AN ABSTRACT OF THE THESIS OF

PAULA ANN HITCHCOCK (Name) for the MASTER OF SCIENCE (Degree)

in COMPUTER SCIENCE (Major) presented on July 30, 1973 (Date)

Title: A BLOCK METHOD APPROACH TO SOLVING THE LAPLACE EQUATION ON A PARALLEL COMPUTER.

Redacted for Privacy

Abstract Approved: Joel Davis

This paper continues exploration in the area of programming for parallel computers. The appendix to the paper contains an extensive survey of the literature related to parallel computers and parallel programming techniques. The paper itself presents a new appraoch to solving the Laplace equation on a parallel computer. A new "block" method, the Accelerated Alternating Halves method, is found to be consistently better than the Accelerated Point Gauss-Seidel method or the Accelerated Line Gauss-Seidel method when used on a parallel processor computer.

A BLOCK METHOD APPROACH TO SOLVING THE LAPLACE EQUATION ON A PARALLEL COMPUTER

by

Paula Ann Hitchcock

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Completed July 30, 1973

Commencement June 1974

APPROVED:

Redacted for Privacy

Professor of Computer Science

Redacted for Privacy

Head of Department of Computer Science

Redacted for Privacy

Dean of Graduate School

Date thesis is presented July 30, 1973

Typed by Diane Glassmire for Paula Ann Hitchcock

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

A BLOCK METHOD APPROACH TO SOLVING THE LAPLACE EQUATION ON A PARALLEL COMPUTER

I. INTRODUCTION

The desire to use computers in the solution of increasingly complex problems has continually motivated the creation of machines with greater computing capabilities. The emphasis of computer technology, until approximately the last ten years, has been on shortening switching times of circuit components, increasing the speed of memory units, and minimizing component size. The achievements in these areas have been impressive — so impressive that computing systems have advanced to "... the point where efforts to increase performance [have encountered] basic physical laws as limiting influences."[1] One attempt to circumvent this apparent dead-end in computer growth has been the exploration of hardware redundancy. In particular, much attention has recently been focused on the "parallel" computer.

The defining feature of the parallel machine is, simply enough, that it can perform more than one com-

---

[1] W.L. Miranker, "A Survey of Parallelism in Numerical Analysis," Society for Industrial and Applied Mathematics Review, 13 (October, 1971), p. 524.

putation at a time. That is, it is capable of performing operations "in parallel." Experiments in parallel computing have been highly varied. In the literature, however, we find three common types of parallel computers.[2] The first, and simplest, involves parallelism at the machine instruction level. Several higher level instructions appear to be processed simultaneously, but actually different phases of the instructions are done at the same time. That is, while a "fetch" for one instruction is being done, an "execute" for a second instruction is underway, a "store" for a third instruction is being done, etc. This type of parallelism is referred to as a "look ahead" capability. The second type of parallel computer contains a single control unit which issues identical instructions to several arithmetic units. Each arithmetic unit has its own data memory. This "array processing" computer is highly suited for working with vectors and matrices. The most complex parallel computer has a control unit which can divide the instruction stream into several substreams to be executed simultaneously. Each substream of instructions is then executed by a separate processor, concurrently with all other substreams.

---

[2] Ibid., p. 525.

Several parallel computers have already been constructed. We find among these the CDC 6600, ILLIAC IV, GAMMA 60, SOLOMON, and IBM SYSTEM/360 MODEL 91. With these parallel machines ready for use, or soon to be available, attention is now turning to finding ways of using them efficiently. In the area of numerical analysis, for example, some techniques are "naturally" parallel. Linear algebra provides many such examples. However, most algorithms have been developed in the traditional sequential manner. Some of these have recently been reconstructed to take advantage of the new parallel processing capabilities. Problems which have been re-examined include matrix inversion, root finding, integration of ordinary differential equations, and polynomial evaluation. (A list of the related literature is given in the appendix to this paper.)

Very little has been published in the area of parallel-computer-oriented methods for solving partial differential equations. An article by Rosenfeld and Driscoll discusses the solution of the Dirichlet problem on the third type of parallel computer described above. Two other relevant articles concerned with solving certain partial differential

equations involved in weather prediction. One of these, the paper by Carroll and Wetherald, investigates the solution of the Laplace equation on the SOLOMON computer.

In this paper, two new "parallel" methods for solving the Laplace equation are presented. The methods are designed to be used on an array-processing computer, although they could be efficiently used on more sophisticated machines. The methods are not limited by being defined about a specific problem, since it seems likely they could be adapted for the solution of other similar problems.

II. COMMON NUMERICAL TECHNIQUES FOR SOLVING THE LAPLACE PROBLEM

2.1. Problem Definition

As stated in the Introduction, this paper examines the effect of parallel computations on the solution of a partial differential equation. The particular problem considered is the solution of the Laplace equation,

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0, \tag{2.1.1}$$

on the rectangle $D = \{(x,y) \mid 0 < x < a,\ 0 < y < b\}$. It is assumed that the function u is continuous and that its value is known on C, the boundary of D. That is, $u = f(x,y)$ for $(x,y) \in C$, where $f(x,y)$ is known.

The usual approach to finding a numerical approximation to u begins by approximating the domain D by a finite set of points.[3] Approximations to $u(x,y)$ are obtained for only those points (x,y) in this finite set. The domain actually considered, then, is defined as

[3] E. Isaacson and H.B. Keller, Analysis of Numerical Methods (New York: John Wiley and Sons, Inc., 1966), 444 ff.

$$D_{IJ} = \{(x_i, y_j) \mid x_i = ih,\ i = 1, \ldots I - 1;$$

$$y_j = jk,\ j = 1, \ldots J - 1\}.$$

Here it is assumed that $h = a/I$, $k = b/J$, where I and J are any positive integers. The boundary of the domain D_{IJ} is defined to be

$$C_{IJ} = \{(x_i, y_j) \mid i \in \{0, I\},\ 1 \leq j \leq J - 1\} \cup$$

$$\{(x_i, y_j) \mid j \in \{0, J\},\ 1 \leq i \leq I - 1\}.$$

We denote the restrictions of f and u to $D_{IJ} \cup C_{IJ}$ by F^* and U, respectively. Finally, second difference quotients are used to approximate the second derivatives of the Laplace equation. The numerical problem thus becomes the following:

$$\frac{U(x_{i+1}, y_j) + U(x_{i-1}, y_j) - 2U(x_i, y_j)}{h^2} +$$

$$\frac{U(x_i, y_{j+1}) + U(x_i, y_{j-1}) - 2U(x_i, y_j)}{k^2} = 0 \tag{2.1.2}$$

$$\text{for } (x_i, y_j) \in D_{IJ}$$

$$U(x_i, y_j) = F^*(x_i, y_j) \text{ for } (x_i, y_j) \in C_{IJ}.$$

By letting $\theta = 2/h^2 + 2/k^2$, $\theta_x = \frac{1}{h^2\theta}$, and $\theta_y = \frac{1}{k^2\theta}$, the problem can be rewritten as

$$\theta_x(U(x_{i+1}, y_j) + U(x_{i-1}, y_j)) +$$

$$\theta_y(U(x_i, y_{j+1}) + U(x_i, y_{j-1})) -$$

(2.1.3)

$$U(x_i, y_j) = 0 \quad \text{for } (x_i, y_j) \in D_{IJ}$$

$$U(x_i, y_j) = F^*(x_i, y_j) \text{ for } (x_i, y_j) \in C_{IJ}.$$

By viewing each value $U(x_i, y_j)$, for $(x_i, y_j) \in D_{IJ}$, as an unknown, 2.1.3 can be seen to be a system of $(I-1)(J-1)$ linear equations in $(I-1)(J-1)$ unknowns. The problem given in 2.1.3 thus can be transformed into a matrix problem of the form $AU = G$. Here U is defined to be the vector

$$U = \begin{bmatrix} U_1 \\ U_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ U_{(I-1)(J-1)} \end{bmatrix} = \begin{bmatrix} U(x_1, y_1) \\ U(x_2, y_1) \\ \cdot \\ \cdot \\ \cdot \\ U(x_{I-1}, y_1) \\ U(x_1, y_2) \\ \cdot \\ \cdot \\ U(x_{I-1}, y_{J-1}) \end{bmatrix}$$

The definitions of A and G are somewhat more

complicated. First we let

$$T_I = \begin{bmatrix} 0 & & & & & \\ 1 & 0 & & & & \\ & 1 & 0 & & & \\ & & 1 & 0 & & \\ & & & \ddots & \ddots & \\ & & & & 1 & 0 \end{bmatrix}$$

be an $(I-1) \times (I-1)$ matrix, and let S_I be an $(I-1) \times (I-1)$ identity matrix. The matrices S and T are each composed of $(J-1) \times (J-1)$ submatrices, with each submatrix being of dimension $(I-1) \times (I-1)$. The arrangement of the submatrices composing S and T are the following:

$$S = \begin{bmatrix} 0 & & & & & \\ S_I & 0 & & & & \\ & S_I & 0 & & & \\ & & S_I & 0 & & \\ & & & \ddots & \ddots & \\ & & & & S_I & 0 \end{bmatrix}$$

and

$$T = \begin{bmatrix} T_I & & & O \\ & T_I & & \\ & & T_I & \\ & & & \ddots \\ O & & & T_I \end{bmatrix}$$

We can now define the matrix A by

$$A = \theta_x(T + T^T) + \theta_y(S + S^T) - I_N.$$

Here, and in the rest of the paper, I_N is an $(I-1)(J-1) \times (I-1)(J-1)$ identity matrix, and the superscript "T" denotes the transpose of the given matrix. We use the conventional notation of representing the element in the i^{th} row and j^{th} column of A by a_{ij}. The matrix A, when $I = J = 5$, is shown in figure 1.

In defining the vector G, we first note that on the boundary C_{IJ} the values of U are known. This means that some of the equations in the system $AU = G$ contain only three or four unknowns. All known values in these equations are included in the vector G.

More precisely, we define

$$\overline{\overline{G}}_{i,j} = 0 \quad \text{if} \quad 1 < i < I-1,$$

$$\overline{\overline{G}}_{1,j} = -\theta_x U(x_0, y_j),$$

$$\overline{\overline{G}}_{I-1,j} = -\theta_x U(x_I, y_j),$$

$$\overline{G}_{i,j} = \overline{\overline{G}}_{i,j} \quad \text{if} \quad 1 < j < J-1,$$

$$\overline{G}_{i,1} = \overline{\overline{G}}_{i,1} - \theta_y U(x_i, y_0),$$

$$\overline{G}_{i,J-1} = \overline{\overline{G}}_{i,J-1} - \theta_y U(x_i, y_J),$$

and finally,

$$G = \begin{bmatrix} G_1 \\ G_2 \\ \vdots \\ G_{(I-1)(J-1)} \end{bmatrix} = \begin{bmatrix} \overline{G}_{1,1} \\ \overline{G}_{2,1} \\ \vdots \\ \overline{G}_{I-1,1} \\ \overline{G}_{1,2} \\ \vdots \\ \overline{G}_{I-1,J-1} \end{bmatrix}.$$

$$
\begin{array}{cccccccccccccccc}
-1 & \theta_x & 0 & 0 & \theta_y & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\theta_x & -1 & \theta_x & 0 & 0 & \theta_y & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \theta_x & -1 & \theta_x & 0 & 0 & \theta_y & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \theta_x & -1 & 0 & 0 & 0 & \theta_y & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\theta_y & 0 & 0 & 0 & -1 & \theta_x & 0 & 0 & \theta_y & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \theta_y & 0 & 0 & \theta_x & -1 & \theta_x & 0 & 0 & \theta_y & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \theta_y & 0 & 0 & \theta_x & -1 & \theta_x & 0 & 0 & \theta_y & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \theta_y & 0 & 0 & \theta_x & -1 & 0 & 0 & 0 & \theta_y & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \theta_y & 0 & 0 & 0 & -1 & \theta_x & 0 & 0 & \theta_y & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \theta_y & 0 & 0 & \theta_x & -1 & \theta_x & 0 & 0 & \theta_y & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \theta_y & 0 & 0 & \theta_x & -1 & \theta_x & 0 & 0 & \theta_y & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \theta_y & 0 & 0 & \theta_x & -1 & 0 & 0 & 0 & \theta_y \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \theta_y & 0 & 0 & 0 & -1 & \theta_x & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \theta_y & 0 & 0 & \theta_x & -1 & \theta_x & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \theta_y & 0 & 0 & \theta_x & -1 & \theta_x \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \theta_y & 0 & 0 & \theta_x & -1
\end{array}
$$

Figure 1. The Matrix A

When I = J = 5.

Before looking at specific ways of solving AU = G, it should be determined that this system does, in fact, have a unique solution. A proof can be found in Isaacson and Keller's Analysis of Numerical Methods (pp. 447-448).

2.2. Common Methods for Solving the Matrix Problem

A variety of methods are available for solving the matrix problem, AU = G, defined in section 2.1. In the following paragraphs, some commonly-used methods are briefly reviewed. New methods which seem particularly suitable for solving partial differential equations on a parallel computer are discussed in Chapter III.

The direct approaches to solving systems of linear equations often are not used to solve the difference schemes which evolve from partial differential equations such as the Laplace equation.[4] These systems tend to become very large — of the order of 2500 or more unknowns. Because of this size problem and because of the very sparse nature of these matrices, iterative methods are often used. In the following discussion, two general types of iterative methods, the point and

[4] Ibid., p. 463.

the block, are considered. The two types are distinguished by the composition of a single iterative step. Each iterative step of a point method determines an estimate for one component of the vector U. In contrast, a group or "block" of estimates is obtained in a single iterative step of a block method.

The first step in each of the point methods for solving $AU = G$ is to write A as $A = D - E - F$ where E is strictly lower triangular, F is strictly upper triangular, and D is a diagonal matrix. In the particular problem defined in section 2.1, for example, we let

$$D = -I_N, \quad E = -\theta_x S - \theta_y T, \quad \text{and} \quad F = -\theta_x S^T - \theta_y T^T.$$

The most straightforward iterative method is the Point Jacobi Iterative method. Here, $AU = G$ is simply rewritten as $DU = (E + F)U + G$. Letting $U^{(0)}$ be the initial estimate of the solution vector, we can write this method as

$$DU^{(m+1)} = (E+F)U^{(m)} + G, \text{ or}$$
$$U^{(m+1)} = D^{-1}(E+F)U^{(m)} + D^{-1}G. \tag{2.2.1}$$

The component equations have the form

$$U_i^{(m+1)} = \frac{1}{a_{ii}} \left(- \sum_{\substack{j=1 \\ j \neq i}}^{(I-1)(J-1)} a_{ij} U_j^{(m)} \right) + \frac{G_i}{a_{ii}} .$$

The Point Gauss-Seidel method is closely related to the Point Jacobi, but it uses estimates of the values U_i, $1 \leq i \leq (I-1)(J-1)$, as soon as they are calculated. In calculating $U_i^{(m+1)}$, for example, the values of $U_1^{(m+1)}, U_2^{(m+1)}, \ldots U_{i-1}^{(m+1)}$ are used instead of $U_1^{(m)}, U_2^{(m)}, \ldots U_{i-1}^{(m)}$. This iterative method is given by the following:

$$(D-E)U^{(m+1)} = FU^{(m)} + G, \quad \text{or}$$
$$U^{(m+1)} = (D-E)^{-1}FU^{(m)} + (D-E)^{-1}G. \tag{2.2.2}$$

The individual equations in this system have the form

$$U_i^{(m+1)} = - \sum_{j=1}^{i-1} \frac{a_{ij}}{a_{ii}} U_j^{(m+1)} - \sum_{j=i+1}^{(I-1)(J-1)} \frac{a_{ij}}{a_{ii}} U_j^{(m)} + \frac{G_i}{a_{ii}} .$$

To obtain the third point method, the Accelerated Point Gauss-Seidel method, we define an intermediate vector $U^{(m+\frac{1}{2})}$ to be the following:

$$U_i^{(m+\frac{1}{2})} = -\sum_{j=1}^{i-1} \frac{a_{ij}}{a_{ii}} U_j^{(m+1)} - \sum_{j=i+1}^{(I-1)(J-1)} \frac{a_{ij}}{a_{ii}} U_j^{(m)} + \frac{G_i}{a_{ii}},$$

for $1 \leq i \leq (I-1)(J-1)$. The $(m+1)^{st}$ estimate of U_i is then obtained from

$$U_i^{(m+1)} = U_i^{(m)} + \omega \{U_i^{(m+\frac{1}{2})} - U_i^{(m)}\},$$

where we have introduced the real acceleration factor ω. The matrix form of this method is

$$(2.2.3) \quad (D-\omega E)U^{(m+1)} = \{(1-\omega)D + \omega F\} U^{(m)} + \omega G.$$

For the block methods, the matrix A is arranged as a system of submatrices,[5] as is shown in the following:

$$(2.2.4) \quad A = \begin{bmatrix} A_{1,1} & A_{1,2} \cdots A_{1,M} \\ A_{2,1} & \\ \vdots & \\ \vdots & \\ A_{M,1} & \cdots A_{M,M} \end{bmatrix}$$

[5] Richard S. Varga, Matrix Iterative Analysis (Englewood Cliffs, New Jersey: Prentice Hall, Inc., 1962). pp. 98-106.

with $1 \leq M \leq (I-1)(J-1)$, and with square diagonal submatrices A_{ii}, $1 \leq i \leq M$. Further specification of M or of the dimensions of the diagonal submatrices depends on the particular problem. In solving the Laplace equation, the dimension of A_{ii}, $1 \leq i \leq M$, is often chosen to be the number of unknowns in a row of the rectangular grid, D_{IJ}. When this is the case, we shall refer to the method as a "line" method rather than as a "block" method. We further define

$$\bar{D} = \begin{bmatrix} A_{1,1} & & & \\ & A_{2,2} & & \\ & & \ddots & \\ & & & A_{M,M} \end{bmatrix},$$

$$\bar{E} = -\begin{bmatrix} 0 & & & \\ A_{2,1} & 0 & & \\ A_{3,1} & A_{3,2} & 0 & \\ \vdots & \vdots & \ddots & \\ A_{M,1} & A_{M,2} & \cdots & 0 \end{bmatrix},$$

and

$$\bar{F} = -\begin{bmatrix} 0 & A_{1,2} & A_{1,3} & \cdots & A_{1,M} \\ & 0 & A_{2,3} & & \\ & & \ddots & \ddots & \\ & & & & A_{M-1,M} \\ & & & & 0 \end{bmatrix}.$$

Using these definitions, the Block Jacobi Iterative method is written as

$$\bar{D}U^{(m+1)} = (\bar{E}+\bar{F})U^{(m)} + G, \text{ or}$$

$$U^{(m+1)} = \bar{D}^{-1}(\bar{E}+\bar{F})U^{(m)} + \bar{D}^{-1}G. \tag{2.2.5}$$

The Accelerated Block Gauss-Seidel method is given by

$$(\bar{D}-\omega\bar{E})U^{(m+1)} = (\omega\bar{F} + (1-\omega)\bar{D})U^{(m)} = \omega G. \tag{2.2.6}$$

As the formulas 2.2.5 and 2.2.6 suggest, these block methods determine a group of the components of $U^{(m+1)}$ simultaneously. More exactly, a single iterative step produces estimates for more than one component of U. Each such group of estimates corresponds to a submatrix A_{ii}, $1 \leq i \leq M$, and the size of

the group is the dimension of the corresponding submatrix. Determination of a group of estimates is actually the solution of a linear system with coefficient matrix closely related to the corresponding submatrix. For example, if we let $U_{[i]}$ represent the i^{th} group of components of the vector U and apply formula 2.2.5, then the i^{th} group of estimates are obtained by solving the following system:

$$A_{i,i}U_{[i]}^{(m+1)} = -\sum_{\substack{j=1 \\ j\neq i}}^{M} A_{i,j}U_{[j]}^{(m)} + G_{[i]}.$$

Thus, in calculating $U^{(m+1)}$ from $U^{(m)}$, M matrix problems must be solved. Since the dimensions of these linear systems are relatively small, the systems often are solved explicitly using a factoring technique[6] or some other computation-saving method.[7]

2.3 Evaluation of Convergence of Methods for Solving the Matrix Problem

One numerical method is considered to be better, or more efficient, than another numerical method if it requires fewer operation times and yields at least the

[6] Isaacson and Keller, op. cit., p. 52.

[7] Ibid., pp. 55-56.

same degree of accuracy as the second method. In order to compare any two methods described in the previous section, then, we must determine the number of multiplications and additions required to calculate $U^{(m+1)}$ from $U^{(m)}$. We also need to know the rate of convergence for each method — approximately how many iterations are required to achieve a desired accuracy. A technique[8] commonly used for comparing convergence properties of methods is outlined in the following paragraphs.

All of the methods described in section 2.2 are of the general form

$$U^{(m+1)} = BU^{(m)} + H. \tag{2.3.1}$$

We refer to B as the iterative matrix of the method. We define the error vector $e^{(m)}$ by $e^{(m)} = U^{(m)} - U$, where U is the unique solution to $(I-B)U = H$. Then, by subtracting $U = BU + H$ from 2.3.1, we get

$$e^{(m)} = Be^{(m-1)} = \ldots = B^m e^{(0)}.$$

By introducing norms into this string of equalities, we obtain

$$\|e^{(m)}\| \leq \|B^m\| \|e^{(0)}\| \tag{2.3.2}$$

---

[8] Varga, op. cit., pp. 61-67.

Here the norms could be any consistent norms, but we choose the vector norm to be the Euclidean norm, defined to be

$$\|x\| = \left(\sum_{i=1}^{n} |x_i|^2 \right)^{\frac{1}{2}}$$

for $x = (x_1, x_2, \ldots x_n)$. The matrix norm is chosen to be the spectral norm, given by

$$\|A\| = \sup_{\|x\| \neq 0} \frac{\|Ax\|}{\|x\|}$$

for any matrix A.

To determine how many operations are required to reduce the norm of the initial error by a factor of p, we must find the smallest m such that

$$\frac{\|e^{(m)}\|}{\|e^{(0)}\|} \leq p.$$

This inequality will hold for a particular value of m if

$$\|B^m\| \leq p.$$

If the eigenvalues of the $n \times n$ matrix B are λ_i, $1 \leq i \leq n$, then we define

$$\rho(B) = \max_{1 \leq i \leq n} |\lambda_i|.$$

The method having iterative matrix B converges if

and only if $\rho(B) < 1$; we assume from now on, then, that $\rho(B) < 1$. From Isaacson and Keller's discussion of rate of convergence,[9] we find that

$$\|B^m\| \approx (\rho(B))^m$$

for large values of m. If B is real and symmetric, then

$$\|B^m\| = (\rho(B))^m.$$

Thus, instead of working with $\|B^m\|$, we look for the smallest m such that

$$(\rho(B))^m \leq p,$$

$$m \ln \rho(B) \leq \ln p, \text{ and} \tag{2.3.3}$$

$$m \geq \frac{-\ln p}{|\ln \rho(B)|}$$

We define the rate of convergence to be

$$R_\infty(B) = |\ln \rho(B)|.$$

Since larger rates of convergence allow smaller values of m to satisfy 2.3.3, we consider a method to be superior to a second method if the rate of convergence of the first is greater than the rate of convergence of the second. We further note from the definition of rate of convergence that the comparison problem actually reduces to the problem of finding the

---

[9] Isaacson and Keller, op. cit., p. 64.

eigenvalue of maximum modulus of a given iterative matrix.

Before attempting to find the rates of convergence for the methods of the previous section, we establish a relationship between eigenvalues of two types of matrices. This relationship, which is expressed in the following well-known theorem, greatly facilitates comparison of both the block and the point methods among themselves. Before stating the theorem, we give some preliminary definitions. We define the matrix A' to be given by

$$A' = \begin{bmatrix} A'_{1,1} & A'_{1,2} & \cdots & A'_{1,r} \\ A'_{2,1} & A'_{2,2} & \cdots & A'_{2,r} \\ \vdots & \vdots & \ddots & \vdots \\ A'_{r,1} & A'_{r,2} & \cdots & A'_{r,r} \end{bmatrix}$$

where the diagonal submatrices are square and non-singular. It is also convenient to write $A' = D' - E' - F'$, with

$$D' = \begin{bmatrix} A'_{1,1} & & & \\ & A'_{2,2} & & \bigcirc \\ & & \ddots & \\ \bigcirc & & & A'_{r,r} \end{bmatrix},$$

$$E' = -\begin{bmatrix} 0 & & & & \\ A'_{2,1} & 0 & & & \bigcirc \\ A'_{3,1} & A'_{3,2} & 0 & & \\ \vdots & & \ddots & & \\ A'_{r,1} & & & A'_{r,r-1} & 0 \end{bmatrix},$$

and

$$F = -\begin{bmatrix} 0 & A'_{1,2} & A'_{1,3} & \cdots & A'_{1,r} \\ & 0 & & & \vdots \\ & & \ddots & & \vdots \\ & & & \ddots & \vdots \\ & \bigcirc & & & A'_{r-1,r} \\ & & & & 0 \end{bmatrix}.$$

We assume the problem to be solved is given by $A'X = Y$, and define Method I to be given by

$$X^{(m+1)} = T'X^{(m)} + D'^{-1}Y,$$

where $T' = D'^{-1}(E' + F')$. Method II is

$$X^{(m+1)} = T''X^{(m)} + (D' - \omega E')^{-1}\omega Y,$$

with $T'' = (D' - \omega E')^{-1}(\omega F' + (1-\omega)D')$ and $\omega \neq 0$ the acceleration factor. We can now give the following

Theorem.[10] Assume T' can be partitioned in tridiagonal block form, with blocks on the main diagonal composed only of zeros. Let λ be a nonzero eigenvalue of T'', and let μ satisfy

$$(\lambda + \omega - 1)^2 = \lambda\omega^2\mu^2. \tag{2.3.4}$$

Then μ is an eigenvalue of T'. Further, if μ is an eigenvalue of T', and λ satisfies 2.3.4, then λ is an eigenvalue of T''.

Proof: First we establish what we call the "α property" of the matrix T'. By this, we mean that the eigenvalues of T' are the same as those of T_α, where T_α is given by

---

[10] See, for example, Varga's similar theorem, Ibid., pp. 106-107.

$$T_\alpha = \begin{bmatrix} 0 & \frac{1}{\alpha}T'_{1,2} & & & \\ \alpha T'_{2,1} & 0 & \frac{1}{\alpha}T'_{2,3} & & \\ & \alpha T'_{3,2} & 0 & & \\ & & & \ddots & \\ & & & & 0 \end{bmatrix}$$

and α is any nonzero number. Let (η,z) be an eigenpair for T'. Then $T'z = \eta z$, and

$$z = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_s \end{bmatrix},$$

partitioned to correspond to the partitioning of T'. Thus we have

$$T'_{i,i-1}z_{i-1} + T'_{i,i+1}z_{i+1} = \eta z_i \quad \text{for } 1 < i < s,$$

$$T'_{1,2}z_2 = \eta z_1, \text{ and } T'_{s,s-1}z_{s-1} = \eta z_s.$$

We define $\bar{z}$ as

$$\bar{z} = \begin{bmatrix} \alpha z_1 \\ \alpha^2 z_2 \\ \alpha^3 z_3 \\ . \\ . \\ . \\ \alpha^s z_s \end{bmatrix} = \begin{bmatrix} \bar{z}_1 \\ \bar{z}_2 \\ \bar{z}_3 \\ . \\ . \\ . \\ \bar{z}_s \end{bmatrix} .$$

The following, then, holds for $1 < i < s$:

$$\alpha T'_{i,i-1}\bar{z}_{i-1} + \frac{1}{\alpha} T'_{i,i+1}\bar{z}_{i+1}$$

$$= \alpha T'_{i,i-1}\alpha^{i-1}z_{i-1} + \frac{1}{\alpha} T'_{i,i+1}\alpha^{i+1}z_{i+1}$$

$$= \alpha^i(T'_{i,i-1}z_{i-1} + T'_{i,i+1}z_{i+1})$$

$$= \alpha^i(\eta z_i) = \eta\bar{z}_i.$$

A similar result holds for $i = 1$ and $i = s$. We now see that if η is any eigenvalue of T', η is also an eigenvalue of T_α. We further note that the α property holds for any matrix having zeros where T' has zeros.

With this preliminary result, we can proceed with the proof of the theorem. We write $T' = T'_L + T'_U$, where T'_L is strictly lower triangular and T'_U is

strictly upper triangular. T'' can now be expressed as

$$T'' = (I_N - \omega T_L')^{-1}\{\omega T_U' + (1-\omega)I_N\}.$$

Let λ be any nonzero eigenvalue of T'' and assume μ satisfies $(\lambda+\omega-1)^2 = \lambda\omega^2\mu^2$. Then we have the following:

$$\begin{aligned} 0 &= \det[(I_N - \omega T_L')^{-1}\{\omega T_U' + (1-\omega)I_N\} - \lambda I_N] \\ &= \det[\omega T_U' + (1-\omega)I_N - \lambda(I_N - \omega T_L')] \\ &= \det[(1-\omega-\lambda)I_N + \omega T_U' + \lambda\omega T_L']. \end{aligned}$$

The second equality is obtained using the fact that $\det(I_N - \omega T_L') = 1$. The next step is to apply the "α property" to the last expression in the above string of equalities. We recall that

$$\mu = \frac{\lambda+\omega-1}{\lambda^{\frac{1}{2}}\omega} \quad \text{or} \quad \mu = \frac{-(\lambda+\omega-1)}{\lambda^{\frac{1}{2}}\omega}.$$

We choose the sign of $\lambda^{\frac{1}{2}}$ so that the first expression for μ holds. Choosing $\alpha = \lambda^{\frac{1}{2}}$, we obtain

$$\det[(1-\omega-\lambda)I_N + \omega\lambda^{\frac{1}{2}}T_U' + \lambda^{\frac{1}{2}}\omega T_L'] = 0.$$

Since $(\lambda^{\frac{1}{2}}\omega)^{-1} I_N$ is nonsingular,

$$\det[(1-\omega-\lambda)(\lambda^{\frac{1}{2}}\omega)^{-1} I_N + T_U' + T_L'] = 0.$$

Thus we see that μ is an eigenvalue of T'.

The proof of the other half of the theorem follows by reversing the arguments given above.

The Method I and Method II of the theorem are obviously very similar to the Line Jacobi and Accelerated Line Gauss-Seidel methods. We now show that the theorem can, in fact, be applied to the methods given by 2.2.5 and 2.2.6. With the partitioning of A given in 2.2.4, the diagonal submatrices are square. Assuming $\theta_y \neq 0$, the diagonal submatrices are also strictly diagonally dominant and thus nonsingular. The Line Jacobi and Accelerated Line Gauss-Seidel are clearly of the proper form to be classified as Method I and Method II types, respectively. Thus we need only check for the property of $T' = \overline{D}^{-1}(\overline{E} + \overline{F})$ required by the theorem. By returning to the original definition of the matrix A given in section 2.1, we see that

$$A_{i,i} = S_I + \theta_x(T_I + T_I^T), \quad 1 \leq i \leq J-1,$$

$$A_{i,i-1} = \theta_y S_I, \quad 2 \leq i \leq J-1, \text{ and}$$

$$A_{i+1,i} = \theta_y S_I, \quad 1 \leq i \leq J-2.$$

All other blocks contain only zeros. Thus $\bar{D}^{-1}(\bar{E} + \bar{F})$ is block tridiagonal with zero blocks on the main diagonal. The theorem can, therefore, be used to relate the eigenvalues of the Line Jacobi iterative matrix to those of the Accelerated Line Gauss-Seidel iterative matrix. In particular, the largest eigenvalues, and consequently, the rates of convergence, can now be easily compared. For example, by choosing $\omega = 1$, we see that the eigenvalues of the Line Gauss-Seidel iterative matrix are the squares of those of the Line Jacobi iterative matrix. This means that the rate of convergence of the Line Gauss-Seidel method is equal to twice the rate of convergence of the Line Jacobi method. For this reason, and because the Line Jacobi requires twice as much storage, the Line Gauss-Seidel is clearly more efficient. The Accelerated Line Gauss-Seidel, as we see from the theorem, is even better. The optimal acceleration parameter has been shown to be

$$\omega = \frac{2}{1 + \sqrt{1-\upsilon^2}} ,$$

where $\upsilon = (2\theta_y \cos\frac{\pi}{I})(1 - 2\theta_x \cos\frac{\pi}{J})^{-1}$ is the largest eigenvalue of the Line Jacobi iterative matrix. The

rate of convergence[11] when this value is used for ω is:

$$2\pi k\sqrt{\left(\frac{1}{a^2} + \frac{1}{b^2}\right)} + \mathcal{O}\left(\frac{1}{\theta}\right).$$

The point methods of section 2.2 do not satisfy the hypotheses of the theorem, but a similar result is true for them, also.[12] Thus we find that, as in the case of the block methods, the Point Gauss-Seidel method is superior to the Point Jacobi method. The optimal acceleration factor for the Accelerated Point Gauss-Seidel method is

$$\omega = \frac{2}{1+\sqrt{1-\lambda^2}},$$

where $\lambda = 1-4\theta_x\sin^2(\frac{\pi}{2I}) - 4\theta_y\sin^2(\frac{\pi}{2J})$ is the largest eigenvalue of the Point Jacobi iterative matrix. Using this value of ω, the rate of convergence[13] is

$$2\pi\sqrt{\frac{2}{\theta}\left(\frac{1}{a^2} + \frac{1}{b^2}\right)} + \mathcal{O}\left(\frac{1}{\theta}\right).$$

---

[11] Isaacson and Keller, op. cit., p. 474.

[12] Ibid., pp. 465-467.

[13] Ibid., p. 470.

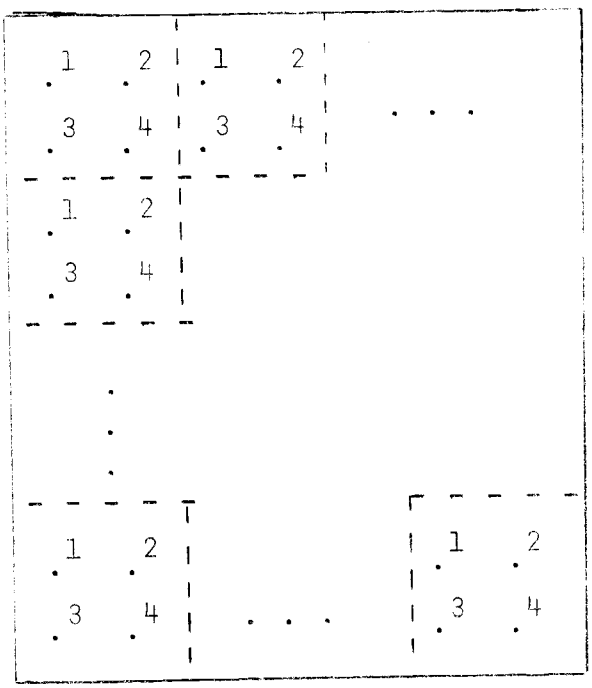
III. THE ALTERNATING HALVES METHODS

3.1 The New Matrix Problem

The methods described in the previous chapter are based on a sequential approach to computations. Employing such methods to solve the Laplace problem on a parallel computer makes little use of the special capabilities of the machine. As will be seen in the next chapter, some economies are possible if one of the conventional block methods is used. Even these block methods, however, cannot take full advantage of the number of processors available.

A new method for solving the Laplace equation on a parallel computer appears in a 1967 paper by Carroll and Wetherald.[14] They use the standard finite difference approach explained in Chapter II. The solution is obtained via a modified Point Gauss-Seidel method. The grid D_{IJ} is divided into groups of four points in the following way:

[14] A.B. Carroll, and R.T. Wetherald, "Application of Parallel Processing to Numerical Weather Prediction." Journal of the Association for Computing Machinery, 14(July, 1967), 591-614.



All points labeled with the same number can be processed simultaneously. That is,

$$\frac{(I-1)(J-1)}{4}$$

unknowns are estimated in each iterative step.

No formal analysis of the method is given, but tests conducted by the authors show the convergence rate of the new method to be slightly better than than of the Point Gauss-Seidel. The real advantage to the method is that it processes one-fourth of the unknowns simultaneously. In the following paragraphs we describe two new block-type methods designed to even more completely employ the processors of a

parallel computer. These methods process half of the unknowns simultaneously. Due to the lack of analysis in the paper by Carroll and Wetherald, we are not able to compare the rate of convergence of their method to the rates of convergence of the Alternating Halves methods presented below.

The Alternating Halves methods are similar to the line methods. In the latter methods, each block corresponds to a row of the grid, D_{IJ}, and a single block is processed in each iterative step. The new methods, in contrast, treat the unknowns of approximately half of the rows of the grid in each iterative step. More specifically, the first iterative step, and every odd-numbered step thereafter, finds estimates for the unknowns corresponding to the grid points on all the odd-numbered rows of the grid. Each even-numbered iterative step treats all unknowns corresponding to points in the even-numbered rows of the grid.

Since the Alternating Halves methods process the unknowns in a different order from that assumed in section 2.1, a reformulation of the matrix problem is necessary. This new system is denoted by $A^*U^* = G^*$. A^* is obtained from A by permuting certain rows and columns. In particular, $A^* = PAP^T$, with the permutation

matrix P defined as follows. Let S_I be an $(I-1)\times(I-1)$ identity matrix. Let P be composed of $(J-1)\times(J-1)$ submatrices $P_{i,j}$, with each submatrix being square of dimension $I-1$. If we further define 0_I to be the $(I-1)\times(I-1)$ matrix composed entirely of zeros, and define K to be the largest integer satisfying $K \leq J/2$, then

$$P = \begin{bmatrix} P_{1,1} & P_{1,2} & \cdots & P_{1,J-1} \\ \vdots & & & \\ P_{J-1,1} & & \cdots & P_{J-1,J-1} \end{bmatrix}, \text{ with}$$

$$P_{i,j} = \begin{cases} S_I & \text{if } j = 2i - 1,\ 1 \leq i \leq K, \\ S_I & \text{if } j = 2i - 2K,\ K + 1 \leq i \leq J - 1, \\ 0_I & \text{otherwise.} \end{cases}$$

Now to exactly specify the matrix A^*, we define S_1 to be the $(I-1)(J-1-K)\times(I-1)K$ matrix given by

$$S_1 = \begin{bmatrix} S_I & S_I & & & & \\ S_I & S_I & & & & \\ & & \ddots & & & \\ & & & \ddots & & \\ & & & & S_I & S_I \\ & & & & & S_I \end{bmatrix}$$

if J is an odd number, or

$$S_1 = \begin{bmatrix} S_I & S_I & & & & \\ S_I & S_I & & & & \\ & & \ddots & & & \\ & & & \ddots & & \\ & & & & S_I & S_I \end{bmatrix}$$

if J is even. Then the $(I-1)(J-1)\times(I-1)(J-1)$ matrix S_2 is given by

$$S_2 = \begin{bmatrix} 0 & 0 \\ S_1 & 0 \end{bmatrix}.$$

Letting T be defined as in section 2.1, we have

$$A^* = \theta_x(T + T^T) + \theta_y(S_2 + S_2{}^T) - I_N.$$

A^* and P, for the case $I = J = 5$, are shown ex-

plicitly in figures 2 and 3.

The last step in reordering the equations of the system $AU = G$ is to reorder the vectors U and G. The new order still numbers the unknowns from left to right along each row of the grid. The difference, as stated before, is that the rows are considered in the following order: 1, 3, 5, 7, ..., 2, 4, 6, ... We define U^*, then, to be $U^* = PUP^T$. G^* is similarly defined.

3.2 Definition and Evaluation of the Alternating Halves Methods.

Using the definitions of the previous section, the Alternating Halves method is given by the following:

(3.2.1) $U^{*(m+1)} = (H^*-I_N)^{-1}V^*U^{*(m)} + (H^*-I_N)^{-1}G^*$,

with $H^* = \theta_x(T + T^T)$ and $V^* = -\theta_y(S_2 + S_2{}^T)$. Assuming $\theta_y \neq 0$, $(H^*-I_N)^{-1}$ exists because H^*-I_N is strictly diagonally dominant. In order to compare this method with those discussed in Chapter II, we must determine the eigenvalues of maximum modulus of $(H^*-I_N)^{-1}V^*$. To this end, we also define

$$
\begin{array}{cccccccccccccccc}
-1 & \theta_x & 0 & 0 & 0 & 0 & 0 & 0 & \theta_y & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\theta_x & -1 & \theta_x & 0 & 0 & 0 & 0 & 0 & 0 & \theta_y & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \theta_x & -1 & \theta_x & 0 & 0 & 0 & 0 & 0 & 0 & \theta_y & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \theta_x & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \theta_y & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & \theta_x & 0 & 0 & \theta_y & 0 & 0 & 0 & \theta_y & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \theta_x & -1 & \theta_x & 0 & 0 & \theta_y & 0 & 0 & 0 & \theta_y & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \theta_x & -1 & \theta_x & 0 & 0 & \theta_y & 0 & 0 & 0 & \theta_y & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \theta_x & -1 & 0 & 0 & 0 & \theta_y & 0 & 0 & 0 & \theta_y \\
\theta_y & 0 & 0 & 0 & \theta_y & 0 & 0 & 0 & -1 & \theta_x & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \theta_y & 0 & 0 & 0 & \theta_y & 0 & 0 & \theta_x & -1 & \theta_x & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \theta_y & 0 & 0 & 0 & \theta_y & 0 & 0 & \theta_x & -1 & \theta_x & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \theta_y & 0 & 0 & 0 & \theta_y & 0 & 0 & \theta_x & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \theta_y & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & \theta_x & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \theta_y & 0 & 0 & 0 & 0 & 0 & 0 & \theta_x & -1 & \theta_x & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \theta_y & 0 & 0 & 0 & 0 & 0 & 0 & \theta_x & -1 & \theta_x \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \theta_y & 0 & 0 & 0 & 0 & 0 & 0 & \theta_x & -1
\end{array}
$$

Figure 2. The Matrix A*

When $I = J = 5$.

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 3. The Matrix P

When I = J = 5.

$H = \theta_x(T + T^T)$ and $V = -\theta_y(S + S^T)$. After noting that $\bar{D} = \theta_x(T + T^T) - I_N$ and $(\bar{E} + \bar{F}) = -\theta_y(S + S^T)$, we see that the iterative matrix of the Line Jacobi method can be expressed as $(H - I_N)^{-1}V$. By setting $\theta_x = 0$, it follows that

$$-V^* - I_N = PAP^T = P(-V - I_N)P^T =$$

$$-PVP^T - I_N$$

so that $V^* = PVP^T$. A similar argument shows that $PHP^T = H^*$. Starting with the Line Jacobi iterative matrix, then, we observe the following relationships:

$$\begin{aligned} & P[(H - I_N)^{-1}V]P^T \\ = \; & P(H - I_N)^{-1}P^TPVP^T \\ = \; & [P(H - I_N)P^T]^{-1}PVP^T \\ = \; & (H^* - I_N)^{-1}V^*. \end{aligned}$$

The Line Jacobi iterative matrix is thus seen to be similar to the Alternating Halves iterative matrix. Since the two matrices have the same eigenvalues, it follows that the corresponding methods have the same rate of convergence.

In preparation for introducing the second new

method, the iterative matrix of the Alternating Halves method is further examined. First, we partition A^* into four submatrices:

$$A^* = \begin{bmatrix} A^*_{1,1} & A^*_{1,2} \\ A^*_{2,1} & A^*_{2,2} \end{bmatrix}.$$

We let $A^*_{2,1} = S_1$ and $A^*_{1,2} = S_1^T$ and we note that with this specification, $A^*_{1,1}$ and $A^*_{2,2}$ are also defined and are square. Further, we define

$$D^* = \begin{bmatrix} A^*_{1,1} & 0 \\ 0 & A^*_{2,2} \end{bmatrix}$$

By comparing D^* to the definition of $H^* - I_N$ and recalling that

$$V^* = -\theta_y(S_2 + S_2^T),$$

we see that the method given by

(3.2.2) $D^*U^{*(m+1)} = -\theta_y(S_2 + S_2^T)U^{*(m)} + G^*$

is identical to 3.2.1.

We now define the second new method, the Accelerated Alternating Halves method, to be

(3.2.3) $(D^*+\omega\theta_y S_2)U^{*(m+1)} = (-\theta_y\omega S_2^T+(1-\omega)D^*)U^{*(m)}+\omega G^*.$

Assuming $\theta_y \neq 0$, it follows that D^* is nonsingular since it is strictly diagonally dominant. Observing that $D^* + \omega\theta_y S_2 = D^*(I + D^{*-1}\omega\theta_y S_2)$ and that $D^{*-1}\omega\theta_y S_2$ is strictly lower triangular and thus nonsingular, we have that $D^* + \omega\theta_y S_2$ is nonsingular.

The method given by 3.2.3 is written in that particular form, using the "2x2" partitioning of A^*, in order to use the theorem of Chapter II to find the eigenvalues of the iterative matrix. We now show that the hypotheses of that theorem are satisfied. It was already noted that the diagonal submatrices $A^*_{1,1}$ and $A^*_{2,2}$ are square. Assuming $\theta_y \neq 0$, they are also strictly diagonally dominant and thus nonsingular.

Letting $D' = D^*$, $E' = -\theta_y S_2$, and $F' = -\theta_y S_2^T$, we see that 3.2.2 is in "Method I" form and 3.2.3 is of the proper form to be classified as "Method II." It is equally evident that T^* is of the following form:

$$T^* = \begin{bmatrix} 0 & T^*_{1,2} \\ T^*_{2,1} & 0 \end{bmatrix}.$$

Thus the theorem can be applied to find the eigenvalues of the iterative matrix of 3.2.3. We recall that the eigenvalues of the Alternating Halves iterative matrix are the same as those of the Line Jacobi iterative matrix. From this, it follows that, for any ω, $0 < \omega < 2$, the rate of convergence for the Accelerated Alternating Halves method is the same as that of the Accelerated Line Gauss-Seidel. It can also be shown that the optimal acceleration factor is the same for both methods.[15]

[15] Varga, op. cit., pp. 109-112.

IV. THE EFFECT OF PARALLEL COMPUTATIONS ON METHOD EFFICIENCY

4.1 Overview and Assumptions of the Evaluation

In this chapter we compare, on the basis of number of arithmetic operations required, three of the methods described in the preceding chapters. Because, in Chapter III, the Point and Line Jacobi and the Alternating Halves methods were found to be inferior to their respective accelerated versions, in this chapter we consider only the following methods: the Accelerated Point Gauss-Seidel, the Accelerated Line Gauss-Seidel, and the Accelerated Alternating Halves. In the following three sections, we estimate the number of parallel arithmetic operations required for a single iterative step of each of these methods. By way of explaining what is meant by parallel arithmetic operations, we first state the assumptions of the following sections.

It is assumed in the remainder of this chapter that the methods compared are to be used on a computer with 64 parallel processors. It is also assumed that, although each processor has its own data set, all processors must perform the same operation at a given time. The user would have the option, however, to select a different subset of the processors to be "on" at any time. In

such a case, those processors selected to be on would all perform the same operation, while the rest would do nothing to their data sets. In this paper, then, we use the phrase "one parallel arithmetic operation" to refer to the total action of all the processors during the time necessary for an arithmetic operation on a single processor. One parallel addition, for example, may consist of one addition on each of as many as 64 processors.

One definition is necessary before looking at the individual methods. In the following sections, we use the function σ, where $\sigma(x)$ is defined to be the smallest integer greater than or equal to x (eg. $\sigma(15/2) = 8$, $\sigma(3) = 3$). This notation allows calculation of the number of necessary operations for general I and J.

4.2 Summary of Arithmetic Operations Required by the Accelerated Point Gauss-Seidel Method

The Accelerated Point Gauss-Seidel method requires the calculation of $U_i^{(m+1)}$ before $U_{i+1}^{(m+1)}$ can be calculated for all $1 \leq i \leq (I-1)(J-1)$, $m > 0$. Thus each iterative step involves finding only $U_i^{(m+1)}$ which is given by:

$$U_i^{(m+1)} = U_i^{(m)} + \omega\{\theta_x U_{i-1}^{(m+1)} + \theta_y U_{i-I+1}^{(m+1)} + \theta_x U_{i+1}^{(m)} + \theta_y U_{i+I-1}^{(m)} - G_i - U_i^{(m)}\},$$

$$I \leq i \leq (I-1)(J-2). \qquad (4.2.1)$$

In the case of equations involving boundary points, some of the terms shown in 4.2.1 are omitted. Regardless of whether terms are omitted, and assuming that the quantities $\omega\theta_y$ and $\omega\theta_x$ are available, the following parallel operations are required for one iterative step:

1) One parallel multiplication calculating simultaneously the following:

$t_1 \leftarrow (\omega\theta_x) * U_{i-1}^{(m+1)}$

$t_2 \leftarrow (\omega\theta_y) * U_{i-I+1}^{(m+1)}$

$t_3 \leftarrow (\omega\theta_x) * U_{i+1}^{(m)}$

$t_4 \leftarrow (\omega\theta_y) * U_{i+I-1}^{(m)}$

$t_5 \leftarrow -\omega * G_i$

$t_6 \leftarrow (1-\omega) * U_i^{(m)}$

2) One parallel addition calculating:

$t_1 \leftarrow t_1 + t_2$

$t_3 \leftarrow t_3 + t_4$

$t_5 \leftarrow t_5 + t_6$

3) One parallel addition calculating:

$t_1 \leftarrow t_1 + t_3$

4) one parallel addition calculating:

$$U_i^{(m+1)} \leftarrow t_1 + t_5$$

We conclude that calculating $U^{(m+1)}$ from $U^{(m)}$ requires the following:

(I-1)(J-1) parallel multiplications and

3(I-1)(J-1) parallel additions.

4.3 Summary of Arithmetic Operations Required by the Accelerated Line Gauss-Seidel Method

We recall from section 2.2 that each iterative step of the Accelerated Line Gauss-Seidel method consists of the solution of J-1 linear systems. Each of these systems has the form

$$A_{i,i}U_{[i]}^{(m+1)} = -\omega\theta_y U_{[i-1]}^{(m+1)} - \omega\theta_y U_{[i+1]}^{(m)} + (1-\omega)A_{i,i}U_{[i]}^{(m)} + \omega G_{[i]}, \qquad (4.3.1)$$

$$1 \leq i \leq J-1,$$

if we assume $U_{[0]}$ and $U_{[J]}$ are vectors of zeros. Because $A_{i,i}$ is tridiagonal and the same for all i, $1 \leq i \leq J-1$, we rewrite 4.3.1, the iterative step as

$$A_L A_U U_{[i]}^{(m+1)} = -\omega\theta_y U_{[i-1]}^{(m+1)} - \omega\theta_y U_{[i+1]}^{(m)} +$$

$$(1-\omega)A_{1,1}U_{[i]}^{(m)} + \omega G_{[i]}. \qquad (4.3.2)$$

Here $A_L A_U = A_{1,1}$, with

$$A_U = \begin{bmatrix} u_1 & \theta_x & & & & \\ & u_2 & \theta_x & & & \\ & & \cdot & \cdot & & \\ & & & \cdot & \cdot & \\ & & & & \cdot & \cdot \\ & & & & U_{I-2} & \theta_x \\ & & & & & U_{I-1} \end{bmatrix},$$

and

$$A_L = \begin{bmatrix} 1 & & & & & \\ \ell_2 & 1 & & & & \\ & \ell_3 & 1 & & & \\ & & \cdot & \cdot & & \\ & & & \cdot & \cdot & \\ & & & & \cdot & \cdot \\ & & & & \ell_{I-1} & 1 \end{bmatrix}.$$

The procedure to solve 4.3.2 involves first performing the multiplications and additions on the right side of the equation, and then using an "LU" decomposition al-

gorithm to solve for $U_{[i]}^{(m+1)}$. The matrices A_U and A_L are calculated initially, and because they remain constant, they do not have to be calculated more than once.

The nonzero elements of A_U and A_L are specified by the following equations:

$$u_1 = -1,\ u_j = -1 - (\theta_x^2/u_{j-1}),\ 2 \leq j \leq I-1,$$

$$\ell_2 = -\theta_x,\ \ell_j = \theta_x/u_{j-1},\ 3 \leq j \leq I-1.$$

Rather than solve these sequentially as the formulas seem to require, we use the technique of "recursive doubling." This technique was recently shown, in a paper by H.S. Stone,[16] to expedite all phases of the solution of a tridiagonal system on a parallel computer. Typically, this technique is applied to a system of equations of the following form:

$$y_1 = b_1,\ y_i = b_i + m_i y_{i-1},\ 2 \leq i \leq n.$$

We note that y_i can be written as

$$(4.3.3) \qquad y_i = \sum_{j=1}^{i} b_j \prod_{k=j+1}^{i} m_k,\ 1 \leq i \leq n,$$

[16] Harold S. Stone, "An Efficient Parallel Algorithm for the Solution of a Tridiagonal Linear System of Equations," Journal of the Association for Computing Machinery, 20 (January, 1973), pp. 27-38.

with the assumption that an empty product of the m_k is equal to one. Applying the recursive doubling technique here, we have the following algorithm:

(4.3.4)

1) $y_1 \leftarrow b_1$, $y_2 \leftarrow b_2$, ... $y_n \leftarrow b_n$.

2) for $j = 1$ step j until z (where z is defined below) do the following:

   a) $y_i \leftarrow y_i + y_{i-j} * m_i$, for $j+1 \leq i \leq n$.

   b) $m_i \leftarrow m_i * m_{i-j}$, for $2j + 1 \leq i \leq n$.

In step 2a. it is assumed that all the right-hand sides, $j+1 \leq i \leq n$, are calculated in parallel. This means that all the values of y_i, $j+1 \leq i \leq n$, are changed simultaneously. The right-hand sides of step 2b. are similarly done in parallel. They can be calculated after step 2a. is completed, or the two steps can be done concurrently. If we have $2^{k-1} < n \leq 2^k$, then we define z to be 2^{k-1}. When step 2-a is completed for $j = z$, the y_i, $1 \leq i \leq n$, have the values given in 4.3.3. The parallelism of the algorithm comes from the fact that for a given j, all y_i, and/or all m_i, $j+1 \leq i \leq n$, can be calculated simultaneously. The result is that, instead of the number of operations being proportional to n as is the case with the traditional sequential algorithm, the number of operations is proportional to k.

To give a better picture of what is involved in the algorithm of 4.3.4, we assume that $m_j=1$, $2 \leq j \leq n$. The first step is initialization. The second step adds together pairs of the b_i in the following way:

$$y_2 \leftarrow b_2 + b_1,\ y_3 \leftarrow b_3 + b_2,\ \ldots\ y_n \leftarrow b_n + b_{n-1}.$$

Each succeeding step doubles the number of b_i included in y_j. The process is completed for a particular y_i when y_i is the sum of all b_j, $1 \leq j \leq i$.

In a similar way Stone has implemented the construction of A_U. We see that we need the following operations to calculate u_i, $2 \leq i \leq I-1$ (with $z=2^{k-1} < I-1 \leq 2^k$):

1) one parallel multiplication to calculate

$$t_I \leftarrow -\theta_x * \theta_x.$$

2) one parallel addition to calculate

$$t_{I+1} \leftarrow t_I + 1.$$

Then the following assignments are made:

$$t_1 \leftarrow 0,\ t_i \leftarrow t_I,\ 2 \leq i \leq I-1,$$

$$r_i \leftarrow 1,\ -1 \leq i \leq I-1,$$

$$q_0 \leftarrow 1,\ q_i \leftarrow -1,\ 1 \leq i \leq I-1,$$

$$p_1 \leftarrow -1,\ p_i \leftarrow t_{I+1},\ 2 \leq i \leq I-1.$$

For $i = 2$ step i until z, do 3,4,5.

3) $\sigma((2I - 2i + 2)/64) + \sigma((I - i + 1)/64)$

parallel multiplications and

$\sigma((I-i)/64)$

parallel additions to get

$s_j \leftarrow q_j * q_{j-i+1} + t_{j-i+2} * r_j * r_{j-1}$,

$i - 1 \leq j \leq I-1$.

4) $\sigma((2I - 2i)/64) + \sigma((I-i)/64)$

parallel multiplications and

$\sigma((I - i)/64)$

parallel additions to calculate

$q_j \leftarrow p_j * q_{j-i} + t_{j-i+1} * q_j * r_{j-i-1}$,

$i \leq j \leq I-1$.

Then assign

$r_j \leftarrow s_j$, $i - 1 \leq j \leq I-1$.

5) $\sigma((I-i-1)/64)$ parallel multiplications and

$\sigma((I-i-1)/64)$ parallel additions to get

$p_j \leftarrow -q_{j-1} + t_j * r_{j-2}$, $i + 1 \leq j \leq I-1$.

6) $\sigma((I - 2)/64)$ parallel divisions to calculate

$u_j \leftarrow p_j/p_{j-1}$, $2 \leq j \leq I-1$.

Calculating the lower diagonal of A_L now requires only the following:

1) $\sigma((I - 3)/64)$ parallel divisions to obtain

$\ell_j \leftarrow \theta_x/u_{j-1}$, $3 \leq j \leq I-1$.

After the preliminary calculations, each iterative step, as given by 4.3.2, is easy to perform. Assuming that $1-\omega$, $\omega\theta_y$, $(1-\omega)\theta_x$, and $\omega G_{[i]}$, $1 \leq i \leq J-1$, are also initially available, the iterative step which calculates $U_{[i]}^{(m+1)}$ requires the following operations:[17]

1) $\sigma((4I - 4)/64)$ parallel multiplications and $\sigma((3I - 5)/64) + 2\sigma((I-1)/64)$ parallel additions to calculate

$$t_{[1]} \leftarrow (-\omega\theta_y) * U_{[i-1]}^{(m+1)} + (-\omega\theta_y) * U_{[i+1]}^{(m)} + (1-\omega)A_{1,1} * U_{[i]}^{(m)} + \omega G_{[i]}.$$

Assign the following:

$Y_j \leftarrow t_j$, $1 \leq j \leq I-1$

$m_j \leftarrow -\ell_j$, $2 \leq j \leq I-1$

For $i = 1$ step i until z do 2,3,4

2) $\sigma((I - i - 1)/64)$ parallel multiplications to calculate

$t_j \leftarrow Y_{j-i} * m_j$, $i + 1 \leq j \leq I-1$

3) $\sigma((I - i - 1)/64)$ parallel additions to get

$Y_j \leftarrow Y_j + t_j$, $i + 1 \leq j \leq I-1$

---

[17] We use $t_{[i]}$ to represent the i^{th} subvector of the temporary storage vector t. Here each subvector is of dimension $I-1$.

4) $\sigma((I - 2i - 1)/64)$ parallel multiplications

to calculate

$$m_j \leftarrow m_j * m_{j-i},\ 2i + 1 \leq j \leq I-1.$$

5) $\sigma((2I - 3)/64)$ parallel divisions

to calculate

$$Y_j \leftarrow Y_j / u_j,\ 1 \leq j \leq I - 1.$$
$$n_j \leftarrow -\theta_x / u_j,\ 1 \leq j \leq I - 2.$$

Assign the following:

$$n_{I-1} \leftarrow 1$$

For i = 1 step i until z do 6, 7

6) $\sigma((I - i - 1)/64)$ parallel multiplications

and

$\sigma((I - i - 1)/64)$ parallel additions

to calculate

$$Y_j \leftarrow Y_j + Y_{j+i} * n_j,\ 1 \leq j \leq I - i - 1.$$

7) $\sigma((I - i - 1)/64)$ parallel multiplications

to calculate

$$n_j \leftarrow n_j * n_{j+i},\ 1 \leq j \leq I - i - 1.$$

After execution of step 7 with i = z, the vector Y contains $U^{(m+1)}_{[i]}$. For the special cases i = 1 and i = J - 1, where either $\theta_y U_{[i-1]}$ or $\theta_y U_{[i+1]}$ is included in $G_{[i]}$, fewer operations are needed. It is

also important to note that the subvectors of $U^{(m+1)}$ must be estimated sequentially since calculation of each $U_{[i]}^{(m+1)}$ involves the values of $U_{[i-1]}^{(m)}$. We further observe that the values of m_j and n_j, $1 \leq j \leq I - 1$, are the same for each iteration. We thus calculate the m_j and n_j, $1 \leq j \leq I - 1$, on only the first iteration, saving all intermediate values since these also must be used during each iteration.

To aid in calculating the total operations for an iteration, we define the following:

$$(i) \sum_{i=1}^{2^k} \alpha(i) = \alpha(1) + \alpha(2) + \alpha(4) + \alpha(8) + \ldots + \alpha(2^k),$$

for any function α. We can now give the total number of operations required to calculate $U^{(m+1)}$ from $U^{(m)}$, after the initial calculation of A_U, A_L, m_j and n_j, $1 \leq j \leq I - 1$:

$$(J-3)\sigma((4I - 4)/64) + 2\sigma((3I - 3)/64) + (J-1)(i) \sum_{i=1}^{z} 2\sigma((I - i - 1)/64)$$

parallel multiplications,

$$(J-3)\sigma((3I - 5)/64) + 2\sigma((2I - 4)/64) +$$

$$(J-1)[(i)\sum_{i=1}^{z} 2\sigma((I - i - 1)/64) + 2\sigma((I - 1)/64)]$$

parallel additions, and

$(J-1)\sigma((I-1)/64)$ parallel divisions.

4.4 Summary of Arithmetic Operations Required by the Accelerated Alternating Halves Method

Estimating the number of operations required by the Accelerated Alternating Halves method is not a great deal different from the procedure of section 4.3 The one significant difference is that U^* is partitioned into only two subvectors, $U^*_{[1]}$ and $U^*_{[2]}$. An iterative step thus involves calculating either

$$(4.4.1)\quad A^*_{1,1}U^{*(m+1)}_{[1]} = -\omega\theta_y S_1{}^T U^{*(m)}_{[2]} + \omega G^*_{[1]} + (1-\omega)A^*_{1,1}U^{*(m)}_{[1]},$$

or

$$(4.4.2)\quad A^*_{2,2}\,U^{*(m+1)}_{[2]} = -\omega\theta_y S_1 U^{*(m+1)}_{[1]} + \omega G^*_{[2]} + (1-\omega)A^*_{2,2}U^{*(m)}_{[2]}.$$

If we assume J is an odd number, then $A^*_{1,1} = A^*_{2,2}$. In this case, the initial calculations involve finding $\omega\theta_y$, ωG^*,$(1-\omega)\theta_x$,$1-\omega$,A^*_L, and A^*_U, where $A^*_L A^*_U = A^*_{1,1}$,

$$A_L^* = \begin{bmatrix} 1 & & & & \\ \ell_2 & 1 & & & \\ & \ell_3 & 1 & & \\ & & \ddots & \ddots & \\ & & & \ell_{K(I-1)} & 1 \end{bmatrix},$$

and

$$A_U^* = \begin{bmatrix} u_1 & \theta_x & & & \\ & u_2 & \theta_x & & \\ & & u_3 & \theta_x & \\ & & & \ddots & \ddots \\ & & & & u_{K(I-1)} \quad \theta_x \end{bmatrix}.$$

We recall that the tridiagonal matrix $A_{1,1}^*$ has the form

$$A_{1,1}^* = \begin{bmatrix} A_{1,1} & & & & \\ & A_{2,2} & & & \\ & & A_{3,3} & & \\ & & & \ddots & \\ & & & & A_{K,K} \end{bmatrix},$$

with the diagonal submatrices defined as they were in connection with the Line Gauss-Seidel method. We know, then, that

$$A_L^* = \begin{bmatrix} A_L & & & \\ & A_L & & \\ & & A_L & \\ & & & \ddots \end{bmatrix},$$

and

$$A_U^* = \begin{bmatrix} A_U & & & \\ & A_U & & \\ & & A_U & \\ & & & \ddots \end{bmatrix}.$$

Here, as in section 4.3, the initial operations are those necessary to find A_L and A_U. These are given in detail in the preceding section.

Again assuming $A^*_{1,1} = A^*_{2,2}$, it is clear that

4.4.1 and 4.4.2 require the same number of arithmetic operations. To calculate 4.4.1, we do the following:

1)$\sigma(3K(I-1)/64)$ parallel multiplications and

$\sigma((3KI-I-5K+1)/64) + 2\sigma(K(I-1)/64)$

parallel additions to calculate

$$t_{[1]} \leftarrow (-\omega\theta_y S_1^T) * U^{*(m)}_{[2]} + \omega G^*_{[1]} + (1-\omega)A^*_{1,1} * U^{*(m)}_{[1]}.$$

Assign the following:

$Y_{k+j} \leftarrow t_{k+j}$, $1 \leq j \leq I-1$, $k = k'(I-1)$, $0 \leq k' \leq K-1$.

$m_j \leftarrow -\ell_j$, $2 \leq j \leq I-1$.

For i = 1 step i until z do 2,3, and 4.

2) $\sigma(K(I-i-1)/64)$ parallel multiplications to calculate

$t_{k+j} \leftarrow Y_{k+j-i} * m_j$, $i+1 \leq j \leq I-1$, $k = k'(I-1)$, $0 \leq k' \leq K-1$.

3) $\sigma(K(I-i-1)/64)$ parallel additions to calculate

$Y_{k+j} \leftarrow Y_{k+j} + t_{k+j}$, $i+1 \leq j \leq I-1$, $k = k'(I-1)$, $0 \leq k' \leq K-1$.

4) $\sigma((I-2i-1)/64)$ parallel multiplications to calculate

$m_j \leftarrow m_j * m_{j-i}$, $2i+1 \leq j \leq I-1$.

Assign the following:

$n_{I-1} \leftarrow 1$

5) $\sigma((KI+I-K-2)/64)$ parallel divisions

to calculate

$Y_{k+j} \leftarrow Y_{k+j}/u_{k+j}$, $1 \leq j \leq I-1$, $k = k'(I-1)$,

$0 \leq k' \leq K-1$.

$n_j \leftarrow -\theta_x/u_j$, $1 \leq j \leq I-2$.

For $i = 1$ step i until z do 6,7.

6) $\sigma(K(I-i-1)/64)$ parallel multiplications and

$\sigma(K(I-i-1)/64)$ parallel additions

to calculate

$Y_{k+j} \leftarrow Y_{k+j} + Y_{k+j+i} * n_j$, $1 \leq j \leq I-1-i$,

$k = k'(I-1)$, $0 \leq k' \leq K-1$.

7) $\sigma((I-i-1)/64)$ parallel multiplications

to calculate

$n_j \leftarrow n_j * n_{j+i}$, $1 \leq j \leq I-i-1$.

Again, we note that the values of m_j and n_j, $1 \leq j \leq I-1$, do not need to be recalculated after the first iteration. After the initial calculations, then, the total parallel operations required to calculate $U^*(m+1)$ from $U^*(m)$ are the following:

$$(i)\sum_{i=1}^{z} 4\sigma(K(I-i-1)/64) + 2\sigma(3K(I-1)/64)$$

parallel multiplications,

$$2\sigma((3KI-I-5K+1)/64) + 4\sigma(K(I-1)/64)$$

$$+ (i)\sum_{i=1}^{z} 4\sigma(K(I-i-1)/64)$$

parallel additions, and

$$2\sigma(K(I-1)/64)$$

parallel divisions.

V. EXAMPLES, EVALUATIONS, AND CONCLUSIONS

5.1 A Closer Look at the Operation Count and the Effect of the Values of I and J.

Because the operation count formulas of Chapter IV are written in a general way to avoid specification of I and J, they are difficult to compare. In this section we see how the methods compare when the formulas are evaluated using a variety of values for I and J. We begin with an example.

To determine the total effect of combining the operation counts of Chapter IV with the rates of convergence for the three accelerated methods, we examine the case where $I = J = 33$ and $h = k = .1$. It is assumed that the acceleration parameter for each method is set to the optimal value. From Isaacson and Keller's discussion of rate of convergence,[18] we find that the number of iterations required to reduce the error by a factor of 10^{-t} is the least value of m for which $(\rho(B))^m \leq 10^{-t}$, where B is the iterative matrix of the method.

---

[18] Isaacson and Keller, op. cit., p. 64.

If we assume that the initial error $e^{(0)}$ is to be improved by a factor of 10^{-3}, then we have that

$$(\rho(B))^m \leq 10^{-3},\ m \ln \rho(B) \leq -3 \ln 10,$$

$$\text{and}\quad m \geq \left|\frac{3 \ln 10}{\ln \rho(B)}\right| = \frac{3 \ln 10}{R_\infty(B)} .$$

The minimal values of m for the three methods, together with the respective total numbers of operations required to achieve the specified accuracy, are given in table 1. For this particular example, we see that the Accelerated Alternating Halves method is approximately twice as fast as the Accelerated Line Gauss-Seidel and over eight times as fast as the Accelerated Point Gauss-Seidel. The percentage of improvement of the Accelerated Alternating Halves method over either of the other two is generally not as good. As $I = J$ increases, we find that the percentage of improvement gradually decreases. When $I = J = 257$, for example, the number of operations required by the Accelerated Alternating Halves method is about 95% of the number of operations required by the Accelerated Line Gauss-Seidel and about 20% of the number of operations required by the Accelerated Point Gauss-Seidel. It is important to note, however, that the decrease in the percentage of improvement is not monotone. The case where

TABLE 1.

Total Parallel Operations Required by Each of Three Methods to Improve $e^{(0)}$ by a Factor of 10^{-3} When $I = J = 33$, $h = k = .1$.

Method	Rate of Convergence	Number of Iterations (m)	Parallel Operations Per Iteration	Total Parallel Operations
Accelerated Point Gauss-Seidel	.189	37	1024 mult. 3072 add.	37888 mult. 113664 add.
Accelerated Line Gauss-Seidel	.269	26	384 mult 446 add. 32 div.	10004 mult. 11609 add. 834 div.
Accelerated Alternating Halves	.269	26	180 mult. 210 add. 16 div.	4700 mult. 5473 add. 418 div.

$I - 1 = J - 1 = 64q$, for some positive integer q, generally yields a lower percentage of improvement than the case where $I - 1 = J - 1 = 64q + p$, $1 \leq p \leq 10$, p an integer. In fact, any value r of $I = J$, for which $I - 1 = J - 1$ is not a multiple of 64 is likely to yield a better percentage of improvement than s or t, where $s = 64s'+1<r<64(s'+1)+1 = t$, for any positive integer s'. This is, of course, a direct consequence of the way in which the Alternating Halves methods are constructed. Each iteration of the Accelerated Alternating Halves method requires nearly as many operations as does the Accelerated Line Gauss-Seidel. If a step of the Accelerated Alternating Halves method consists of Kn operations which can be done simultaneously, doing the equivalent operations using the Accelerated Line Gauss-Seidel method requires K steps, each consisting of n operations which can be done simultaneously. If n is close to a multiple of 64, which is often the case when $I - 1 = J - 1$ is a multiple of 64, then the percentage of processors used by the two methods does not greatly differ. When n is not close to a multiple of 64, however, $K * \sigma(n/64)$ parallel operations are required by the Accelerated Line Gauss-Seidel method, but the $\sigma(Kn/64)$ parallel operations required by the Accelerated Alternating Halves may

be considerably less. For example, if $n = 96$ and $K = 16$, the Accelerated Line Gauss-Seidel method requires 32 parallel operations to do what the Accelerated Alternating Halves method can do in only 24 parallel operations.

Tables 2 and 3 give the number of parallel operations required by the three methods for a variety of values of $I = J$. Table 2 covers a wide range of values for $I = J$, while table 3 provides a closer look at how the operation counts vary for neighboring values of $I = J$. To aid in evaluating the improvement of the Accelerated Alternating Halves method over the other two methods, the last two columns of each table contain a ratio of the number of parallel operations required by the Accelerated Alternating Halves method to the number of parallel operations required by one of the other methods.

Tables 2 and 3 deal only with the cases where $I = J$. Some operation counts for $I \neq J$ have also been made. All the cases calculated with $I > J$ show only about 1% difference, from the case where J has the value of I, in the percentage of improvement of the Accelerated Alternating Halves method over either of the other two methods.

TABLE 2

Operations Required For Accelerated Point Gauss-Seidel (APGS), Accelerated Line Gauss-Seidel (ALGS), and Accelerated Alternating Halves (AAH) Methods When $33 \leq I(=J) \leq 257$

Operation Type	I(=J)	Number of Operations Per Iteration for APGS	ALGS	AAH	$\frac{AAH}{APGS}$	$\frac{AAH}{ALGS}$
*	33	1024	384	180	.18	.47
+	33	3072	446	210	.07	.47
÷	33	0	32	16		.50
*	49	2304	720	452	.20	.63
+	49	6912	814	522	.08	.64
÷	49	0	48	36		.75
*	65	4096	1022	836	.20	.82
+	65	12288	1086	960	.08	.88
÷	65	0	64	64		1.00
*	81	6400	2158	1388	.22	.64
+	81	19200	2398	1584	.08	.66
÷	81	0	160	100		.62
*	97	9216	2878	2072	.22	.72
+	97	27648	3164	2354	.09	.74
÷	97	0	192	144		.75
*	113	12544	3694	2896	.23	.78
+	113	37632	4028	3282	.09	.81
÷	113	0	224	196		.87
*	129	16384	4348	3844	.23	.88
+	129	49152	4604	4348	.09	.94
÷	129	0	256	256		1.00
*	145	20736	6764	5012	.24	.74
+	145	62208	7340	5652	.09	.77
÷	145	0	432	324		.75
*	161	25600	7996	6328	.25	.79
+	161	67800	8634	7118	.09	.82
÷	161	0	480	400		.83

TABLE 2 - Continued

Operation Type	I(=J)	Number of Operations Per Iteration for APGS	ALGS	AAH	AAH/APGS	AAH/ALGS
*	177	30976	9324	7800	.25	.84
+	177	92928	10026	8758	.09	.87
÷	177	0	528	484		.92
*	193	36864	10362	9416	.26	.91
+	193	110592	10938	10556	.10	.97
÷	193	0	576	576		1.00
*	209	43264	13930	11192	.26	.80
+	209	129792	14970	12532	.10	.84
÷	209	0	832	676		.81
*	225	50176	15674	13116	.26	.84
+	225	150528	16792	14670	.10	.87
÷	225	0	896	784		.88
*	241	57600	17514	15196	.26	.87
+	241	172800	18712	16982	.10	.91
÷	241	0	960	900		.94
*	257	65536	18936	17416	.27	.92
+	257	196608	19960	19448	.10	.97
÷	257	0	1024	1024		1.00

TABLE 3

Operations Required for Accelerated Point Gauss-Seidel (APGS), Accelerated Line Gauss-Seidel (ALGS), and Accelerated Alternating Halves (AAH) Methods When $129 \leq I(=J) \leq 161$.

Operation Type	I(=J)	Number of Operations Per Iteration for APGS	ALGS	AAH	AAH / APGS	AAH / ALGS
*	129	16384	4348	3844	.23	.88
+	129	49152	4604	4348	.09	.94
÷	129	0	256	256		1.00
*	131	16900	5326	3998	.24	.75
+	131	50700	5846	4522	.09	.77
÷	131	0	390	266		.68
*	133	17424	5672	4142	.24	.73
+	133	52272	6200	4682	.09	.76
÷	133	0	396	274		.69
*	135	17956	6026	4282	.24	.71
+	135	53868	6562	4838	.09	.74
÷	135	0	402	282		.70
*	137	18496	6116	4424	.24	.72
+	137	55488	6660	4996	.09	.75
÷	137	0	408	290		.71
*	139	19044	6482	4570	.24	.71
+	139	57132	7034	5158	.09	.73
÷	139	0	414	298		.72
*	141	19600	6576	4724	.24	.72
+	141	58800	7136	5330	.09	.75
÷	141	0	420	308		.73
*	143	20164	6670	4874	.24	.73
+	143	60492	7238	5498	.09	.76
÷	143	0	426	316		.74

TABLE 3 - Continued

Operation Type	I(=J)	Number of Operations Per Iteration for APGS	ALGS	AAH	AAH / APGS	AAH / ALGS
*	145	20736	6764	5012	.24	.74
+	145	62208	7340	5652	.09	.77
÷	145	0	432	324		.75
*	147	21316	7294	5184	.24	.71
+	147	63948	7734	5844	.09	.76
÷	147	0	438	334		.76
*	149	21904	7394	5340	.24	.72
+	149	65712	7840	6018	.09	.77
÷	149	0	444	344		.77
*	151	22500	7496	5504	.24	.73
+	151	67500	7946	6198	.09	.78
÷	151	0	450	352		.78
*	153	23104	7596	5664	.25	.75
+	153	69312	8202	6378	.09	.78
÷	153	0	456	362		.79
*	155	23716	7696	5832	.25	.76
+	155	71148	8310	6568	.09	.79
÷	155	0	462	372		.81
*	157	24336	7796	6002	.25	.77
+	157	73008	8418	6756	.09	.80
÷	157	0	468	382		.82
*	159	24964	7896	6176	.25	.78
+	159	74892	8526	6950	.09	.82
÷	159	0	474	392		.83
*	161	25600	7996	6328	.25	.79
+	161	76800	8634	7118	.09	.82
÷	161	0	480	400		.83

Cases where $J > I$ yield much different results, however. For example, when $I = 65$ and J assumes values ranging from 65 to 257, the percentage of improvement remains almost constant, at the percentage of improvement when $I = J = 65$. Thus, although the number of operations required depends on J, the percentage of improvement of the Accelerated Alternating Halves method over either of the other methods appears to be a function of only the value of I.

If we assume $h = k$, the rate of convergence of the Accelerated Point Gauss-Seidel method is less than the rate of the other two methods. From this and from the calculation of operation requirements, we conclude that the Accelerated Alternating Halves and the Accelerated Line Gauss-Seidel methods are both generally superior to the Accelerated Point Gauss-Seidel in a parallel processor environment. We also note that the Accelerated Alternating Halves method is consistently better than the Accelerated Line Gauss-Seidel. As we have just seen, the degree of improvement varies considerably and tends to zero as I increases. Careful selection of I and J can increase the improvement of the Accelerated Alternating Halves method over the Accelerated Line Gauss-Seidel.

5.2 Parallel versus Nonparallel

In the previous section we saw how the methods compare for particular values of the parameters. The basic assumption there was that the methods would be used on a parallel computer. We now return to the example of section 5.1 to see how the methods compare on a non-parallel computer and how much the operations counts for a method differ using a parallel as compared to a non-parallel machine. As in section 5.1, we assume $I = J = 33$, $h = k = .1$, and that the initial error is to be decreased by a factor of 10^{-3}. The Accelerated Point Gauss-Seidel method requires the following non-parallel operations:

189,440 multiplications, and

189,440 additions.

The Accelerated Line Gauss-Seidel method requires the following non-parallel operations:

156,447 multiplications,

181,407 additions, and

26,686 divisions,

if $\omega\theta_y U_{[i-1]}^{(m+1)}$, $2 \leq i \leq J - 2$, is not stored for use in the next iteration. If this quantity is retained to be used in the calculation of $U^{(m+2)}$, then the Accelerated Line Gauss-Seidel requires the same number of non-parallel

operations as does the Accelerated Alternating Halves:

131,487 multiplications,

176,415 additions, and

26,686 divisions.

Assuming divisions and multiplications require approximately the same time, the latter two methods are slightly faster than the Accelerated Point Gauss-Seidel.

If we compare the non-parallel operation counts with the parallel operation counts of table 1, we see a significant difference. Even if the non-parallel computer is as much as six times faster than the parallel machine, the latter could perform the calculations of the example in about 1/10 the time required by the non-parallel computer.

5.3 Further Improvements

The conclusions of the previous sections about the Accelerated Alternating Halves method are highly dependent on the particular algorithm used to determine operation counts. The goal in the selection of Chapter IV's algorithms was the maximization of processor use. Stone's algorithm appears to be the most efficient technique available for solving 4.3.1 or 4.4.1 with LU decomposition. If LU decomposition is not used, the

most likely approach involves initially finding $A_{1,1}^{-1}$ and then multiplying $A_{1,1}^{-1}$ by the sum of products on the right of 4.3.1 to obtain $U_{[i]}^{(m+1)}$. When this technique is used with both the Accelerated Alternating Halves method and the Accelerated Line Gauss-Seidel method, the Accelerated Alternating Halves is faster than the other method. The large number of multiplications necessary with this technique, however, makes it considerably less efficient than the recursive doubling approach to LU decomposition.

To determine how fully the algorithm of section 4.4 uses the 64 processors, we look at the individual steps. The first step can fully utilize all processors if, for example, $I - 1$ is a multiple of 64 and K is a multiple of 32. A similar result follows for steps 2, 3, and 4 if K is a multiple of 64. When neither $I - 1$ nor K is a power of two, 100% use of the processors is not possible. Because of this latter fact, some improvement of the algorithm would be possible. However, because there are cases where the processors are fully used, significant improvement does not seem likely.

As the above argument indicates, the conclusions about the Accelerated Alternating Halves method are

very dependent on the choice of 64 as the number of processors. When $I - 1$ is greater than 64, for example, the Accelerated Line Gauss-Seidel method is able to use all 64 processors most of the time. Since the Accelerated Alternating Halves method requires almost as many operations as does the Accelerated Line Gauss-Seidel, the difference between their processor use is small in such a case. In the example of section 5.1, the number of processors was greater than I. There we saw that the Accelerated Alternating Halves method requires only about 50% of the number of parallel operations required by the Accelerated Line Gauss-Seidel method. By increasing the number of processors to 256, the common range of I should be less than the number of processors. The Accelerated Alternating Halves method, on the average, allows $K(I-1)$ operations to be done simultaneously, while the Accelerated Line Gauss-Seidel allows only $I - 1$. Thus, the Accelerated Alternating Halves method would keep more of 256 processors busy and would complete an iteration with fewer parallel operations than would the other method. From this argument, we conclude that as the number of available processors increases, the Accelerated Alternating Halves will become a more valuable method.

BIBLIOGRAPHY

Carroll, A.B., and Wetherald, R.T. "Application of Parallel Processing to Numerical Weather Prediction." Journal of the Association for Computing Machinery, 14(July, 1967), 591-614.

Isaacson, E., and Keller, H.B. Analysis of Numerical Methods. New York: John Wiley and Sons, Inc., 1966.

Miranker, W. L. "A Survey of Parallelism in Numerical Analysis." SIAM Review, 13(October, 1971), 524-545.

Rosenfeld, J.L., and Driscoll, G.C. "Solution of the Dirichlet Problem on a Simulated Parallel Processing System." Information Processing 68 Proceedings of the IFIP Congress 68, Edinburgh, Scotland, 5-10 August, 1968, 1, 499-507.

Stone, H.S. "An Efficient Parallel Algorithm for the Solution of a Tridiagonal Linear System of Equations." Journal of the Association for Computing Machinery, 20(January, 1973), 27-38.

Varga, Richard S. Matrix Iterative Analysis. Englewood Cliffs, N.J.: Prentice Hall, Inc., 1962.

APPENDIX

APPENDIX

Bibliography on Parallel Computers and Parallel Programming Techniques

The following bibliography is a survey of the last ten years' publications in the area of parallel computing. The listings have been organized into several subject groups, ranging from parallel programming in numerical analysis to parallel computer architecture. Most of the articles, it should be noted, deal with more than one subject area, but are listed only according to one area of emphasis.

Not included in the listings of the following pages is the book, Parallel Processor Systems, Technologies and Applications.[19] Because this large collection of articles covers a wide range of topics, it is recommended as a convenient source of general information about parallel computing.

[19] L.C. Hobbs, et. al. ed., Parallel Processor Systems, Technologies, and Applications (London: McMillan, 1970).

Existing Parallel Computers

Alsberg, P.; Gaffney, J.; Grossman, C.; Mason, T.; and Westlund, G., A Description of the ILLIAC IV Operating System. Report ILLIAC-IV-212. Urbana, Ill.: University of Ill., 1969.

Barnes, George H.; Brown, Richard M.; Kato, Maso; Kuck, David J.; Slotnick, Daniel L.; and Stokes, Richard A. "The ILLIAC IV Computer." IEEE Transactions on Computers, C-27 (August, 1968), 746-757.

Burroughs Corporation. ILLIAC-IV Systems Characteristics and Programming Manual. Document No. 66000A, June, 1969.

Chen, F.T. Linear Program Implementation in ILLIAC-IV: Revised Simplex Method. Report ILLIAC-IV-177. Urbana, Ill.: University of Ill., 1968.

Davis, R.L. "The ILLIAC IV Processing Element." IEEE Transactions on Computers, C-28 (September, 1969), 800-816.

Gallagher, R.J. "An Experimental Parallel Logic Pattern Recognition Machine." Unpublished thesis, State University of New York at Stony Brook, 1971. (Available from Univ. Microfilms, Ann Arbor, Mich., Order No. 72—6380.)

Graham, W.R. The Parallel, Pipeline and Conventional Computer. Rand Corporation Report No. P-4221. Santa Monica, Cal.: Rand Corporation. 1969. (Available from CFSTI, Springfield, Va.)

Graham, W.R. "The Parallel and the Pipeline Computers." Datamation, 16 (April, 1970), 68-71.

Kato, M.; Koga, Y.; and Naemura, K. Diagnostic Test Patterns and Sequences for ILLIAC IV Processing Element. Report ILLIAC-IV-180. Urbana, Ill.: University of Ill., 1968.

Knapp, M.A. "Parallel Processing Computer Systems." Government Reports Announcement, 72 (May 25, 1972), 97.

Knapp, M.A. "Evolution of Computer Systems to Perform Parallel Processing." Star, 5 (April, 1967), 1043-1044.

McIntyre, D.E. "An Introduction to ILLIAC IV Computer." Datamation, 16 (April, 1970), 60-67.

Matsushita, Y. Diagnostic Sequence Generator for ILLIAC IV Processing Element. Report ILLIAC-IV-187. Urbana, Ill.: University of Ill., 1968.

Schwartz, J. "Large Parallel Computers." Journal of the Association for Computing Machinery, 13 (January, 1966), 25-32.

Schwartz, J. "AT-1 Parallel Computer--Second Preliminary Version." Unpublished report, New York University, 1964.

Stern, L. "PDP-11 Parallel Multiprocessor." Decuscope, 9(1970),7-8.

Stokes, R.A. "ILLIAC IV Route to Parallel Computers." Electronic Design, 26(December, 1967), 64-69.

Thurber, K.J., and Berg, R.O. "Applications of Associative Processor." Computer Design, 10 (November, 1971), 103-110.

Westinghouse Defense and Space Center. Parallel Network Computer (SOLOMON) Applications Analyses. August, 1964.

Westinghouse Electric Corporation. SOLOMON Parallel Network Processor. Internal Report, 1962.

Parallel Programming

Anderson, J.P. "Program Structures for Parallel Processing." Communications of the ACM, 8 (December, 1965), 786-788.

Ashcroft, E., and Manna, Z. "Formalization of Properties of Parallel Programs." Machine Intelligence, 6 (July, 1970), 17-41.

Baer, J.L., and Russell, E.C. "Modelling and Scheduling of Computer Programs for Parallel Processing Systems." In: Second Conference on Application of Simulation, New York, 2-4 December, 1968. New York: Institute of Electrical and Electronics Engineers, Inc., 1968. 278-281.

Bekishev, G.A. "Parallelization of Computational Algorithms." Star, 5 (July 8, 1967), 2287.

Bernstein, A.J. "Analysis of Programs for Parallel Processing." IEEE Transactions on Electronic Computers, EC-15 (October, 1966), 757-763.

Bingham, H.W., and Seward, J.W. "Plan for Detection of Parallelism in Computer Programs." United States Government Research and Development Reports, 67 (October 10, 1967), 75.

Fateman, Richard J. "Optimal Code for Serial and Parallel Computers." Communications of the ACM, 12 (December, 1969), 694-695.

Gilmore, P.A. "Structuring of Parallel Algorithms." Journal of the Association for Computing Machinery, 15 (April, 1968), 176-192.

Gonzales, M.J., Jr., and Ramamoorthy, C.V. "Program Suitability for Parallel Processing." IEEE Transactions on Computers, C-20 (June, 1971), 647-654.

Gonzales, M.J., Jr., and Ramamoorthy, C.V. "Recognition of Parallel Processable Streams in Multiprocessor Computers." In: Twenty-first Annual Southwestern IEEE Conference and Exhibition, San Antonio, Texas, 23-25 April, 1969. New York: Institute of Electrical and Electronics Engineers, Inc., 1969.

Han, J.C., and Han, C. "Tree Height Reduction for Parallel Processing of Blocks of Fortran Assignment Statements." Government Reports Announcement, 72 (May 25, 1972), 98.

Karp, R.M., and Miller, R.E. "Parallel Program Schemata: A Mathematical Model for Parallel Computations." In: IEEE Conference Record of Eighth Annual Symposium on Switching and Automata Theory, October, 1967. New York: Institute of Electrical and Electronics Engineers, Inc., 1967. 55-61.

Kotov, V.E., and Narinyani, A.S. "On Transformation of Sequential Programs into Asynchronous Parallel Programs." Information Processing 68 Proceedings of the IFIP Congress 68, Edinburgh, Scotland, 5-10 August, 1968, 1, 351-357.

Kuck, J. "ILLIAC IV Software and Application Programming." IEEE Transactions on Computers, C-17 (August, 1968), 758-770.

Larson, R.E.; Richardson, M.H.; and Bree, D.W., Jr. "Dynamic Programming in Parallel Computers." In: Proceedings of the Fourth Hawaii International Conference on System Sciences, Honolulu, Hawaii, 12-14 January, 1971. Hollywood, Cal.: Western Periodicals Co.,1971.256-9.

Ramamoorthy, C.V., and Gonzales, M.J., Jr. "Subexpression Ordering in the Execution of Arithmetic Expressions." Communications of the ACM, 14 (July, 1971), 479-485.

Ramamoorthy, C.V., and Gonzales, M.J., Jr. "A Survey of Techniques for Recognizing Parallel Processable Streams in Computer Programs." AFIPS Conference Proceedings 1969 Fall Joint Computer Conference, Las Vegas, 18-20 November, 1969, 35,1-15.

Rosenfeld, J.L. "A Case Study in Programming for Parallel-Processors." Communications of the ACM, 12 (December, 1969), 645-655.

Schwartz, J. "Algorithms in Parallel Computation." Unpublished report, New York University, 1964.

Squire, S.J. "Programming and Design Considerations of of a Highly Parallel Computer." AFIPS Conference Proceedings 1963 Spring Joint Computer Conference, Detroit, May, 1963, 23, 359-400.

Stone, H.S. "Parallel Processing with the Perfect Shuffle." IEEE Transactions on Computers, C-20 (February, 1971), 153-161.

Wishner, R.P.; Downs, H.R.; and Shechter, J. "Real-Time Computing Techniques for Parallel Processors." Information Processing 71 Proceedings of IFIP Congress 71, Ljubljana, Yugoslavia, 23-28 August, 1971, 1, 704-710.

Parallel Computing in Numerical Analysis

Dorn, W.S. "Generalizations of Horner's Rule for Polynomial Evaluation." IBM Journal of Research and Development, 6 (April, 1962), 239-245.

Hellerman, H. "Parallel Processing of Algebraic Expressions." IEEE Transactions on Electronic Computers, EC-15 (February, 1966), 82-91.

Miranker, W.L. "A Survey of Parallelism in Numerical Analysis." SIAM Review, 13 (October, 1971), 524-545.

Miranker, W. L. "Parallel Methods for Approximating the Root of a Function." IBM Journal of Research and Development, 13 (May, 1969), 297-301.

Shedler, G.S. "Parallel Numerical Methods for Solutions of Equations." Communications of the ACM, 10 (May, 1967), 286-290.

Winograd, Shumel. "Parallel Iteration Methods." In: Complexity of Computer Computations. Edited by R.E. Miller and J.W. Thatcher. New York: Plenum Publ., 1972.

Parallel Computing in Solution of Ordinary Differential Equations

Downes, H.R. "Parallel Computation of a Differential Equation." SIAM Review, 13 (April 1971), 265.

Miranker, Willard L., and Liniger, Werner. "Parallel Methods for the Numerical Integration of Ordinary Differential Euqations." Mathematics of Computation, 21 (July, 1967), 303-320.

Nievergelt, J. "Parallel Methods for Integrating Ordinary Differential Equations." Communications of the ACM, 7 (December, 1964), 731-733.

Parallel Computing in Solution of Partial Differential Equations

Carroll, A. B., and Wetherald, R.T. "Application of Parallel Processing to Numerical Weather Prediction." Journal of the Association for Computing Machinery, 14(July, 1967), 591-614.

Gilmore, P.A. "Numerical Solution of Partial Differential Equations by Associative Processing." AFIPS Conference Proceedings 1971 Fall Joint Computer Conference, Las Vegas, 16-18 November, 1971, 34, 411-418.

Rosenfeld, J.L., and Driscoll, G.C. "Solution of the Dirichlet Problem on a Simulated Parallel Processing System." Information Processing 68 Proceedings of the IFIP Congress 68, Edinburgh, Scotland, 5-10 August, 1968, 1, 499-507.

Parallel Computing in Linear Algebra

Matsushita, Y. Sparse Matrix Inversion on ILLIAC IV Report ILLIAC-IV-193. Urbana, Ill.: University of Ill., 1968.

Pease, Marshall C. "Matrix Inversion Using Parallel Processing." Journal of the Association for Computing Machinery, 14 (October, 1967), 757-764.

Sameh, A.H. "On Jacobi and Jacobi-Like Algorithms for a Parallel Computer." Mathematics of Computation, 25 (July, 1971), 579-590.

Stone, H.S. "An Efficient Parallel Algorithm for the Solution of a Tridiagonal Linear System of Equations." Journal of the Association for Computing Machinery, 20 (January, 1973), 27-38.

Compilers and Languages for Parallel Computers

Allard, R.W.; Wolf, K.A.; and Zemlin, R.A. "Some Effects of the 6600 Computer on Language Structures." Communications of the ACM, 7 (February, 1964), 112-119.

Baer, J.L., and Bovet, D.P. "Compilation of Arithmetic Expressions for Parallel Computations." Information Processing 68 Proceedings of the IFIP Congress 68, Edinburgh, Scotland, 5-10 August, 1968, 1, 340-347.

Betourne, C.; Ferrie, J.; Kaiser, C.; Krakowiak, S.; and Mossiere, J. "System Design and Implementation Using Parallel Processes." Information Processing 71 Proceedings of the IFIP Congress 71, Ljubljana, Yugoslavia, 23-28 August, 1971, 1, 345-352.

Betourne, C.; Boulenger, J.; Ferrie, J.; Kaiser, C.; Krakowiak, S.; and Mossiere, J. "Process Management and Resource Sharing in the Multiaccess System ESOPE." Communications of the ACM, 13 (December, 1970), 727-733.

Chamberlin, D.D. "Parallel Implementation of a Single-Assignment Language." Unpublished thesis, Stanford University, 1971. (Available from Univ. Microfilms, Ann Arbor, Mich., Order No. 71-23494.)

Chamberlin, D.D. "The 'Single-Assignment' Approach to Parallel Processing." AFIPS Conference Proceedings 1971 Fall Joint Computer Conference, Las Vegas, 16-18 November, 1971, 39, 263-269.

Dennis, Jack B., and Van Horn, Earl C. "Programming Semantics for Multiprogrammed Computations." Communications of the ACM, 9 (March, 1966), 143-155.

Ellis, Clarence A. "Parallel Compiling Techniques." In: Proceedings of the 1971 Annual Conference Association for Computing Machinery, Chicago, 3-5 August, 1971. New York: Association for Computing Machinery, 1971. 508-519.

Gosden, J.A. "Explicit Parallel Processing Description and Control in Programs for Multi- and Uni-Processor Computers." AFIPS Conference Proceedings 1966 Fall Joint Computer Conference, San Francisco, 7-10 November 1966, 29, 651-660.

Lawrie, D.H. GLYPNIR: A List Processing Language for ILLIAC IV. Report ILLIAC-IV-322. Urbana, Ill.: University of Ill., 1969.

Nielsen, N.R. "Controlling a Real-Time Parallel Processing Computer." Simulation, 17 (September, 1971), 97-103.

Parallel Processor Simulation

Findler, N.V. "On a Computer Language which Simulates Associative Memory and Parallel Processing." Cybernetics, 10, No. 4 (1967), 229-254.

Katz, Jesse H. "Simulation of a Multiprocessor Computer System." AFIPS Conference Proceedings 1966 Spring Joint Computer Conference, Boston, April, 1966, 28, 127-139.

Parallel Computer Architecture

Bredt, T. H. "Analysis of Parallel Systems." IEEE Transactions on Computers, C-20 (November, 1971), 1403-1407.

Critchlow, A.J. "Generalized Multiprocessing and Multiprogramming Systems." AFIPS Conference Proceedings 1963 Fall Joint Computer Conference, Las Vegas, November, 1963, 24, 107-125.

Dennis, J.B. "Programming Generality, Parallelism and Computer Architecture." Information Processing 68 Proceedings of the IFIP Congress 68, Edinburgh, Scotland, 5-10 August, 1968, 1, 484-492.

Estrin, G.· Russell, B.; Turn, R.; and Bibb, J. "Parallel Processing in a Restructurable Computer System." IEEE Transactions on Electronic Computers, EC-12 (December, 1963), 747-755.

Fateman, R.J. "Optimal Code for Serial and Parallel Computation." Communications of the ACM, 12 (December, 1969), 694-695.

Fuller, R.H. "Associative Parallel Processing." Computer Design, 6 (December, 1967), 43-46.

Haberman, A. Nico. "Synchronization of Communicating Processes." Communications of the ACM, 15 (March, 1972), 171-176.

Huttenhoff, J.H., and Shively, R.R. "Arithmetic Unit of a Computing Element in a Global, Highly Parallel Computer." IEEE Transactions on Computers, C-18 (August, 1969), 695-698.

Koczela, L.J., and Wang, G.Y. "The Design of a Highly Parallel Computer Organization." IEEE Transactions on Computers, C-18 (June, 1969), 520-529.

Kuck, D.J. "A Preprocessing High-Speed Memory System." IEEE Transactions on Computers, C-19 (September, 1970), 793-802.

Langsford, A. "Reliable Inter-Process Communication." In: Proceedings of the First European Seminar on Computing with Real-Time Systems, Harwell, Berks, England, 1971. London, England: Transcripta Books, 1972. 123-126.

Nuspl, S.J., and Johnson, M.D. "The Effect of Input/Output Characteristics on the Performance of a Parallel Processor." In: Fifth Annual 1971 IEEE International Computer Society Conference on Hardware, Software, Firmware, and Trade-Offs, Boston, 22-24 September, 1971. New York:Institute of Electrical and Electronics Engineers, Inc., 1971. 127-128.

Pomerene, J. H. "An Approach to Parallel Processing." Information Processing 1965 Proceedings of IFIP Congress 65, New York, 24-29 May, 1965, 2, 322.

Popova, G.M., and Prangishvili, I.V. "Associative Parallel Processor for Grouped Processing of Data." Automation and Remote Control, 30 (January, 1972), 152-162.

Shechter, J. "System Design Criteria for the Use of ILLIAC IV in Real-Time Environment." In: Proceedings of the Fourth Hawaii International Conference on System Sciences, Honolulu, 12-14 January, 1971. North Hollywood, Cal.: Western Periodicals,Co., 1971. 260-262.

Shemier, D., and Gupta, S.C. "A Simplified Analysis of Processor 'Look-Ahead' and Simultaneous Operation of a Multi-Module Main Memory." IEEE Transactions on Computers, C-18(January 1969), 64-71.

Sofer, D., and Sproul, W.W. III. "Parallel Pipeline Organization of Execution Unit." IBM Technical Disclosures Bulletin, 14 (March, 1972), 2930-2933.

Stone, H.S. "Parallel Processing with the Perfect Shuffle." IEEE Transactions on Computers, C-20 (February, 1971), 153-161.

Parallel Computer Theory

Baer, J.L.E., and Estrin, G. "Bounds for Maximum Parallelism in a Bilogic Graph Model of Computations." IEEE Transactions on Computers, C-18 (November, 1969), 1012-1014.

Baer, J.L.E.; Bovet, D.P.; and Estrin, G. "Legality and Other Properties of Graph Models of Computations." Journal of the Association for Computing Machinery, 17 (July, 1970), 543-544.

Baker, K.R., and Morton, A.G. "Scheduling with Parallel Processors and Linear Delay Costs." Bulletin of Operations Research Society of America, 20, Supplement 1 (Spring, 1972), B/86.

Banks, E.R. "Universality in Cellular Automata." In: IEEE Conference Record of 1970 Eleventh Annual Symposium on Switching and Automata Theory, Santa Monica, Cal., 28-30 October, 1970. New York: Institute of Electrical and Electronics Engineers, Inc., 1970. 194-215.

Barskiy, A.B. "Dynamic Parallel Ordering of Computations." Cybernetics, 9 (September-October, 1971), 879-883.

Berkling, K.J. "A Computing Machine Based on Tree Structures." IEEE Transactions on Computers, C-20 (April, 1971), 404-418.

Bredt, T.H., and McCluskey, E.J. A Model for Parallel Computer Systems. Report NASA-CR-110465. Palo Alto, Cal.: Stanford University, April, 1970. (Available from CFSTI, Springfield, Va.)

Budnik, P., and Kuck, D.J. "The Organization and Use of Parallel Memories." IEEE Transactions on Computers, 20 (December, 1971), 1566-1569.

Burkhardt, W.H. "Automation of Program Speed-Up on Parallel Processor Computers." Computing, 3, No. 4 (1968), 297-310.

Chang, Shi-Kuo. "On the Parallel Computation of Local Operations." In: Proceedings of the Third Annual ACM Symposium on Theory of Computing, Shaker Heights, Ohio, 3-5 May, 1971. New York: Association for Computing Machinery, 1971. 101-115.

Cohen, D. "On Parallel Processing Networks." In: Proceedings of the Third Hawaii International Conference on System Sciences, Honolulu, Hawaii, 14-16 January 1970. Hollywood, California: Western Periodicals Co., 1970. 580-583.

Cohen, D. "A Parallel Process Definition and Control System." AFIPS Conference Proceedings 1968 Fall Joint Computer Conference, San Francisco, Cal., 9-11 December, 1968, 33, pt. 2, 1043-1050.

Deutsch, E.S. "On Parallel Operations on Hexagonal Arrays." IEEE Transactions on Computers, C-19 (October, 1970), 982-983.

Gilbert, P., and Chandler, W.J. "Interference Between Communicating Parallel Processes." In: Proceedings of the Fourth Hawaii International Conference on System Sciences, Honolulu, Hawaii, 12-14 January, 1971. Hollywood, Cal.: Western Periodicals Co., 1971. 399-401.

Hamacher, V.G. "A Class of Parallel Processing Automata." Unpublished thesis. Syracuse University, New York. (Available from Univ. Microfilms, Ann Arbor, Mich. Order No. 69-7744.)

Hawkins, J.K. "A Parallel Computer Organization and Mechanizations." IEEE Transactions on Electronic Computers, EC-12 (June, 1963), 251-262.

Hebalkar, P.G. "A Graph Model for Analysis of Deadlock Prevention in Systems with Parallel Computations." Information Processing 71 Proceedings of the IFIP Congress 71, Ljubljana, Yugoslavia, 23-28 August, 1, 498-503.

Karp, R. M.; Miller, R.E.; and Winograd, S. "Two Graph-Theoretic Studies of Parallel Computation." In: Theory of Graphs--International Symposium, Rome, 5-9 July, 1966. Paris: Dunod, 1967. 193-200.

Karp, R.M.; Miller, R.E. "Properties of a Model for Parallel Computations: Determinacy, Terminations, Queueing." SIAM Journal on Applied Mathematics, 14 (November, 1966), 1390-1411.

Keller, Robert M. "On Maximally Parallel Schemata." In: IEEE Conference Record of 1970 Eleventh Annual Symposium on Switching and Automata Theory, Santa Monica, Cal., 28-30 October, 1970. New York: Institute of Electrical and Electronics Engineers, Inc., 1970. 32-50.

Korn, G.A. "Back to Parallel Computation: Proposal for a Completely New On-Line Simulation System Using Standard Minicomputers for Low-Cost Multiprocessing." Simulation, 19 (August, 1972), 37-45.

Kotov, V.E., and Narin'yani, A.S. "Asynchronous Processes and the Computer Memory." Cybernetics, 2 (May-June, 1966), 54-61.

Mommens, J.H., and Wesley, M.A. "Masking Technique for Control of an Associative Parallel Processor." IBM Technical Disclosure Bulletin, 14 (June, 1971), 125-127.

Murtha, J.C. "Highly Parallel Information Processing Systems." Advances in Computers, 7 (1966), 1-116.

Ofman, Yu. P. "On a Parallel Machine." Problems in Information Transmission, 4 (Fall, 1968), 46-48.

Patil, S. "An Abstract Parallel Processing System," Unpublished S.M. thesis. Massachusetts Institute of Technology Department of Electrical Engineering, 1967.

Reiter, Raymond. "Scheduling Parallel Computations." Journal of the Association for Computing Machinery, 15 (October, 1968), 590-599.

Rodriguez, Beza J.E. "A Graph Model for Parallel Computations." Unpublished Ph.D. thesis, Massachusetts Institute of Technology Department of Electrical Engineering, 1967.

Rothkopf, M.H. "Scheduling Independent Tasks on Parallel Processors." Management Science, 12 (January, 1966), 437-447.

Schwartz, E.S. "An Automatic Sequencing Procedure with Application to Parallel Programming." Journal of the Association for Computing Machinery, 8 (October, 1961), 513-537.

Seeber, R., and Lindquist, A. "Associative Logic for Highly Parallel Systems." AFIPS Conference Proceedings 1963 Fall Joint Computer Conference, Las Vegas, November, 1963, 24, 489-493.

Shooman, William. "Parallel Computing with Vertical Data." In: Proceedings of the Eastern Joint Computer Conference, 13-15 December, 1960. New York: Eastern Joint Computer Conference, 1960. 111-115.

Slotnick, D.L. "Parallel and Concurrent Computer Systems Symposium Summary." Information Processing 1965 Proceedings of the IFIP Congress 65, New York, 24-29 May, 1965, 1, 319-322.

Tjaden, G.S., and Flynn, M.J. "Detection and Parallel Execution of Independent Instructions." IEEE Transactions on Computers, C-19 (October, 1970), 889-895.

Wilkerson, L.J. "A Simple Approach for Scheduling on Single and Parallel Processors." In: Proceedings of the Fourth Hawaii International Conference on System Sciences, Honolulu, Hawaii, 12-14 January, 1971. Hollywood, Cal.: Western Periodicals Co., 1971, 338-340.