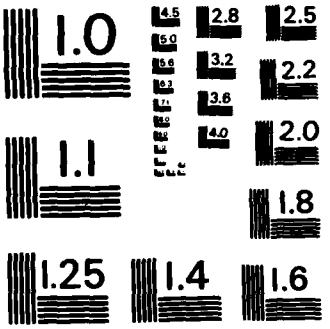
AD-A160 240 PARALLEL MATRIX COMPUTATIONS(U) MARYLAND UNIV COLLEGE PARK DEPT OF COMPUTER SCIENCE G W STEWART ET AL. APR 85 AFOSR-TR-85-0820 AFOSR-82-0078 1/1

UNCLASSIFIED F/G 9/2 NL

END

FILMED

DTIC



UNCLASS

SECURITY CLASSIFIC

NTATION PAGE

1a. REPORT SECURIT
UNCLASSIFIED

AD-A160 240

1b. RESTRICTIVE MARKINGS

2a. SECURITY CLASSIFICATION AUTHORITY

3. DISTRIBUTION/AVAILABILITY OF REPORT
Approved for public release; distribution unlimited.

2b. DECLASSIFICATION/DOWNGRADING SCHEDULE

4. PERFORMING ORGANIZATION REPORT NUMBER(S)

5. MONITORING ORGANIZATION REPORT NUMBER(S)
AF[illegible] -85-0820

6a. NAME OF PERFORMING ORGANIZATION	6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION
University of Maryland		Air Force Office of Scientific Research

6c. ADDRESS (City, State and ZIP Code)	7b. ADDRESS (City, State and ZIP Code)
College Park, Maryland 20742	Directorate of Mathematical & Information Sciences, Bolling AFB DC 20332-6448

8a. NAME OF FUNDING/SPONSORING ORGANIZATION	8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
AFOSR	NM	AFOSR-82-0078

8c. ADDRESS (City, State and ZIP Code)	10. SOURCE OF FUNDING NOS.			
	PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.
Bolling AFB DC 20332-6448	61102F	2304		

11. TITLE (Include Security Classification)
PARALLEL MATRIX COMPUTATIONS

12. PERSONAL AUTHOR(S)
G.W.Stewart, Dianne P. O'Leary

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
Interim	FROM 1984 TO April 1985	April 85	11

16. SUPPLEMENTARY NOTATION

17 COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB GR.	

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

This project concerns the design and analysis of algorithms to be run in a processor-rich environment. We focus primarily on algorithms that require no global control and that can be run on systems with only local connections among processors. We investigate the properties of these algorithms both theoretically and experimentally. The experimental work is done on the ZMOB, a working parallel computer operated by the Laboratory for Parallel Computation of the Computer Science Department at the University of Maryland.

To give our work direction, we have ~~focused~~ on two areas:

1) Dense problems from numerical linear algebra;

2) The iterative and direct solution of sparse linear systems.

DTIC FILE COPY

1 5 1985

A

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE NUMBER (Include Area Code)	22c. OFFICE SYMBOL
John Thomas	(202) 767-5026	NM

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

19. ABSTRACT cont.

We discuss in this summary the ZMOB hardware and the research projects we have pursued under this grant support.

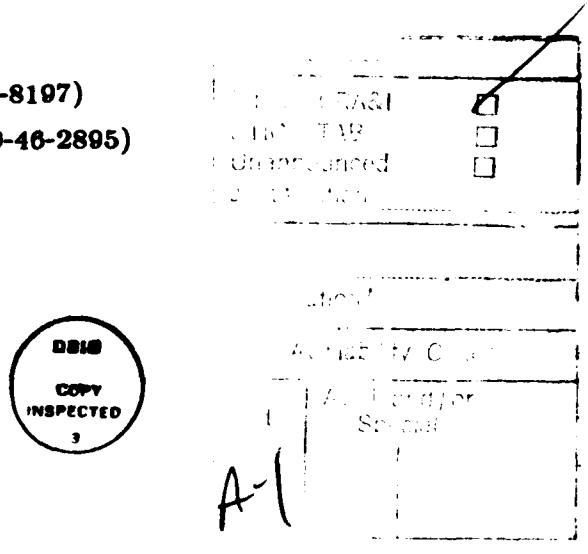
AFOSR-TR- 85-0820

Technical Summary Report
AFOSR 82-0078
Parallel Matrix Computations
April, 1984 - April, 1985

Supported by
Air Force Office of Scientific Research
Numerical Mathematics
Bolling Air Force Base, D.C. 20332

Research conducted at
Department of Computer Science
University of Maryland
College Park, MD 20742
(301) 454-2001

Principal Investigators
Professor G. W. Stewart (410-68-8197)
Assoc. Prof. Dianne P. O'Leary (359-46-2895)



A-1

1. INTRODUCTION

This project concerns the design and analysis of algorithms to be run in a processor-rich environment. We focus primarily on algorithms that require no global control and that can be run on systems with only local connections among processors. We investigate the properties of these algorithms both theoretically and experimentally. The experimental work is done on the ZMOB, a working parallel computer operated by the Laboratory for Parallel Computation of the Computer Science Department at the University of Maryland.

To give our work direction, we have focused on two areas:

1. Dense problems from numerical linear algebra,
2. The iterative and direct solution of sparse linear systems.

We discuss in this summary the ZMOB hardware and the research projects that we have pursued under this grant support.

2. The ZMOB Computer

This is a configuration of Z80 processors, which are connected by a slotted ring. Here we summarize the features important to our project.

1. The basic unit is a Z80 processor board, called a moblet, with 64K bytes of RAM, 2K bytes of ROM, an Intel 8232 floating point processor, a serial port, and, in some cases, a parallel port.
2. Although moblets are connected by a ring, the ring moves so fast that any two processors can communicate as quickly as they can move information on and off the ring. Moreover, the ring has an output-bearing slot for each moblet, which means that two moblets can communicate without blocking the communication of any other moblets. Thus, the ZMOB looks like a completely connected network of processors.
3. Messages can be sent under a number of protocols, which include a broadcast mode, a pattern matching mode, and sending to a specific moblet.
4. The ZMOB has a control moblet which can broadcast nonmaskable control interrupts.

5. A small ROM monitor supports communication activities such as loading the processors.

We have worked on a ZMOB consisting of 64 processors. This configuration will be extended to at least 128 processors, and perhaps 256. An advantage of the ZMOB architecture is that small ZMOBs can be split off for debugging purposes.

It is important to be precise about how we use the ZMOB in our research. What we do *not* do is to investigate algorithms for the ZMOB itself. Instead we use the fact that the ZMOB appears to be a completely connected network to simulate various locally connected networks of processors. Thus we can investigate, in a realistic setting, the effects on our algorithms of various processor interconnections.

3. Summary of Work

Our activities may be conveniently divided into four categories: algorithms, software development, theoretical analysis, and experimental analysis. Since experiments on the ZMOB have been preliminary in nature, we discuss only the first three in detail. For details, consult the annotated list of references in Appendix A.

3.1 Algorithms

We have based most of our work in this area on the notion of a *data-flow algorithm*. The computations in a data-flow algorithm are done by independent *computational nodes,* which cycle between requesting data from certain nodes, computing, and sending data to certain other nodes. More precisely, the nodes lie at the vertices of a directed graph whose arcs represent lines of communication. Each time a node sends data to another node, the data is placed in a queue on the arc between the two nodes. When a node has requested data from other nodes, it is blocked from further execution until the data it has requested arrives at the appropriate input queues. An algorithm organized in this manner is called a data-flow algorithm because the times at which nodes can compute is controlled by the flow of data between nodes.

Data-flow algorithms are well suited for implementation on Multiple-Instruction/Multiple-Data networks of processors. Each node in a computational network is regarded as a process residing on a fixed member of a network of processors. We allow more than one node on a processor. Since many nodes will be performing essentially the same functions, we allow nodes which share a processor also to share pieces of reentrant code, which we call *node programs*. Each processor has a resident operating system to receive and transmit messages from other processors and to awaken nodes

when their data has arrived. We will discuss this operating system in greater detail later.

Data-flow algorithms have a number of advantages.

1. The approach eliminates the need for global synchronization.
2. Parallel matrix algorithms, including all algorithms for systolic arrays, have data-flow implementations.
3. Data-flow algorithms can be coded in a high-level sequential programming language, augmented by two communication primitives for sending and receiving data.
4. Data-flow computations can be supported by a very simple operating system.
5. The approach allows the graceful handling of missized problems, since several nodes can be mapped onto one processor.
6. By mapping all nodes in a data-flow algorithm onto a single processor, one can debug parallel algorithms on an ordinary sequential processor.

The chief difficulty with the data-flow approach is that the behavior of the algorithms cannot be analyzed purely from the local viewpoint of the node programs. This is one reason for supplementing theory with experiment.

In addition to delineating a general approach to parallel matrix computations, we devised a number of new parallel algorithms. For dense matrices we developed parallel algorithms for the computation of the singular value decomposition, for the computation of the Schur decomposition, for the computation of congruence transformations, and for the solution of Liapunov equations. We developed iterative algorithms for the solution of large sparse systems and for the solution of nearly uncoupled Markov chains.

3.2 Software Development

A major part of our efforts was devoted to building an operating system to implement data-flow algorithms. The system consists of three parts: the node communication and control system (NCC), the front end, and the snapshotter.

NCC is the heart of our system. A copy of it resides on each processor. It is responsible for matching incoming messages with data requests from nodes on the processor. Whenever a node's requests are satisfied, NCC can awaken the node, permitting it to compute.

The front end is a loader that assigns nodes to processors and loads the appropriate node programs and data. The front end also constructs address tables that are used by NCC for interprocessor communication.

The snapshotter is our main measurement tool for evaluating algorithms and scheduling strategies. It is triggered by the control interrupt, which causes all computations to cease and control to be transfered to the snapshotter. The snapshotter then reports the status of the computation to the control processor. By repeatedly invoking the snapshotter, we can get an execution profile of our algorithms. It is an example of the flexibility of the data-flow approach that the snapshotter itself is implemented as a set of computational nodes and uses NCC to communicate with the control processor.

Since the system is adaptable to any Multiple-Instruction/Multiple-Data network of processors, we took care to code it so that the machine dependent parts are isolated in functionally defined segments of code. Thus we hope that the system will prove useful to others doing research in parallel computation, and, in fact, other research groups have expressed interest in using it. Complete documentation on the system is in preparation.

3.3 Theoretical Analysis

The analysis of parallel numerical algorithms has to be understood in two senses. In the first place there are the conventional analyses that must be done on any numerical algorithm; rounding error analyses, proofs of convergence, and determination of rates of convergence are typical examples. In the course of developing algorithms we have done a number of these. Beyond these analyses there is the problem of determining how well a parallel implementation works. This is analogous to the computation of operations counts and other performance measurements for sequential algorithms. The main part of our theoretical work is devoted to the study of this problem. We have considered three issues: determinacy, assignment, and scheduling.

The determinacy issue arises from the fact that in the specification of a data-flow algorithm, there may be no unique order of execution for the nodes. Thus it was necessary to show that whatever the order, the computation produces essentially the same results.

The issues of assignment and scheduling are closely related. When a computational network is to be mapped onto a smaller network of processors, it may happen that there are several ways of assigning the nodes to processors. The question then arises of which way is best. Once several nodes are executing on a processor, an operating system such as NCC must schedule the nodes which are ready for execution according to some fixed strategy. Again the question arises of which scheduling strategy is best. The assignment and scheduling issues are related because an optimal scheduling strategy for one

assignment may not be optimal for another.

We investigated these issues for a class of algorithms for matrix factorization, including implementations of the Cholesky algorithm, the LU decomposition, and the QR decomposition. We identified several good assignment and scheduling strategies for problems in which the number of matrix elements exceeds the number of processors, and computed upper and lower bounds on the execution times. This permits choice of a good algorithm for a particular machine, once the ratio of computation time to communication time is known.

4. Summary

Our work has resulted in a collection of parallel algorithms for matrix computations, a data-flow operating system to support experiments, and theoretical investigation into complexity and determinacy issues in parallel matrix computations.

Appendix A
Accomplishments under Grant AFOSR 82-0078

I. Technical Reports

(1) G. W. Stewart, *Computing the CS Decomposition of a Partitioned Orthonormal Matrix*, TR-1159, May, 1982.

This paper describes an algorithm for simultaneously diagonalizing by orthogonal transformation the blocks of a partitioned matrix having orthonormal columns.

(2) G. W. Stewart *A Note on Complex Division*, TR-1206, August, 1982.

An algorithm (Smith, 1962) for computing the quotient of two complex numbers is modified to make it more robust in the presence of underflows.

(3) D. P. O'Leary, *Solving Sparse Matrix Problems on Parallel Computers*, TR-1234, December, 1982.

This paper has a dual character. The first part is a survey of some issues and ideas for sparse matrix computation on parallel processing machines. In the second part, some new results are presented concerning efficient parallel iterative algorithms for solving mesh problems which arise in network problems, image processing, and discretization of partial differential equations.

(4) G. W. Stewart, *A Jacobi-like Algorithm for Computing the Schur Decomposition of a Non-Hermitian Matrix*, TR-1321, August, 1983.

This paper describes an iterative method for reducing a general matrix to upper triangular form by unitary similarity transformations. The method is similar to Jacobi's method for the symmetric eigenvalue problem in that it uses plane rotations to annihilate off-diagonal elements, and when the matrix is Hermitian it reduces to a variant of Jacobi's method. Although the method cannot compete

with the QR algorithm in serial implementation, it admits of a parallel implementation in which a double sweep of the matrix can be done in time proportional to the order of the matrix.

(5) D. P. O'Leary and G. W. Stewart, *Data-Flow Algorithms for Matrix Computations,* TR-1366, January, 1984.

In this work we develop some algorithms and tools for solving matrix problems on parallel processing computers. Operations are synchronized through data-flow alone, which makes global synchronization unnecessary and enables the algorithms to be implemented on machines with very simple operating systems and communications protocols. As examples, we present algorithms that form the main modules for solving Liaponuv matrix equations. We compare this approach to wavefront array processors and systolic arrays, and note its advantages in handling missized problems, in evaluating variations of algorithms or architectures, in moving algorithms from system to system, and in debugging parallel algorithms on sequential machines.

(6) G. W. Stewart, W. F. Stewart, D. F. McAlister, *A Two Stage Iteration for Solving Nearly Uncoupled Markov Chains,* TR-1384, 1984.

This paper presents and analyses a parallizable algorithm for solving Markov chains that arise in queuing models of loosely coupled systems.

(7) Dianne P. O'Leary and Robert E. White, *Multi-Splittings of Matrices and Parallel Solution of Linear Systems,* TR-1362, December, 1983.

We present two classes of matrix splittings and give applications to the parallel iterative solution of systems of linear equations. These splittings generalize regular splittings and P-regular splittings, resulting in algorithms which can be implemented efficiently on parallel computing systems. Convergence is established, rate of convergence is discussed, and numerical examples are given.

(8) David C. Fisher, *In Three-Dimensional Space, the Time Required to Add N Numbers is* $O(N^{1/4})$, TR-1431, August, 1984.

How quickly can the sum of N numbers be computed with sufficiently many processors? The traditional answer is $t = O(log\ N)$. However, if the processors are in R^d (usually $d \leq 3$), addition time and processor volume are bounded away from zero, and transmission speed and processor length are bounded, $t \geq O(N^{1/d+1})$.

(9) Dianne P. O'Leary, G. W. Stewart, *On the Determinacy of a Model for Parallel Computation,* TR-1456, November, 1984.

In this note we eextend a model of Karp and Miller for parallel computation. We show that the model is deterministic, in the sense under different scheduling regimes each process in the computation consumes the same input and generates the same output. Moreover, if the computation halts, the final state is independent of scheduling.

(10) Dianne P. O'Leary, *Systolic Arrays for Matrix Transpose and Other Reorderings,* TR-1481, March, 1985.

In this note, a systolic array is described for computing the transpose of an $n \times n$ matrix in time $3n-1$ using n^2 switching processors and n^2 bit buffers. A one-dimensional implementation is also described. Arrays are also given to take a matrix in by rows and put it out by diagonals, and vice versa.

(11) Dianne P. O'Leary, G. W. Stewart, *Assignment and Scheduling in Parallel Matrix Factorization,* TR-1486, April, 1984.

We consider in this paper the problem of factoring a dense $n \times n$ matrix on a network consisting of P MIMD processors when the network is smaller than the number of elements in the matrix ($P < n^2$). The specific example analyzed is a computational network that arises in computing the LU, QR, or Cholesky factorizations. We prove that if the nodes of the network are evenly distributed among processors and if computations are scheduled by a round-robin or a least-recently-executed scheduling algorithm, then optimal order of speed-up is achieved.

However, such speed-up is not necessarily achieved for other scheduling algorithms or if the computation for the nodes is inappropriately split across processors, and we give examples of these phenomena. Lower bounds on execution time for the algorithm are established

II. Technical report in preparation

(1) R. van de Geijn, D. P. O'Leary, G. W. Stewart, *A Node Communication System for Data-Flow Computation.*

III. Presentations during 1983-85

(1) D. P. O'Leary, *Solving Mesh Problems on Parallel Computers,*
Bell Laboratory, Murray Hill, N.J., January, 1983
IBM T. J. Watson Laboratory, Yorktown Heights, N.Y., January, 1983.

(2) G. W. Stewart, *A Jacobi-like Algorithm for Computing the Schur Decomposition of a Non-Hermitian Matrix* (invited), Symposium on Numerical Analysis and Computational Complex Analysis, Zurich, Switzerland, August, 1983. Also presented at North Carolina State University, September, 1983, and at University of Houston, November, 1983,

(3) G. W. Stewart, *The Structure of Nearly Uncoupled Markov Chains* (invited), International Workshop on Systems Modeling, Pisa, Italy, September, 1983.

(4) G. W. Stewart, *Data Flow Algorithms for Parallel Matrix Computations* (invited), SIAM Conference on Parallel Processing for Scientific Computing, Norfolk, VA, November, 1983.

(5) D. P. O'Leary, *Parallel Computations for Sparse Linear Systems* (minisymposium invitation), SIAM 1983 Fall Meeting, Norfolk, VA, November, 1983.

(6) D. C. Fisher, *Numerical Computations on Multiprocessors with Only Local Communications* (poster session), SIAM Conference on Parallel Processing for Scientific Computing, Norfolk, VA, November, 1983.

(7) G. W. Stewart, *Parallel Computations on the ZMOB,* Annual meeting of CER participants, University of Utah, March, 1984.

(8) D. P. O'Leary, *Data-flow Algorithms for Matrix Computations* (minisymposium invitation), ACM SIGNUM Conference on Numerical Computations and Mathematical Software for Microcomputers, Boulder, Colorado, March, 1984.

(9) D. P. O'Leary, *Solution of Matrix Problems on Parallel Computers* (invited presentation), Gatlinburg IX Meeting on Numerical Linear Algebra, Waterloo, Ontario, Canada, July, 1984. Also presented at Oak Ridge National Laboratory, September, 1984; National Bureau of Standards, Boulder, Colorado, March, 1984; Yale University, November, 1984; Cornell University, January, 1985; Courant Institute, February, 1985.

(10) G. W. Stewart, *The Data-Flow Approach to Matrix Computations,* Los Alamos Scientific Laboratory, October, 1984.

(11) G. W. Stewart, *The Impact of Computer Architecture on Statistical Computing,* (invited) SIAM/ISA/ASA Conference on Frontiers of Statistical Computing, October, 1984.

(12) G. W. Stewart, *Determinacy,* (invited) Symposium in Honor of G. Dahlquist, Stockholm, January, 1985.

(13) D. C. Fisher, *Fast Matrix Multiplication on Square and Cubic Grids of Processors,* SIAM Conference on Applied Linear Algebra, Raleigh, April, 1985.

IV. Publications

(1) G. W. Stewart, "Computing the CS Decomposition of a Partitioned Orthonormal Matrix," *Numerische Mathematik* 40 (1982) 297-306.

(2) D. P. O'Leary, "Ordering schemes for parallel processing of certain mesh problems," *SIAM Journal on Scientific and Statistical Computing* 5 (1984) 620-632.

(3) D. P. O'Leary, R. E. White, "Multi-splittings of matrices and parallel solution of linear systems," *SIAM Journal on Algebraic and Discrete Methods*, to appear.

(4) D. P. O'Leary, G. W. Stewart, "Data-flow algorithms for parallel matrix computations," *Communications of the ACM*, to appear.

(5) G. W. Stewart, "A Jacobi-like Algorithm for Computing the Schur Decomposition of a Non-Hermitian Matrix," *SIAM Journal on Scientific and Statistical Computing*, to appear.

END

FILMED

11-85

DTIC