NASA Contractor Report 172418

ICASE REPORT NO. 84-34

ICASE

A MODEL OF ASYNCHRONOUS ITERATIVE ALGORITHMS FOR SOLVING LARGE, SPARSE, LINEAR SYSTEMS

Daniel A. Reed

and

Merrell L. Patrick

Contract Nos. NAS1-17070, NAS1-17130

July 1984

INSTITUTE FOR COMPUTER APPLICATIONS IN SCIENCE AND ENGINEERING
NASA Langley Research Center, Hampton, Virginia 23665

Operated by the Universities Space Research Association

NASA
National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665

```
           DISPLAY 25/2/1
      84N31988*#   ISSUE 21   PAGE 3469   CATEGORY 62   RPT#: NASA-CR-172418
      ICASE-84-34 NAS 1.26:172418   CNT#: NAS1-17070 NAS1-17130   84/07/00  29
      PAGES   UNCLASSIFIED DOCUMENT
UTTL: A model of asynchronous iterative algorithms for solving large, sparse,
      linear systems
AUTH: A/REED, D. A.;  B/PATRICK, M. L.   PAA: A/(North Carolina Univ., Chapel
      Hill);  B/(Duke Univ.)
CORP: National Aeronautics and Space Administration. Langley Research Center,
      Hampton, Va.    AVAIL.NTIS   SAP: HC A03/MF A01
MAJS: /*ALGORITHMS/*COMPUTATION/*ITERATION/*LINEAR EQUATIONS/*LINEAR SYSTEMS/*
```

A Model of Asynchronous Iterative Algorithms for Solving Large, Sparse, Linear Systems

Daniel A. Reed[†]

Department of Computer Science
University of North Carolina
Chapel Hill, North Carolina 27514

Merrell L. Patrick[†]

Department of Computer Science
Duke University
Durham, North Carolina 27706

ABSTRACT

Solving large, sparse, linear systems of equations is one of the fundamental problems in large scale scientific and engineering computation. A model of a general class of asynchronous, iterative solution methods for linear systems is developed. In the model, the system is solved by creating several cooperating tasks that each compute a portion of the solution vector. This model is then analyzed to determine the expected intertask data transfer and task computational complexity as functions of the number of tasks. Based on the analysis, recommendations for task partitioning are made. These recommendations are a function of the sparseness of the linear system, its structure (i.e., randomly sparse or banded), and dimension.

[†]The research reported here was supported in part by the National Aeronautics and Space Administration under NASA Contracts No. NAS1-17070 and No. NAS1-17130 and was performed while the authors were visitors at ICASE, NASA Langley Research Center, Hampton, VA 23665. In addition, the second author was also supported by Control Data Grant No. 80D05.

N84-32988#

Introduction

In this paper we focus on *iterative* methods for solving *large, sparse* linear systems on MIMD computers. In related work, Adams [1] has studied parallel implementations of iterative methods for the linear systems arising from finite element analysis, with particular emphasis on mapping the methods on the FEM [3], an MIMD machine being developed at the NASA Langley Research Center. She has also developed models for predicting the performance of these algorithms and validated them using the FEM [2]. Gannon and Van Rosendale [6] have also recently proposed a parallel architecture for another class of iterative algorithms based on multigrid methods. Finally, Amano, Yoshida, and Aiso [5] have proposed a parallel architecture, called the *S*parse *M*atrix *S*olving *M*achine, $(SM)^2$, for iteratively solving sparse linear systems.

In the remainder of the paper, we precisely define both the problem and the class of iterative methods used to solve it, and we discuss one possible implementation. We then define a probabilistic model for predicting iteration time and an optimal number of data partitions given the dimension and sparsity of the coefficient matrix and the costs of computation, synchronization, and communication. We conclude with graphs and analyses of execution time as a function of the number of matrix partitions for various parameter values.

Problem Definition

Consider a linear system of equations of the form

$$Kx = f \tag{1}$$

where K is a large $N \times N$ sparse matrix and x and f are vectors of length N. Such systems are frequently rewritten in the form

$$x = Ax + c$$

and solved using the iteration formula

$$x^{(i+1)} = Ax^{(i)} + c \tag{2}$$

where x and c are N-vectors and A is another sparse $N \times N$ matrix. Although A is a function of K, and c is a function of K and the f vector, there are many ways to choose A and c such that (2) describes a convergent iterative scheme for (1). We only assume that they are chosen such that the sequence of iterates $<x^{(i)}>$ converges to the solution. Henceforth, we consider only the parallel implementation of the computation defined by (2).

Parallel Solution Technique

One parallel computation schema for (2) is illustrated by the diagram in Figure I. The matrix A and the vectors c and $x^{(i+1)}$ (denoted by XN) are partitioned into sets of rows. A basic iteration step of the computation is then partitioned into the set of computations defined symbolically by

$$\text{XSET}[\text{I}] = \text{ASET}[\text{I}] * \text{XO} + \text{CSET}[\text{I}] \quad \text{I} = 1,\ldots,\text{M}.$$

After each basic iteration, a norm of the vector XN − XO must be checked for convergence. If convergence has occurred or the maximum allowable number of iterations has been exceeded, the iteration halts; otherwise XO is replaced by XN, and the iteration step is repeated.

This computation schema can be realized as follows. In the main program, the data objects and their types are declared. In addition, worker tasks, called X_TASKs, and their controlling task, the C_TASK, are defined. The body of the main program reads the input data, initiates the control task, which in turn initiates the worker tasks, and prints the solution vector after the control task has terminated.

X_TASK[I] computes the components of the vector XN corresponding to XSET[I]. To accomplish this, X_TASK[I] needs the non-zero elements of A corresponding to ASET[I], the elements of c corresponding to CSET[I], and a portion of the vector XO, specifically those elements whose subscripts are the same as the column subscripts of the non-zero elements of A in ASET[I]. Initially, this information is sent to each of the X_TASKs by the C_TASK. After each iteration, if convergence has not occurred, the

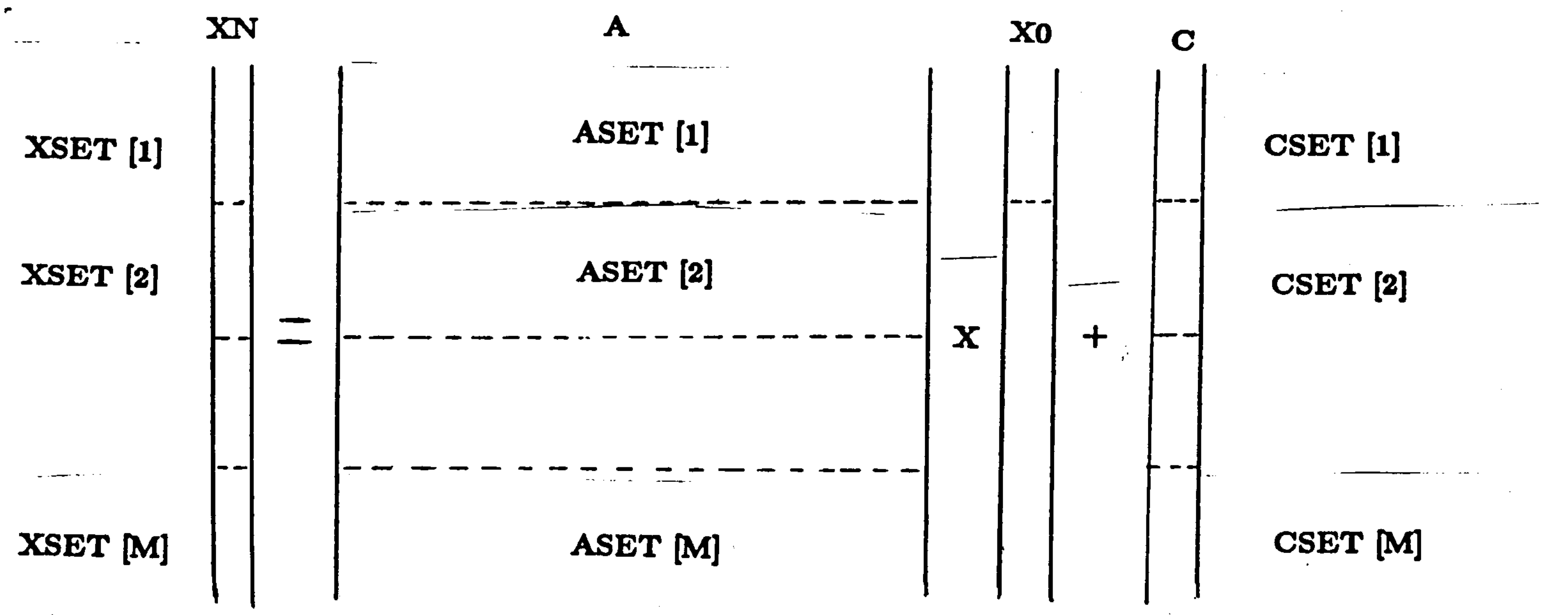


Figure I Partitioning of a linear system

vector XO is replaced with the vector XN. Because of this replacement, each X_TASK must send those components of XN that it computes to the other X_TASKs that need them. Each X_TASK then determines if the XN components just computed have converged and notifies the C_TASK accordingly by sending a boolean flag. When an X_TASK is notified by the C_TASK that all X_TASKs report convergence of their components, it sends the current values of its XN components to the C_TASK and terminates.

The role of the C_TASK is now clear. After initializing the X_TASKs and sending them the initial data they need to iterate, it receives local convergence information from each X_TASK, determines if global convergence has occurred, and notifies the X_TASKs accordingly. If global convergence has occurred, the C_TASK then receives the components of XN and terminates.

An Analytical Model of the Computation

Objectives

Because the intent of parallel computation is a reduction of the expected execution time, we must consider the *performance* of the parallel, sparse, linear systems iteration algorithm just described. Unlike sequential algorithms, the performance of a parallel algorithm depends not only on the number of arithmetic operations but also on the amount and frequency of intertask data transfer. Consequently, we derive formulae describing

- the amount of data transfer among X_TASKs needed for each iteration,
- the computational complexity of each X_TASK, and
- the time to synchronize the X_TASKs.

Based on these formulae, we create a model for predicting performance as a function of both the number and size of matrix partitions and the matrix sparsity. This performance prediction model is then applied to both the general case of randomly sparse matrices and an important special case, the band matrix.

Notation and Assumptions

Unless otherwise specified, we assume the elements of the matrix A are randomly non-zero with probability P (i.e., $p(a_{ij} \neq 0) = P$). In our model of matrix sparsity, the probability function P is determined by imposing two very weak conditions on A. First, we require each row of A to contain at least Z non-zero elements, each randomly distributed throughout the row. Second, each row element not known to be one of the Z non-zero values is itself assumed to be non-zero with probability q.

Given the two conditions above, the value of P can be derived using a straightforward application of conditional probabilities. We define two events:

A: a_{ij} is one of the Z non-zero elements in row i

B: a_{ij} is a non-zero element with probability q

Then

$$\begin{aligned} P(N, Z, q) &= p(a_{ij} \neq 0) \\ &= p(A) + p(B) - p(A \text{ and } B) \\ &= \frac{Z}{N} + q - q\left(\frac{Z}{N}\right) \\ &= \frac{Z}{N}(1 - q) + q. \end{aligned}$$

Finally, we require Z to be greater than zero. Otherwise, this sparsity model includes matrices containing one or more identically zero rows; the consequent singularity must be avoided.

Throughout our discussion, M denotes the number of partitions of A (i.e., the number of X_TASKs), and b_j and e_j respectively denote the indices of the beginning and ending rows of partition j. This notation, and that introduced throughout the remainder of our analysis, is summarized in Table I.

Table I Notation

Quantity	*Definition*
A	arbitrary $N \times N$ sparse matrix
b	matrix semi-bandwidth
b_j	initial row of partition j
c	arbitrary constant N-vector
$C_b(M)$	transmission time for a boolean as a function of the number of partitions
C_p	computation time for arithmetic operations
C_s	startup time for data transmission
$C_t(M)$	transmission time for one datum as a function of the number of partitions
e_j	final row of partition j
M	number of matrix partitions
N	matrix dimension
$P(N, Z, q)$	probability that a matrix element is non-zero
$P_{Tr}(k, j)$	probability that partition k transfers data to partition j
q	probability that a matrix element is non-zero given that it is not known to be zero
S	fixed partition width
$t_j(comm)$	communication time for one iteration at partition j
$t_j(comp)$	computation time for one iteration at partition j
$Tr(k, j)$	expected data transfer from partition k to j
x	N-vector of unknowns
Z	number of known non-zero elements in a row

Data Transfer for Sparse Matrices

Given a sparse matrix A whose elements are randomly non-zero with probability $P(N, Z, q)$ and two partitions j and k, we wish to determine the data transfer from partition k to partition j needed to perform one iteration. For pedagogic purposes, we consider three cases of increasing generality.

Case I: j and k are single row partitions

Partition j requires the single value x_{b_k} if and only if $a_{b_j b_k} \neq 0$. Since this occurs with probability $P(N, Z, q)$, the expected data transfer from k to j is simply $P(N, Z, q)$.

Case II: j is a multiple row partition; k is not

Clearly, partition j does not need x_{b_k} if and only if $a_{ib_k} = 0$ for all i in the range $b_j \leq i \leq e_j$. By assumption, each matrix element is randomly non-zero. Hence, the probability that at least one element of the column b_k in partition j is non-zero is

$$1 - \left[1 - P(N, Z, q)\right]^{e_j - b_j - 1}.$$

Because partition k contains only one row, the expected data transfer from k to j is the same.

Case III: both j and k are multiple row partitions

This case is illustrated in Figure II. An immediate generalization of the previous case, partition j does not need any x_l if and only if $a_{il} \neq 0$ for i in the range $b_j \leq i \leq e_j$ and all l in the range $b_k \leq l \leq e_k$. Consequently, the expected data transfer from partition k to partition j is just $(e_k - b_k + 1)$ times that of case II, namely,

$$Tr(k, j) = \quad (3)$$
$$(e_k - b_k + 1)\left[1 - \left[1 - P(N, Z, q)\right]^{e_j - b_j + 1}\right].$$

Finally, the probability, $P_{Tr}(k, j)$, that partition j needs at least one element from

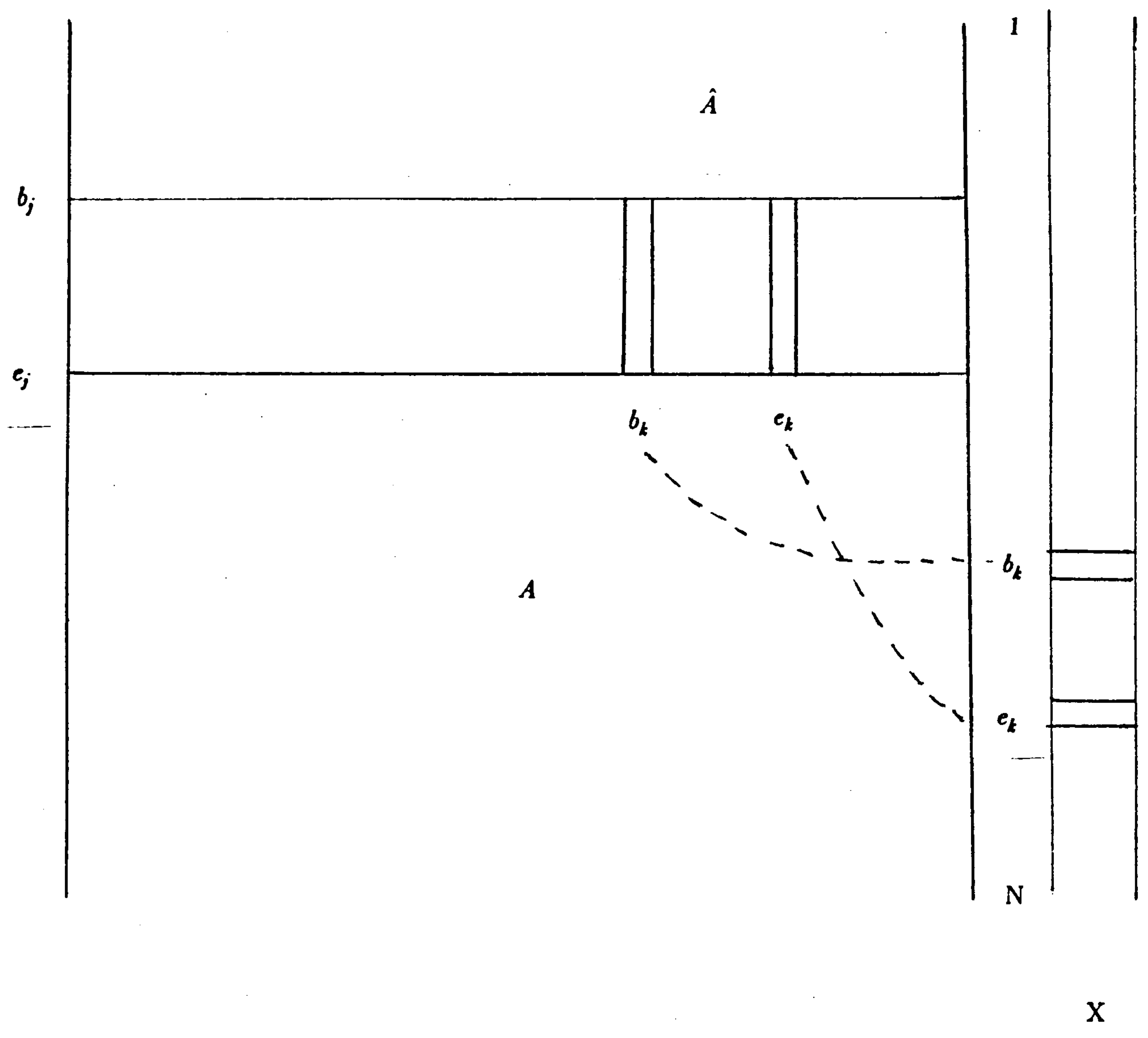


Figure II

Data transfer between multiple row partitions partition k is just the probability that the submatrix delimited by rows b_j and e_j and columns b_k and e_k, matrix $\hat{A}$ in Figure II, is not identically zero. This probability is just

$$P_{Tr}(k, j) = 1 - \Big[1 - P(N, Z, q)\Big]^{(e_j - b_j + 1)(e_k - b_k + 1)} \tag{4}$$

Although general, (3) and (4) provide little insight or intuition about data transfer as a function of either $P(N, Z, q)$ or M. If the partition size is constant, simpler expressions can be obtained. Hence, we fix $(e_j - b_j + 1)$, the partition size, at a constant $S = N/M$ for all partitions. Then replacing $P(N, Z, q)$ by its definition, we obtain

$$Tr(k, j) = \frac{N}{M}\left[1 - \left[1 - q\right]^{\frac{N}{M}}\left[1 - \frac{Z}{N}\right]^{\frac{N}{M}}\right]$$

and

$$P_{Tr}(k, j) = 1 - \left[\left[1 - q\right]^{\left(\frac{N}{M}\right)^2}\left[1 - \frac{Z}{N}\right]^{\left(\frac{N}{M}\right)^2}\right].$$

Data Transfer for Band Matrices

In addition to randomly sparse matrices, there are many other sparse matrices with discernible structure, notably band matrices. For a band matrix A with semi-bandwidth b, $a_{ij} \neq 0$ only if $|i - j| \leq b$.

Applying the sparsity model derived for the random sparsity case, the probability $P_{ij}^{band}(N, Z, q)$ that $a_{ij} \neq 0$ is given by

$$P_{ij}^{band}(N, Z, q) = \begin{cases} P(2b, Z, q) & \text{if } |i - j| \leq b \\ 0 & \textit{otherwise.} \end{cases}$$

Unlike the random sparsity case, all elements of the band matrix are not non-zero with equal probability. Hence, a direct substitution of $P_{ij}^{band}(N, Z, q)$ into (3) is inappropriate. Consider, however, a single column m of partition j where $b_k \leq m \leq e_k$. As with random sparsity, partition j does not need element x_m if and only if column m is identically zero. This occurs with probability

$$\prod_{l=b_j}^{e_j} \left[1 - P_{lm}^{band}(N, Z, q)\right] \qquad b_k \le m \le e_k.$$

Hence, the probability that partition j needs x_m is

$$1 - \prod_{l=b_j}^{e_j} \left[1 - P_{lm}^{band}(N, Z, q)\right],$$

and the expected data transfer from partition k to partition j is

$$\sum_{m=b_k}^{e_k} \left[1 - \prod_{l=b_j}^{e_j} \left[1 - P_{lm}^{band}(N, Z, q)\right] \right]. \tag{5}$$

Now consider column m, shown in Figure III. It can only contain non-zero elements if it lies between columns $b_j - b$ and $e_j + b$ inclusive. Otherwise, it would lie outside the intersection of the matrix band and partition j. Moreover, column m can only cause data transfer from partition k to partition j if it lies between columns b_k and e_k inclusive. Hence, the structure of the band matrix implies that

$$\max\left\{ b_k , b_j - b \right\} \le m \le \min\left\{ e_k , e_j + b \right\}.$$

Now consider the rows l associated with column m. By the definition of a band matrix, non-zero elements in column m must lie between rows $m - b$ and $m + b$ inclusive. Moreover, the rows are constrained to lie within the partition j. Hence,

$$\max\left\{ b_j , m - b \right\} \le l \le \min\left\{ e_j , m + b \right\}.$$

Within these constraints on l and m, $P_{lm}^{band}(N, Z, q)$ is just $P(2b, Z, q)$. Hence, (5) reduces to

$$\sum_{m=s_l}^{s_u} \left[1 - \prod_{l=p_l}^{p_u} \left[1 - P(2b, Z, q)\right] \right] \tag{6}$$

where

$$s_l = \max\left\{ b_k , b_j - b \right\},$$

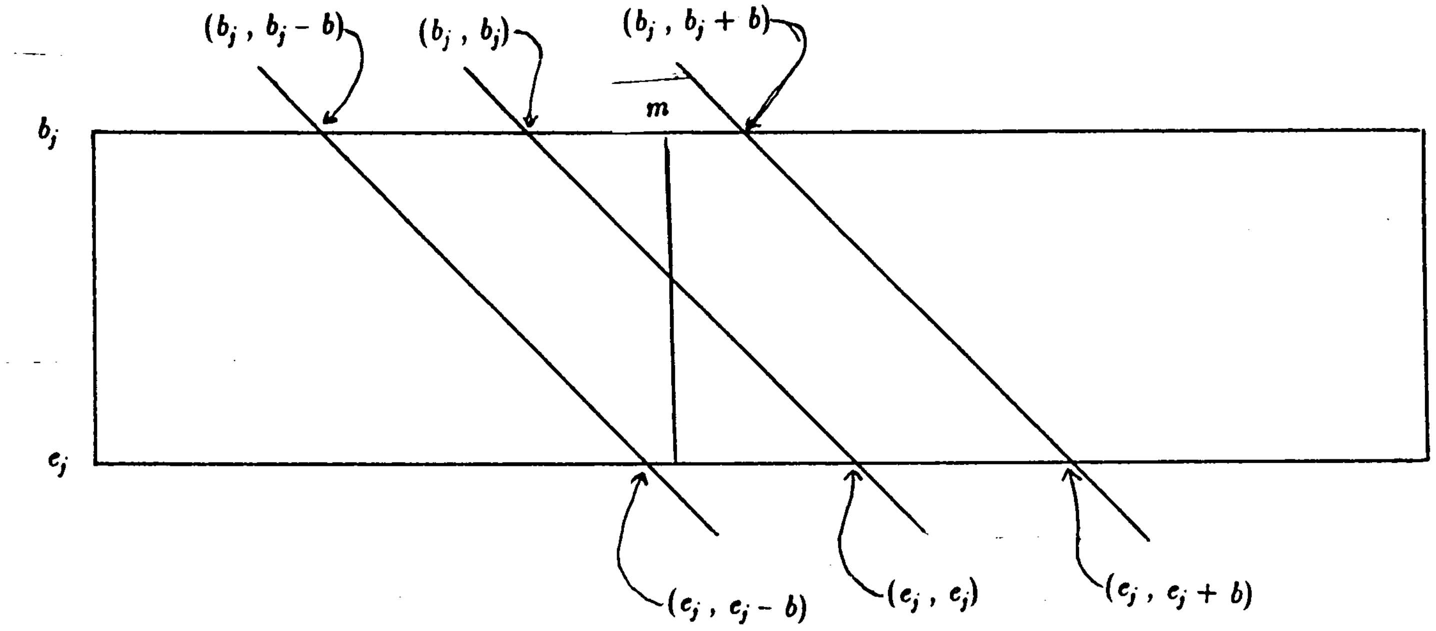


Figure III

Band of non-zero elements intersecting partition j

$$s_u = \min\left\{e_k , e_j + b\right\},$$

$$p_l = \max\left\{b_j , m - b\right\},$$

and

$$p_u = \min\left\{e_j , m + b\right\}.$$

To obtain a closed form, we again reduce the problem to one of fixed size partitions S. Then the limits on (6) simplify to

$$s_l = \max\left\{(k-1)S + 1 , (j-1)S + 1 - b\right\},$$

$$s_u = \min\left\{kS , jS + b\right\},$$

$$p_l = \max\left\{(j-1)S + 1 , m - b\right\},$$

and

$$p_u = \min\left\{jS , m + b\right\}.$$

Further simplification of this summation unfortunately requires enumeration of several cases. These cases are a function of the relationships among the matrix bandwidth, the partition size, and the relative positions in the matrix of the partitions j and k. Fortuitously, those cases where $j > k$ are symmetric with $j < k$. Hence, we consider only the case $j < k$. Derivation of the remaining cases is still a lengthy endeavor, providing little insight. Consequently, we simply describe the cases, using Figure IV, and enumerate the results.

Case I: $(k-1)S + 1 > jS + b$

Here, the submatrix determining possible data transfer from partition k to partition j lies outside the matrix band. Consequently, the submatrix is identically zero and no data transfer occurs. This case arises if

$$k - j > \frac{b-1}{R} + 1.$$

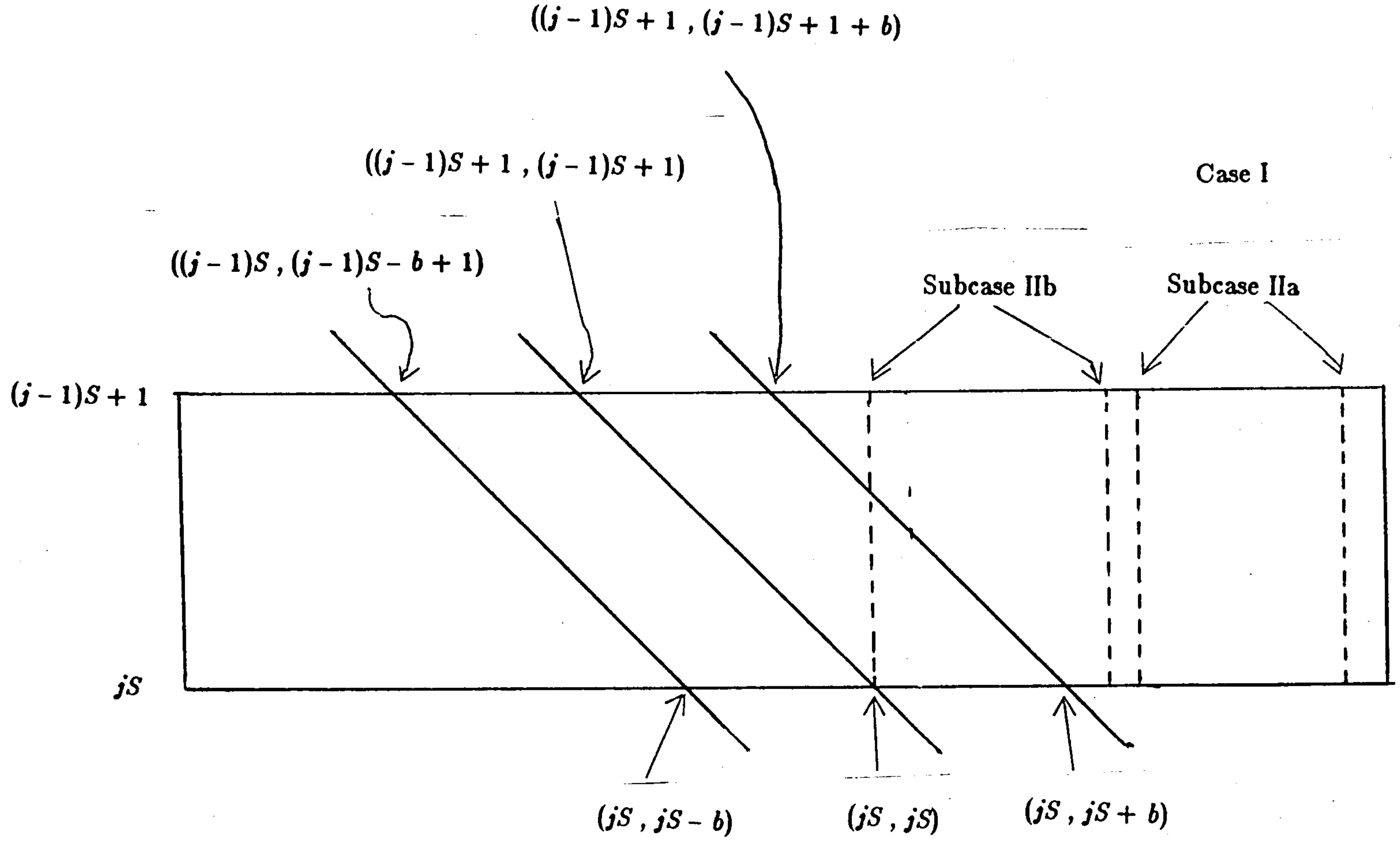


Figure IV

Data transfer cases for band matrix with fixed partitions (part I)

For the remainder of the cases, we implicitly assume that *some* data transfer occurs (i.e., $(k-1)S+1 \leq jS+b$).

Case II: $b \leq S$

In this case, the partition width exceeds half the bandwidth. Two subcases, based on possible positions of k, arise.

Subcase IIa: $k > (j+1)$

This condition, coupled with that of case II, places the determining submatrix outside the band, and no data transfer occurs.

Subcase IIb: $k = (j+1)$

Partitions j and k are adjacent. Moreover, partition k is the *only* partition transferring data to j such that $k > j$. The expected data transfer is

$$Tr(k, j) = b + \frac{\left[1 - P(N, Z, q)\right]\left[P(N, Z, q)^b - 1\right]}{P(N, Z, q)}. \tag{7}$$

Similarly, only partition $j-1$ transfer data to j from the other side. Hence, if $b \leq S$, only adjacent partitions must exchange data. This suggests that this partition size for band matrices might be well suited to a ring architecture.

The probability that partition k must transfer data to partition j is again just the probability that the submatrix is not identically zero, or

$$P_{Tr}(k, j) = \prod_{m=jR+1}^{jR+b} \prod_{l=m-b}^{jR} \left[1 - P(N, Z, q)\right]$$
$$\left[1 - P(N, Z, q)\right]^{\frac{b(b+1)}{2}}.$$

Case III: $b > S$

The converse of case II, the partition size is less than half the matrix bandwidth. As before, subcases based on the possible positions of partition k arise.

Subcase IIIa: $kS < (j-1)S + b$

Here, the determining submatrix lies *completely* within the band, and the expected

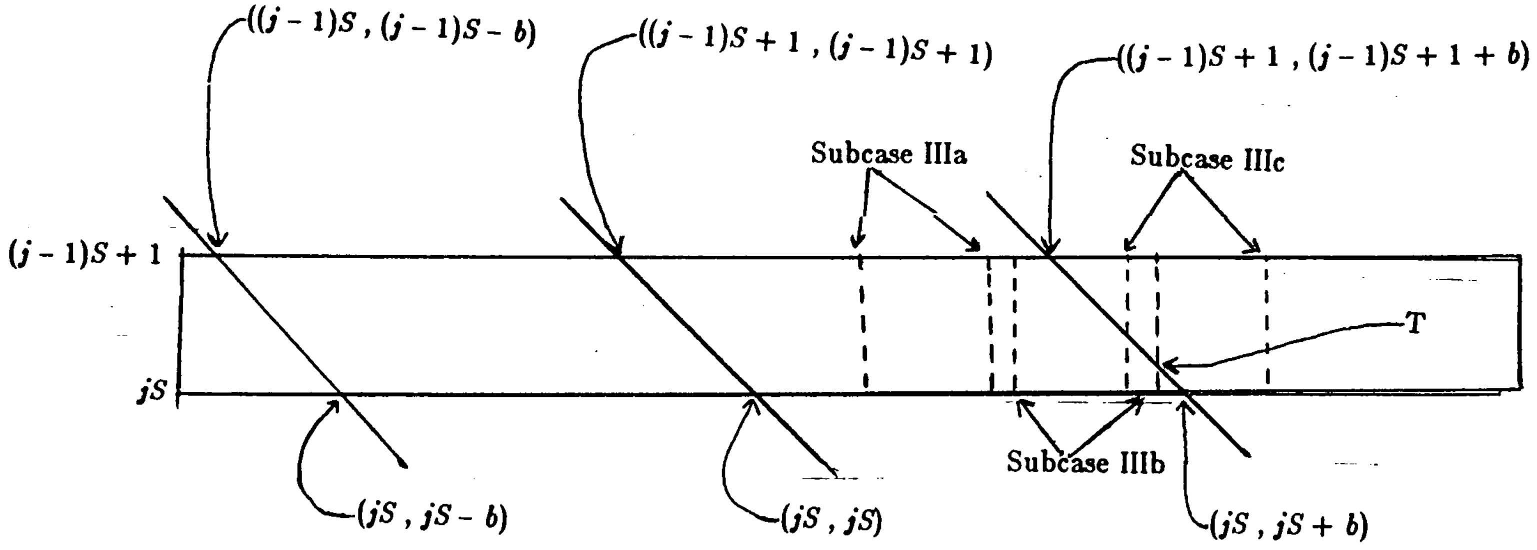

((j - 1)S , (j - 1)S - b)
((j - 1)S + 1 , (j - 1)S + 1)
((j - 1)S + 1 , (j - 1)S + 1 + b)
Subcase IIIa
Subcase IIIc
(j - 1)S + 1
T
jS
Subcase IIIb
(jS , jS - b)
(jS , jS)
(jS , jS + b)

Figure IV

Data transfer cases for band matrix with fixed partitions (part II) data transfer is

$$S\left[1 - \left[1 - P(N, Z, q)\right]^{S}\right].$$

Subcase IIIb: $(j-1)S + b + 1 \leq kS < jS + b$

The determining submatrix lies partially within the matrix band, and every column also lies partially within the band. If T denotes the column of partition j where the last column of the determining submatrix intersects the right edge of the band, the expected data transfer from partition k to j is

$$S - T\left[1 - \left[1 - P(N, Z, q)\right]^{S}\right] - \frac{\left[1 - P(N, Z, q)\right]^{jS+b+1}}{P(N, Z, q)}\left[\left(\frac{1}{1 - P(N, Z, q)}\right)^{kS} - \left(\frac{1}{1 - P(N, Z, q)}\right)^{T-1}\right].$$

Subcase IIIc: $kS \geq jS + b$

Finally, the determining submatrix can lie partially within the band with some columns entirely outside the band. This leads to an expected data transfer of

$$(j-k+1)S + \left[\frac{1 - P(N, Z, q)}{P(N, Z, q)}\right]\left[\left[1 - P(N, Z, q)\right]^{(j-k+1)S+b-1} - 1\right].$$

Parallel Computational Complexity

As noted earlier, the performance of a parallel algorithm depends on both the intertask data transfer *and* the amount of computation performed by each task. Having considered the former, we turn our attention to the latter.

Each of the parallel X_TASKs is itself just a sequential code whose two primary constituents, inner product and convergence test, were described earlier. Consequently, we can apply standard techniques [4] to determine the complexity of each X_TASK. The results of this analysis are shown in Table II.

We assume that all indexing and arithmetic operations require the same amount of time C_p. Combining the results for the inner product and convergence test, the computational complexity of an arbitrary X_TASK is

Table II Computational Complexity of X_TASKs

Loop Cost	*Statement Cost*	*Statement*
(1)	$2C_p$	FOR I := E [J] to B [J] DO
		BEGIN
	C_p	SUM := 0;
(2)	$2C_p$	FOR K := L [J] to U [J] DO
	$6C_p$	SUM := SUM + ANZ [K] * X0 [COLSUB [K]]†
	$4C_p$	XN [I] := SUM + C [I]
		END;
	C_p	CONVERGED := TRUE;
(3)	$2C_p$	FOR I := E [J] to B [J] DO
		BEGIN
	$5C_p$	IF ABS (XN [I] - X0 [I]) > EPS THEN
	C_p	CONVERGED := FALSE
	$3C_p$	X0 [I] := XN [I]
		END;

(1): $(e_j - b_j + 1)C_p\Big[6(N \text{ or } 2b)P(N, Z, q) + 7\Big] + 2C_p$

(2): $6C_p(N \text{ or } 2b)P(N, Z, q) + 2C_p$

(3): $(e_j - b_j + 1)9C_p + 3C_p$

†ANZ and COLSUB are vectors of the non-zero elements of A and the corresponding column subscripts, respectively. L [J] and U [J] denote the beginning and ending indices of components of these vectors belonging to partition J.

$$(e_j - b_j + 1)C_p(6NP(N, Z, q) + 16) + 5C_p \tag{8}$$

for the random matrix and

$$(e_j - b_j + 1)C_p(12bP(N, Z, q) + 16) + 5C_p$$

for the band matrix. The C_TASK must also check for global convergence after each iteration. This consists of ANDing the M local convergence flags received from the X_TASKs and requires

$$(3M + 1)C_p$$

operations.

Model Description

Having just determined the expected amount of data transfer among X_TASKs (partitions), and their computational complexity, we can now define an execution time model of the parallel, sparse matrix algorithm. This model can then be used to predict the execution time of one iteration.

Let $t_j(comp)$ denote the computational complexity of X_TASK j, $t_j(comm)$ denote the time required for task j to send and receive all data needed for the next iteration, and $t(sync)$ be the time required for the C_TASK to receive and test all local synchronization flags. Then the total execution time for one sparse matrix iteration is

$$t(sync) + \max_{1 \leq j \leq M} \left\{ t_j(comp) + t_j(comm) \right\}. \tag{9}$$

Clearly, the time required to transmit or receive a datum is some function of the number of partitions (X_TASKs) concurrently operating (e.g., if only two X_TASKs were operating in parallel, they should be able to exchange data more quickly than if fifty additional X_TASKs were also operating). Hence, we make both the time needed to transmit a boolean, $C_b(M)$, and the time to transmit an x value, $C_f(M)$, functions of M.

We now consider each component of the execution time. Given that $C_b(M)$ denotes the time needed to transmit a single boolean value, then $t(sync)$ is given by

$$\begin{array}{ccccc} \textit{RECEIVE FLAGS} & & \textit{TEST FLAGS} & & \textit{SEND FLAGS} \\ MC_b(M) & + & (3M+1)C_p & + & MC_b(M). \end{array}$$

Of course, $t_j(comp)$ is given by (8). The communication component, $t_j(comm)$ is, however, somewhat more complicated. In addition to including the interpartition data transfer, it should also include startup costs for data transmission. That is, two partitions exchanging ten data values should require less time than four partitions exchanging five data values. This intent is reflected by the formula

$$\begin{aligned} t_j(comm) &= \textit{send to other partitions} \\ &+ \textit{receive from other partitions} \\ &= \sum_{\substack{k=1 \\ k\neq j}}^{M} \Big[C_s P_{Tr}(k,j) + C_t(M) Tr(k,j) \Big] \\ &+ \sum_{\substack{k=1 \\ k\neq j}}^{M} C_t(M) Tr(k,j) \end{aligned} \tag{10}$$

where C_s is the startup cost for initiating a data transfer.

Given these formulae, consider the two matrix cases for which we derived closed forms for $P_{Tr}(k,j)$ and $Tr(k,j)$, the randomly sparse matrix and the band matrix both with fixed partition size.

Randomly Sparse Matrix

Substituting values in (10) for $P_{Tr}(k,j)$ and $Tr(k,j)$ gives

$$t_j^{random}(comm) =$$

$$C_s(M-1)\left[1 - \left[(1-q)\left[1-\frac{Z}{N}\right]\right]^{\left[\frac{N}{M}\right]^2}\right] +$$

$$\frac{2C_t(M)N(M-1)}{M}\left[1 - \left[(1-q)\left[1-\frac{Z}{N}\right]\right]^{\frac{N}{M}}\right].$$

Band Matrix

For the band matrix, case IIb, we have

$$t_j(comm) = 2C_s\left[(1-q)\left[1-\frac{Z}{2b}\right]\right]^{\frac{b(b+1)}{2}} +$$

$$2C_t(M)\left[b + \frac{(1-q)\left[1-\frac{Z}{2b}\right]\left[\left[\left[1-\frac{Z}{2b}\right](1-q)\right]^{b} - 1\right]}{\frac{Z}{2b}(1-q)+q}\right].$$

Conclusions Based on the Model

As we have seen, the total execution time for one sparse matrix iteration is given by (9). For equal sized partitions, (9) simplifies to

$$t(sync) + t_j(comp) + t_j(comm). \tag{11}$$

There are two primary means of implementing communication in a parallel system, shared memory and communication networks. In both cases, the delays incurred for data transfer increase as the number of parallel tasks increase. (Shared memory suffers from memory access conflicts, and communication networks, being necessarily incomplete connections, require additional routing of data.) Hence, it seems appropriate to make the synchronization and data transmission costs functions of the number of partitions M (i.e., the number of parallel X_TASKs). We used the functions

$$f(M) = \begin{cases} 1 \\ \log_2(M) \\ \sqrt{M} \\ M \end{cases}$$

in the communication component of (11) to reflect the possible range of communication costs one might encounter in a complete connection, tree, square mesh, and ring, respectively.

Using (11) and the communication cost function, $f(M)$, we then plotted total execution time as a function of matrix sparsity, $P(N, Z, q)$, computation time, C_p, communication time, C_t and C_s, and synchronization cost, C_b, for the random sparsity case. These plots, shown in Figures V-VII, are discussed in detail below. In all cases, the smallest number of partitions chosen was $M = 5$.

Figure V

This figure shows iteration time as a function of the number of matrix partitions (X_TASKs) for varying communication costs. Each matrix row contains 14 non-zero elements, a typical number for a matrix arising from a finite element method.

As can be seen, there exists an optimal level of parallelism in each case. Not surprisingly, the optimum level of parallelism declines as the communication costs increase. Even the complete connection cannot support as many parallel tasks as there are matrix rows. The reason is quite simple, as the number of partitions grows, synchronization costs become prohibitive.

Figure VI

This figure shows the effect of matrix sparsity on iteration time for communication costs proportional to $\sqrt{M}$; the lowest curve corresponds to greatest sparsity. As expected, increasing the number of non-zero elements results in increased iteration time. In addition, the optimum level of parallelism increases as the number of non-zero elements increases.

Figure VII

Finally, this figure shows iteration time for varying matrix sizes, again with communication costs proportional to $\sqrt{M}$.

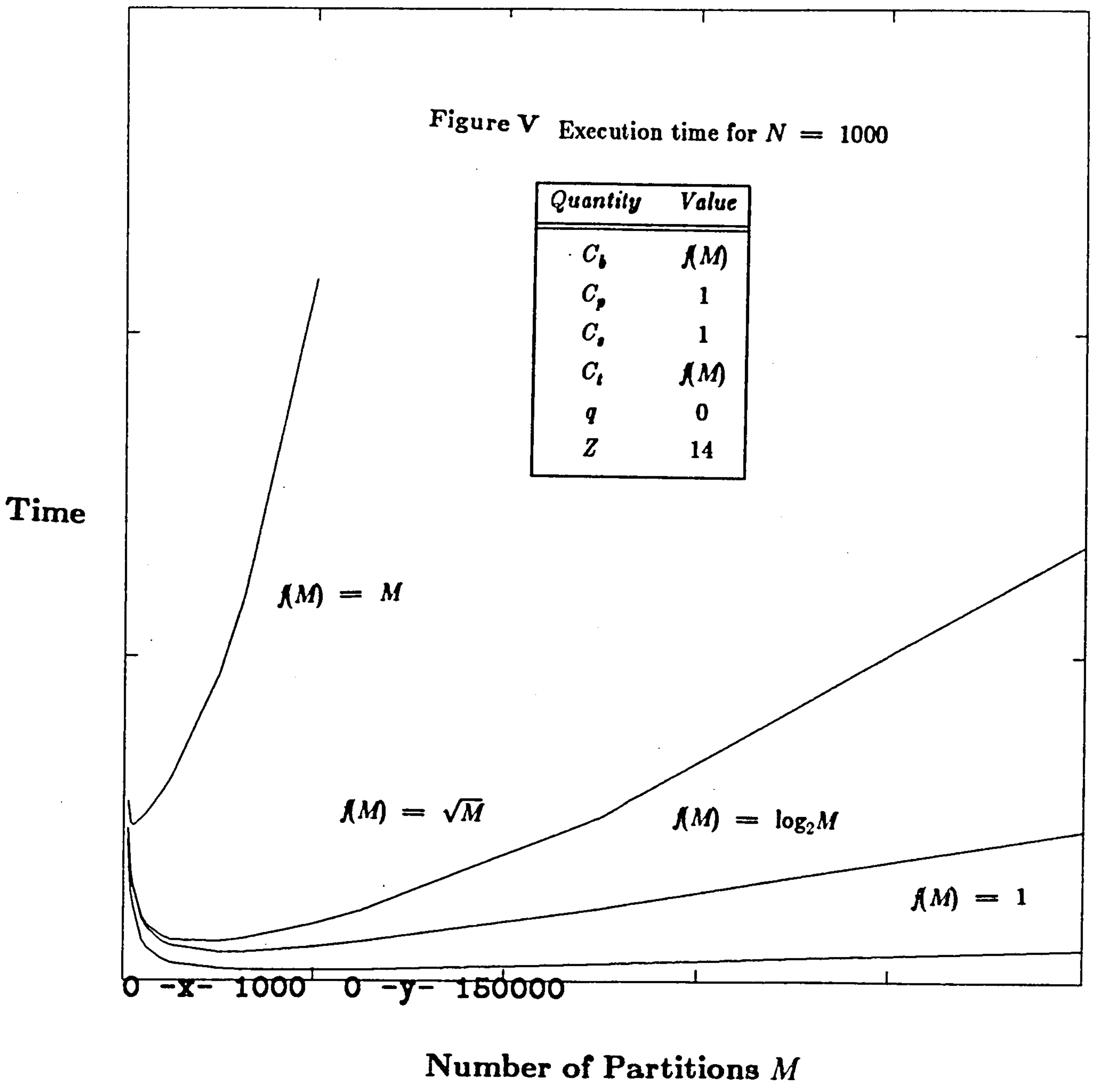


Figure V Execution time for $N = 1000$

Quantity	*Value*
C_b	$f(M)$
C_p	1
C_s	1
C_t	$f(M)$
q	0
Z	14

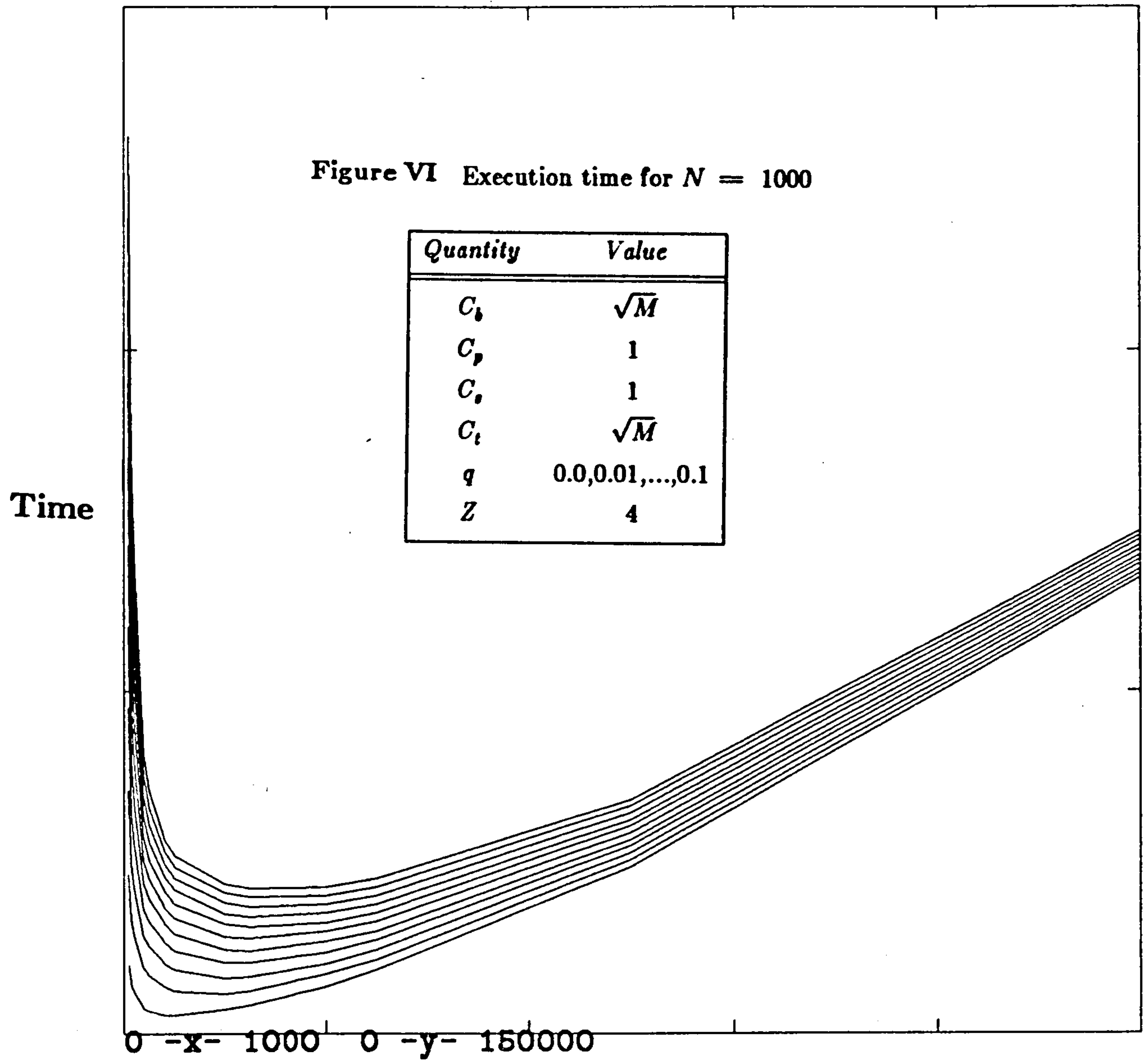
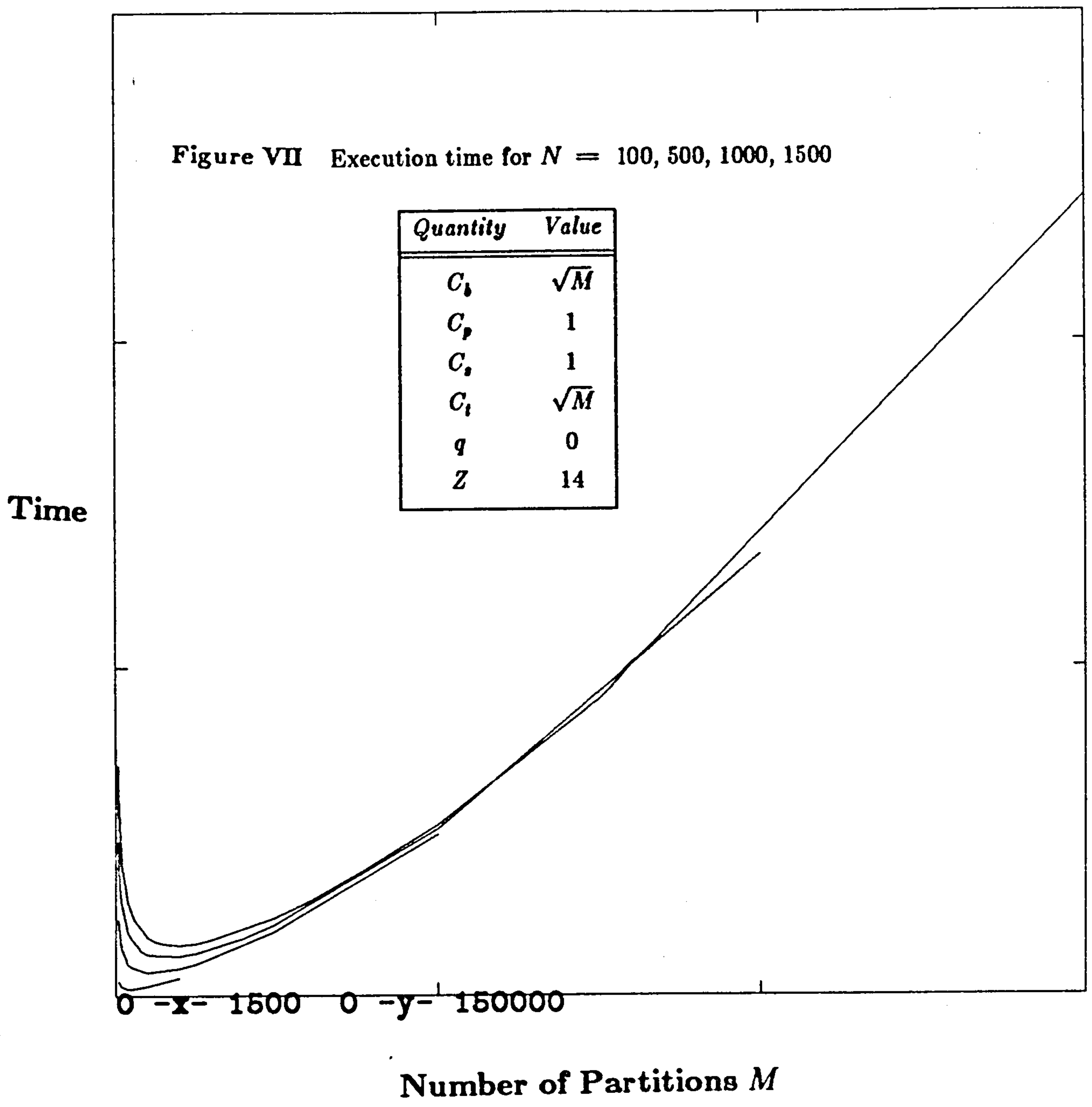


Figure VI Execution time for $N = 1000$

Quantity	*Value*
C_b	$\sqrt{M}$
C_p	1
C_s	1
C_t	$\sqrt{M}$
q	0.0,0.01,...,0.1
Z	4

Figure VII Execution time for $N = 100, 500, 1000, 1500$

Quantity	*Value*
C_b	$\sqrt{M}$
C_p	1
C_s	1
C_t	$\sqrt{M}$
q	0
Z	14



Band Matrices

The execution time model for banded matrices, case A, is easier to analyze. We have seen that intertask data transfer occurs only between adjacent tasks if the width of a partition is ***at least as large*** as the matrix semi-bandwidth. If this condition is met, the optimum number of partitions (X_TASKs) depends on the relative costs of computation and communication.

Summary

As we have seen, the performance of a parallel algorithm depends not only on the amount of computation performed by each task but also on the amount and frequency of intertask data transfer and task synchronization.

For a parallel implementation of iterative methods for solving sparse linear systems of equations, we have derived the expected intertask data transfer and defined an execution time model that can be used to predict iteration time. We have applied the model to both the general case of randomly sparse matrices and one important special case, banded matrices.

Results of the model clearly show that the execution time of the solution methods can be reduced by partitioning the computation into parallel subtasks. However, the optimum number of partitions is very dependent on synchronization and communication costs.

Acknowledgments

We are particularly indebted to Loyce Adams, Piyush Mehrotra, Terry Pratt, John van Rosendale and Robert Voigt, our colleagues in the XFEM Research group at ICASE, NASA Langley Research Center, for many helpful discussions. We would also like to thank Dennis Gannon of Purdue University and Mike Leuze of Vanderbilt University for their ideas.

References

[1] L. Adams, "Iterative Algorithms for Large Sparse Linear Systems on Parallel Computers," NASA CR-166027, NASA Langley Research Center, November 1982, (also published as a Ph.D. dissertation, University of Virginia).

[2] L. Adams and T. Crockett, "Modeling Algorithm Execution Time on Processor Arrays," IEEE Computer, Vol. 17, No. 7, July 1984, pp. 38-44.

[3] L. Adams and R. Voigt, "Design, Development, and Use of the Finite Element Machine," ICASE Report No. 83-56, NASA CR-172250, NASA Langley Research Center, October 1983, (also published in Proc. of Conference on Large Scale Scientific Computations, University of Wisconsin, 1983, Academic Press).

[4] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, The Design and Analysis of Computer Algorithms, Addison-Wesley, 1974.

[5] H. Amano, T. Yoshida, and H. Aiso, "$(SM)^2$: Sparse Matrix Solving Machine," The 10th Annual International Symposium on Computer Architecture, ACM Sigarch Newsletter, Vol. 11, No. 3, June 1983, pp. 213-220.

[6] D. Gannon and J. Van Rosendale, "Parallel Architectures for Iterative Methods of Adaptive, Block Structured Grids," ICASE Report No. 83-39, NASA CR-172195, NASA Langley Research Center, August 1983, (also published in the Proceedings of the Monterey Elliptic Solver Conference, Academic Press, G. Birkhoff, (ed.)).

1. Report No. NASA CR-172418 ICASE Report No. 84-34	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle A Model of Asynchronous Iterative Algorithms for Solving Large, Sparse, Linear Systems		5. Report Date July 1984
		6. Performing Organization Code
7. Author(s) Daniel A. Reed and Merrell L. Patrick,		8. Performing Organization Report No. 84-34
9. Performing Organization Name and Address Institute for Computer Applications in Science and Engineering Mail Stop 132C, NASA Langley Research Center Hampton, VA 23665		10. Work Unit No.
		11. Contract or Grant No. NAS1-17070, NAS1-17130
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, D.C. 20546		13. Type of Report and Period Covered Contractor Report
		14. Sponsoring Agency Code 505-31-83-01

15. Supplementary Notes

Langley Technical Monitor: R. H. Tolson
Final Report

16. Abstract

Solving large, sparse, linear systems of equations is one of the fundamental problems in large scale scientific and engineering computation. A model of a general class of asynchronous, iterative solution methods for linear systems is developed. In the model, the system is solved by creating several cooperating tasks that each compute a portion of the solution vector. This model is then analyzed to determine the expected intertask data transfer and task computational complexity as functions of the number of tasks. Based on the analysis, recommendations for task partitioning are made. These recommendations are a function of the sparseness of the linear system, its structure (i.e., randomly sparse or banded), and dimension.

17. Key Words (Suggested by Author(s)) Sparse linear algebraic systems, asynchronous iterative algorithms, computational and data transfer model	18. Distribution Statement 62 - Computer Systems 64 - Numerical Analysis Unclassified - Unlimited

19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 28	22. Price A03

N-305