUSFAC (A, N, M, IA)

where A is the array containing the full $\underline{A}^{(k)}$ matrices of (1)

N is the dimension of the $\underline{A}^{(k)}$

M is the number (m) of $\underline{A}^{(k)}$ matrices

IA is the address displacement in array A between $a_{ij}^{(k)}$ and $a_{ij}^{(k+1)}$

Subroutine call to forward and back substitute

CALL FUSSOL (B, IB)

where B is the array containing the right hand sides $\underline{b}$ and the solutions $\underline{x}$,

IB is the address displacement in array B between $b_i^{(k)}$ and $b_i^{(k+1)}$

Restrictions:

(a) $M \leq 64$

(b) $IA \geq N^2$, i.e., Map I is used.

(c) no err monitoring of pivot reciprocation.

Performance (simulated)

Equations per system (N)	Number of Systems (M)				
	4	8	16	32	64
2	3.1/.52	5.8/.55	11/.61	16/.80	21/1.2
4	5.7/2.1	11/2.2	19/2.6	28/3.5	35/5.6
8	9.6/11	18/11	32/13	45/18	53/31

(a) Factorization

Equations per system (N)	Number of Systems (M)				
	4	8	16	32	64
2	3.1/.62	6.1/.63	11/.68	19/.80	26/1.2
4	6.7/1.3	13/1.4	23/1.6	35/2.0	44/3.2
8	11/3.4	21/3.6	37/4.1	53/5.8	62/9.8

(b) Substitution

[Execution rate (MFLOPS)]/[time (kiloclocks)] of simultaneous equation solver.

C. Solution of banded unsymmetric systems of equations (Ames/Calahan)

Description:

m simultaneous systems of the form of (1) are solved when the $\underline{A}^{(k)}$ are banded. The $\underline{A}^{(k)}$ are stored in compressed form.

Subroutine call to triangularly factor $\hat{A}$ and forward and back substitute

CALL BANSIM (A, B, N, NB, M)

where A is an M*(2*NB+1)*N array of elements of the banded $\underline{A}^{(k)}$ matrices of (1)

B is an M*N array of elements of the right hand side $\underline{b}^{(k)}$ and the solution $\underline{x}^{(k)}$

N is dimension of the matrices $\underline{A}^{(k)}$

NB is the half bandwidth (i.e., 2*NB+1 is the full bandwidth)

M is the number of systems.

Comments

$\hat{\underline{A}}$, a band matrix, is stored in packed form so that the (i,j) position of $\underline{A}^{(k)}$ is stored in the k + (i - j + NB)*M + (j - 1)*M*(2*NB+1) address of A.

$\hat{\underline{b}}$ is stored so that the ith position of $\underline{b}^{(k)}$ is stored in the k + (i - 1)*M address of B

Performance (simulated)

Half-bandwidth (NB)	Number of systems (M)					
	2	4	8	16	32	64
2	2.1/47.3	4.1/47.8	7.9/49.3	14.5/53.9	23./68.0	31.3/100.
4	3.1/80.7	6.2/81.5	12./84.0	21.6/93.6	32.9/123.	43.0/188.
8	4.6/156.	9.2/157.	17.9/161.	31.1/186.	45.1/256.	56.5/409.

[Execution rates (MFLOPS)]/[time (kiloclocks)] for factorization, forward, and back substitution of simultaneous banded systems. Each system has 32 equations.

D. Simultaneous block tridiagonal systems (Sesek)

Descriptions

Consider the simultaneous block tridiagonal system $\hat{\underline{A}}\,\hat{\underline{x}} = \hat{\underline{b}}$ with kth system $\hat{\underline{A}}^{(k)}\,\hat{\underline{x}}^{(k)} = \hat{\underline{b}}^{(k)}$, viz

$$\begin{bmatrix} \underline{A}_{11}^{(k)} & \underline{A}_{12}^{(k)} & & & \\ \underline{A}_{21}^{(k)} & \underline{A}_{22}^{(k)} & \underline{A}_{23}^{(k)} & & \\ & \underline{A}_{32}^{(k)} & \underline{A}_{33}^{(k)} & \ddots & \\ & & \ddots & \underline{A}_{n-1,n-1}^{(k)} & \underline{A}_{n-1,n}^{(k)} \\ & & & \underline{A}_{n,n-1}^{(k)} & \underline{A}_{n,n}^{(k)} \end{bmatrix} \begin{bmatrix} \underline{x}_1^{(k)} \\ \underline{x}_2^{(k)} \\ \underline{x}_3^{(k)} \\ \vdots \\ \underline{x}_{n-1}^{(k)} \\ \underline{x}_n^{(k)} \end{bmatrix} = \begin{bmatrix} \underline{b}_1^{(k)} \\ \underline{b}_2^{(k)} \\ \underline{b}_3^{(k)} \\ \vdots \\ \underline{b}_{n-1}^{(k)} \\ \underline{b}_n^{(k)} \end{bmatrix}$$

where the size of the square $A_{ij}^{(k)}$ is of fixed size to achieve high execution rates.

Subroutine call to triangularly factor $\hat{A}$.

3 x 3 block size

```
CALL BT3FAC (A, B, C, M, N)
```

5 x 5 block size

```
CALL BT5FAC (A, B, C, M, N)
```

where A, B, and C are defined below

M is the number of systems

N is the number of diagonal blocks

Subroutine call to forward and back substitution

3 x 3 block size

```
CALL BT3SOL (A, B, C, D, M, N)
```

5 x 5 block size

```
CALL BT5SOL (A, B, C, D, M, N)
```

where D is defined below

Comments

Storage Map II is used for all $\underline{\hat{A}}_{ij}$, $\underline{\hat{x}}_i$, and $\underline{\hat{b}}_i$. For block size ℓ, the storage is given below for n diagonal blocks.

Array A $(m \times \ell \times \ell n)$

$$[\hat{A}_{11}\ \hat{A}_{22} \cdots \hat{A}_{nn}]$$

Array B $(m \times \ell \times \ell(n-1))$

$$[\hat{A}_{12}\ \hat{A}_{23} \cdots \hat{A}_{n-1,n}]$$

Array C $(m \times \ell \times \ell(n-1))$

$$[\hat{A}_{21}\ \hat{A}_{32} \cdots \hat{A}_{n,n-1}]$$

Array D $(m \times \ell n)$

$[\hat{b}_1\ \hat{b}_2 \cdots \hat{b}_n]$ on entry to code

$[\hat{x}_1\ \hat{x}_2 \cdots \hat{x}_n]$ on exit from code

Performance (simulated)

Execution rates (MFLOPS)

Number of Systems	3x3 FACTOR.	3x3 SUBS.	5x5 FACTOR.	5x5 SUBS.
8	28.8	27.6	40.8	40.1
16	46.6	46.1	54.7	53.6
32	60.3	58.7	65.5	64.1
64	67.9	66.3	72.0	70.0

MFLOP rates for solution of block tridiagonal systems for two block sizes, as a function of the number of simultaneous systems (= vector length).

IV. Code for the Solution of a Single Tridiagonal System (Calahan/Sesek)

Description

A single tridiagonal system is completely solved by cyclic (odd-even) reduction. This method requires ≈2.7 times the floating point computation of a scalar solution (important when evaluating the MFLOPS rates below), but yields, for large matrices, nearly all 64-length vector operations.

Subroutine call to triangularly factor A.

```
CALL TRIFAC (A, B, C, N)
```

where A contains the diagonal stripe
B contains the super-diagonal stripe
C contains the sub-diagonal stripe
N is the number of diagonal elements

Subroutine call to forward and back substitute.

```
CALL TRISOL (A, B, C, D, N)
```

where D contains the right hand side on entry and the solution on exit.

Performance (simulated)

Matrix size (N)	Factorization	Forward & Back Substitution
15	8.22/1.30	4.80/2.02
31	13.5/1.79	7.65/2.75
63	21.1/2.45	11.8/3.71
127	30.3/3.55	17.3/5.18
255	38.6/5.69	23.5/7.76
511	45.2/9.85	28.7/12.8
1023	49.7/18.1	32.7/22.5
2047	52.4/34.3	35.6/41.3
4095	53.9/66.9	37.6/78.4

Timings of a cyclic reduction of a single tridiagonal system; results given as [exectuion rate (MFLOPS)]/[time (kiloclocks)]

Restrictions:*

(1) No premature termination of reduction [8]

(2) Single precision (10 digit) reciprocation of pivots.

Comments

The coding has been optimized for the large matrix case. The execution is then easily shown to be memory-path bound. The algorithm was therefore chosen to make a minimum number of memory accesses per loop, utilizing shifting instead to align operand vectors in the vectors registers. The resultant coding was then optimized to achieve full memory-path utilization. Simulation shows that 94, 87, 86% utilization is achieved for the factorization, forward, and back substitutions, respectively, as $N \to \infty$. Simulation also shows that as $N \to \infty$, approximately 2/3 of the execution time is needed during factorization for this CAL version vis-a-vis a Fortran version written independently.†

*The present version of the code requires $N = 2^r-1$, r a positive integer.

†By Dave Kersnaw, Lawrence Livermore Laboratory; full precision reciprocation was used in this version.

References

[1]. Boley, D., B. L. Buzbee, and S. V. Parter, "On Block Relaxation Techniques," TR #318, Computer Sciences Dept., University of Wisconsin, Madison, June, 1978.

[2]. Orbits, D. A., and D. A. Calahan, "A CRAY-1 Simulator and Its Application to the Development of High Performance Codes," Proc. 1978 LASL Workshop on Vector and Parallel Processors, Los Alamos Scientific Laboratories, pp. 42-56.

[3]. Orbits, D. A., "A CRAY-1 Timing Simulator," SEL Report No. 118, Systems Engineering Laboratory, University of Michigan, September, 1978.

[4]. Ames, W. G., et al, "Sparse Matrix and Other High Performance Algorithms for the CRAY-1," Report #124, Systems Engineering Laboratory, University of Michigan, January 24, 1979.

[5]. Calahan, D. A., "Vectorized Sparse Equation Solution on the CRAY-1," Proc. 1979 International Conf. on Parallel Computing, Wayne State University, August, 1979.

[6]. Fong, K., and T. L. Jordan, "Some Linear Algebraic Algorithms and Their Performance on the CRAY-1," Report LA-6774, Los Alamos Scientific Laboratory, June, 1977.

[7]. Calahan, D. A., "Performance of Linear Algebra Codes on the CRAY-1," Fifth SPE Symposium on Reservoir Simulation, Denver, February, 1979.

[8]. Lambiotte, J. J., and R. G. Voigt, "The Solution of Tridiagonal Linear Systems on the CDC STAR-100 Computer," ACM Trans. Math. Soft., vol 1, no 4, December, 1975, pp. 308-329.