ITERATIVE SOLUTION OF TRIDIAGONAL SYSTEMS ON PARALLEL OR VECTOR COMPUTERS

J. F. Traub

Department of Computer Science
Carnegie-Mellon University
Pittsburgh, Pennsylvania

January, 1973
Revised May, 1973

To appear in Complexity of Sequential and Parallel Numerical Algorithms, Academic Press, 1973.

Part of this work was performed under the auspices of the U.S. Atomic Energy Commission while the author was a consultant at Lawrence Livermore Laboratory. Part was performed while the author was a visiting scientist at the National Center for Atmospheric Research. The work was also supported by the National Science Foundation under Grant GJ-32111 and the Office of Naval Research under Contract N00014-67-A-0314-0010, NR 044-422.

ABSTRACT

We study the iterative solution of a tridiagonal linear system of size m on a parallel or vector computer. Such systems arise commonly in the numerical solution of partial differential equations.

The Gauss algorithm takes time linear in m. We introduce a Parallel Gauss iteration and show it can be used to solve a tridiagonal system in time independent of m. Furthermore, the error norm is reduced at each iteration step. A Parallel LR decomposition is also defined.

Parallel Gauss is based on "multiplicative splitting". We introduce parallel algorithms based on "additive splitting". These are Jacobi, JOR, Parallel Gauss-Seidel and Parallel SOR.

We compare these parallel algorithms on a model problem and conclude that JOR, Gauss-Seidel, and SOR are not competitive. If the matrix has only "limited" diagonal dominance, Parallel Gauss is far superior to Jacobi. If the matrix is very diagonally dominant, Jacobi is somewhat better than Parallel Gauss.

TABLE OF CONTENTS

1. INTRODUCTION*

We study the solution of a tridiagonal linear system $A\underline{u} = \underline{v}$ on a parallel or vector computer. Tridiagonal and block tridiagonal systems arise commonly in the numerical solution of partial differential equations. We shall report on the extension of our results to block tridiagonal systems in a later paper. Stronger results are achieved in the special case of constant diagonals; we do not report these results here.

A substantial number of tridiagonal systems have to be solved in the course of solving a partial differential equation. In regions where the coefficients of the partial differential equations are not rapidly varying, the coefficient matrices of successive tridiagonal systems will enjoy the same property. This makes the iterative solution of the tridiagonal systems attractive. Final iterates of the previous system can be taken as initial iterates of the current system.

Alternatively, tridiagonal systems can be solved by parallel direct methods (Stone [72], Kogge and Stone [72]). We expect that both parallel direct and parallel iterative methods will prove useful.

Our interest in parallel algorithms is threefold.

1. Parallel algorithms are needed for some of the computers now becoming available (Carnegie-Mellon C.mmp,

*Part of this paper was first presented at a Colloquium at the National Center for Atmospheric Research in July, 1972.

Control Data Corporation STAR, ILLIAC IV, Texas Instrument ASC).

2. Semiconductor technology is near the point where a small processor can fit in a single integrated-circuit package. Already, large memory arrays and microprocessors are being manufactured on a single chip. Drastic cost reductions are accompanying this miniaturization and computer systems with a very large number of processors are becoming practical.

3. Constructive mathematics has always been sequential. The possibility of concurrent processing is opening up new worlds of possible algorithms.

The parallel algorithms introduced here can be run on single instruction stream multiple data stream (SIMD) or multiple instruction stream multiple data stream (MIMD) machines. A model of a SIMD machine is given by Kogge and Stone [72]. ASC, ILLIAC IV, and STAR are examples of SIMD machines while C.mmp is an example of a MIMD computer.

Let the size of the linear system be m. The time estimates presented in this paper assume that the number of processors (or the length of the vectors for a vector computer such as STAR) is also m. If only p processors are available, time estimates should be multiplied by $\lceil m/p \rceil$. We neglect the overhead needed to load vectors or due to using m processors. Such overhead should not be ignored in applying our conclusions to a particular machine.

We summarize the results of this paper. For tridiagonal systems, the Gauss algorithm takes time linear in m. We introduce a Parallel Gauss iteration which permits us to solve a tridiagonal system in time independent of m. The time depends only on the diagonal dominance enjoyed by the matrix, the initial errors, and the final accuracy desired. Furthermore, the error norm is reduced at each iteration step.

Parallel LR decomposition and a Parallel Gauss algorithm are defined in Section 2. In Section 3 we show that the rate of convergence of Parallel Gauss depends on the zeros of a quadratic polynomial which can be easily calculated from the given matrix A. Theorem 3.3 gives bounds on the quantities calculated during Parallel Gauss while Theorem 3.4 and Theorem 3.5 give bounds on the error norms. Corresponding results for the Parallel LR decomposition may be found in Section 4.

Parallel Gauss is based on "multiplicative splitting". In Section 5 we introduce parallel algorithms based on additive splitting. These are Jacobi (Jacobi is a parallel algorithm), JOR, Parallel Gauss-Seidel, and Parallel SOR. In Section 6 we compare these parallel algorithms on a two-parameter model problem and conclude that JOR, Gauss-Seidel, and SOR are not competitive. If the matrix has only "limited" diagonal dominance, Parallel Gauss is far superior to Jacobi. If the matrix is very diagonally dominant, Jacobi is somewhat better than Parallel Gauss. Both are superior to Sequential Gauss under the assumptions of the paper. Numerical experimentation is reported in the concluding section.

2. PARALLEL LR AND PARALLEL GAUSS ALGORITHMS

We begin by deriving the Sequential LR and Sequential Gauss algorithms in matrix form. We assume that the diagonal elements of A are non-zero. It is no restriction to assume the diagonal elements are normalized to unity. Let

$$A = A_L + I + A_R = LR, \tag{2.1}$$

$$L = A_L + D,\ R = I + E, \tag{2.2}$$

where D is a diagonal matrix, A_L is a matrix whose only non-zero elements are those on the first subdiagonal, A_R and E are matrices whose only non-zero elements are those on the first superdiagonal. Since $A_L E$ is diagonal,

$$D + A_L E = I, \tag{2.3}$$

$$DE = A_R. \tag{2.4}$$

Hence

$$(I - A_L E)E = A_R. \tag{2.5}$$

To solve $A\underline{u} = \underline{v}$, we write

$$(A_L + D)(I + E)\underline{u} = \underline{v}.$$

Define $\underline{f}$ by $(A_L + D)\underline{f} = \underline{v}$. Hence

$$(I + A_L(I - E))\underline{f} = \underline{v} \tag{2.6}$$

and

$$(I + E)\underline{u} = \underline{f}. \tag{2.7}$$

Let

$$A = \begin{pmatrix} 1 & s_1 & & & & \\ t_2 & 1 & s_2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & t_{m-1} & 1 & s_{m-1} \\ & & & & t_m & 1 \end{pmatrix}. \tag{2.8}$$

For convenience, define $t_1 = s_m = 0$. Let

$$L = \begin{pmatrix} d_1 & & & & \\ t_2 & d_2 & & & \\ & \ddots & \ddots & & \\ & & \ddots & \ddots & \\ & & & t_{m-1} & d_{m-1} \\ & & & & t_m \quad d_m \end{pmatrix}, \tag{2.9}$$

$$R = \begin{pmatrix} 1 & e_1 & & & \\ & 1 & e_2 & & \\ & & \ddots & \ddots & \\ & & & \ddots & \ddots \\ & & & & 1 \quad e_{m-1} \\ & & & & 1 \end{pmatrix}. \tag{2.10}$$

The sequential LR decomposition is defined by

Sequential LR

$$(I - A_L E)E = A_R,$$

$$D = I - A_L E.$$

In components,

$$d_1 = 1, \quad e_1 = s_1,$$

$$d_j = 1 - t_j e_{j-1}, \quad j = 2,\ldots,m,$$

$$e_j = \frac{s_j}{d_j}, \quad j = 2,\ldots,m-1.$$

The Sequential Gauss algorithm for the solution of $A\underline{u} = \underline{v}$ is defined by equations (2.5) - (2.7). We repeat the definition here.

Sequential Gauss

(2.11) $(I - A_L E)E = A_R,$

(2.12) $(I + A_L(I - E))\underline{f} = \underline{v},$

(2.13) $(I + E)\underline{u} = \underline{f}.$

In components,

$$e_1 = s_1, \; e_j = \frac{s_j}{1-t_j e_{j-1}}, \quad j = 2,\ldots,m-1,$$

$$f_1 = v_1, \; f_j = \frac{v_j - t_j f_{j-1}}{1-t_j e_{j-1}}, \quad j = 2,\ldots,m,$$

$$u_m = f_m, \; u_j = f_j - e_j u_{j+1}, \quad j = m-1,\ldots,1.$$

The component representation is well-known (Young [71, Section 14.2]). Clearly the LR and Gauss algorithms are sequential. We now define a Parallel LR and two Parallel Gauss iterations.

Parallel LR

Let $E^{(0)}$ be given. Then

(2.14) $(I - A_L E^{(i-1)})E^{(i)} = A_R, \quad i = 1,\ldots,M,$

(2.15) $D = I - A_L E^{(M)}.$

Let the non-zero elements of $E^{(i)}$ be labeled $e_1^{(i)},\ldots,e_{m-1}^{(i)}$. Then in components,

$$e_j^{(0)} \text{ given}, \quad j = 2,\ldots,m-1,$$

$$s_i^{(1)} = s_i, \quad i = 0,\ldots,M,$$

$$e_j = \frac{s_j}{1 - t_j e_{j-1}}, \quad j = 2,\ldots,m-1. \tag{2.16}$$

$$d = 1,$$

$$d = d_j + d_{j-1} e_j, \quad j = 2,\ldots,m. \tag{2.17}$$

The Parallel Gauss algorithm for the solution of Au = v is defined by

Parallel Gauss

Let $E^{(1)}$, $f^{(1)}$, $u^{(1)}$ be given. Then

$$(I - A_1 E^{(1)}) E^{(2)} = A_2, \quad i = 1,\ldots,M, \tag{2.18}$$

$$(I - A_1 E^{(1)}) f^{(2)} = v - A_1 f^{(1)}, \quad i = 1,\ldots,N, \tag{2.19}$$

$$u_l = r_l - E_l u_{l+1}, \quad l = 1,\ldots,P. \tag{2.20}$$

In components,

$$e_1 \text{ given}, \quad j = 2,\ldots,m-1,$$

$$e_{j1} = \quad i = 0,\ldots,M,$$

$$e_j = \frac{s_j}{t_j (1 - e_{j-1})}, \quad j = 2,\ldots,m-1. \tag{2.21}$$

$$f_j \text{ given}, \quad j = 2,\ldots,m,$$

$$f_1^{(2)} = v_i, \quad i = 0,\ldots,N,$$

$$f_j^{(2)} = \frac{v_i - t_j f_{j-1}}{1 - t_j e_j}, \quad i = 1,\ldots,N, \quad j = 2,\ldots,m. \tag{2.22}$$

$$u_j^{(0)} \text{ given, } j = 1,\ldots,m-1,$$

$$u_m^{(i)} = f_m, \; i = 0, 1,\ldots,P,$$

$$(2.23) \quad u_j^{(i)} = f_j^{(N)} - e_j^{(M)} u_{j+1}^{(i-1)}, \; i = 1,\ldots,P, \; j = 1,\ldots,m-1.$$

This is a parallel algorithm since all components of $E^{(i)}$ may be computed in parallel from the components of $E^{(i-1)}$. The following variation of the Parallel Gauss algorithm can also be defined. It differs from the algorithm defined above in that the E, $\underline{f}$, $\underline{u}$ iterations are done simultaneously.

Parallel Gauss Variation

Let $E^{(0)}$, $\underline{f}^{(0)}$, $\underline{u}^{(0)}$ be given. Then for $i = 1,\ldots,M$,

$$(I - A_L E^{(i-1)}) E^{(i)} = A_R,$$

$$(I - A_L E^{(i)}) \, \underline{f}^{(i)} = \underline{v} - A_L \underline{f}^{(i-1)},$$

$$\underline{u}^{(i)} = \underline{f}^{(i)} - E^{(i)} \underline{u}^{(i-1)}.$$

In this paper we shall not analyze this variation.

3. ANALYSIS OF THE PARALLEL GAUSS ALGORITHM

The rate of convergence of Parallel Gauss depends on the zeros of a quadratic polynomial. Properties of the zeros are given by Theorem 3.1. Sufficient conditions for Sequential and Parallel Gauss to be well-defined and bounds on quantities occurring in these algorithms are covered by Theorems 3.2 and 3.3 while Theorems 3.4 and 3.5 give bounds on the error norms.

At the end of the Section we analyze norm reduction, prove monotonicity properties of the iteration rates of convergence, briefly discuss stability, and simplify the main results for an interesting special case.

We use the vector norm

$$||\underline{x}|| = ||\underline{x}||_\infty = \max_{j=1,\ldots,m} |x_j|$$

and the matrix norm

$$||A|| = ||A||_\infty = \max_{i=1,\ldots,m} \sum_{j=1}^{m} |a_{ij}|.$$

Note that for a matrix such as A_R, which has only one non-zero diagonal,

$$||A_R|| = \max_{j=1,\ldots,m} |s_j|$$

which is the norm of a vector with components $s_1,\ldots,s_m$.

Let

$$(3.1) \quad s = ||A_R|| = \max_{j=1,\ldots,m} |s_j|,$$

$$(3.2) \quad t = ||A_L|| = \max_{j=1,\ldots,m} |t_j|.$$

If s or t were zero, A would be bidiagonal rather than tridiagonal. Hence we assume throughout this paper that $st > 0$.

Define

$$(3.3) \quad f(x) = tx^2 - x + s.$$

The zeros of this quadratic play a key role in the analysis of the Parallel Gauss algorithm. Some of the properties of the zeros are covered by the following theorem, additional properties are given at the end of this Section and at the beginning of Section 4.

THEOREM 3.1. Let $s+t < 1$. Then $F(x) = tx^2 - x + s$ has two real distinct zeros

$$a_- = \frac{1-\sqrt{1-4st}}{2t}, \quad a_+ = \frac{1+\sqrt{1-4st}}{2t},$$

and

$$(3.4) \quad 0 < s < a_- < s + t < 1 < a_+,$$

$$(3.5) \quad t < a_+^{-1} < s + t < 1,$$

$$(3.6) \quad st < \frac{a_-}{a_+} < (s+t)^2 < 1.$$

PROOF. Since

$$1 > (s+t)^2 \geq 4st,$$

the zeros are real and distinct. The inequalities (3.4) follow from

$$f(s) > 0,\ f(s+t) < 0,\ f(1) < 0,\ f(\infty) > 0.$$

The inequalities (3.5) follow from

$$f(\frac{1}{s+t}) < 0,\ f(\frac{1}{t}) > 0.$$

Finally, (3.6) follows from (3.4) and (3.5).

In the following theorems we shall assume

$$||A_L|| + ||A_R|| = s+t < 1.$$

This is a stronger assumption than the usual (Young [71, Section 14.2]) diagonal dominance condition. We could refine our assumption and obtain partial conclusions but this would obscure our primary focus on the Parallel Gauss algorithm. However, in Section 4 we shall use the weaker condition $4st < 1$ in our study of the Parallel LR algorithm. We begin by establishing bounds on the quantities occurring in the Sequential Gauss algorithm.

THEOREM 3.2. Let

$$s+t < 1.$$

Then

(3.7) $\quad ||E|| < a_-,$

Sequential Gauss algorithm is well-defined,

(3.8) $\quad ||\underline{f}|| < \dfrac{||\underline{y}||}{1-t(1+a_-)},$

(3.9) $\quad ||\underline{u}|| < \dfrac{||\underline{y}||}{1-s-t}.$

PROOF. We establish the bound on $||E||$ by showing $|e_i| < a_-$, $i = 1,\ldots,m-1$. Note that $|e_1| = |s_1| \leq s < a_-$ by definition and Theorem 3.1. Assume $|e_i| < a_-$. Then

$$|e_{i+1}| = \frac{|s_{i+1}|}{|1-t_{i+1}e_i|} < \frac{s}{1-ta_-} = a_-,$$

which completes the induction.

To establish the second conclusion, note that the Sequential Gauss algorithm is well-defined if $D = I - A_L E$ is nonsingular, that is, if $||A_L E|| < 1$. Now

$$||A_L E|| \le ta_- < 1$$

by Theorem 3.1.

To establish the bound on $\underline{f}$, from (2.12),

$$||\underline{f}|| \le \frac{||\underline{v}||}{1-||A_L(I-E)||} < \frac{||\underline{v}||}{1-t(1+a_-)}.$$

To establish the bound on $\underline{u}$, from (2.13),

$$||\underline{u}|| \le \frac{||\underline{f}||}{1-||E||} < \frac{||\underline{f}||}{1-a_-}$$

$$< \frac{||\underline{v}||}{(1-a_-)(1-t(1+a_-))} = \frac{||\underline{v}||}{1-s-t}.$$

The next theorem gives analogous results for the Parallel Gauss algorithm.

THEOREM 3.3. Let

$$s+t < 1, \quad ||E^{(0)}|| < a_-.$$

Then

(3.10) $||E^{(i)}|| < a_-, \quad i = 1,\ldots,M,$

Parallel Gauss algorithm is well-defined,

(3.11) $$||\underline{f}^{(i)}|| < \frac{||\underline{v}||}{1-t(1+a_-)} + \frac{||\underline{f}^{(0)}||}{a_+^i}, \quad i = 1,\ldots,N,$$

(3.12) $$||\underline{u}^{(i)}|| < \frac{||\underline{v}||}{1-s-t} + \frac{||\underline{f}^{(0)}||}{a_+^N(1-a_-)} + a_-^i||\underline{u}^{(0)}||, \quad i = 1,\ldots,P.$$

PROOF. The bound on $||E^{(i)}||$ is established by induction. By

hypothesis, $||E^{(0)}|| < a_-$. Assume $||E^{(i-1)}|| < a_-$. Then from (2.18),

$$||E^{(i)}|| \le \frac{||A_R||}{1-||A_L E^{(i-1)}||} < \frac{s}{1-ta_-} = a_-.$$

To show that the algorithm is well-defined, observe that $I - A_L E^{(i)}$ is non-singular since

$$||A_L E^{(i)}|| < ta_- < 1.$$

To establish the bound on $||\underline{f}^{(i)}||$, from (2.19),

$$||\underline{f}^{(i)}|| \le \frac{||\underline{v}||}{1-ta_-} + \frac{t}{1-ta_-}||\underline{f}^{(i-1)}||$$

$$= \frac{||\underline{v}||}{ta_+} + \frac{1}{a_+}||\underline{f}^{(i-1)}||.$$

Hence

$$||\underline{f}^{(i)}|| \le \frac{||\underline{v}||}{ta_+}\left(\frac{1-a_+^{-i}}{1-a_+^{-1}}\right) + \frac{||\underline{f}^{(0)}||}{a_+^i}$$

$$< \frac{||\underline{v}||}{ta_+(1-a_+^{-1})} + \frac{||\underline{f}^{(0)}||}{a_+^i}$$

$$= \frac{||\underline{v}||}{1-t(1+a_-)} + \frac{||\underline{f}^{(0)}||}{a_+^i}.$$

To establish the bound on $||\underline{u}^{(i)}||$, from (2.20),

$$||\underline{u}^{(i)}|| \le ||\underline{f}^{(N)}|| + ||E^{(M)}|| \cdot ||\underline{u}^{(i-1)}||.$$

Then

$$||\underline{u}^{(i)}|| < \frac{||\underline{f}^{(N)}||}{1-a_-} + ||E^{(M)}||^i \cdot ||\underline{u}^{(0)}||.$$

From (3.10), (3.11),

$$||\underline{u}^{(i)}|| < \frac{||\underline{v}||}{1-s-t} + \frac{||\underline{f}^{(0)}||}{a_+^N(1-a_-)} + a_-^i\,||\underline{u}^{(0)}||.$$

COMMENT. Since $a_+^{-1} < 1$, $a_- < 1$, this shows that the bounds on $\underline{f}^{(i)}$, $\underline{u}^{(i)}$ are "close" to those on $\underline{f}$, $\underline{u}$.

The next two theorems, which give upper bounds on the iterate errors, are the major results of this Section.

THEOREM 3.4. Let

$$s+t < 1, \quad ||E^{(0)}|| < a_-.$$

Then

$$(3.13)\quad ||E^{(i)}-E|| < \frac{a_-}{a_+}\,||E^{(i-1)}-E||, \quad i = 1,\ldots,M,$$

$$(3.14)\quad ||\underline{f}^{(i)}-\underline{f}|| < \frac{1}{a_+}\,||\underline{f}^{(i-1)}-\underline{f}|| + \delta_M, \quad i = 1,\ldots,N,$$

$$\delta_M = \frac{||\underline{v}||\cdot||E^{(M)}-E||}{(a_+-1)(1-ta_-)},$$

$$(3.15)\quad ||\underline{u}^{(i)}-\underline{u}|| < a_-||\underline{u}^{(i-1)}-\underline{u}|| + \epsilon_{M,N}, \quad i = 1,\ldots,P,$$

$$\epsilon_{M,N} = ||\underline{f}^{(N)}-\underline{f}|| + \frac{||\underline{v}||}{1-s-t}\,||E^{(M)}-E||.$$

PROOF. From

$$(I-A_LE^{(i-1)})E^{(i)} = A_R,$$

$$(I-A_LE)E = A_R,$$

we have

$$0 = E^{(i)} - E - A_L(E^{(i-1)}E^{(i)} - EE),$$

$$(I - A_LE^{(i-1)})(E^{(i)} - E) = A_L(E^{(i-1)} - E)E.$$

Hence

$$||E^{(i)}-E|| \le \frac{||A_L||\cdot||E||\cdot||E^{(i-1)}-E||}{1 - ||A_L||\cdot||E^{(i-1)}||}$$

$$< \frac{ta_-}{1-ta_-}\,||E^{(i-1)}-E||$$

$$= \frac{a_-}{a_+}\,||E^{(i-1)}-E||,$$

which establishes (3.13).

From

$$(I - A_L E^{(M)})\underline{f}^{(i)} = \underline{v} - A_L\underline{f}^{(i-1)},$$

$$(I - A_L E)\underline{f} = \underline{v} - A_L\underline{f},$$

we have

$$(I - A_L E^{(M)})(\underline{f}^{(i)}-\underline{f}) = -A_L(\underline{f}^{(i-1)}-\underline{f}) + A_L(E^{(M)} - E)\underline{f}.$$

Hence

$$||\underline{f}^{(i)}-\underline{f}|| < \frac{t}{1-ta_-}\,||\underline{f}^{(i-1)}-\underline{f}|| + \frac{t||f||\cdot||E^{(M)}-E||}{1-ta_-}.$$

From (3.8),

$$||\underline{f}^{(i)}-\underline{f}|| < \frac{||f^{(i-1)}-f||}{a_+} + \frac{||v||\cdot||E^{(M)}-E||}{(a_+-1)(1-ta_-)},$$

which establishes (3.14).

From

$$\underline{u}^{(i)} = \underline{f}^{(N)} - E^{(M)}\underline{u}^{(i-1)},$$

$$\underline{u} = \underline{f} - E\,\underline{u},$$

we have

$$\underline{u}^{(i)} - \underline{u} = - E^{(M)}(\underline{u}^{(i-1)}-\underline{u}) - (E^{(M)} - E)\underline{u} + \underline{f}^{(N)} - \underline{f}.$$

Hence

$$||\underline{u}^{(i)}-\underline{u}|| < a_-||\underline{u}^{(i-1)}-\underline{u}|| + \frac{||v||}{1-s-t} \cdot ||E^{(M)}-E|| + ||\underline{f}^{(N)}-\underline{f}||,$$

which establishes (3.15).

The next theorem gives bounds on the iterate errors in terms of the initial errors. The bounds depend only on the diagonal dominance, the initial errors and the final accuracy desired. They are independent of the size of the system.

THEOREM 3.5. Let

$$s+t < 1, \quad ||E^{(0)}|| < a_-.$$

Then

$$(3.16) \quad ||E^{(i)}-E|| < \left(\frac{a_-}{a_+}\right)^i ||E^{(0)}-E||, \quad i = 1,\ldots,M,$$

$$(3.17) \quad ||\underline{f}^{(i)}-\underline{f}|| < \left(\frac{1}{a_+}\right)^i ||\underline{f}^{(0)}-\underline{f}|| + \left(\frac{a_-}{a_+}\right)^M \left(\frac{a_++a_-}{(a_+-1)^2}\right) ||v|| \cdot ||E^{(0)}-E||,$$

$$i = 1,\ldots,N,$$

$$(3.18) \quad ||\underline{u}^{(i)}-\underline{u}|| < a_-^i ||\underline{u}^{(0)}-\underline{u}|| + \frac{1}{1-a_-}\left(\frac{1}{a_+}\right)^N ||\underline{f}^{(0)}-\underline{f}|| + \frac{a_+^2-a_-^2}{(1-a_+)^2(1-a_-)^2}\left(\frac{a_-}{a_+}\right)^M ||v|| \cdot ||E^{(0)}-E||,$$

$$i = 1,\ldots,P.$$

PROOF. Equation (3.16) follows easily from (3.13). To obtain

(3.17), we use (3.14) to get

$$\|E^{(i)}\| \le \|I - LU\| \, \|E^{(0)}\| + \frac{(1-\alpha)}{(\alpha - \beta)} \|E\|$$

$$\le \|I - LU\| \, \|E^{(0)}\| + \frac{(1-\alpha)}{(\alpha - \beta)} \|E\|$$

Now

$$\le \|E^{(0)}\| + \frac{(1-\alpha)}{(\alpha - \beta)(1-\alpha+\beta)} \|E\|$$

and the result follows.

To obtain (3.18), we use (3.15) to get

$$\|E^{(i)}\| \le \|I - LU\| \, \|E^{(0)}\| + \frac{\beta}{1-\alpha+\beta} \|E\|$$

Now

$$\|E^{(i)} - E\| \le \left(\frac{\beta}{(\alpha - \beta)} \right)^2 \frac{\beta}{1-\alpha+\beta}$$

and the result follows after some algebraic manipulation.

The Parallel Gauss algorithm is finite in the following sense. Since

$$e_{ii} = s = e$$

the first component of $E^{(i)} - E$ is zero. It can be shown that this implies

(3.19) $E^{(i)} - E = 0$, $i \le m-2$.

If the iteration for is carried out until (3.19) holds,

then

$$f_1^{(i)} = v_1 = f_1$$

implies that

(3.20) $\underline{f}^{(i)} - \underline{f} = 0, \ i \geq m-1.$

If the iteration for $\underline{f}^{(i)}$ is carried out until (3.20) holds, then

$$u_m^{(i)} = f_m = u_m$$

implies that

(3.21) $\underline{u}^{(i)} - \underline{u} = 0, \ i \geq m-1.$

Thus if each of the three iterations which comprise parallel Gauss are carried to completion, the algorithm is finite. This fact is of course not of algorithmic interest. What is important is that P, M, N can be chosen so that each iteration is norm reducing.

To see this consider the inequalities of Theorems 3.4 and 3.5. The E iteration is always norm reducing. The $\underline{u}$ iteration is norm reducing while

(3.22) $e_{M,N} < ||\underline{u}^{(i-1)} - \underline{u}|| \ (1-a_-).$

Fix P. Choose M, N so that (3.22) holds for $i \leq P$. Equations (3.15) - (3.17) show this can always be done. The $\underline{f}$ iteration is norm reducing while

(3.23) $\delta_{M'} < ||\underline{f}^{(i-1)} - \underline{f}|| \ (1 - \frac{1}{a_+}).$

Choose $M' \geq M$ such that (3.23) holds for $i = 1,\ldots,N$. With this choice of P, M', N, all three iterations are norm reducing.

The rates of convergence are determined by a_-, a_+^{-1}. We expect these quantities to increase as s and t increase. That this is indeed the case is the content of the following theorem. Since the results are easily obtained from calculus, the proof is omitted.

THEOREM 3.6. Let

$$s + t < 1.$$

Then

$\frac{a_-}{a_+}$ is a monotonically increasing function of st.

For s fixed, a_- and a_+^{-1} are monotonically increasing functions of t.

For t fixed, a_- and a_+^{-1} are monotonically increasing functions of s.

To get some feel for the size of a_-, a_+^{-1}, $a_-a_+^{-1}$, take

(3.24) $s = t = .4$, $s+t = .8$.

Then

$$a_-a_+^{-1} = \frac{1}{4},\ a_- = a_+^{-1} = \frac{1}{2}.$$

Hence if the system has the diagonal dominance of (3.24), each E iteration gains two bits while each $\underline{f}$ or $\underline{u}$ iteration gains approximately one bit.

A stability analysis of the Parallel Gauss algorithm has not yet been performed. We shall show that under our assumptions no pivoting is required for the Sequential Gauss algorithm, which leads us to expect that Parallel Gauss also has desirable stability properties.

THEOREM 3.7. Let

$$s+t < 1.$$

Then

$$|d_j| = |1-t_je_{j-1}| > |s_j|,\ |d_j| > |t_j|,\ j = 1,\ldots,m,$$

where for convenience we define $e_0 = 0$.

PROOF.

$$\left|\frac{s_j}{1-t_je_{j-1}}\right| < \frac{s}{1-ta_-} = a_- < 1,$$

$$\left|\frac{t_j}{1-t_je_{j-1}}\right| < \frac{t}{1-ta_-} = \frac{1}{a_+} < 1.$$

If

(3.25) $t = s$

our results simplify. In particular, (3.25) holds if $s_i = t_{i+1}$, which holds for symmetric matrices. One may easily verify the following

THEOREM 3.8. Let

$$t = s, \; s < \frac{1}{2}.$$

Then

(3.26) $$\frac{1}{a_+} = a_-, \; \frac{a_-}{a_+} = a_-^2.$$

Theorem 3.5 simplifies to

THEOREM 3.9. Let

$$t = s, \; s < \frac{1}{2}, \; ||E^{(0)}|| < a_-.$$

Then

$$||E^{(i)}-E|| < a_-^{2i}||E^{(0)}-E||, \; i = 1,\ldots,M,$$

$$||\underline{f}^{(i)}-\underline{f}|| < a_-^i||\underline{f}^{(0)}-\underline{f}|| + \frac{a_-^{2M+1}}{1-2s}||\underline{v}||\cdot||E^{(0)}-E||,$$

$$i = 1,\ldots,N.$$

$$||\underline{u}^{(i)}-\underline{u}|| < a_-^i||\underline{u}^{(0)}-\underline{u}|| + \frac{a_-^N}{1-a_-}||\underline{f}^{(0)}-\underline{f}|| + \frac{1-a_-^4}{(1-a_-)^4}a_-^2||v||\cdot$$

$$||E^{(0)}-E||, \; i = 1,\ldots,P.$$

4. ANALYSIS OF PARALLEL LR ALGORITHM

We turn to the problem of calculating only the LR decomposition of A without solving the linear system $A\underline{u} = \underline{v}$. A Parallel LR decomposition was defined in Section 2. Our basic assumption in this section is $4st < 1$. Since

$$4st \le (s+t)^2,$$

this is a weaker condition than $s+t < 1$. The following theorem plays the analogous role that Theorem 3.1 played in the analysis of Parallel Gauss.

THEOREM 4.1. Let

$$4st < 1.$$

Then $f(x) = tx^2 - x + s$ has two real distinct zeros a_- and a_+ and

(4.1) $0 < s < a_- < 2s < a_+$,

(4.2) $t < \frac{1}{a_+} < 2t$,

(4.3) $st < \frac{a_-}{a_+} < 4st < 1$.

PROOF. The inequalities (4.1) follow from

$$f(s) > 0,\ f(2s) < 0,\ f(\infty) > 0.$$

The inequalities (4.2) follow from

$$f(\tfrac{1}{2t}) < 0,\ f(\tfrac{1}{t}) > 0,\ 2s < \tfrac{1}{2t}.$$

Finally, (4.3) follows from (4.1) and (4.2).

COMMENT. As in Theorem 3.1, we conclude that $a_- a_+^{-1} < 1$. However, we <u>cannot</u> conclude $a_- < 1$, $a_+^{-1} < 1$.

The following theorem is the counterpart of Theorem 3.2. Since the proof is the same as for Theorem 3.2, with Theorem 4.1 replacing Theorem 3.1, it is omitted.

THEOREM 4.2. Let

$$4st < 1.$$

Then

$$||E|| < a_-,$$

and Sequential LR is well-defined.

The proofs of the next two theorems are analogous to the proofs of Theorems 3.3 and 3.4 and are therefore omitted.

THEOREM 4.3. Let

$$4st < 1, \quad ||E^{(0)}|| < a_-.$$

Then

$$||E^{(i)}|| < a_-$$

and Parallel LR is well-defined.

THEOREM 4.4. Let

$$4st < 1, \quad ||E^{(0)}|| < a_-.$$

Then

$$||E^{(i)}-E|| < \frac{a_-}{a_+} ||E^{(i-1)}-E||.$$

From Theorem 4.1, $a_- a_+^{-1} < 1$. Hence the E iteration is norm-reducing.

5. PARALLEL ALGORITHMS BASED ON ADDITIVE SPLITTINGS

The Parallel LR and Parallel Gauss Algorithms are based on the "multiplicative splitting"

$$A = LR.$$

We now define parallel algorithms based on the "additive splitting"

$$A = A_L + I + A_R,$$

where A_L and A_R are defined as in Section 2. We consider four parallel algorithms: Jacobi, JOR (Jacobi Over-Relaxation), Parallel Gauss-Seidel, Parallel SOR (Parallel Successive Over-Relaxation).

Jacobi

$$(5.1) \quad \underline{u}^{(i+1)} = -(A_L + A_R)\, \underline{u}^{(i)} + \underline{v}.$$

JOR

$$(5.2)\quad \underline{u}^{(i+1)} = [(1-\omega)I-\omega(A_L + A_R)]\underline{u}^{(i)} + \omega\underline{v},\ \omega > 0$$

These two classic algorithms are both parallel since, when $\underline{u}^{(i)}$ is known, all components of $\underline{u}^{(i+1)}$ may be calculated in parallel.

We now define parallel analogues of Gauss-Seidel and SOR iteration.

Parallel Gauss-Seidel

Outer iteration:

$$(5.3)\quad (I + A_R)\underline{u}^{(i)} = -A_L\underline{u}^{(i-1)} + \underline{v}.$$

Let

$$\underline{z} = -A_L\,\underline{u}^{(i-1)} + \underline{v},\ \underline{y} = \underline{u}^{(i)}.$$

Note that the components of $\underline{z}$ can be formed in parallel. Then

$$(I + A_R)\underline{y} = \underline{z}.$$

We solve this bidiagonal system by the following parallel iteration.

Inner Iteration:

$$(5.4)\quad \underline{y}^{(j)} = \underline{z} - A_R\underline{y}^{(j-1)}.$$

Observe that (5.4) is a Jacobi iteration for the solution of the bidiagonal system. Furthermore, this iteration has the same form as the third iteration of the Parallel Gauss Algorithm defined by (2.20).

Parallel SOR

Outer Iteration:

$$(5.5)\quad (I + \omega A_R)\underline{u}^{(i)} = [(1-\omega)I - \omega A_L]\underline{u}^{(i-1)} + \omega\underline{v},\ \omega > 0.$$

Let

$$\underline{z}_\omega = [(1-\omega)I - \omega A_L]\underline{u}^{(i-1)} + \omega\underline{v},\ y_\omega = \underline{u}^{(i)}.$$

Thus

$$(I + \omega A_R)\underline{y}_\omega = \underline{z}_\omega .$$

We solve this bidiagonal system by the following parallel iteration.

Inner Iteration:

$$(5.6)\quad \underline{y}_\omega^{(j)} = \underline{z}_\omega - \omega A_R \underline{y}_\omega^{(j-1)}.$$

We now have defined parallel algorithms based on multiplicative and additive splittings. In the next section we give theoretical comparisons among these algorithms for a model problem.

6. COMPARISON OF PARALLEL ALGORITHMS ON A MODEL PROBLEM

We compare the parallel algorithms we have defined on the two parameter model problem

$$A_{s,m} = \begin{pmatrix} 1 & s & & & & \\ s & 1 & s & & & \\ & \cdot & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & & & s & 1 & s \\ & & & & s & 1 \end{pmatrix},$$

where A is an (m,m) matrix and $0 < s < \frac{1}{2}$. We first dispose of JOR by proving that the optimal JOR iteration is Jacobi iteration. We shall let $\rho(B)$ denote the spectral radius of a matrix B. Let

$$(6.1)\quad J_\omega = (1-\omega)I - \omega(A_L + A_R).$$

THEOREM 6.1. For the model problem

$$\rho(J_\omega) \geq \rho(J_1),\ 0 < \omega.$$

PROOF. Denote the eigenvalues of J_1 and J_ω by μ_i and λ_i respectively. Then

$$ix = 2s\cos(--), \quad i = 1,\ldots,m.$$

Let

Then

From (6.1),

$$X_{.} = 1 - a)(1 \quad ,$$

Hence

$$X_{1} - 1 - u)(1\text{-}2sTp$$

and

$$|X_{.}1 \hat{} |Xj I, \quad j = 2,\ldots,m, \text{ for } 0 < u) \pounds 1.$$

Furthermore,

$$X = 1 - U)(1+2sTl)$$

and

$$|X_{m}| \hat{} IVjL J\cdot 1,\ldots,m\text{-}1, \text{ for } 1 \hat{} a).$$

It may be verified that for $0 < u)$

$$\max C|xJ_{,} |\backslash J) \hat{} 2sT| = pC\hat{})$$

and the result follows.

We compare the following four parallel algorithms:

Parallel Gauss
Jacobi
Parallel Gauss-Seidel
Parallel Optimal SOR

We then compare the best of the parallel algorithms with the Sequential Gauss algorithm.

We have two types of estimates for how the norm of the error decreases. For the three iterations which comprise the Parallel Gauss algorithm, the errors a satisfy relations of the form

(6.2) $$*C \quad ||a^{(\cdot)}||.$$

The error formulas for the remaining algorithms are typified by the error formula for Jacobi iteration,

(6.3) $\underline{\sigma}^{(i+1)} = J\,\underline{\sigma}^{(i)}$,

where $J = -(A_L + A_R)$.

After I iterations, assume the norm of the error has been reduced by 2^{-b}. Thus

$$||\underline{\sigma}^{(I)}|| = 2^{-b}||\underline{\sigma}^{(0)}||.$$

For $\underline{\sigma}^{(i)}$ satisfying a relation of the form (6.2), we estimate the number of iterations to reduce the error norm by 2^{-b} as

(6.4) $$I \sim \frac{-b}{\log C}.$$

(In this paper, log denotes logarithm to the base 2.) For $\underline{\sigma}^{(i)}$ satisfying a relation of the form (6.3), we estimate

(6.5) $$I \sim \frac{-b}{\log \rho(J)}.$$

Assume that all arithmetic operations take unit time and that z arithmetic operations must be done sequentially per iteration. Then the total time T is estimated by

(6.6) $T \sim zI$.

We define the rate of convergence R by either

$$R = -\log C \text{ or } R = -\log \rho(J).$$

We take

$$\ln 2 \doteq .69 \doteq 2/3.$$

We deal with three values of s: s near $\frac{1}{2}$, s small, and $s = .25$. More precisely,

1. s near $\frac{1}{2}$. Set

(6.7) $$1 - 4s^2 = \epsilon^2$$

and assume ϵ^2 is negligible compared to unity.

2. s small. Neglect s^2 relative to unity.

3. $s = .25$. This is an intermediate value of s.

In the analysis that follows, we assume that the E and $\underline{f}$ iterations are performed long enough so that the quantities δ_M and $\epsilon_{M,N}$ which occur in Theorem 3.4 are negligible. This is equivalent to assuming that (3.16) - (3.18) are replaced by

$$||E^{(i)}-E|| < \left(\frac{a_-}{a_+}\right)^i ||E^{(0)}-E||,$$

$$||\underline{f}^{(i)}-\underline{f}|| < \left(\frac{1}{a_+}\right)^i ||\underline{f}^{(0)}-\underline{f}||,$$

$$||\underline{u}^{(i)}-\underline{u}|| < a_-^i ||\underline{u}^{(0)}-\underline{u}||.$$

We remind the reader of the assumption, stated in the Introduction, that no overhead is assessed for using m processors or for loading vectors of length m.

We estimate the total time required for our four algorithms over the three values of s. Results are summarized in Table 6.1.

1.1 Parallel Gauss Algorithm, s near $\frac{1}{2}$.

Let $1 - 4s^2 = \epsilon^2$. Then

$$a_- = \frac{1-\sqrt{1-4s^2}}{2s} \sim \frac{1-\epsilon}{1-\frac{1}{2}\epsilon^2} \sim e^{-\epsilon},$$

and

$$\frac{1}{a_+} \sim e^{-\epsilon}, \quad \frac{a_-}{a_+} \sim e^{-2\epsilon}.$$

Assume for simplicity that b is the same in each of the three iterations which comprise the algorithm. (This would be the case if each iteration is done until full machine accuracy is achieved and if the initial relative errors are all around unity.)

For the model problem, the Parallel Gauss iterations simplify to

$$e_j^{(i)} = \left(\frac{1}{s} - e_{j-1}^{(i-1)}\right)^{-1}$$

$$f_j^{(i)} = \left(\frac{v_j}{s} - f_{j-1}^{(i-1)}\right)\cdot\left(\frac{s}{1-se_{j-1}^{M}}\right),$$

$$u_j^{(i)} = f_j^{(N)} - e_j^{(M)}\, u_{j+1}^{(i-1)}.$$

Since v_j/s is independent of i, we shall not count it in our iteration costs.

We have:

$\underline{e}$ iteration: $I \sim \frac{-b}{\log(a_-/a_+)} \sim \frac{b}{3\varepsilon}$, $z = 2$,

$\underline{f}$ iteration: $I \sim \frac{-b}{\log(1/a_+)} \sim \frac{2}{3}\frac{b}{\varepsilon}$, $z = 2$,

$\underline{u}$ iteration: $I \sim \frac{-b}{\log a_-} \sim \frac{2}{3}\frac{b}{\varepsilon}$, $z = 2$.

Hence for the Parallel Gauss algorithm,

$$T \sim \frac{b}{3\varepsilon}(2 + 2\cdot 2 + 2\cdot 2) = \frac{10}{3}\frac{b}{\varepsilon}.$$

1.2 Jacobi algorithm, s near $\frac{1}{2}$.

For the matrix $A_{s,m}$,

$$\rho(J) = 2s\cos\left(\frac{\pi}{m+1}\right)$$

with s near $\frac{1}{2}$ and m large,

$$\log \rho(J) \sim -\frac{3}{4}\sigma^2,$$

where

(6.8) $\sigma^2 = \varepsilon^2 + \left(\frac{\pi}{m+1}\right)^2$.

For the model problem, Jacobi iteration simplifies to

$$u_j^{(i)} = v_j - s(u_{j-1}^{(i-1)} + u_{j+1}^{(i-1)}).$$

Hence

$$I \sim \frac{-b}{\log \rho(J)} \sim \frac{4}{3}\frac{b}{\sigma^2}, \quad z = 3$$

and

$$T \sim \frac{4b}{\sigma^2}.$$

1.3 Parallel Gauss-Seidel algorithm, s near $\frac{1}{2}$.

$A_{s,m}$ is irreducible and weakly diagonally dominant. Hence, if G denotes the Gauss-Seidel matrix,

$$\rho(G) = \rho^2(J).$$

The number of outer iterations is then given by

$$I \sim \frac{-b}{\log \rho(G)} \sim \frac{2}{3}\frac{b}{\sigma^2}.$$

We next estimate how many inner iterations we must perform for each outer iteration. We estimate the rate of convergence of the inner iteration by

$$R = -\log ||A_R|| = -\log s.$$

Hence the number of inner iterations per outer iteration is

$$I \sim \frac{-b}{\log s} \sim b$$

and the total number of inner iterations is estimated by

$$I \sim \frac{2}{3}\left(\frac{b}{\sigma}\right)^2.$$

Since z = 2 for the inner iteration, the total time spent on inner iterations is

$$T \sim \frac{4}{3}\left(\frac{b}{\sigma}\right)^2.$$

This dominates the total time required.

1.4 Parallel Optimal SOR Algorithm, s near $\frac{1}{2}$.

Let Ω denote the optimal value of the over-relaxation parameter ω. Then (Young [71]),

$$\Omega = \frac{2}{1+\sqrt{1-\rho^2(J)}}.$$

Let G_Ω denote the optimal SOR matrix. Then

$$N(C) \;\approx\; \frac{1-\rho^{2}(J)}{1+\sqrt{1-\rho^{2}(J)}}$$

Note that Q is the optimal parameter for the outer iteration only. (There is no inner iteration for sequential SOR.) Numerical experimentation indicates that adjusting a) so as to optimize over outer and inner iterations does not significantly affect the results.

Now,

The number of outer iterations is given by

$$I \sim \log \rho(G_{\omega})^{-b} \quad 3_{\omega}{}^{b} *$$

The rate of convergence of the inner iteration is

$$R = -\log \|\ln A_{\omega}\| J = -\log Q s.$$

Hence

$$3$$

where as before

$$2 \qquad 2 \bullet / T \backslash 2$$

The total number of inner iterations per outer iteration is

$$T \quad 2\ b \\ 3\ a$$

and the total number of inner iterations is

Since z = 2, the total time for inner iterations is

$$T_{m} \sim 9 < a > \quad 4,b.2$$

*2.*1 Parallel Gauss algorithm, s small.

For s small,

$$a \sim s, \quad \frac{1}{-} \sim s, \quad \frac{*}{-} \sim s^{2}.$$

Let

(6.9) $\mu = -\log s.$

Then we have:

$\underline{e}$ iteration: $I \sim \frac{b}{2\mu}$, $z = 2$,

$\underline{f}$ iteration: $I \sim \frac{b}{\mu}$, $z = 2$,

$\underline{u}$ iteration: $I \sim \frac{b}{\mu}$, $z = 2$.

Hence

$$T \sim 5\frac{b}{\mu}.$$

2.2 Jacobi algorithm, s small.

Since

$$\rho(J) = 2s \cos(\tfrac{\pi}{m+1}) \sim 2s,$$
$$T \sim \frac{3b}{\mu-1}.$$

The analysis of Parallel Gauss-Seidel and Parallel SOR for the case of s small introduces no new ideas and is omitted.

The analysis of the four algorithms for $s = .25$ is straightforward and is also omitted.

The results are summarized in Table 6.1. We repeat the definitions of the quantities appearing in the Table.

b: $||\underline{\sigma}^{(I)}|| = 2^{-b} ||\underline{\sigma}^{(0)}||$, where $\underline{\sigma}^{(i)}$ denotes error at ith stage and I is the total number of iterations.

ϵ: $\epsilon^2 = 1 - 4s^2$.

σ: $\sigma^2 = \epsilon^2 + (\frac{\pi}{m+1})^2$. Since m is large, $\sigma \sim \epsilon$ unless ϵ is very small (s very close to $\frac{1}{2}$).

μ: $\mu = -\log s$.

The following conclusions concerning the model problem may be drawn from Table 6.1.

1. Parallel SOR is superior to Parallel Gauss-Seidel.

2. Neither of these algorithms is competitive with Jacobi or Parallel Gauss. (Note that we draw this conclusion only for the particular Parallel Gauss-Seidel and Parallel SOR defined in this paper. It need not of course apply to other parallel SOR algorithms which might be defined.)

3. For s small (the matrix very diagonally dominant), Jacobi is somewhat superior to Parallel Gauss.

4. For moderate values of s ($s \sim .25$), Parallel Gauss is about equal to Jacobi.

5. For s close to $\frac{1}{2}$ (very little diagonal dominance), Parallel Gauss is far superior to Jacobi.

Table 6.1. Estimates of Total Time

	$s \sim \frac{1}{2}$	s small	s = .25
Parallel Gauss	$\frac{10}{3}\frac{b}{\varepsilon}$	$5\frac{b}{\mu}$	$\frac{5}{2}b$
Jacobi	$4\frac{b}{\sigma^2}$	$3\frac{b}{\mu-1}$	$3b$
Parallel Gauss-Seidel	$\frac{4}{3}(\frac{b}{\sigma})^2$	$\frac{b^2}{\mu(\mu-1)}$	$\frac{1}{2}b^2$
Parallel Optimal SOR	$\frac{4}{9}(\frac{b}{\sigma})^2$	$(\frac{b}{\mu})^2$	$\frac{b^2}{4}$

The switch-over point between Jacobi and Parallel Gauss is determined by

$$(6.10)\quad 5\frac{b}{\log a_-} = \frac{3b}{\log(2s\cos(\frac{\pi}{m+1}))}.$$

For m large, equality holds for $s \doteq .19$.

Thus for $s \leq .19$ Jacobi is the best of our parallel algorithms while for $s \geq .19$ Parallel Gauss is best. This result agrees with our intuition since multiplicative splitting is "more implicit" than additive splitting and is therefore superior for matrices with little dominance.

Finally we compare the best parallel algorithm with the Sequential Gauss. For the model problem Sequential Gauss requires total time

(6.11) $T \sim 7m.$

For s small, Jacobi is clearly superior to Sequential Gauss since

$$\frac{3b}{\mu-1} < 7m,$$

for reasonable b and almost all m. For s near $\frac{1}{2}$, we compare Parallel and Sequential Gauss. The times are equal when

(6.12) $$\frac{10}{3}\frac{b}{\epsilon} = 7m.$$

If m is large, this will only hold when s is very close to 1/2. For example, if $m = 1000$, $b = 27$, then $s \doteq .499959$. Thus, under the assumptions of this paper, Parallel Gauss is faster than Sequential Gauss until s is very close to 1/2.

7. NUMERICAL EXPERIMENTATION

Extensive testing on the Jacobi and Parallel Gauss algorithms was performed for the model problem with $m = 100$ and 1000 and a wide range of values of s. All testing was carried out on the CMU Computer Science Department's PDP-10 in APL. On this sequential machine, the vectors are generated one component at a time. Iteration counts for a parallel or vector computer are obtained by counting the generation of a vector as one iteration.

The numerical results are consistent with the theoretical estimates developed in this paper. We limit ourselves here to reporting several examples.

According to (3.13)

$$||E^{(i)}-E|| < a_*^2 \, ||E^{(i-1)}-E||.$$

In Table 7.1, we exhibit

$$||E^{(i)}-E||/||E^{(i-1)}-E||, \quad i = 1,\ldots,10$$

for m = 100, s = .48. Then $a = .75$, $a^2 = .5625$.

Table 7.1. Reduction of $||E^{(i)}-E||$

i	$\|\|E^{(i)}-E\|\|/\|\|E^{(i-1)}-E\|\|$
1	.47
2	.51
3	.54
4	.55
5	.55
6	.56
7	.56
8	.56
9	.56
10	.56

The system $Au = v$ is solved by Jacobi and Parallel Gauss for m = 100, $v_i = 1$, $i = 1,\ldots,m$. The initial estimate for Jacobi is $u^{(0)} = v$. The initial estimates for Parallel Gauss are $e_i^{(0)} = s$, $f^{(0)} = v$, $u^{(0)} = f^{(N)}$. To obtain the "true" solution, the system is solved by Sequential Gauss. Iterations are terminated when the initial error is reduced by 2^{-b}, b = 27. The total time on a parallel machine is estimated by multiplying the number of iterations by the number of sequential arithmetic operations per iteration. Results for selected s are given in Table 7.2. Total times are underlined.

Since Parallel Gauss terminates in m iterations, the data for Parallel Gauss when s = .49 is obtained with m = 1000. The number of iterations is essentially independent of m as long as that number is less than m.

Table 7.2. Total Time for Parallel Gauss and Jacobi

s	E Iterations	f Iterations	u Iterations	Total Time Parallel Gauss	Jacobi Iterations	Total Time Jacobi
.10	3	8	9	40	11	33
.20	6	11	13	60	20	60
.24	6	13	13	64	25	75
.40	13	26	26	130	82	246
.49	41	79	82	404	705	2115

These numerical results agree with the theoretical predictions. For s very small, Jacobi is slightly faster. They are equal at s - ,20, close to the predicted value of s' .19. By s = .40 Parallel Gauss is substantially faster and its advantage increases sharply as s = I is approached.

ACKNOWLEDGMENTS

I want to thank D. Heller, Carnegie-Mellon University, for performing all numerical experimentation and for his helpful suggestions. I want to acknowledge Dr. F. N. Fritsch, Dr. A. Hindmarsh and Dr. N. Madsen, LLL, Dr. P. Swartztrauber, NCAR, D. Stevenson and Professor G. W. Stewart, Carnegie-Mellon University, for their comments.

BIBLIOGRAPHY

Kogge and Stone [72] Kogge, P. M. and Stone, H. S., A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations. Report 25, Digital Systems Laboratory, Stanford University, March 1972.

Stone [73] Stone, H. S., An Efficient Parallel Algorithm for the Solution of a Tridiagonal Linear System of Equations. JACM, 2J0 (1973), 27-38.

Young [71] Young, D., Iterative Solution of Large Linear Systems, Academic Press, 1971.