General Disclaimer

One or more of the Following Statements may affect this Document

- This document has been reproduced from the best copy furnished by the organizational source. It is being released in the interest of making available as much information as possible.

- This document may contain data, which exceeds the sheet parameters. It was furnished in this condition by the organizational source and is the best copy available.

- This document may contain tone-on-tone or color graphs, charts and/or pictures, which have been reproduced in black and white.

- This document is paginated as submitted by the original source.

- Portions of this document are not fully legible due to the historical nature of some of the material. However, it is the best reproduction available from the original submission.

SYSTEM BALANCE ANALYSIS FOR VECTOR COMPUTERS

John C. Knight, et al

College of William and Mary

Prepared for:

Office of Naval Research
National Aeronautics and Space Administration

May 1975

Unclassified
Security Classification

DOCUMENT CONTROL DATA - R & D AD-A009 430

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)
College of William and Mary
Department of Mathematics
Williamsburg, VA. 23185

2a. REPORT SECURITY CLASSIFICATION
Unclassified

2b. GROUP

3. REPORT TITLE
System Balance Analysis for Vector Computers

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)
Technical Report No. 7, May 1975

5. AUTHOR(S) (First name, middle initial, last name)
John C. Knight
William G. Poole, Jr.
Robert G. Voight

6. REPORT DATE
May, 1975

7a. TOTAL NO. OF PAGES
29

7b. NO. OF REFS
9

8a. CONTRACT OR GRANT NO.
N00014-73-A0374-0001

b. PROJECT NO.
NR044-459

c.

d.

9a. ORIGINATOR'S REPORT NUMBER(S)
Technical Report 7

9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)

10. DISTRIBUTION STATEMENT
Approved for public release,
distribution unlimited

11. SUPPLEMENTARY NOTES

12. SPONSORING MILITARY ACTIVITY
Mathematics Program
Office of Naval Research
Arlington, VA. 22217

13. ABSTRACT

The availability of vector processors capable of sustaining computing rates of 10^8 arithmetic results per second has raised the question of whether peripheral storage devices representing current technology can keep such processors supplied with data. By carefully examining the solution of a large banded linear system on these computers it is found that even under ideal conditions the processors will frequently be waiting for problem data.

DD FORM 1 NOV 68 1473 (PAGE 1)

S/N 0101-807-6801

Unclassified
Security Classification

System Balance Analysis for Vector Computers

John C. Knight
William G. Poole, Jr.
Robert G. Voight

Technical Report No. 7
May 1975

SYSTEM BALANCE ANALYSIS FOR VECTOR COMPUTERS

John C. Knight
Analysis and Computation Division
NASA Langley Research Center

William G. Poole, Jr.
Institute for Computer Applications
in Science & Engineering (ICASE)
and College of William and Mary

Robert G. Voigt
Institute for Computer Applications in
Science and Engineering (ICASE)

ABSTRACT

The availability of vector processors capable of sustaining computing rates of 10^8 arithmetic results per second has raised the question of whether peripheral storage devices representing current technology can keep such processors supplied with data. By carefully examining the solution of a large banded linear system on these computers it is found that even under ideal conditions the processors will frequently be waiting for problem data.

This paper is a result of work performed, in part, under NASA Grant NGR 47-102-001 while the latter two authors were in residence at ICASE, NASA Langley Research Center. The work of the second author also was partially supported by the Office of Naval Research Contract N00014-73-A-0374-0001, NR 044-459.

SYSTEM BALANCE ANALYSIS FOR VECTOR COMPUTERS

1. INTRODUCTION

The availability of vector processors such as the Control Data Corporation STAR-100 and Texas Instruments, Inc. Advanced Scientific Computer (ASC) provides the scientific community with considerably expanded computing power. These processors are able to approach sustained rates of 10^8 arithmetic results per second and this raises the question of whether available peripheral storage devices can keep such processors supplied with problem data. It is frequently true that even with a third generation scalar system, central processor efficiency is limited by the performance of the associated peripheral equipment. However, improved peripheral hardware and improved operating system performance have made this problem manageable.

A study by Lynch [1974] using a simple model of matrix multiplication as a representative problem has indicated that conventional rotating bulk storage probably cannot supply data to vector processors fast enough to keep them from being idle most of the time. In the present paper a detailed analysis of the input/output (I/O) requirements for a problem of interest to the scientific community, that of solving a banded linear system, is given for an algorithm specifically designed for vector computers. The results obtained show that even under ideal conditions, guaranteeing an adequate supply of data for the processor is difficult with present technology.

In the following section the central processor of a vector computer is described and a model for rotating bulk storage is developed. In Section 3 the particular application problem including the solution algorithm is discussed. Section 4 includes a timing analysis for the two vector computers mentioned above and Section 5 contains several implications of this study.

2. HARDWARE

2.1 Central Processor

For the purposes of this paper, it is assumed that the arithmetic section of a vector computer contains one or more segmented execution units or "pipelines" which are used for vector arithmetic. A pipeline is functionally divided into a sequence of subunits or segments, each of which performs only part of a given arithmetic operation. A vector instruction initiates the flow of data from one or two source vectors to the pipeline. Assuming that the instruction involves two source vectors, each subunit accepts two elements, performs its particular function, passes the result, and possibly the source operands, to the next subunit, and receives the next two elements from the stream of vector operands. Thus, because of simultaneous execution in the subunits, a set of identical operations may be executed very quickly even though the time for a particular single result is much larger.

The execution time associated with any vector instruction is composed of two parts. The first depends on the hardware status and is assumed here to be constant; this part is usually known as the "start up" time. The second part is variable and is a function of the length of the vectors involved. The start up time is the delay which occurs between the issue of the instruction and the appearance of the first result. Factors affecting this delay include the need to read operands from storage before arithmetic can begin, the fact that the first operands must proceed through all the segments before the first result appears and hardware housekeeping. The variable component of the execution time consists of a constant "per result" time multiplied by the number of elements in the vector.

In general vector instruction execution times may be expressed as:

$$E = S + PL$$

where E = total time, S = start-up time, P = time per result and L = vector length. Since most pipelines operate synchronously, it is convenient to express times as multiples of the machine's basic processor cycle. In this paper it will be sufficient to consider only the operations of addition (componentwise), multiplication (componentwise), transmit (move a vector within main storage), and inner product.

2.2 Peripheral Storage

In this analysis it is assumed that the bulk storage device is managed by a sophisticated satellite processor which operates independently from the central processor. In addition, in the interest of efficiency, it is further assumed that the user has complete control of the data layout on the storage device.

For the problem to be described here, and in many large scientific computing problems, the data bases in use are only referenced sequentially. This property is frequently used to improve the overall efficiency of the data transfer operations. In the model of a peripheral device which follows, the use of only sequential access allows some quantities to be treated as constants rather than as random variables. Also, the storage device being modeled is patterned after a disk although adjustment of the parameters allows the modeling of several devices. For example, setting the parameter corresponding to seek time equal to zero provides a reasonable model of a drum. Head switching time is not explicitly considered because it can be regarded as part of the rotational delay.

The following parameters are used:

D_S The maximum seek delay when reading a large sequential file. This will correspond to the time required to move a disk head between adjacent cylinders.

D_R The disk rotation time.

γ That proportion of the rotation time which actually constitutes a delay when switching tracks. Careful layout of data or judicious hardware design can cause γ to be very small.

R The peak transfer rate which can be attained during the transfer of a single block of data.

δ That proportion of the transfer rate which can be sustained during the transfer of a complete track.

M The number of bits per track.

C The number of tracks per cylinder.

Assuming that a data set consists of B bits which are to be read or written sequentially, and that the disk arm is correctly positioned as the transfer operation begins, the time T required to transfer this data to or from a single device as modeled is approximately:

$$(2.2.1) \quad T = \left(\left\lceil \frac{B}{MC} \right\rceil - 1\right) D_S \qquad \text{time required to move head between cylinders}$$

$$+ \left(\left\lceil \frac{B}{M} \right\rceil - 1\right) D_R \gamma \qquad \text{rotational delay}$$

$$+ \frac{B}{R\delta} \qquad \text{transfer time.}$$

For any given disk drive, the parameters γ and δ are extremely difficult to quantify. They are greatly affected by the efficiency of the controlling software and many other factors. For the purposes of this paper it is assumed that the entire data file will be used sequentially by the algorithm. It is further assumed that the independent processor controlling the disk has sufficient capabilities to store and reorder, if necessary, a track of information. Consequently, in principle, reading a track load of data may begin anywhere on

the track; in practice reading will begin at the next available sector. Thus the maximum rotational delay is the time for one sector to pass under the read head. The estimate of γ used here is one half of the sector size as a fraction of the track size.

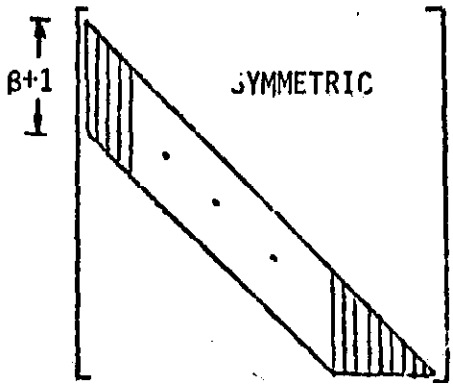
It is virtually impossible to give a reasonable estimate of δ. In this analysis it is assumed that $\delta = 1$ although in practice it will always be less.

3. THE APPLICATION PROBLEM

3.1 Choice of Problem

In choosing a particular problem to investigate, two goals were kept in mind. One was to make a selection which was representative of actual problems to be solved on vector computers and which might be common to many different applications. The second goal was to choose a problem for which the computation and I/O demands were simple enough to permit a meaningful analysis.

The problem that was chosen is one which is common to the many areas of science and engineering that require the numerical solution of partial differential equations; namely, solving a system of linear equations, $Ax = b$. The restrictions placed on the NxN matrix A are that it is real, symmetric, positive definite and banded (i.e., all nonzeros are clustered near the main diagonal). Because of the interest in the possible inefficiencies caused by I/O for problems too large for main memory, the order N of the matrix is assumed to be quite large, perhaps in the range of 5,000 to 50,000. The bandwidth, β (see Figure 1), is usually some fractional power of N; for example, β is approximately $\sqrt{N}$ for the problem of solving the Poisson equation on a square region. For a set of matrices arising in various practical problems, it was noted by Gibbs, Poole and Stockmeyer [1974] that the bandwidth averaged about $3\sqrt{N}$. These values of β, $\sqrt{N}$ and $3\sqrt{N}$, are assumed to be representative for the purpose of this study.



The Matrix A
FIGURE 1

3.2 Algorithm and Timing

For the solution of the linear system, $Ax = b$, a modification of Gaussian elimination is used. The method of solution can be described mathematically as follows:

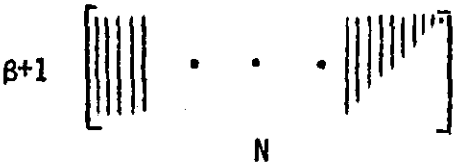
(a) factor $A = LDL^T$ where L is unit lower triangular and D is diagonal with positive diagonal elements;

(b) solve $Lz = b$;

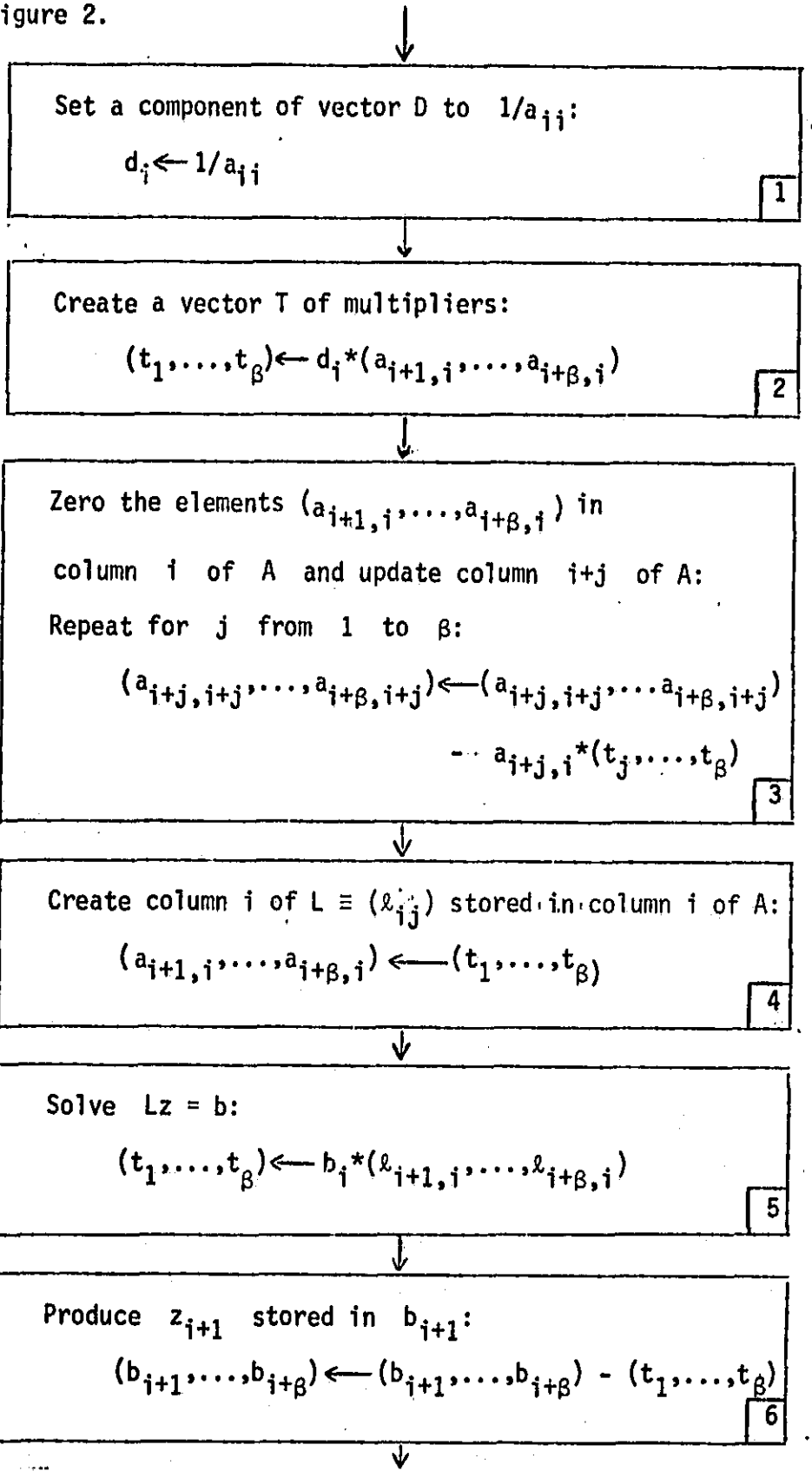
(c) solve $Dy = z$;

(d) solve $L^Tx = y$.

On a serial computer, the factors can be found by any of several algorithms, see, for example, Martin and Wilkinson [1965]. Some of these have been analyzed for vector computers by Lambiotte [1975]. One of the fastest algorithms for problems of the size of interest here is a vectorized version of symmetric Gaussian elimination. For this algorithm it is assumed that the lower half of A is stored by columns in a $(\beta+1)$ by N array and will be overwritten by L. Thus the matrix in Figure 1 will be stored as:



There are N major stages of the algorithm, one for each column of the matrix. A flowchart combining the i^{th} stages of parts (a) and (b) is given in Figure 2.



Flowchart of i^{th} Stage

FIGURE 2

Parts (c) and (d) can be implemented in a straight forward manner. For further detail the reader should see Appendix A which contains a program for the entire algorithm.

The timing formulas that follow are applicable to any computer that has the characteristics described in Section 2.1. An outline of the derivation of the timing formulas is given in Appendix B.

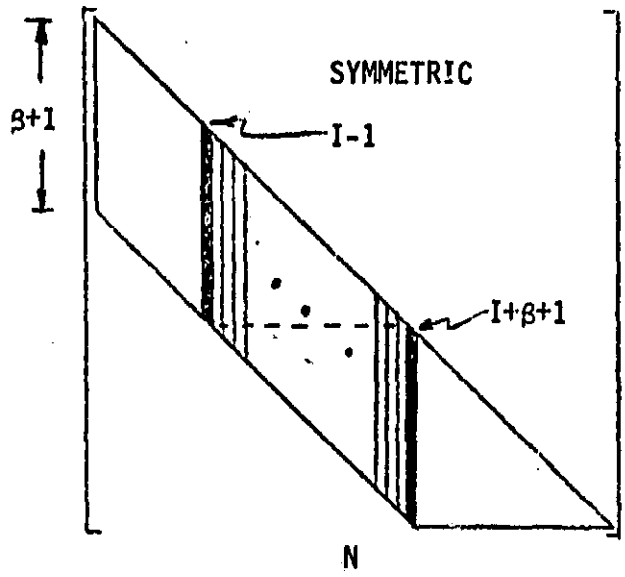
Formula (3.2.1) is for parts (a), (b) and (c) and corresponds to procedure BANSYG in Appendix A; formula (3.2.2) is for part (d) and procedure UPBSOL. They are given in units of machine cycles. The DA term refers to the time required to perform the scalar divide and for storage of - 1's in steps such as (1) and (2) of procedure BANSYG. The start up times for addition, multiplication, transmit and inner product are designated by S_A, S_M, S_T and S_I, respectively; the per result times are denoted by P_A, P_M, P_T, and P_I.

$$(3.2.1)\quad T_B(N,\beta) = .5(P_M+P_A)N\beta^2 + (S_M+S_A+2.5P_M+1.5P_A+P_T)N\beta + (DA+2S_M+S_A+S_T+P_M)N - .333(P_M+P_A)\beta^3 - .5(S_M+S_A+3P_M+2P_A+P_T)\beta^2 - .5(S_M+S_A+2.333P_M+1.333P_A+P_T)\beta - (S_M+S_A+S_T).$$

$$(3.2.2)\quad T_U(N,\beta) = P_IN\beta + (S_I+P_I)N - .5P_I\beta^2 - .5P_I\beta - (S_I+P_I).$$

3.3 Data Management

None of parts (a), (b), (c) or (d) of the algorithm requires that all of the matrix A be in main memory at one time. By examining step 3 of the flowchart in Figure 2, it is clear that for the ith stage of the factorization only columns i, i + 1,..., i + β are required. Furthermore, after step 6 of the flowchart is executed, column i is not needed again in parts (a) or (b). Thus while computation is being performed at the ith stage, the (i-1)st column may be removed from memory and the (i+β+1)st column may be brought in (see Figure 3).



Data Flow for the Factorization

FIGURE 3

In summary, the requirements on main memory for parts (a), (b) and (c) consist of an array of size $(\beta+1)^2$ for the active columns of A; at least two buffers of size $\beta + 1$: one for the column of A for which computation has been completed and one for the column for which computation has not yet begun; two vectors of length N for D and b; and one of length β for the temporary vector needed. Thus the minimum main memory required for efficient operation is

(3.3.1) $(\beta+1)^2 + 2N + 3\beta + 2$ words.

It is assumed that the data is laid out on the peripheral storage device in the following manner. Starting with the first column of the matrix, columns are stored sequentially on a track. Tracks are then filled in sequence. For a device organized as a set of concentric cylinders, after a track is filled, the next track on the same cylinder is used. Upon filling an entire cylinder, the adjacent cylinder is used.

The algorithm effectively treats a column as the logical record size, although in practice one track of data, perhaps containing more than one column, would be transferred as a single physical record. This allows maximum device utilization. It is assumed that the satellite processor which manages the

peripheral storage device will reformat the tracks of data into the columns required by the algorithm.

It should be pointed out that step 3 of the flow chart actually requires that only two columns of A and a temporary vector be in main memory at one time; however, this would increase the amount of data to be moved in one stage of the factorization to $\beta(\beta+1)$ words. The time required to move this amount of data is at least an order of magnitude larger than the corresponding computation time. Furthermore, the algorithm used here will work if only the region above the dotted line in Figure 3 is in main memory; however, this makes it impractical to keep all of the elements of a given column in contiguous storage locations, a requirement for maximum efficiency of vector operations.

Ordinarily the factorization and the lower solve are organized as separate modules because of their independence; this requires two complete passes through the data. However, in the flowchart, the lower solve is embedded in the factorization module so that only one pass is necessary.

Part (d) of the algorithm requires another pass through the matrix with the columns required in reverse order, one at a time. It should be pointed out that under the assumptions of Section 2.2, this causes only minor problems for the satellite processor since data management can take place in its own storage. Thus the reverse order requirement is met by reading the tracks in reverse order and using a buffer to sort out the data taken from a given track.

4. RESULTS FOR TYPICAL COMPUTERS

The various assumptions made on hardware in Section 2 are satisfied by the CDC STAR-100 and the TI ASC. The only assumption made about system software is that it does not interfere with the user's control of relevant hardware features as described in Sections 2 and 3. In particular there are no other active tasks competing for resources.

This section includes detailed timings of computation and I/O for the two computers. These are based on the procedures BANSYG and UPBSOL found in Appendix A. Procedure BANSYG corresponds to parts (a), (b) and (c) of the algorithm given in Section 3.2; procedure UPBSOL corresponds to part (d). For the sake of simplicity the procedure names are used throughout this section.

4.1 CDC STAR-100

The STAR-100 central processing unit (CPU) is equipped with two arithmetic pipelines although the user has no control over how they are utilized for a particular instruction. Table 1 contains timing information from Control Data Corporation [1974A] for the instructions pertinent to this study. The times are in CPU cycles and are for 64 bit operands.

$S_A = 94$	$P_A = \frac{1}{2}$
$S_M = 156$	$P_M = 1$
$S_T = 90$	$P_T = \frac{1}{2}$
$S_I = 100$	$P_I = 4$

STAR-100 Instruction Timings
(In CPU cycles of 40 nanoseconds)

TABLE 1

Based on scalar instruction timing, DA is estimated to be 106. If these times are used in expressions (3.2.1) and (3.2.2), the following timing formulas are obtained for the procedures BANSYG and UPBSOL respectively:

$$T_B(N,\beta) = .75N\beta^2 + 253.75N\beta + 603N - .5\beta^3 - 127.25\beta^2 - 126.75\beta - 340$$

$$T_U(N,\beta) = 4N\beta + 104N - 2\beta^2 - 2\beta - 104$$

Table 2 presents the time required for computation for the two procedures for various values of N and β. It should be emphasized that the times are based on manufacturer's documentation rather than on actual computer timings.

		BANSYG			UPBSOL		
		Computation	Input/Output		Computation	Input/Output	
N	β		844	819		844	819
5,000	70	4.38	4.73	0.70	0.076	4.73	0.70
5,000	210	16.93	14.07	2.09	0.185	14.07	2.09
10,000	100	13.32	13.47	2.01	0.201	13.47	2.01
10,000	300	56.69	40.17	5.99	0.514	40.17	5.99
25,000	160	60.19	53.72	8.02	0.742	53.72	8.02
25,000	480	291.82	160.50	23.96	2.006	160.50	23.96
50,000	225	190.84	150.82	22.52	2.004	150.82	22.52
50,000	675	1018.73	451.13	67.36	5.571	451.13	67.36

Computation and Input/Output Times (in seconds) for STAR-100

TABLE 2

Two disk drives are available for use with the STAR-100. They are CDC models 844 and 819, and for these drives the values of the parameters used in the model of section 2.2 are given in Table 3, as taken from Control Data Corporation [1974B].

	844	819
D_S	10 ms.	15 ms.
D_R	33 ms.	33 ms.
R	6.8×10^6 bps.	38×10^6 bps.
M	9.8×10^4 bits	52.4×10^4 bits
C	19	10

STAR-100 Disk Characteristics

TABLE 3

An 844 disk track is divided into 3 sectors; thus γ is taken to be 1/6. An 819 disk track consists of 16 sectors and γ is taken to be 1/32.

It is assumed that separate dedicated disks and associated controlling equipment are available for the input of the matrix A and for the output of the matrix L of the factorization. This permits the data to be transferred with minimal disk arm movement. The times required in seconds in terms of N and β are given by the following expressions obtained from formula (2.2.1).

$$T_{844}(N,\beta) = (\lceil(3.44\times10^{-5})N(\beta+1)\rceil - 1)(10^{-2}) + (\lceil(6.53\times10^{-4})N(\beta+1)\rceil - 1)(5.5\times10^{-3}) + (9.41\times10^{-6})N(\beta+1)$$

$$T_{819}(N,\beta) = (\lceil(1.22\times10^{-5})N(\beta+1)\rceil - 1)(1.5\times10^{-2}) + (\lceil(1.22\times10^{-4})N(\beta+1)\rceil - 1)(1.03\times10^{-3}) + (1.68\times10^{-6})N(\beta+1)$$

Table 2 contains the total I/O time for the two procedures.

In Table 2, the I/O times for the 844 disk exceed the computation times for procedure BANSYG in some cases. Furthermore, these I/O time estimates may be overly optimistic because of the ideal conditions assumed about the secondary storage environment (e.g., δ=1 in (2.2.1)). One approach to handling this probable I/O-computation imbalance is multiprogramming. However, the limited amount of main memory available on the STAR-100 (either 524K or 1048K words) may make multiprogramming impractical: for most of the cases shown in Table 2, the main memory requirement of procedure BANSYG is a significant portion of the smaller memory. For the larger memory, multiprogramming is more feasible for handling several of the smaller problems.

If 819 disk drives are used, the question of multiprogramming is not as important for procedure BANSYG since the columns of the matrix can be read or written at a rate which exceeds the computation by a reasonable margin. Careful buffering will permit the I/O operations to totally overlap the processing. However, it must be noted that separate drives may still be needed for each of the two data streams in order to avoid disk arm movement.

The main storage requirement of procedure UPBSOL consists of just one row of the matrix and one vector, $N + \beta + 1$ words. Thus this phase uses relatively little processor time or main memory but the data input requirements are the same as for procedure BANSYG. The entire matrix must be read sequentially into main storage with the rows in reverse order as the processing proceeds. Even the 819 disk unit is too slow by an order of magnitude on this procedure. If multiprogramming involving processes of this type is to be used to increase processor utilization then sufficiently many independent high performance disk drives must be available to permit each process to have dedicated drives.

4.2 TI ASC

The central processor of the ASC is available with up to four pipelines and in this analysis a four pipeline configuration is assumed. In addition the way in which the pipelines are used on a given sequence of instructions is determined by the programmer (or compiler). A single vector instruction may be executed by a single pipeline, or the operand vectors may be split and shared between the pipelines. In this analysis, it is assumed that all vector instructions are shared. With this assumption, the timing of vector instructions in terms of CPU cycles for 64 bit operands is shown in Table 4. This data is from Texas Instruments, Inc. [1973A].

$S_A = 109$	$P_A = 7/16$
$S_M = 110$	$P_M = 3/4$
$S_T = 109$	$P_T = 7/16$
$S_I = 120$	$P_I = 1$

ASC Instruction Timings
(In CPU cycles of 60 nanoseconds)

TABLE 4

Based on scalar instruction timing, DA is estimated to be 27. If these times are used in expressions (3.2.1) and (3.2.2), then the following timing formulas are obtained for the procedures BANSYG and UPBSOL respectively.

$$T_B(N,\beta) = .594N\beta^2 + 221.969N\beta + 465.75N - .396\beta^3 - 111.281\beta^2 - 110.885\beta - 328$$

$$T_U(N,\beta) = N\beta + 121N - .5\beta^2 - .5\beta - 121$$

Table 5 presents the time required for computation for the two procedures for various values of N and β. Again, the times are not based on actual computer timings. It should be noted that the same algorithm is used for the ASC as for the STAR. Because the relative speeds of instructions may vary from computer to computer, the algorithm used here may not be the fastest for the ASC. Consequently, no conclusions regarding the relative speeds of the two computers should be drawn from the data of Tables 2 and 5.

		B A N S Y G			U P B S O L		
		Computation	Input/Output		Computation	Input/ Output	
N	β		PAD	HPT		PAD	HPT
5,000	70	5.63	4.73	1.52	0.057	4.73	1.52
5,000	210	21.46	14.07	4.51	0.098	14.07	4.51
10,000	100	17.07	13.47	4.32	0.132	13.47	4.32
10,000	300	71.05	40.17	12.87	0.250	40.17	12.87
25,000	160	76.50	53.72	17.21	0.421	53.72	17.21
25,000	480	361.55	160.50	51.40	0.895	160.50	51.40
50,000	225	240.79	150.82	48.31	1.036	150.82	48.31
50,000	675	1252.12	451.13	144.49	2.374	451.13	144.49

Computation and Input/Output times (in seconds) for ASC
TABLE 5

The two disk drives available with the ASC are the Head Per Track Disk (HPT) and the Positioning Arm Disk (PAD). They have the following parameter values (see Texas Instruments, Inc. [1973B]):

	PAD	HPT
D_S	10 ms.	0
D_R	33 ms.	34 ms.
R	6.8×10^6 bps.	15×10^6 bps.
M	9.8×10^4 bits	5.2×10^5 bits
C	19	NA

ASC Disk Characteristics

TABLE 6

Each track of the PAD disk is divided into 3 sectors; thus γ is taken to be 1/6. An HPT disk track is divided into 256 sectors and γ is taken to be 1/512. Note that D_S is zero for the HPT disk so that the first term of expression (2.2.1) vanishes.

If it is assumed that the data is transferred in a continuous stream with two independent dedicated disk units, then the time required in terms of N and β is given by the following expressions for the PAD and HPT disks respectively.

$$T_{PAD}(N,\beta) = (\lceil (3.44\times10^{-5})N(\beta+1) \rceil - 1)(10^{-2}) + (\lceil (6.53\times10^{-4})N(\beta+1) \rceil - 1)(5.5\times10^{-3}) + (9.41\times10^{-6})N(\beta+1)$$

$$T_{HPT}(N,\beta) = (\lceil (1.23\times10^{-4})N(\beta+1) \rceil - 1)(6.64\times10^{-5}) + (4.27\times10^{-6})N(\beta+1)$$

Table 5 contains the total I/O times for the two procedures.

The overall results are similar to those obtained for the STAR-100. A major difference however is the availability of much larger main storage sizes for the ASC. It can be delivered with up to 8 million 64 bit words which exceeds

the maximum STAR-100 memory by a factor of 8. This has two important implications: first, multiprogramming of large scale scientific programs is much more feasible; and secondly, for the smaller values of N and β, the entire factored matrix can be held in main storage. The latter alternative eliminates the need for the output stream for procedure BANSYG and the input stream for procedure UPBSOL. Because it is procedure UPBSOL which is severely limited by its input requirements, if the entire matrix is available in storage that phase can proceed uninterrupted.

5. CONCLUSIONS

Achieving adequate input/output rates is a significant problem with high performance vector computers. The analysis presented above indicates that currently available disk units are able to provide data at a sufficient rate only in cases when a sizable amount of computation is performed with each item of data. For instance, in procedure BANSYG $O(\beta)$ computations are performed with each element of the matrix, whereas there are only 2 computations per element in procedure UPBSOL. In the general mix of jobs at a scientific computing facility, one might encounter many problems whose order of computation is similar to that of UPBSOL.

An approach to handling the I/O imbalance for third generation computers is multiprogramming. It was indicated in Section 4 that multiprogramming the STAR-100 is not always practical for the problems considered here, whereas it may be fruitful for these problems on the ASC because of the availability of a much larger main memory.

Another possible solution is the use of multiple disk drives for each data stream. Since only sequential files need be considered here, the necessary synchronization is possible in principle. A longer term solution may be offered by memories using the newer technologies such as magnetic bubbles.

Perhaps an even greater problem than supplying the central processor with data is that of supplying the entire system. Once the computations have been completed for sets of data residing on disks, new data sets must be transferred

to the disk system from other peripherals. For example, ten of the above problems with $N = 10{,}000$ and $\beta = 300$ will fill an 819 disk, yet all computation will be completed in approximately ten minutes. Loading new data sets onto an 819 will take considerably longer than this. Very little attention seems to have been given to this question.

Finally, since there is an excess of computing power and a lack of I/O capabilities, it would behoove the designers of numerical algorithms to consider techniques which reduce I/O requirements at the, perhaps considerable, expense of extra computation. For example, by examining Table 2, one sees that an I/O decrease of 50% and a computation increase of fivefold would halve the total time required by procedure UPBSOL on the STAR-100. Note that the I/O time requirements were reduced by 1/3 simply by organizing the factorization and the lower solve into one module, thus eliminating the need for a third pass through the matrix.

6. ACKNOWLEDGEMENT

The authors gratefully acknowledge Dan Siffert and John Schier of Texas Instruments, Inc. for some valuable details on the ASC hardware.

REFERENCES

Basili, V. R. and Knight, J. C. [1975]. A Language Design for Vector Machines, SIGPLAN Notices, 10, 3, 39-43.

Control Data Corporation [1974A]. CDC STAR-100 Instruction Execution Times, Revision 2, Arden Hills, Minn.

Control Data Corporation [1974B]. Control Data STAR Peripheral Stations, Revision B, Arden Hills, Minn.

Gibbs, N. E., Poole, W. G., and Stockmeyer, P. K. [1974]. An Algorithm for Reducing the Bandwidth and Profile of a Sparse Matrix, ICASE Report, July 22.

Lambiotte, J. J. [1975]. The Solution of Linear Systems of Equations on a Vector Computer, Ph.D. Thesis, University of Virginia.

Lynch, W. C. [1974]. How to Stuff an Array Processor, Third Texas Conference on Computing Systems, Nov. 7-8. Proceedings published by IEEE.

Martin, R. S. and Wilkinson, J. H. [1965]. Symmetric Decomposition of Positive Definite Band Matrices, Numer. Math. 7, 355-361.

Texas Instruments, Inc. [1973A]. The ASC System Central Processor, Austin, Texas.

Texas Instruments, Inc. [1973B]. Description of the ASC System Hardware, Austin, Texas.

APPENDIX A. PROGRAM FOR ALGORITHM

The algorithm is presented in a new programming language called SL/1 which is being developed explicitly for the CDC STAR-100. The details of SL/1 may be found in Basili and Knight [1975]; however, pertinent points are included here so that the reader may follow the algorithm.

All declarations in SL/1 are explicit and are similar to those in ALGOL. Thus

```
REAL VECTOR [2..BETA+1]  T;
```

is the declaration of a vector named T with BETA real components subscripted from 2 to BETA + 1. The lower subscript bound may be omitted in which case it defaults to one.

The other declaration of interest in SL/1 is used to describe an array of vectors:

```
REAL VECTOR [BETA+1] ARRAY (N) A;
```

This declaration describes a data structure named A consisting of N vectors of real numbers each of length BETA + 1. SL/1 also permits the user to reference subvectors and subarrays; thus, for example

```
A(I) [2..N-I+1]
```

refers to elements 2 through N - I + 1 of the I^{th} vector of array A.

The only nonstandard arithmetic operator of SL/1 used here is the inner product. Thus the statement:

```
B[I] := -U(I)[1..N-I+1] .DOT. B[I..N]
```

stores the inner product of the negative of the first N - I + 1 components of the I^{th} vector of the array U with the I through N^{th} elements of the vector B in the I^{th} location of vector B.

The program is divided into two procedures: BANSYG implements parts (a), (b), and (c) of Section 3.2 and UPBSOL corresponds to part (d). It should be pointed out that both procedures consist almost entirely of vector operations, and are thus extremely well suited for a vector processor.

```
PROCEDURE BANSYG;
   /*THIS PROCEDURE PERFORMS AN  LDL^T  FACTORIZATION OF THE GIVEN SYMMETRIC,
   POSITIVE DEFINITE, N BY N MATRIX A WITH (SEMI) BANDWIDTH BETA.  IT ALSO
   SOLVES  LZ = B  AND  DY =, Z.  THUS TO COMPLETE THE SOLUTION OF  AX = B,
   ONE MUST SOLVE  L^TX = Y.
   THE LOWER PART OF  A  IS STORED BY COLUMN, THE LOWER TRIANGLE, L, IS
   COMPUTED A COLUMN AT A TIME AND IS STORED OVER  A.  THE VECTOR  D
   CONTAINS THE INVERSES OF THE DIAGONAL ELEMENTS OF THE MATRIX D.  UPON
   EXIT, THE VECTOR  B  CONTAINS THE ELEMENTS OF  Y, OVERWRITING THE RIGHT
   HAND SIDE OF THE ORIGINAL LINEAR SYSTEM.*/
   REAL VECTOR [BETA+1] ARRAY (N)              A;
   REAL VECTOR [N]                          B,D;
   REAL VECTOR [2..BETA+1]                     T;
   INTEGER                           BETA,I,J,N;

   /*FIRST PERFORM THE ELIMINATION FOR COLUMNS 1,2,···,N-BETA-1 */

   FOR  I  FROM  1  TO  N - BETA - 1  DO
(1)    D[I] := 1.0/A(I)[1];
(2)    A(I)[1] :=  -1.0;                    /* REDEFINE DIAGONAL ELEMENTS
                                               FOR USE IN PROCEDURE UPBSOL */
(3)    T := D[I]*A(I)[2..BETA+1];           /* CREATE THE MULTIPLIERS */
       FOR  J  FROM  1  TO BETA DO          /* MODIFY RELEVANT ELEMENTS */
(4)      A(I+J)[1..BETA-J+1] := A(I+J)[1..BETA-J+1]
                  - T[J+1..BETA+1]*A(I)[J+1];

       ENDF  J;
(5)    A(I)[2..BETA+1] := T;
(6)    T := B[I]*A(I)[2..BETA+1];           /* NOW SOLVE  LZ = B  FOR Z[I+1] */
```

```
(7)     B[I+1..I+BETA] := B[I+1..I+BETA] - T;     /* Z[I+1] IS IN B[I+1] */
      ENDF   I;

      /* NEXT PERFORM THE ELIMINATION OF THE DENSE TRIANGULAR BETA+1 SYSTEM
      IN THE LOWER RIGHT CORNER */

      FOR  I  FROM  N - BETA  TO  N - 1  DO
(8)     D[I] := 1.0/A(I)[1];
(9)     A(I)[1] := -1.0;
(10)    T[2..N-I+1] := D[I]*A(I)[2..N-I+1];
        FOR J FROM I + 1 to  N  DO
(11)       A(J)[1..N-J+1] := A(J)[1..N-J+1]
                           - T[J-I+1..N-I+1]*A(I)[J-I+1];
        ENDF   J;

(12)    A(I)[2..N-I+1] := T[2..N-I+1];
(13)    T[2..N-I+1] := B[I]*A(I)[2..N-I+1];
(14)    B[I+1..N] := B[I+1..N] - T[2..N-I+1];
      ENDF    I;
(15) D[N] := 1.0/A(N)[1];
(16) A(N)[1] := -1.0;

      /* FINALLY SOLVE  DY = Z */

(17) B := D*B;
ENDP BANSYG;
```

```
PROCEDURE UPBSOL;
   /* THIS PROCEDURE SOLVES THE UNIT UPPER TRIANGULAR SYSTEM  UX=B.
   STORAGE IS SIMILAR TO BANSYG.  THE RESULT, X, IS OVERWRITTEN ON  B. */
   REAL VECTOR [BETA+1] ARRAY (N)              U;
   REAL VECTOR [N]                             B;
   INTEGER                                BETA,I,N;

   /* FIRST SOLVE DENSE LOWER RIGHT CORNER TRIANGLE */
   FOR  I  FROM  N - 1  TO  N - BETA + 1  DO
(1)    B[I] := -U(I)[1..N-I+1] .DOT. B[I..N];      /* U(I)[1] CONTAINS -1.0 */
   ENDF   I;
   /* NEXT SOLVE REMAINING BAND */
   FOR  I  FROM N - BETA TO  1  DO
(2)    B[I] := -U(I) .DOT. B[I..BETA+I];
   ENDF    I;
ENDP   UPBSOL;
```

APPENDIX B. TIMINGS OF PROCEDURES

This appendix contains an outline of the derivation of the timing formulas (3.2.1) and (3.2.2). T_i represents the timing of statement i in procedure BANSYG.

B.1 Timing of BANSYG

$$T_1 + T_2 = DA$$

$$T_3 = S_M + P_M\beta$$

$$T_4 = S_M + P_M(\beta-j+1) + S_A + P_A(\beta-j+1)$$

$$T_5 = S_T + P_T\beta$$

$$T_6 = S_M + P_M\beta$$

$$T_7 = S_A + P_A\beta$$

$$T_8 + T_9 = DA$$

$$T_{10} = S_M + P_M(N-i)$$

$$T_{11} = S_M + P_M(N-j+1) + S_A + P_A(N-j+1)$$

$$T_{12} = S_T + P_T(N-i)$$

$$T_{13} = S_M + P_M(N-i)$$

$$T_{14} = S_A + P_A(N-i)$$

$$T_{15} + T_{16} = DA$$

$$T_{17} = S_M + P_M N$$

Thus the total computation time required by BANSYG is

$$T_B(N,\beta) = \left[\sum_{i=1}^{N-\beta-1} T_1 + T_2 + T_3 + \left(\sum_{j=1}^{\beta} T_4\right) + T_5 + T_6 + T_7\right]$$

$$+\left[\sum_{i=N-\beta}^{N-1} T_8 + T_9 + T_{10} + \left(\sum_{j=i+1}^{N} T_{11}\right) + T_{12} + T_{13} + T_{14}\right]$$

$$+ T_{15} + T_{16} + T_{17}.$$

After considerable manipulation, it can be shown that

$$T_B(N,\beta) = .5(P_M+P_A)N\beta^2 + (S_M+S_A+2.5P_M+1.5P_A+P_T)N\beta$$

$$+ (DA+2S_M+S_A+S_T+P_M)N - .333(P_M+P_A)\beta^3$$

$$- .5(S_M+S_A+3P_M+2P_A+P_T)\beta^2$$

$$- .5(S_M+S_A+2.333P_M+1.333P_A+P_T)\beta - (S_M+S_A+S_T).$$

B.2 Timing of UPBSOL

$$T_U(N,\beta) = \left[\sum_{i=N-1}^{N-\beta+1} S_I + P_I(N-i+1)\right] + \left[\sum_{i=N-\beta}^{1} S_I + P_I\,(\beta+1)\right]$$

$$= P_I N\beta + (S_I+P_I)N - .5P_I\beta^2 = .5P_I\beta - (S_I+P_I).$$