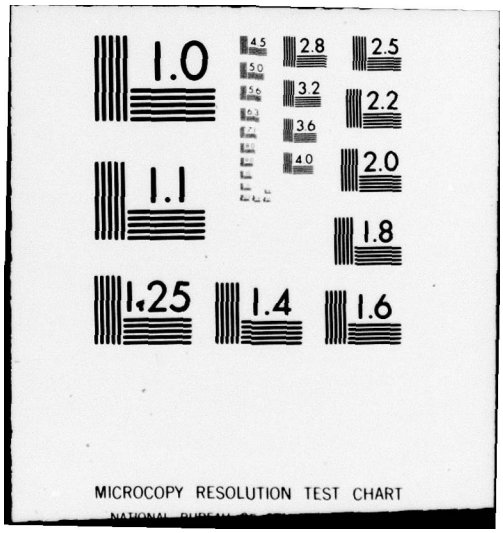
AD-A074 091 MICHIGAN UNIV ANN ARBOR DEPT OF ELECTRICAL AND COMPU--ETC F/G 12/1
VECTORIZED SPARSE EXAMINATION.(U)
AUG 79 D A CALAHAN AFOSR-75-2812

UNCLASSIFIED AFOSR-TR-79-0980 NL

| OF |
ADA
074091

END
DATE
FILMED
10 -79
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
MICROCOPY RESOLUTION TEST CHART

VECTORIZED SPARSE ELIMINATION

D. A. Calahan
Department of Electrical and Computer Engineering
University of Michigan
Ann Arbor, MI 48109

ABSTRACT

Vectorizable sparse equation solution algorithms are classified by the matrix structure which they favor. The state-of-the-art for solution of relatively dense systems is then reviewed. A hybrid vector construct is defined for the increasingly common structure of both moderate local matrix density and global matrix regularity. Estimates are made of CRAY-1 speedup achievable with this construct. A finite difference matrix is studied as an example.

INTRODUCTION

Direct solution of sparse systems has enjoyed wide application to simulation of lumped physical systems described by ordinary differential equations. Also, the last decade has seen a movement toward implicit solution of partial differential equations away from explicit procedures. An excellent example is Navier Stokes aerodynamic simulation codes, which have changed from the purely explicit, through hybrid explicit-implicit[1] and now purely implicit procedures[2].

The vectorization of direct solution portions of large codes has an immediate aspect related to the recoding of specific equation solvers for a particular architecture. Although most vector architectures have at least a minimal provision for sparse vector operations, an overhead is inevitably incurred in reduced memory bandwidth and/or the loading of associated bit maps and linked lists. It is the goal of research in sparse matrix algorithms to reduce this overhead by re-organization of the computation either (1) to obtain longer vectors, or (2) to reduce data flow, and thus achieve an overall speedup.

This paper (1) classifies sparse matrix characteristics amenable to vector processing, (2) reviews the state-of-the-art in solving certain of these problems, and (3) presents new results in the detection of vectors in patterned sparse systems. All of the experimental results were obtained from the CRAY-1; even the algorithm classifications to be made are useful only for a memory-hierarchical processor of the CRAY-1 class with a range of scalar, short vector, and long vector capabilities.

CLASSIFICATION

Consider the linear system $Ax = b$ solved by triangular factorization of A into L and U. Assume that the factorization has proceeded by outer product column-row operations so that an nxn unreduced system remains. The structure of this unreduced system--which includes fill from the completed portion of the reduction--then becomes the principal issue in determination of the sparsity algorithm to be used during the remainder of the reduction. This is an important generalization beyond examination of only the structure of A, since it suggests the use of different algorithms (polyalgorithms) as the reduction proceeds and fill increases the density of the unreduced portion.

Four sparsity structures will be considered at various parts of this paper; they are listed below to assist in unifying the later discussion. These distinguishing

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: AFOSR-TR-79-0980	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): VECTORIZED SPARSE EXAMINATION	5. TYPE OF REPORT & PERIOD COVERED: Interim rept.
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): D.A. Calahan	8. CONTRACT OR GRANT NUMBER(s): AFOSR-75-2812
9. PERFORMING ORGANIZATION NAME AND ADDRESS: University of Michigan, Electrical & Computer Engineering, Ann Arbor, Michigan 48109	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 61102F 2304/A3
11. CONTROLLING OFFICE NAME AND ADDRESS: Air Force Office of Scientific Research/NM, Bolling AFB, Washington, DC 20332	12. REPORT DATE: August 1979
	13. NUMBER OF PAGES: 11
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): 13p.	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

PROC. SCIENTIFIC COMPUTER INFORMATION EXCHANGE MEETING IN LIVERMORE, CALIFORNIA, SEPTEMBER 12-13, 1979

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

Sparse matrices
Vector processing
Parallel processing

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

Vectorizable sparse equation solution algorithms are classified by the matrix structure which they favor. The state-of-the-art for solution of relatively dense systems is then reviewed. A hybrid vector construct is defined for the increasingly common structure of both moderate local matrix density and global matrix regularity. Estimates are made of CRAY-1 speedup achievable with this construct. A finite difference matrix is studied as an example.

DD FORM 1 JAN 73 1473 — 407400 — UNCLASSIFIED

attributes are related to local and global sparsity characteristics:

(a) locally and globally dense, partitioned;
(b) locally dense, globally unpatterned;
(c) locally dense, globally patterned;
(d) locally sparse, globally patterned;
(e) locally sparse, globally unpatterned.

The last is the least vectorizable. Its scalar solution is probably amenable to speedup only by using a MIMD architecture[3] and so will not be discussed further.

BLOCK-ORIENTED SPARSE SOLUTION

INTRODUCTION

Two classes of relatively dense matrices benefit from solution by a general sparse solver which is oriented toward the solution of block structures. Although algorithmically less challenging than the sparser case to be studied later, such structures are becoming more common due to the aforementioned increase in the implicitness of PDE solution codes.

THE DENSE, PARTITIONED CASE

The utility of a general sparse solver in the analysis of full, banded, and other dense systems arises from vector length limitations of the processor, which in turn results from a relatively small cache memory in a hierarchial memory system. Such dense systems must be block-partitioned; in the case of the CRAY-1, these partitions must be limited in one dimension to 64, the maximum vector length of the machine. Using a general solver avoids the writing of specialized assembly language routines for dense systems with globally different density patterns but which are partitioned into locally similar 64-length or smaller dense blocks.

The processing of such large blocks with a sparse solver can be carried out on the CRAY-1 with >99.9% of the solution time in numeric kernels, and with <.1% in processing of lists resulting from the general sparsity assumption. A variety of common compressed storage schemes can also be accomodated[4].

LOCAL DENSITY

Moderate-sized dense blocks occur naturally from the representation of variable and equation coupling, from nodal coupling in a grid, and from coordinate transformations, among other causes. In the absence of other vectorization strategies (to be discussed shortly), it becomes necessary to reduce the system a block at a time with dense matrix kernels of a block-oriented sparse solver. Descriptors of the location and size of the block suffice to guide the solution of such a system[4].

The overhead of list processing of the blocks may be compensated by finely-tuned numeric kernels, with the net result that a general solver can execute at a higher rate than a conventionally-coded specialized solver[4].

The execution rate is of course highly dependent on the matrix sparsity structure. However, a timing model of the numeric kernels and the list processing overhead[4] allows the establishment of MFLOPS bounds for matrices of constant block sizes but arbitrary block sparsity patterns. Such bounds are given in Table 1 for the CRAY-1. The minimum rate is achieved with a single off-diagonal block (e.g., block tridiagonal) and the maximum with r off-diagonal blocks (Figure 1), as $r \to \infty$.

LOCALLY DENSE, GLOBALLY PATTERNED SPARSE SYSTEMS

INTRODUCTION

Table 1 shows that processing block sizes with dimensions below 10 utilizes a small fraction of the CRAY-1 processor speed. To regain a high processing rate, another structural property be-

Block sizes	MFLOPS range
2	1.9 - 7.6
3	5.0 - 17.
4	10. - 26.
6	21. - 43.
8	32. - 60.
12	54. - 84.
16	69. - 98.
32	102. - 124.
64	126. - 141.

Table 1. Performance of general block sparse system solver on the CRAY-1

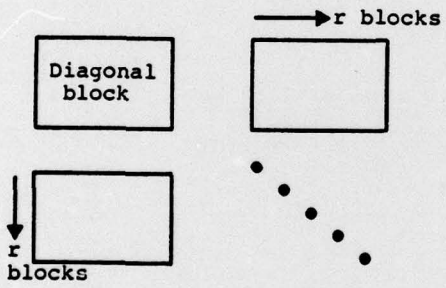


Figure 1. Model of block pivot step

sides density should be exploited. It is proposed to utilize global similarities or patterns to lengthen density-related vectors. These will be termed hybrid vectors and are the subject of the remainder of this paper.

A "bottom-up" approach will be used. After defining and illustrating the model hybrid problem, it will first be demonstrated that the CRAY-1 can achieve considerably higher execution rates on hybrid-related kernels. Then it will be shown how such hybrid vectors can be achieved with common finite difference (or finite element) structures.

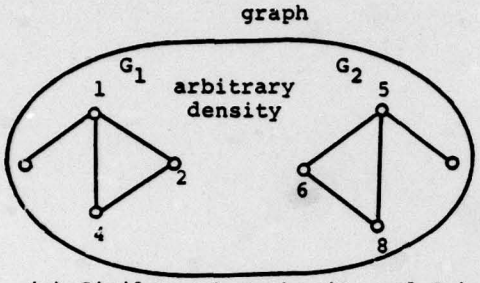
GLOBAL vs. LOCAL PROPERTIES

In establishing the hybrid vector concept, it will be useful to use the notion of the graph of a matrix.

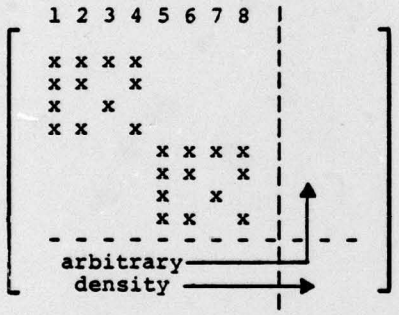
The non-zero structure of a matrix A, where A is structurally symmetric, has a convenient graph theoretic formulation. Assume that $a_{ii} \neq 0$, $i=1,2,\ldots n$. Let $V = \{v_1, v_2, \ldots v_n\}$, with the v_i termed vertices and V the vertex set. Define a set P of ordered pairs of V, called edges, by $(v_i, v_j) \in P$ if and only if $a_{ij} 0$ and $k \neq j$. Then $G=G(V,P)$ is called the graph of A. Note that, because the matrix is structurally symmetric, $(v_i, v_j) \in P$ if and only if $(v_j, v_i) \in P$.

To illustrate the relationship between local and global properties, consider the subgraphs G_1 and G_2 of Figure 2(a). These subgraphs are possibly connected by paths through vertices not shown, but are assumed to be not directly connected. If the associated equations are arranged in the numbered order, the partial matrix structure of Figure 2(b) results. This structure is locally dense (contrast full) but globally sparse, since the two dense submatrices are not coupled in the northwest matrix partition. If the equations are reordered so that similarly-connected nodes are consecutively ordered, then each of the resulting 16 partitions is either a diagonal or a null submatrix (Figure 2(c)). Because most sparse blocks are coupled to other sparse blocks by diagonal coupling blocks, the matrix structure is now termed globally dense. (It may be noted that the local density pattern of each dense block of Figure 2(b) is identical to the global density pattern of Figure 2(c).) The factorization of the northwest corner of the system matrix may utilize any algorithm, independently of the algorithms used to reduce the remainder of the matrix.

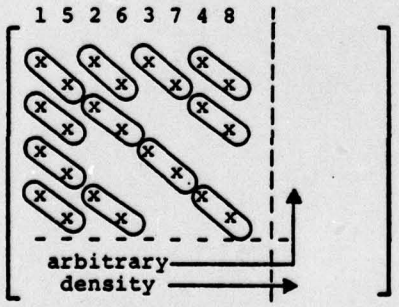
If the connection symmetry between the two sets of nodes undergoing reduction extends to their interconnections to other unreduced nodes as in Figure 3(a), and if these unreduced nodes are properly ordered as shown, then the northeast and southwest parti-



(a) Similar subgraphs (G_1 and G_2)

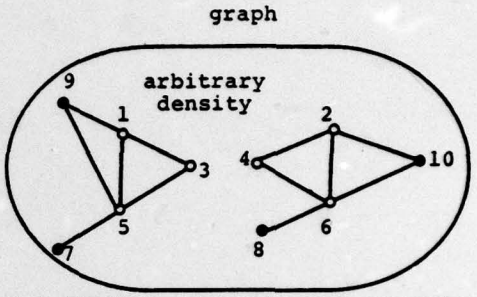


(b) Locally dense, globally sparse corner

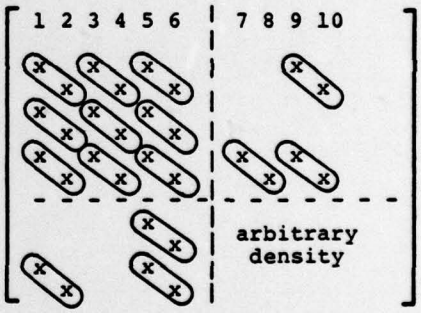


(c) Locally sparse, globally dense corner.

Figure 2. Relationships between local, global matrix properties



(a) Similar subgraphs and connections (o-node undergoing reduction; •-unreduced node).



(b) Associated matrix

Figure 3. Similar subgraphs with similar connections to rest of graph.

tions can be made to contain similar diagonal coupling matrices (Figure 3(b).

KERNEL STUDY

In solution of large sparse systems, the multiplication/accumulation (M/A) kernel dominates other numeric kernels. Elimination of a strip of row and column blocks symmetrically coupling a diagonal block to r other diagonal blocks (Figure 1) requires (a) fac-

torization of a diagonal block, (b) r block forward and back substitutions, and (c) r^2 multiplications/accumulations. For r=3 (a common number for dissected finite element and finite difference grids), 69% of the operations are of the M/A type. The M/A kernel therefore warrants principle study.

The nature of the M/A model kernel with both local diagonal sparsity and global density is illustrated in Figure 4. It is proposed to study the execution of the kernel

$$C \leftarrow C \pm A*B \qquad (1)$$

where A, B, and C are illustrated in the figure.

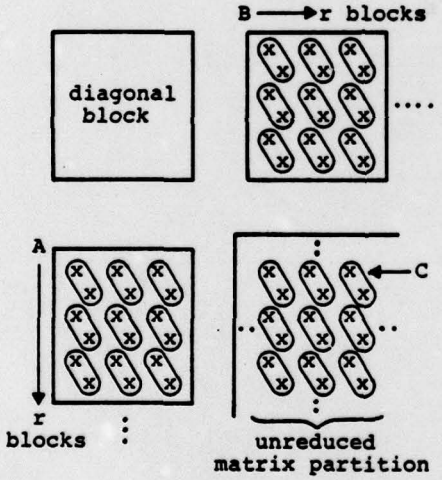


Figure 4. Example of model problem

The preference for processing hybrid kernels can be expected to arise from the interconnection or, more generally, the data flow protocol of the processor. For the CRAY-1, two recursive features of the vector registers permit high performance M/A kernels.

4-Matrix M/A. The pattern illustrated by the matrix multiply

$$\begin{matrix} & A & & * & & B \\ \begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \end{bmatrix} & & & \begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \end{bmatrix} & & \end{matrix} \qquad (2)$$

represents, on equation reordering, the simultaneous multiplication of four 3x3 full matrices. It is proposed to implement the associated accumulation kernel by forming

$$\begin{bmatrix} C_{1j} \\ C_{2j} \\ C_{3j} \end{bmatrix} \leftarrow \begin{bmatrix} C_{1j} \\ C_{2j} \\ C_{3j} \end{bmatrix} \pm \sum_{k=1}^{3} B_{kj} \begin{bmatrix} A_{1k} \\ A_{2k} \\ A_{3k} \end{bmatrix} \qquad (3)$$

i.e., by accumulating a column of diagonal blocks of C. To perform each term of the summation by a single chained multiply-add vector operation with the CRAY-1 requires chain replications of the 4-length B_{kj} to a 12-length vector so that the overhead of the replication does not seriously impact the overall timing.

The basis of this replication is the recursive feature of the vector logical pipeline, whereby, if the same vector register is both operand and result register--usually prohibited in register allocation--data will be delayed four clocks in the pipeline and the desired replication achieved. Figure 5 gives the CAL instruction sequence and the clock level report of a part of the accumulation loop, as reproduced by a CRAY-1 timing simulator[5].

Table 2 gives the execution rates of a complete 4-matrix multiply, in comparison with the rates of two full matrix multiply kernels previously studied. The standard full kernel for short vectors produces large gaps in the floating point pipelines due to the chaining[6,7]. The high-performance matrix multiply kernel avoids chaining and the consequent gaps

but suffers from register and pipeline reservations and addressing overhead resulting from four separate invocations of the full matrix multiply. Table 2 shows that nearly three times the execution rate is achieved for multiplication of four 4x4 matrices with the specialized kernel, in comparison with the standard CAL kernel.

8 Matrix M/A. A similar recursive feature of the addition pipeline allows the rapid accumulation of 8-length vectors and consequently the simultaneous multiplication of 8 matrices. This is a well known feature described in [8] and will not be discussed here. It suffices to note in Table 2 the extraordinary speedups achievable with very small matrices. However, the execution rate has a large discontinuity between n=7 and n=8, due to the nature of the algorithm, and is less desirable beyond n=7 than a 4-matrix multiply.

```
T                                    FFF      V. REG   BSRRR S S. REG   A A. REG
A INSTRUCTION      P-ADDR   CP       PPPVVV             SCKKK R         R
G                                    */&>< 01234567 F1ABC A01234567 A01234567

  V1  ,A0,A0         54C    628!0Z     ! 50  00 !5     !   4     !
  A3  A0+A3          54D    629!0Z     ! 50  00 !5     !   4     !    6
                            630!0Z     ! 50  00 !5     !   4     !6   6
  A3  A3+A1          55A    631!0      ! 50  00 !5     !4  4     !    7
  VL  A2             55B    632!0      ! 50  00 !5     !         !7   7
  A0  A0+A5          55C    633!0      ! 50  00 !5     !         ! 9
  A5  A5+A6          55D    634!0      ! 50  00 !5     !         !99    A
                            635!0      ! 50  00 !5     !         !A     A
  A5  A0+A5          56A    636!0      ! 5   0  !      !         !      B
  V0  ,A0,A0         56B    637!0      !C*   0  !C     !         !B     B
                            638!0      !C5   0  !C     !         !
                            639!0      !C5   0  !C     !         !
                            640!       !C5   0  !C     !         !
                            641!       !C5   0  !C     !         !
  V3  V3!V1&VM       56C    642!   D   !CD D 0  !C     !         !
                            643!   D   !CD D 0  !C     !         !
                            644!   D   !CD D    !C     !         !
                            645!   D   !CD D    !C     !         !
  V4  V3*RV0         56D    646! E D   !ED EE   !C     !         !
  A0  A0+A4          57A    647! E D   !ED EE   !C     !         ! F
  A4  A4+A1          57B    648! E D   !ED EE   !C     !         !FF   G
                            649! E D   !ED EE   !C     !         !G     G
                            650! E D   !ED EE   !C     !         !
                            651! E D   !ED EE   !C     !         !
                            652! E D   !ED EE   !C     !         !
                            653! E D   !ED EE   !C     !         !
                            654! E D   !ED EE   !C     !         !
  V6  V7+FV4         57C    655!HE D   !ED EH HH!C     !         !
  VL  A1             57D    656!HE D   !ED EH HH!C     !         !
                            657!HE D   !ED EH HH!C     !         !
                            658!HE D   !ED EH HH!C     !         !
                            659!HE D   !ED EH HH!C     !         !
                            660!HE D   !ED EH HH!C     !         !
                            661!HE D   !ED EH HH!C     !         !
                            662!HE D   !ED EH HH!C     !         !
                            663!HE D   !ED EH *H!C     !         !
                            664!HE D   !ED EH HH!C     !         !
...                                      .......
                            704!HE D   !ED EH HH!C     !         !
                            705!HE D   !ED EH HH!      !         !
  V1  ,A0,A0         60A    706!HE D   !EJ EH HH!J     !         !
```

Figure 5. Simulator output for 4-matrix accumulation instruction sequence for VL = 64. V1 is replicated into V3 with VM = $0077..7_8$.

Full matrix size	Stand. full	High-perf. full	4-matrix hybrid	8-matrix hybrid**
2	7.6	8.7	17.2	22.1
4	19.	30.	54.0	55.7
6	NT*	NT	76.3	88.5
8	43.	64.	90.2	72.5
10	NT	NT	97.8	90.8
16	88.	102.	119.	--

* Not tested
** Positive accumulation only

Table 2. Execution rates (MFLOPS) of matrix multiply kernels. Subroutine entry and exit overhead is not included.

AN ALGORITHMIC OVERVIEW

An important algorithmic property of the above hybrid vector construct is that the kernel vector length is proportional to the product of factors related to (1) the local matrix density and (2) the global matrix patterns. In less precise terms, one may claim the length is the product of local coupling and global decoupling.

In solution of large initially sparse patterned systems, as the reduction progresses fill causes the coupling to increase and the decoupling to decrease, leaving the possibility that their product remains a relative constant.

Such a result could produce a very useful generalization of previous work (ref. [9][10]) where vector lengths were assumed a function of the local density only or global patterns only[11]. To lengthen vectors by increasing local density, previous algorithms were inevitably driven to an increase in the arithmetic computational complexity[10].

The following study can be considered an initial investigation into the production of hybrid vectors for finite difference grids. A number of algorithmic questions will be left unanswered, a topic for continuing research.

GENERATION OF HYBRID VECTORS

INTRODUCTION

Given the graph of a matrix, it is proposed to perform operations on this graph which yield hybrid vectors in the matrix reduction with either no increase or a determinable increase in the arithmetic operation count. The example of a 5-point 2-D finite difference grid will be used to illustrate the procedure, because of its connection regularity and because its solution by nested dissection is characterized by exploitation of decoupling to achieve a reduced arithmetic operation count for large grids. The reader is assumed to be familiar with this dissection process[12,13].

FOLDING AND ROTATION

The (diagonal) nested dissection of a 5x5 grid proceeds by recursively dividing the grid into quadrants until each quadrant consists of a single node. This division is performed along diagonal separators, which are lines of nodes whose removal divides the graph into unconnected parts.

It is clear from Figure 6(a) that, since the quadrants have a similar structure, "similarly-positioned" vertices not on a separator may be eliminated simultaneously with vector operations, without increasing arithmetic computation. These vectors will be of length four, as required for the 4-matrix kernel of Figure 5.

"Similarly-positioned" nodes can be generated by overlaying the quadrants so that a single node in the overlay represents 4 nodes. This single-quadrant representation of the 4 quadrants may be achieved by folding or rotating the original graph. This rotation process is illustrated in Figure 6(a)-(b); a recursive folding process - which generates vectors of decreasing length - is discussed in ref. [14].

Because the non-separator nodes are eliminated first in the nested dissection process, these interior nodes are represented by the northwest corner of the system matrix; this corner is consequently guaranteed to consist of 4x4 blocks with either diagonal or null structure. The southeast corner of LU, representing the reduction of the separator nodes, is dense and can be reduced at execution rates exceeding 100 MFLOPS. The northeast and southwest partitions, however, represent coupling between the separators and the interior nodes of the quadrants. Asymptotically in the grid dimensions, operations involving these two partitions consist of approximately 30% of the total, so that the choice of a proper kernel is important (Figure 7). Irregularity in these coupling matrices results in part from the separator nodes shared by Q1 and Q4 (nodes #1, #7 and #13 in Figure 6(a)) in the rotation sequence. The regularity may be restored by cutting the graph along the boundary, adding nodes and associated unknowns, and adding equations that relate the new nodes to the originally shared ones. This cutting process is illustrated in Figure 6(c)-(d).

The structure of L and U resulting from application of such cutting to a 17 x 17 finite difference grid is shown in Figure 8(b). The conventional nested dissection ordering for the same matrix yields the LU map of Figure 8(a). Coupling in the northwest partition appears as 4x4 diagonal blocks, as predicted; coupling in the northeast and southwest partitions appears as 4-length stripes, but not necessarily as 4x4 diagonal blocks. Thus a somewhat modified 4-matrix accumulation kernel would have to be used. The addition of the 8 nodes along the cut also increases each dimension of the dense southeast corner of LU by approximately 25%. The total increase in computation resulting from cutting is as yet undetermined.

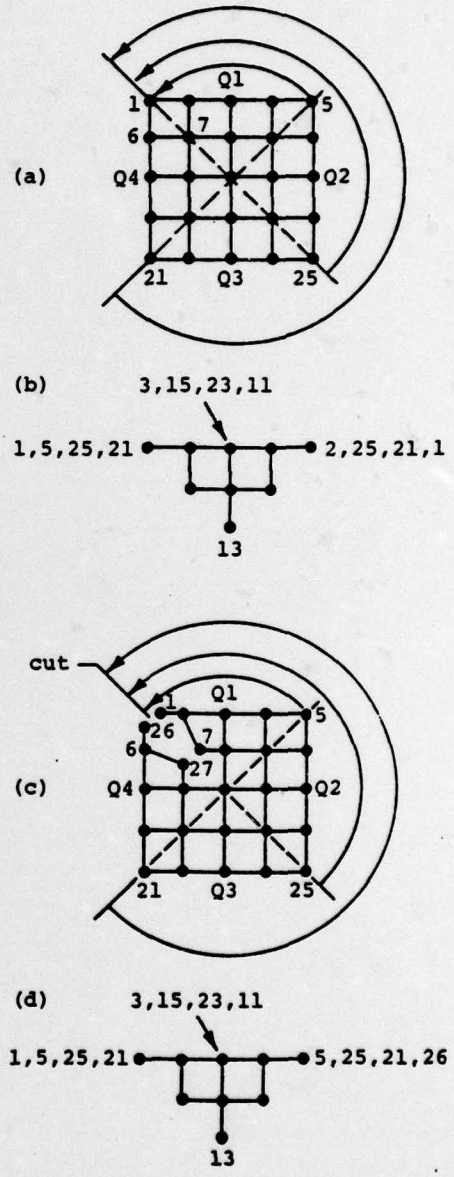
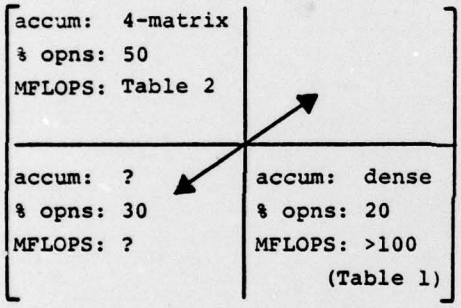


Figure 6. Rotation of quadrants (a) into single-quadrant representation (b); rotation with cut and creation of nodes ((b)-(c)).

SUMMARY

Vectorization and data flow

for a memory hierarchical processor add two new issues to be considered in the development of codes for the direct solution of 2-D finite difference grids. What is a single algorithm class--nested dissection--for a scalar machine now divides into subclasses of algorithms, of which the above proposal is only one. Checkerboard and related ordering strategies are also attractive; preliminary estimates indicate such codes will execute over 100 MFLOPS on the CRAY-1[15], which can partially compensate for increased arithmetic computation.



*Approx. percent of total arithmetic operations.

Figure 7. Estimated asymptotic perf. of polyalgorithm to perform nested dissection in four matrix partitions.

SPARSE, PATTERNED SYSTEMS

As the coupling in A, B, and C of Figure 4 decreases, each approaches a diagonal matrix. The accumulation then involves at least two vector loads (and usually one vector store) for each floating point M/A operation and sufficient list processing to locate at least two of the matrices in memory. An accumulation kernel written for the CRAY-1, including list processing and a vector store for each accumulation, executes at the rate

$$\text{MFLOPS} = 53.3 \left(\frac{1}{1 + 31.3/\ell}\right)$$

with the maximum value of 35.8 for $\ell = 64$. This is less than 1/4 the asymptotic rate of a dense accumulation. The kernel is memory bound and involves significant start up time for the relatively small floating point computation involved.

CONCLUSION

While the speed of vector processors encourages the formulation of denser systems, their increasingly parallel design favors the construction of longer vectors that can be distributed across many pipelines operating concurrently. In this paper, the vector-lengthening advantages of the hybrid vector construct have been shown at the kernel level and methods have been proposed to produce such vectors directly from the problem structure.

From the algorithm viewpoint, the direct relationship between problem and processor structure offers novel insight possibly useful in developing a family of equation ordering techniques based on folding, rotation, etc. It is also hoped that a high performance software package may be developed for specific 2D grid geometries.

From the viewpoint of processor architecture, this paper has quantified the motion that the less dense the system, the more data flow and other accumulation kernel overhead is required. A patterned system may permit the lengthening of vectors - which reduces the influence of overhead - but does not significantly alter the data flow problem.

ACKNOWLEDGMENT

The author acknowledges the programming assistance of M. Yatchman, and the insight provided by a CRAY-1 simulator developed by D. A. Orbits. This work was sponsored jointly by the Mathematical and Information Sciences Directorate of the Air Force Office of Scientific Research, and by the Air Force Flight Dynamics Laboratory, Wright Patterson AFB under

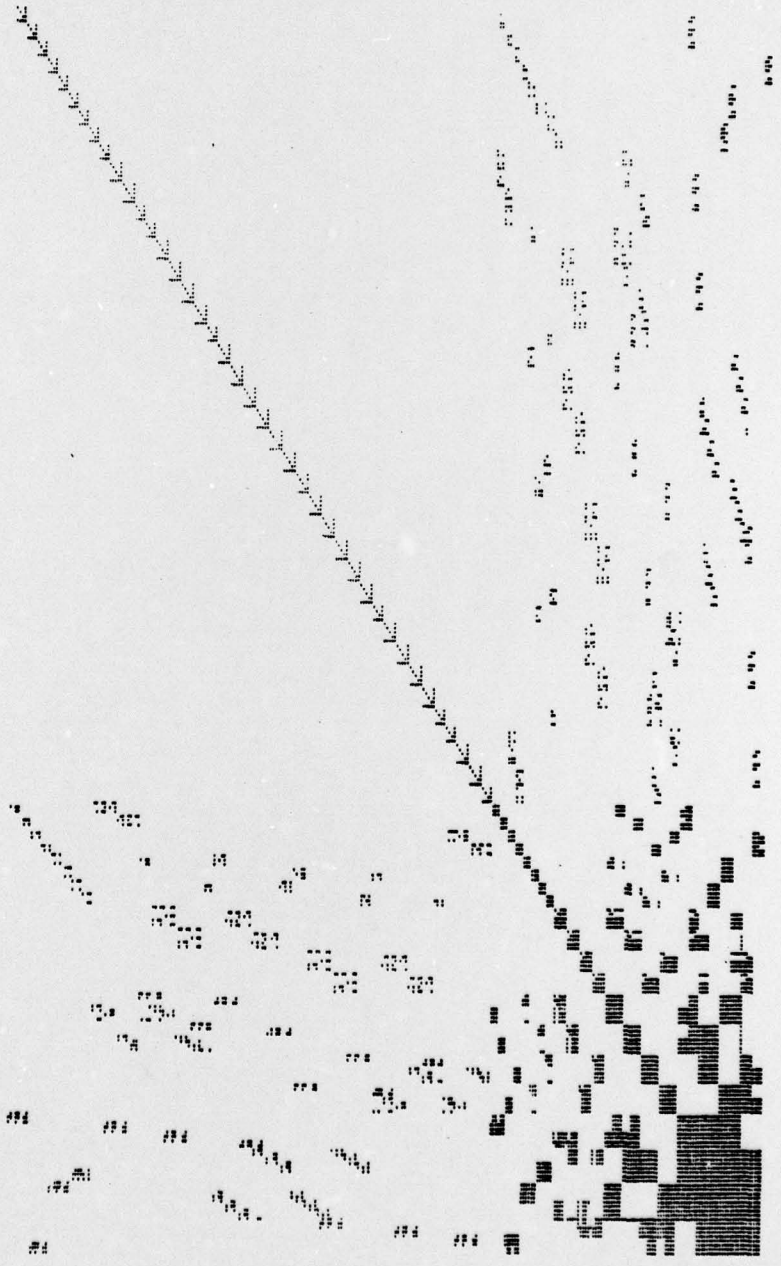


Figure 8(a). Matrix of dissected 17x17 5-point finite difference grid; before rotation.

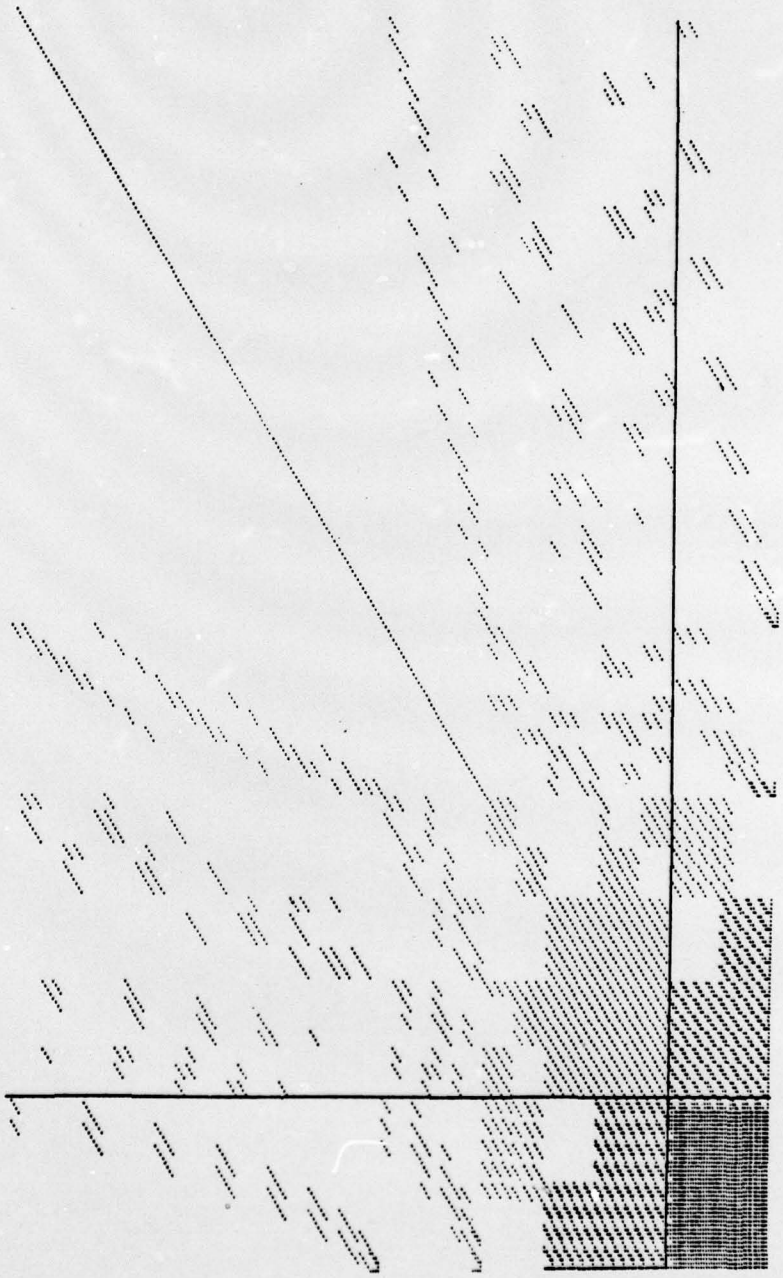


Figure 8(b). Matrix of dissected 17x17 5-point finite difference grid; after rotation.

Grant 75-2812.

References

[1] R. W. MacCormack, "An Efficient Numerical Method for Solving the Time-Dependent Compressible Navier-Stokes Equations at High Reynolds Number," NASA Report TMX-73.129, Ames Research Center, Moffett Field, CA, (July, 1976).

[2] R. M. Beam and R. F. Warming, "An Implicit Factored Scheme for the Compressible Navier-Stokes Equations II: The Numerical ODE Connection," Paper No. 79-1446, AIAA 4th Computational Fluid Dynamics Conf., Williamsburg, VA., (July, 1979).

[3] O. Wing, and J. W. Huang, "An Experiment in Parallel Processing of Gaussian Elimination of a Sparse Matrix," Proc. IEEE 1976 International Symposium on Circuits and Systems, Munich, Germany (April, 1976).

[4] D. A. Calahan, "A Block-Oriented Sparse Equation Solver for the CRAY-1," Proc. 1979 Intl. Conf. on Parallel Processing, Bellaire, MI. (August, 1979).

[5] D. A. Orbits, "A CRAY-1 Simulator," Report #118, Systems Engineering Laboratory, Univ. of Michigan, (Sept., 1978).

[6] D. A. Orbits, and D. A. Calahan, "A CRAY-1 Simulator and Its Use in Development of High Performance Algorithms," Proc. Workshop on Vector and Parallel Processing, Los Alamos Scientific Laboratory, 42-56, (Sept., 1978).

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

[8] CRAY-1 Reference Manual, Pub. #2240004, Cray Research, Inc., Chippawa Falls, Wisc.

[9] D. A. Calahan, "Complexity of Vectorized Solution of 2-Dimensional Finite Element Grids," Report #91, Systems Engineering Laboratory, Univ. of Michigan, (November, 1975).

[10] A. George, W. E. Poole, Jr., and R. G. Voigt, "Analysis of Dissection Algorithms for Vector Computers," ICASE Report 76-17, NASA Langley Research Center, Hampton, VA (June, 1976).

[11] D. E. Barry, C. Pottle, and K. A. Wirgan, "A Technology Assessment Study of Near Term Computer Capabilities and Their Impact on Power Flow and Stability Simulation Programs," Final report on Research Project EPRI TPS 77-749, General Electric Co., Schenectady (June, 1978).

[12] A. George, "Numerical Experiments Using Dissection Methods to Solve n by n Grid Problems," SIAM J Numer. Anal., vol. 14, 161-179 (April, 1977).

[13] P. T. Woo, S. J. Roberts, and F. G. Gustavson, "Applications of Sparse Matrix Techniques in Reservoir Simulation," SPE 4544 48th Annual Fall Meeting of Soc. of Pet. Engrs., Las Vegas, Nevada (1973)

[14] D. A. Calahan, and W. G. Ames, "Vector Processors: Models and Applications," (To be published, Trans. IEEE on Circuits and Systems, Fall, 1979).

[15] D. A. Calahan, W. G. Ames, and E. J. Sesek, "A Collection of Equation-Solving Codes for the CRAY-1," Report #133, Systems Engineering Laboratory, Univ. of Michigan (August, 1979).