NASA Technical Paper 3585

A Fast Parallel Tridiagonal Algorithm for a Class of CFD Applications

Xian-He Sun and Stuti Moitra

August 1996

NASA Technical Paper 3585

A Fast Parallel Tridiagonal Algorithm for a Class of CFD Applications

Xian-He Sun
Louisiana State University • Baton Rouge, Louisiana

Stuti Moitra
Langley Research Center • Hampton, Virginia

National Aeronautics and Space Administration
Langley Research Center • Hampton, Virginia 23681-0001

August 1996

Available electronically at the following URL address: http://techreports.larc.nasa.gov/ltrs/ltrs.html

Printed copies available from the following:

NASA Center for AeroSpace Information
800 Elkridge Landing Road
Linthicum Heights, MD 21090-2934
(301) 621-0390

National Technical Information Service (NTIS)
5285 Port Royal Road
Springfield, VA 22161-2171
(703) 487-4650

Symbols

A	$n \times n$ tridiagonal system
$\tilde{A}$	$n \times n$ matrix
a, b	real solutions of equation (15)
d	right side of the tridiagonal system
E	$n \times 2(p-1)$ matrix
E^T	transpose of matrix E
m	order of the submatrix ($m = n/p$)
n	order of the matrix
$n1$	number of right sides of the system
p	number of processors
V	$n \times 2(p-1)$ matrix
$\tilde{v}$	vectors
v^i, w^i	vectors
x	solution of the tridiagonal system
$\tilde{x}$	$n \times 1$ vector
x', x^*	notation introduced in accuracy analysis
Y	$n \times 2(p-1)$ matrix
Z	$2(p-1) \times 2(p-1)$ matrix
α	communication latency (start time)
β	transmission rate (bandwidth)
ΔA	$\Delta A = VE^T$
λ	off-diagonal elements of matrix A
μ	diagonal elements of matrix A
$()^{-1}$	inverse of matrix ()

Abbreviations:

ADI	alternating direction implicit
CFD	computational fluid dynamics
MFLOPS	million floating-point operations per second
MIMD	multiple-instruction multiple-data
PDD	parallel diagonal dominant
PDE	partial differential equation
RCD	recursive doubling
RISC	reduced instruction set computer
SIMD	single-instruction multiple-data
SOR	successive over relaxation
SPP	simple parallel prefix
OER	odd-even reduction
STT	symmetric Toeplitz tridiagonal

Abstract

The parallel diagonal dominant (PDD) algorithm is an efficient tridiagonal solver. This paper presents for study a variation of the PDD algorithm, the reduced PDD algorithm. The new algorithm maintains the minimum communication provided by the PDD algorithm, but has a reduced operation count. The PDD algorithm also has a smaller operation count than the conventional sequential algorithm for many applications. Accuracy analysis is provided for the reduced PDD algorithm for symmetric Toeplitz tridiagonal (STT) systems. Implementation results on Langley's Intel Paragon and IBM SP2 show that both the PDD and reduced PDD algorithms are efficient and scalable.

1.0. Introduction

Distributed-memory parallel computers dominate today's parallel computing arena. These machines, such as the Kendall Square KSR-1, Intel Paragon, TMC CM-5, and the recently announced IBM SP2 and Cray T3D concurrent systems, successfully deliver high-performance computing power for solving certain of the so-called "grand-challenge" problems (ref. 1). Despite initial success, parallel machines have not been widely accepted in the production engineering environment. On a parallel computing system, a task has to be partitioned and distributed appropriately among processors to reduce communication cost and to achieve load balance. More importantly, even with careful partitioning and mapping, the performance of an algorithm might still be unsatisfactory because conventional sequential algorithms may be serial in nature and may not be implemented efficiently on parallel machines. In many cases, new algorithms must be introduced to increase parallelism and to take advantage of the computing power of the scalable parallel hardware.

Solving tridiagonal systems is a basic computational kernel of many computational fluid dynamics (CFD) applications. Tridiagonal systems appear in multigrid methods, alternating direction implicit (ADI) method, wavelet collocation method, and in-line successive over relaxation (SOR) preconditioners for conjugate gradient methods (ref. 2). In addition to solving partial differential equations (PDE), tridiagonal systems also arise in digital signal processing, image processing, stationary time series analysis, and spline curve fitting (ref. 3). One direct motivation for developing an efficient kernel for solving tridiagonal systems at the National Aeronautics and Space Administration (NASA) is that the implicit systems of compact schemes (ref. 4), which are relatively new finite-difference schemes widely used in production codes at Langley Research Center and Ames Research Center, are tridiagonal.

Intensive research has been carried out on the development of efficient parallel tridiagonal solvers. Many algorithms have been proposed (refs. 5, 6, and 7), including the recursive doubling reduction method (RCD) developed by Stone (ref. 8) and the cyclic reduction or odd-even reduction method (OER) developed by Hockney (ref. 9). In general, parallel tridiagonal solvers require global communications, which makes them inefficient on distributed-memory architectures. Recently, we have taken a new approach: to increase parallel performance by introducing a bounded numerical error. Two new algorithms, namely the parallel diagonal dominant (PDD) algorithm (ref. 2) and the simple parallel prefix (SPP) algorithm (ref. 10), have been proposed for multiple-instruction multiple-data (MIMD) and single-instruction multiple-data (SIMD) machines, respectively. These two algorithms take advantage of the fact that tridiagonal systems arising in compact schemes are diagonal dominant. Backed by rigorous accuracy analyses, the algorithms truncate communication and computation without degrading the accuracy of the calculations.

In this paper, a new algorithm, the reduced PDD algorithm, is studied based on the same approach: increasing parallel performance by introducing a bounded numerical error. The reduced PDD algorithm, a variation of the PDD algorithm, maintains the minimum communication provided by the PDD algorithm, but has a reduced operation count. The reduced PPD algorithm also has a smaller operation count than the conventional sequential algorithm for many applications. The emphasis of this study is on implementation issues and performance comparisons of the PDD and reduced PDD algorithm. Most of the theoretical results, including the introduction of the PDD and reduced PDD algorithm, can be found in reference 2.

This paper is organized as follows. Section 2 provides the background of the parallel PDD algorithm. Section 3 introduces the new algorithm, the reduced PDD algorithm. Section 4 gives an accuracy analysis for the reduced PDD algorithm. Experimental results on the Intel Paragon and IBM SP2 multicomputer are presented in section 5. Performance comparison of the newly proposed algorithm and other existing algorithms, and of the two parallel platforms are also discussed in this section. Section 6 provides concluding remarks.

2.0. Parallel Diagonal Dominant (PDD) Algorithm

A tridiagonal system is a linear system of equations

$$Ax = d \tag{1}$$

where $x = (x_1, \ldots, x_n)^T$ and $d = (d_1, \ldots, d_n)^T$ are n-dimensional vectors and A is a diagonally dominant tridiagonal matrix with order n:

$$A = \begin{bmatrix} b_0 & c_0 & & & \\ a_1 & b_1 & c_1 & & \\ & \cdot & \cdot & \cdot & \\ & & \cdot & \cdot & \cdot \\ & & a_{n-2} & b_{n-2} & c_{n-2} \\ & & & a_{n-1} & b_{n-1} \end{bmatrix} = [a_i, b_i, c_i] \tag{2}$$

To solve equation (1) efficiently on parallel computers, we partition A into submatrices. We assume that $n = pm$, where p is the number of processors available. The matrix A in equation (1) can be written as

$$A = \tilde{A} + \Delta A$$

where $\tilde{A}$ is a block diagonal matrix with diagonal submatrices $A_i (i = 0, \ldots, p-1)$. The submatrices $A_i (i = 0, \ldots, p-1)$ are $m \times m$ tridiagonal matrices. Let e_i be a column vector with its ith $(0 \le i \le n-1)$ element being one and all the other entries being zero. We have

$$\Delta A = \left[a_m e_m, c_{m-1} e_{m-1}, a_{2m} e_{2m}, c_{2m-1} e_{2m-1}, \ldots, c_{(p-1)m-1} e_{(p-1)m-1}\right] \begin{bmatrix} e_{m-1}^T \\ e_m^T \\ \cdot \\ \cdot \\ e_{(p-1)m-1}^T \\ e_{(p-1)m}^T \end{bmatrix} = VE^T$$

where both V and E are $n \times 2(p-1)$ matrices. Thus, we have

$$A = \tilde{A} + VE^T$$

Based on the matrix modification formula originally defined by Sherman and Morrison (ref. 11) for rank-one changes, and assuming that all A_i's are invertible, equation (1) can be solved by

$$x = A^{-1} d = (\tilde{A} + VE^T)^{-1} d \tag{3}$$

$$x = \tilde{A}^{-1} d - \tilde{A}^{-1} V (I + E^T \tilde{A}^{-1} V)^{-1} E^T \tilde{A}^{-1} d \tag{4}$$

Let

$$\tilde{A}\tilde{x} = d \tag{5}$$

$$\tilde{A} Y = V \tag{6}$$

$$h = E^T \tilde{x} \tag{7}$$

$$Z = I + E^T Y \tag{8}$$

$$Zy = h \tag{9}$$

$$\Delta x = Yy \tag{10}$$

Equation (4) becomes

$$x = \tilde{x} - \Delta x \tag{11}$$

In equations (5) and (6), $\tilde{x}$ and Y are solved by the lower/upper (LU) decomposition method. By the structure of $\tilde{A}$ and V, this is equivalent to solving

$$A_i \left[\tilde{x}^{(i)}, v^{(i)}, w^{(i)}\right] = \left[d^{(i)}, a_{im} e_0, c_{(i+1)m-1} e_{m-1}\right] \tag{12}$$

$i = 0, \ldots, p-1$. Here $\tilde{x}^{(i)}$ and $d^{(i)}$ are the ith block of $\tilde{x}$ and d, respectively, and $v^{(i)}, w^{(i)}$ are possible nonzero column vectors of the ith row block of Y. Equation (12) implies that we only need to solve three linear systems of order m with the same LU decomposition for each i $(i = 0, \ldots, p-1)$.

Solving equation (9) is the major computation involved in the conquer part of our algorithms. Different approaches have been proposed for solving equation (9), which results in different algorithms for solving tridiagonal systems (ref. 5). The matrix Z in equation (9) has the form

$$Z = \begin{bmatrix} 1 & w_{m-1}^{(0)} & 0 & & & & & & \\ v_0^{(1)} & 1 & 0 & w_0^{(1)} & & & & & \\ v_{m-1}^{(1)} & 0 & 1 & w_{m-1}^{(1)} & 0 & & & & \\ & \cdot & \cdot & \cdot & \cdot & \cdot & & & \\ & & \cdot & \cdot & \cdot & \cdot & \cdot & & \\ & & & \cdot & \cdot & 1 & 0 & w_0^{(p-2)} & \\ & & & & v_{m-1}^{(p-2)} & 0 & 1 & w_{m-1}^{(p-2)} & \\ & & & & & 0 & v_0^{(p-1)} & 1 & \end{bmatrix}$$

where $v^{(i)}$, $w^{(i)}$ for $i = 0, \ldots, p-1$ are solutions of equation (12) and the 1's come from the identity matrix I. Here and throughout, the subindex indicates the component of the vector. In practice, especially for a diagonally dominant tridiagonal system, the magnitude of the last component of $v^{(i)}$, $v_{m-1}^{(i)}$ and the first component of $w^{(i)}$, $w_0^{(i)}$ may be smaller than machine accuracy when $p \ll n$. (See section 4 for detailed accuracy analysis.) In this case, $w_0^{(i)}$ and $v_{m-1}^{(i)}$ can be dropped, and Z becomes a diagonal block system consisting of $(p-1)2 \times 2$ independent blocks. Thus, equation (9) can be solved efficiently on parallel computers, which leads to the highly efficient parallel diagonal dominant (PDD) algorithm.

Using p processors, the PDD algorithm consists of the following steps:

Step 1. Allocate A_i, $d^{(i)}$, and elements a_{im}, $c_{(i+1)m-1}$ to the ith node, where $0 \le i \le p-1$.

Step 2. Solve equation (12). All computations can be executed in parallel on p processors.

Step 3. Send $\tilde{x}_0^{(i)}$, $v_0^{(i)}$ from the ith node to the (i − 1)th node, for $i = 1, \ldots, p-1$.

Step 4. Solve

$$\begin{bmatrix} 1 & w_{m-1}^{(i)} \\ v_0^{(i+1)} & 1 \end{bmatrix} \begin{pmatrix} y_{2i} \\ y_{2i+1} \end{pmatrix} = \begin{pmatrix} \tilde{x}_{m-1}^{(i)} \\ \tilde{x}_0^{(i+1)} \end{pmatrix}$$

in parallel on the ith node for $0 \le i \le p-2$. Then send y_{2i} from the ith node to the $(i+1)$ node, for $i = 0, \ldots, p-2$.

Step 5. Compute equations (10) and (11). We have

$$\Delta x^{(i)} = [v^{(i)}, w^{(i)}] \begin{pmatrix} y_{2i-1} \\ y_{2i} \end{pmatrix}$$

$$x^{(i)} = \tilde{x}^{(i)} - \Delta x^{(i)}$$

For each of these calculations, there are only two neighboring communications.

3.0. Reduced PDD Algorithm

The PDD algorithm has efficient communications. It achieves good load balance and is a good choice for solving a large single system. However, for systems with multiple right sides, the PDD algorithm is competitive only with the conventional sequential algorithm, the Thomas algorithm (ref. 12). It is not necessarily superior to the Thomas algorithm for compact schemes and other applications when the order of the matrix is much larger than the number of processors. The PDD algorithm has a larger operation count than the Thomas algorithm. However, for a sufficiently large number of processors and an efficient hardware platform, the computation/communication ratio of the PDD algorithm is high enough to render its performance comparable to the best case performance of the Thomas algorithm which was achieved by solving multiple right-hand sides simultaneously. The reduced PDD algorithm is proposed to further enhance computation because it has the same communication cost as the PDD algorithm but has a reduced operation count. For some applications, the reduced PDD may have a smaller operation count than the Thomas algorithm.

In the last step, step 5, of the PDD algorithm, the final solution x is computed by combining the intermediate results concurrently on each processor:

$$x^{(k)} = \tilde{x}^{(k)} - y_{2k-1} v^{(k)} - y_{2k} w^{(k)}$$

which requires $4(n-1)$ sequential operations and $4m$ parallel operations if $p = n/m$ processors are used. The PDD algorithm drops off the first element of w, w_0 and the last element of v, v_{m-1} in solving equation (9). In reference 2, we showed that, for symmetric Toeplitz tridiagonal systems (see eq. (14)), we have

$$v = \frac{1}{\lambda(a + b\Sigma_{i=0}^{m-1} b^{2i})} \left(\sum_{i=0}^{m-1} b^{2i}, \sum_{i=0}^{m-1} b^{2i}/(-b), \ldots, (-b)^{m-1} \right)^T$$

So, when m is large enough (see theorem 1 for quantitative measurement), we may drop off $v_i, i = j, j+1, \ldots, m-1$, and $w_i, i = 0, 1, \ldots, j-1$, for some integer $j > 0$, while maintaining the required accuracy. If we replace v_i by $\tilde{v}_i$, where $\tilde{v}_i = v_i$, for $i = 0, 1, \ldots, j-1$, $\tilde{v}_i = 0$, for $i = j, \ldots, m-1$; and replace w by $\tilde{w}$, where $\tilde{w}_i = w_i$ for $i = j, \ldots, m-1$, and $\tilde{w}_i = 0$, for $i = 0, 1, \ldots, j-1$; and use $\tilde{v}$, $\tilde{w}$ in step 5, we have

Step 5′:

$$\Delta x^{(k)} = [\tilde{v}, \tilde{w}] \begin{pmatrix} y_{2k-1} \\ y_{2k} \end{pmatrix}$$

$$x^{(k)} = \tilde{x}^{(k)} - \Delta x^{(k)} \tag{13}$$

It only requires $4\frac{j}{p}$ parallel operations. Replacing step 5 of the PDD algorithm by step 5′, we get the reduced PDD algorithm. The key question for the reduced PDD algorithm is how to find the smallest integer $j > 0$ that maintains the required accuracy.

4.0. Accuracy Analysis

The PDD algorithm reduces the communication from global to local. In addition to the reduced communication, the reduced PDD algorithm further reduces the computation. The PDD and reduced PDD algorithms are efficient because they have truncated communication and computation. However, this dropping may lead to an inaccurate solution. Thus, an accuracy study is essential in applying the PDD algorithm. Some preliminary study of the accuracy of the PDD algorithm has been done (refs. 5 and 13); however, the study is for general cases and only provides sufficient conditions to guarantee a given accuracy. Unfortunately, the conditions given in references 5 and 13 are difficult to verify, and the accuracy bound given is quite loose. A practical, tight error bound is given in reference 2 for a class of tridiagonal systems, symmetric Toeplitz tridiagonal (STT) systems. A matrix is Toeplitz if its entries along each diagonal are the same. As a special class of tridiagonal systems, STT systems arise in many applications. For instance, the discretization matrices of the compact scheme (ref. 4) are STT systems. In this section, we extend the recent accuracy analysis on STT systems to the reduced PDD algorithm.

The accuracy analysis of the reduced PDD algorithm is three-fold: first we study the decay rate of the decaying elements $v_{m-1}^{(i)}, w_0^{(i)} (0 \le i \le p-1)$; second, we study the influence of dropping $v_{m-1}^{(i)}, w_0^{(i)} (0 \le i \le p-1)$ on the final solution, which is the accuracy analysis of the PDD algorithm; and third, the truncation of computation is studied, based on the accuracy analysis of the PDD algorithm. The accuracy analysis of the PDD algorithm gives the error bound of the reduced PDD algorithm. The error bound of the reduced PDD algorithm is a recent result. See the following analysis.

A symmetric Toeplitz tridiagonal matrix has the form

$$A = \begin{bmatrix} \mu & \lambda & & & \\ \lambda & \mu & \lambda & & \\ & \cdot & \cdot & \cdot & \\ & & \cdot & \cdot & \lambda \\ & & & \lambda & \mu \end{bmatrix} = [\lambda, \mu, \lambda] = \lambda[1, c, 1] \tag{14}$$

Let a and b be the real solutions of

$$b + a = c, b \bullet a = 1 \tag{15}$$

where c is the diagonal element of matrix $[1, c, 1]$ given by equation (14). Because we assume $|c| > 2$, we can further assume that $|b| < 1$ and $|a| > 1$. For decay rate we have the result (ref. 2),

$$|v_{m-1}| \le b^m, |w_0| \le b^m$$

which leads to theorem 1.

Theorem 1: *For any diagonal-dominant, symmetric Toeplitz tridiagonal matrix,* $A = [\lambda, \mu, \lambda]$ *if* b^{m-1}/a *is less than machine accuracy, where a and b are the solutions of equation (15), the PDD algorithm approximates the true solution to within machine accuracy.*

Theorem 1 states that, if v_{m-1}, w_0 are less than machine accuracy, the PDD algorithm gives a satisfactory solution. In most scientific applications, the accuracy requirement is much weaker than machine accuracy. We need to study how the decay rate of v_{m-1}, w_0 influences the accuracy of the final solution. Let x be the exact solution of equation (1) and let x^* be the corresponding final solution of the PDD algorithm. We have the error bound of the PDD algorithm in l_1 norm (ref. 2):

$$\frac{\|x - x^*\|}{\|x\|} \le \frac{|b|^m}{(1-|b|)(|a|-1)} \tag{16}$$

Let $\tilde{v}, \tilde{w}$ be the vectors defined in equation (13), $\tilde{V}$ be the corresponding matrix in equation (6), consisting of all the $2(p-1)$ vectors, and let x' be the solution of the reduced PDD algorithm. Then

$$x' = \tilde{A}^{-1}d - \tilde{A}^{-1}\tilde{V}(I + E^T\tilde{A}^{-1}V)E^T\tilde{A}^{-1}d$$

similar to the accuracy analysis of the PDD algorithm (ref. 2), we let $y = (I + E^T\tilde{A}^{-1}V)E^T\tilde{A}^{-1}d$. By equation (4) and equation (55) in reference 2,

$$x' - x^* = (\tilde{A}^{-1}\tilde{V} - \tilde{A}^{-1}V)y = (\tilde{A}^{-1}\tilde{V} - \tilde{A}^{-1}V)E^{Tx}$$

Therefore, for a given integer $j > 0$,

$$\frac{\|x' - x^*\|}{\|x\|} \le \|\tilde{A}^{-1}\tilde{V} - \tilde{A}^{-1}V\| \bullet \|E^T\| = \|\tilde{A}^{-1}\tilde{V} - \tilde{A}^{-1}V\| = \|\tilde{V} - V\| = \left|\frac{1}{\lambda(a + b\Sigma_{i=0}^{m-1}b^{2i})}\right| \sum_{i=j}^{m-1} \left|\frac{(-b)^i(1 - b^{2(m-i)})}{1 - b^2}\right|$$

Since

$$\sum_{i=j}^{m-1}\left|\frac{b^i(1 - b^{2(m-i)})}{1 - b^2}\right| \le \frac{1}{|1 - b^2|}\left(\sum_{i=j}^{m-1}|b|^i + \sum_{i=j}^{m-1}|b|^{2m-i}\right) = \frac{1}{|1 - b^2|}\left(\frac{|b|^j(1 - |b|^m)}{1 - |b|} + |b|^m\frac{1 - |b|^{m-j+1}}{1 - |b|}\right)$$

$$= \frac{\left[|b|^j(1 - |b|^m) + |b|^m(1 - |b|^{m-j+1})\right]}{|1 - b^2|(1 - |b|)}$$

$$\frac{\|x' - x^*\|}{\|x\|} \le \left|\frac{1}{\lambda a}\right|\left|\frac{1 - b^2}{1 - b^{2(m+i)}}\right| \bullet \frac{|b|^j(1 - |b|^m) + |b|^m(1 - |b|^{m-j+1})}{|(1 - b^2)|(1 - |b|)} \le \frac{|b|^j + |b|^m}{|\lambda(|a| - 1)|}$$

By inequality (16), inequality (18) gives the error bound of the reduced PDD algorithm.

$$\frac{\|x - x'\|}{\|x\|} \le \frac{\|x - x^*\|}{\|x\|} + \frac{\|x^* - x'\|}{\|x\|} \tag{17}$$

$$\le \frac{|b|^m}{\left|\lambda\left(|\lambda| - \left|\frac{b(1 - b^{2m})}{1 - b^{2(m+1)}}\right|\right)(|a| - 1)\right|} + \frac{|b|^j + |b|^m}{|\lambda(|a| - 1)|} \tag{18}$$

For a given error tolerance $\varepsilon > 0$, the right side of inequality (18)

$$\frac{|b|^m}{\left|\lambda\left(|\lambda| - \left|\frac{b(1 - b^{2m})}{1 - b^{2(m+1)}}\right|\right)(|a| - 1)\right|} + \frac{|b|^j + |b|^m}{|\lambda(|a| - 1)|} < \varepsilon$$

if and only if

$$j > \frac{\log\left[|\lambda|(|a| - 1)\left(\varepsilon - \frac{|b|^m}{|\lambda|(|a| - 1)}\left\{1 + 1/\left[|\lambda| - \left|\frac{b(1 - b^{2m})}{1 - b^{2(m+1)}}\right|\right]\right\}\right)\right]}{\log|b|} \tag{19}$$

When $\left|\frac{b}{\lambda}\right| < 1$

$$\frac{\|x - x'\|}{\|x\|} \le \frac{|b|^m}{|\lambda|(|a| - 1)}\left(1 + \frac{1}{|\lambda| - |b|}\right) + \frac{|b|^j}{|\lambda|(|a| - 1)}$$

and we get a simpler inequality for the minimal number j

$$j > \frac{\log|\lambda|(|a|-1)\left[\varepsilon - |b|^m|\lambda|(|a|-1)\left(1+\frac{1}{|\lambda|-|b|}\right)\right]}{\log|b|} \tag{20}$$

When $|b^m|$ is less than machine accuracy, inequality (19) becomes the same as inequality (20), and we have an even simpler formula:

$$j > \frac{\log|\lambda|(|a|-1)\varepsilon}{\log|b|} \tag{21}$$

Inequality equation (21) gives a lower bound of the number of variables that need to be modified in equation (13) for a given error tolerance $\varepsilon > 0$. Usually, j is quite small. For instance, when error tolerance ε equals 10^{-4}, j equals either 10 or 7 when λ, the magnitude of the off-diagonal elements, equals $\frac{1}{3}$ or $\frac{1}{4}$, respectively, the diagonal elements being equal to 1. The integer j reduces to 4 for $0 < \lambda \leq \frac{1}{9}$.

5.0. Experimental Results

The PDD and the reduced PDD algorithms were implemented on the 48-node IBM SP2 and 72-node Intel Paragon available at Langley Research Center. Both the SP2 and Paragon machines are distributed-memory parallel computers that adopt message-passing communication paradigms and support virtual memory. Each processor (node) of the SP2 is either functionally equivalent to a reduced instruction set computer (RISC) System/6000 desktop system (thin node) or a RISC System/6000 deskside system (wide node). The Paragon XP/S supercomputer uses the i860 XP microprocessor that includes a RISC integer core processing unit and three separate on-chip caches for page translation, data, and instructions. The Langley SP2 has 48 wide nodes with 128 Mbytes local memory and peak performance of 266 million floating-point operations per second (MFLOPS) each. In contrast, the Langley Paragon has 72 nodes with 32 Mbytes of local memory and peak performance of 75 MFLOPS each. The heart of all distributed-memory parallel computers is the interconnection network that links the processors together. The SP2 high-performance switch is a multistage packet-switched Omega network that provides a minimum of four paths between any pair of nodes in the system. The Intel Paragon processors are connected in a two-dimensional (2-D) rectangular mesh topology. The diameter of the 2-D mesh topology increases with the number of processors. Communication delay on a message-passing distributed-memory machine usually can be modeled by using two parameters, the latency (start time) α and transmission rate (in terms of transmission time per byte) β. For the SP2, the latency is 30 μsec and transmission rate is 2 μsec. For Paragon, the latency is 46 μsec and transmission rate is 6 μsec.

Table 1 gives the computation and communication count of the PDD algorithm. The best conventional sequential algorithm for the LU decomposition method

Table 1. Computation and Communication Counts of PDD Algorithm

System	Matrix	Best sequential	PDD	
			Computation	Communication
Single	Nonperiodic	$8n-7$	$17\frac{n}{p}-4$	$2\alpha+12\beta$
	Periodic	$14n-16$	$17\frac{n}{p}-4$	$2\alpha+12\beta$
Multiple right sides	Nonperiodic	$(5n-3)\bullet n1$	$\left(9\frac{n}{p}+1\right)\bullet n1$	$(2\alpha+8n1\bullet\beta)$
	Periodic	$(7n-1)\bullet n1$	$\left(9\frac{n}{p}+1\right)\bullet n1$	$(2\alpha+8n1\bullet\beta)$

for tridiagonal systems is the Thomas algorithm (ref. 14). For most distributed-memory computers, the time to communicate with nearest neighbors varies linearly with problem size. Let S be the number of bytes to be transferred. Then the transfer time to communicate with a neighbor can be expressed as $\alpha + S\beta$. Assuming 4 bytes are used for each real number, steps 3 and 4 of the PDD and reduced PDD algorithm take $\alpha + 8\beta$ and $\alpha + 4\beta$ time, respectively, on any architecture that supports single array topology. Tridiagonal systems arising in both ADI and compact scheme methods, which are two widely used methods in CFD applications, are multiple right-side systems. They are usually "kernels" in much larger codes. The computation and communication counts for solving multiple right-side systems are listed in table 1, in which the factorization of matrix $\tilde{A}$ and computation of Y are not considered (see eqs. (5) and (6) in Section 2). Parameter $n1$ is the number of right-hand sides. Note that, for multiple right-side systems, the communication cost increases with the number of right-hand sides. If the boundary conditions are periodic, the tridiagonal systems arising in CFD applications are periodic tridiagonal systems. As shown in reference 2, the PDD algorithm, and consequently the reduced PDD algorithm, can be extended to solve periodic tridiagonal systems as well. Table 1 also lists computing and communication counts for solving periodic systems.

Table 2 gives the computation and communication counts of the reduced PDD algorithm. As for the PDD algorithm, it has the same parallel computation and communication counts for both periodic and nonperiodic systems. The computational saving of the reduced PDD algorithm is not only in step 5, the final modification step, but also in other steps. Because we only need j elements of vectors v and w for the final modification in the reduced PDD algorithm (eq. (13) in section 3), we only need to compute j elements for each column of V in solving equation (6). The integer j is given by equations (19), (20), or (21), depending on the particular circumstance. Notice that, when $j < n/2$, the reduced PDD algorithm has a smaller operation count than that of the Thomas algorithm for periodic systems with multiple right-hand sides.

Table 2. Computation and Communication Counts of Reduced PDD Algorithm

System	Reduced PDD	
	Computation	Communication
Single	$11\frac{n}{p} + 6j - 4$	$2\alpha + 12\beta$
Multiple right sides	$\left(5\frac{n}{p} + 4j + 1\right) \bullet n1$	$(2\alpha + 8n1 \bullet \beta)$

While the accuracy analyses given in this study are for Toeplitz tridiagonal systems, the PDD algorithm and the reduced PDD algorithm can be applied for solving general tridiagonal systems. The computation counts given in tables 1 and 2 are for general tridiagonal systems. For symmetric Toeplitz tridiagonal systems, a fast method proposed by Malcolm and Palmer (ref. 15) has a smaller computation count than the Thomas algorithm for systems with single right-hand sides. It requires only $5n + 2k - 3$ counts for arithmetic, where k is a decay parameter, depending on the diagonal dominancy of the system. Formulas are available to compute the upper and lower bounds of parameter k (ref. 15). The computation savings of Malcolm and Palmer's method are in the LU decomposition. For systems with multiple right-hand sides, in which the factorization cost is not considered, the Malcolm and Palmer's method and the Thomas method have the same computation count. Table 3 gives the computation and communication counts of the PDD and reduced PDD algorithms based on Malcolm and

Table 3. Computation and Communication Counts for Symmetric Toeplitz Systems

Algorithm	Matrix	Best sequential	Parallel Algorithm	
			Computation	Communication
PDD Algorithm	Nonperiodic	$5n + 2k - 3$	$14\frac{n}{p} + 2k$	$2\alpha + 12\beta$
	Periodic	$11n + 2k - 12$	$14\frac{n}{p} + 2k$	$2\alpha + 12\beta$
Reduced PDD Algorithm	Nonperiodic	$5n + 2k - 3$	$8\frac{n}{p} + 2k + 6j$	$2\alpha + 8\beta$
	Periodic	$11n + 2k - 12$	$8\frac{n}{p} + 2k + 6j$	$2\alpha + 8\beta$

Palmer's algorithm. The computation counts of the two algorithms are reduced with the fast method used in solving the subsystems. Table 3 shows the computation and communication counts for solving systems with a single right-hand side. For systems with multiple right-hand sides, the computation counts remain the same as in tables 1 and 2 for both the PDD and the reduced PDD algorithms, respectively.

As an illustration of the algorithm and theoretical results given in previous sections, a sample matrix is tested here. This sample matrix is a periodic, symmetric, Toeplitz system

$$A = \begin{bmatrix} 1 & \frac{1}{3} & & & & \frac{1}{3} \\ \frac{1}{3} & 1 & \frac{1}{3} & & & \\ & \cdot & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & & & \frac{1}{3} & 1 & \frac{1}{3} \\ \frac{1}{3} & & & & \frac{1}{3} & 1 \end{bmatrix} = \left[\frac{1}{3}, 1, \frac{1}{3}\right] \qquad (22)$$

which arises in the compact scheme. We have

$$A = \left[\frac{1}{3}, 1, \frac{1}{3}\right] = \frac{1}{3} \bullet [1, 3, 1]$$

$$= \frac{1}{3} \bullet ([b, 1, 0] \times [0, a, 0] \times [0, 1, b] - \Delta B)$$

where ΔB is an $n \times n$ zero matrix, except that the first element on the first row is b, and

$$\lambda = \frac{1}{3}, c = 3, a = \frac{3 + \sqrt{5}}{2}, b = \frac{3 - \sqrt{5}}{2} \qquad (23)$$

The reduced PDD algorithm was first implemented on a Sun workstation with double precision to solve the tridiagonal system $Ax = d$ for accuracy checking. The right-side vector d was generated randomly. Figure 1 depicts the accuracy comparison of the reduced PDD algorithm. The measured and predicted data have been converted to a common logarithm scale to make the difference visible. The x-coordinate is the order of matrix A_i, and the y-coordinate is the relative error in the 1-norm. From figure 1, we can see that the accuracy analysis provides a very good error bound.

Speedup, one of the most frequently used performance metrics in parallel processing, is defined as sequential execution time over parallel execution time.

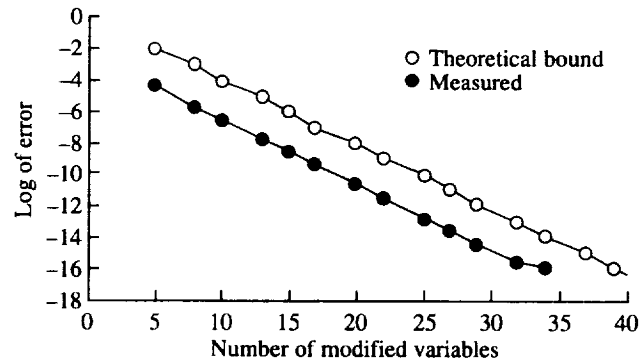

Figure 1. Measured and predicted accuracy of reduced PDD algorithm.

Parallel algorithms often exploit parallelism by sacrificing mathematical efficiency. To measure the true parallel processing gain, the sequential execution time should be based on a commonly used sequential algorithm. To distinguish it from other interpretations of speedup, the speedup measured versus a commonly used sequential algorithm has been called *absolute* speedup (ref. 7). Another widely used interpretation is the *relative* speedup (ref. 7), which uses the uniprocessor execution time of the parallel algorithm as the sequential time. Relative speedup measures the performance variation of an algorithm in terms of the number of processors and is commonly used in scalability studies. Both Amdahl's law (ref. 16) and Gustafson's scaled speedup (ref. 17) are based on relative speedup. In this study, we first use relative speedup to study the scalability of the PDD and reduced PDD algorithms; then, we use the absolute speedup to compare these two algorithms with the conventionally used sequential algorithm.

Because execution time varies with communication/computation ratio on a parallel machine, the problem size is an important factor in performance evaluation, especially for machines supporting virtual memory. Virtual address space separates the user logical memory from physical memory. This separation allows an extremely large virtual memory to be provided (with a much slower memory access time) on a sequential machine when only a small physical memory is available. If the problem size is larger than physical memory, data must be swapped in from and out to secondary memory, which may lead to inefficient sequential processing and unreasonably high speedup. If the problem size is too small, on the other hand, when the number of processors increases, the workload on each processor will drop quickly, which may lead to an extremely high communication/computation ratio and unacceptably low performance. As studied in reference 18, the correct choice of initial problem size is the problem size that reaches the asymptotic speed, the sustained uniprocessor speed corresponding to the main memory access (ref. 18). The nodes of SP2 and Paragon have different processing powers and local memory sizes. For a fixed 1024 right sides, following the asymptotic speed concept, the order of matrix for SP2 was found to be 6400, and the order of matrix for Paragon was found to be 1600. Figures 2 and 3 show the measured speedup of the PDD algorithm when the large problem size $n = 6400$ is solved on Paragon and the small problem size $n = 1600$ is solved on SP2. For comparison, ideal speedup, where speedup equals p when p processors are available, is also plotted with the measured speedups. As indicated previously, the large problem size leads to an unreasonable superlinear speedup on Paragon, and the small problem size leads to a disappointingly low performance on SP2.

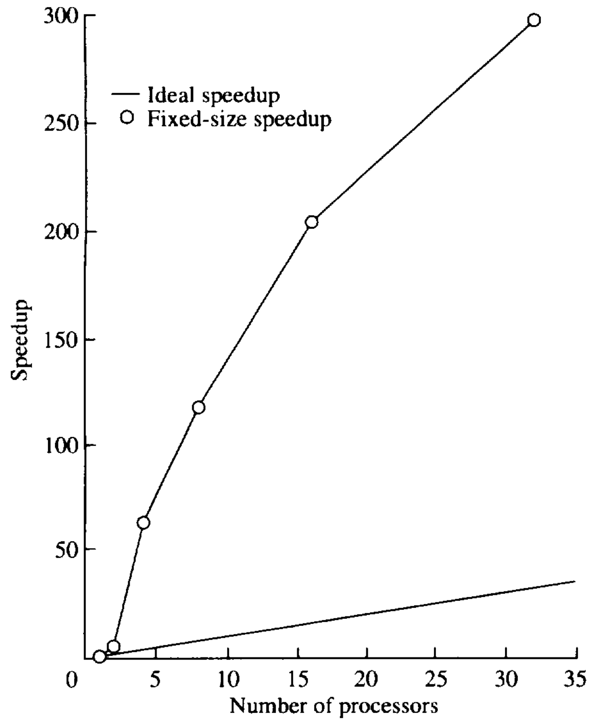

Figure 2. Superlinear speedup with large problem size on Intel Paragon (1024 system of order 6400).

From the problem size point of view, speedup can be divided into the *fixed-size* speedup and the *scaled* speedup. Fixed-size speedup fixes the problem size. Scaled speedup scales the problem size with the number of processors. Fixed-size speedup emphasizes how much execution time can be reduced for a given application with parallel processing. Amdahl's law (ref. 16) is based on the fixed-size speedup. The scaled speedup concentrates on exploring the computational power of parallel computers for solving otherwise intractable large problems. Depending on the scaling restrictions of the problem size, the scaled speedup can be classified as both the *fixed-time* speedup (ref. 17) and the *memory-bounded* speedup (ref. 19). As the number of processors increases, memory-bounded speedup scales problem size to utilize the associated memory increase. In general, operation count increases much faster than memory requirement. Therefore, the workload on each processor will not decrease with the increase in number of processors in memory-bounded speedup. Thus, scaled speedup is more likely to get a higher speedup than that of fixed-size speedup.

Figures 4 and 5 depict the speedup of the fixed-size and memory-bounded speedup of the PDD and the reduced PDD algorithm, respectively, on the Intel Paragon. From figures 4 and 5 we can see that the PDD and the reduced PDD algorithm have the same speedup pattern. This similarity is very reasonable because these two algorithms share the same computation and communication pattern. It has been proven that the PDD algorithm, and therefore the reduced PDD algorithm, are perfectly scalable, in terms of isospeed scalability (ref. 20), on any

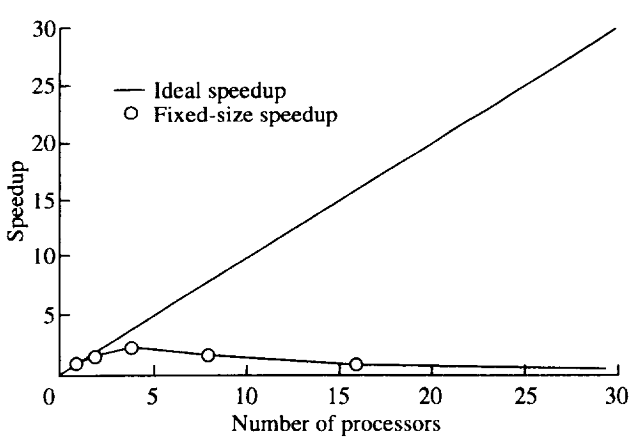

Figure 3. Inefficient performance with small problem size on SP2 (1024 system of order 1600).

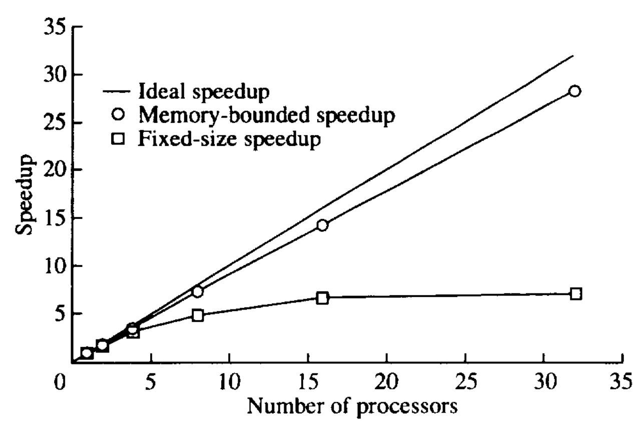

Figure 4. Measured speedup of PDD algorithm on Intel Paragon (1024 system of order 1600).

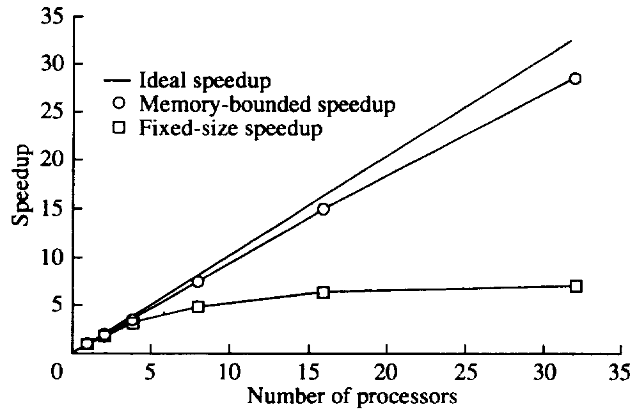

Figure 5. Measured speedup of reduced PDD algorithm on Intel Paragon (1024 system of order 1600).

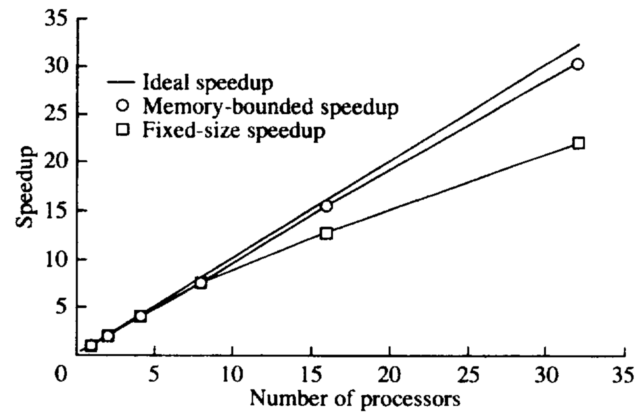

Figure 6. Measured speedup of PDD algorithm on a SP-2 (1024 system of order 6400).

architecture that supports the ring communication network. However, ring communication cannot be embedded in 2-D mesh topologies perfectly unless a wraparound is supported. Thus, the communication cost of the algorithms increases slightly with the increase in the number of processors. The fact that the memory-bounded speedups on the Paragon are slightly below the ideal speedup is very reasonable. The influence of the communication cost has been reflected in the measured speedup.

Figure 6 demonstrates the speedups of the PDD algorithm on the SP2 machine. Because the one-to-one communication of the SP2 multistage Omega network does not increase with the number of processors, the PDD algorithm reaches the ideal memory-bounded speedup. In accordance with the isospeed metric (ref. 20), the PDD algorithm is perfectly scalable in the multistage SP2 machine.

Although the PDD and reduced PDD have similar relative speedup patterns, the execution times of the two algorithms are very different. The reduced PDD algorithm has a smaller execution time than that of the PDD algorithm. For periodic systems the reduced PDD algorithm has an even smaller execution time than the conventional sequential algorithm. The timing of the Thomas algorithm, the PDD algorithm, and the reduced PDD algorithm on a single node of the SP2 and Paragon machine are listed in table 4. The problem size for all algorithms on SP2 is $n = 6400$ and $n1 = 1024$ and on

Table 4. Sequential Timing (in seconds) on Paragon and SP2 Machines

	Size	Thomas algorithm	PDD algorithm	Reduced PDD algorithm
Paragon	1600	0.8265	0.9026	0.6432
SP2	6400	0.7387	0.856	0.5545

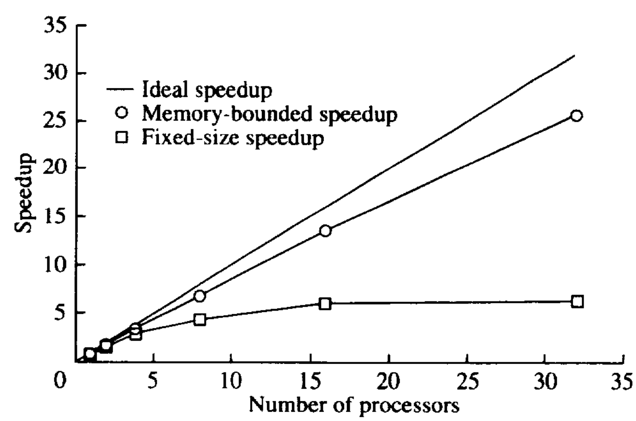

Figure 7. Speedup of PDD algorithm over Thomas algorithm (1024 systems of order 1600).

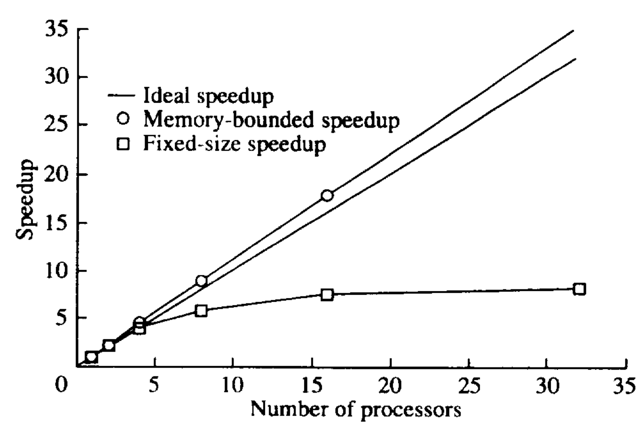

Figure 8. Speedup of reduced PDD algorithm over Thomas algorithm (1024 systems of order 1600).

Paragon is $n = 1600$ and $n1 = 1024$. The measured results confirm the analytical results given in tables 1 and 2.

Figures 7 and 8 show the speedup of the PDD and reduced PDD algorithms over the conventional sequential algorithm, the Thomas algorithm, respectively. The

PDD algorithm increases computation count for high parallelism. The reduced PDD reduces computation count by taking advantage of diagonal dominance. Compared to the Thomas algorithm, while the absolute speedup of the PDD algorithm is worse than its relative speedup, the reduced PDD algorithm has a better absolute speedup than its relative speedup. The reduced PDD algorithm achieves a superlinear speedup over the Thomas algorithm. Experimental results confirm that the reduced PDD algorithm maintains the good scalability of the PDD algorithm and delivers an efficient performance in terms of execution time as well.

6.0. Concluding Remarks

Computational fluid dynamics (CFD) constantly demands higher computing power than is currently available. With current technology, the feasible approach to continually increasing computing power seems to be through parallel processing: construct a computer that consists of hundreds and thousands of processors working concurrently. Parallel computers have become commercially available; however, unlike their sequential counterparts, efficient parallel algorithms require a high degree of parallelism and low cost of communication.

Tridiagonal systems arise in many CFD applications. They are usually kernels in much larger codes. Although multiple tridiagonal systems are available in the larger codes, in many situations it is often more efficient to use a parallel tridiagonal solver for these systems than to remap data among processors to be able to perform a serial solve, especially for distributed memory machines where communication cost is high.

Experimental and theoretical results show that both the PDD and reduced PDD algorithms are efficient and scalable, even for systems with multiple right sides. For periodic systems, as confirmed by our implementation results, the reduced PDD algorithm even has a smaller sequential execution time than that of the best sequential algorithm. The two algorithms are good candidates for parallel computers.

The accuracy analysis and implementation given in this study are for Toeplitz systems. The PDD and reduced PDD algorithms, however, are applicable for general tridiagonal and narrow band linear systems. The common merit of these two algorithms is the minimum communication required, which makes them even more valuable in a distributed computing environment, such as the environment of a cluster of a network of workstations.

NASA Langley Research Center
Hampton, VA 23681-0001
March 19, 1996

References

1. Committee on Physical, Mathematical, and Engineering Sciences: *Grand Challenges 1993—High Performance Computing and Communications.* PB92-16530, Natl. Sci. Found., Jan. 1992.

2. Sun, X.-H.: Application and Accuracy of the Parallel Diagonal Dominant Algorithm. *Parallel Comput.*, vol. 21, no. 8, Aug. 1995, pp. 1241–1268.

3. Taha, Thiab R.; and Jiang, Peiqing: A Parallel Algorithm for Solving Periodic Tridiagonal Toeplitz Linear Systems. *Proceedings of the Sixth SIAM Conference on Parallel Processing for Scientific Computing*, Richard Sincovec, et al., eds., SIAM, Mar. 1993, pp. 491–496.

4. Lele, Sanjiva K.: Compact Finite Difference Schemes With Spectral-Like Resolution. *J. Comput. Phys.*, vol. 103, no. 1, Nov. 1992, pp. 16–42.

5. Sun, X.-H.; Zhang, H.; and Ni, L. M.: Efficient Tridiagonal Solvers on Multicomputers. *IEEE Trans. Comput.*, vol. 41, no. 3, Mar. 1992, pp. 286–296.

6. Lambiotte, Jules J., Jr.; and Voigt, Robert G.: The Solution of Tridiagonal Linear Systems on the CDC STAR-100 Computer. *ACM Trans. Math. Softw.* vol. 1, no. 4, Dec. 1975, pp. 308–329.

7. Ortega, J. M.; and Voigt, R. G.: Solution of Partial Differential Equations on Vector and Parallel Computers. *SIAM Rev.*, June 1985, pp. 149–240.

8. Stone, Harold S.: An Efficient Parallel Algorithm for the Solution of a Tridiagonal Linear System of Equations. *J. Assoc. Comput. Mach.*, vol. 20, no. 1, Jan. 1973, pp. 27–38.

9. Hockney, R. W.: A Fast Direct Solution of Poisson's Equation Using Fourier Analysis. *J. Assoc. Comput. Mach.*, vol. 12, no. 1, Jan. 1965, pp. 95–113.

10. Sun, X.-H.; and Joslin, R.: A Massively Parallel Algorithm for Compact Finite Difference Schemes. *Proceedings of the 1994 International Conference on Parallel Processing*, Aug. 1994, pp. III-282–III-289.

11. Sherman, Jack; and Morrison, Winifred J.: Adjustment of an Inverse Matrix Corresponding to a Change in One Element of a Given Matrix. *Ann. Math. Stat.*, vol. 21, 1950, pp. 124–127.

12. Hirsch, Charles: *Numerical Computation of Internal and External Flows.* John Wiley & Sons, 1988.

13. Zhang, H.: On the Accuracy of the Parallel Diagonal Dominant Algorithm. *Parallel Comput.*, vol. 17, nos. 2–3, June 1991, pp. 265–272.

14. Strikwerda, John C.: *Finite Difference Schemes and Partial Differential Equations.* Wadsworth & Brooks/Cole Advanced Books & Software, 1989.

15. Malcolm, Michael A.; and Palmer, John: A Fast Method for Solving a Class of Tridiagonal Linear Systems. *Commun. ACM*, vol. 17, no. 1, Jan. 1974, pp. 14–17.

16. Amdahl, Gene M.: Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities. *AFIPS Conference Proceedings—Volume 30, Spring Joint Computer Conference*, Apr. 1967, pp. 483–485.

17. Gustafson, John L.: Reevaluating Amdahl's Law. *Commun. ACM*, vol. 31, no. 5, May 1988, pp. 532–533.

18. Sun, Xian-He; and Zhu, Jianping: Shared Virtual Memory and Generalized Speedup. *Proceedings of the Eighth International Parallel Processing Symposium*, Apr. 1994, pp. 637–643.

19. Sun, Xian-He; and Ni, Lionel M.: Scalable Problems and Memory-Bounded Speedup. *J. Parallel & Distrib. Comput.*, vol. 19, 1993, pp. 27–37.

20. Sun, Xian-He; and Rover, D. T.: Scalability of Parallel Algorithm-Machine Combinations. *IEEE Trans. Parallel and Distrib. Syst.*, vol. 5, issue 6, pp. 599–613.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	August 1996	Technical Paper

4. TITLE AND SUBTITLE
A Fast Parallel Tridiagonal Algorithm for a Class of CFD Applications

5. FUNDING NUMBERS
WU 509-10-24-01

6. AUTHOR(S)
Xian-He Sun and Stuti Moitra

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
NASA Langley Research Center
Hampton, VA 23681-0001

8. PERFORMING ORGANIZATION REPORT NUMBER
L-17510

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
NASA TP-3585

11. SUPPLEMENTARY NOTES
Sun: Louisana State University, Baton Rouge, LA; Moitra: Langley Research Center, Hampton, VA.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified–Unlimited
Subject Category 59
Availability: NASA CASI (301) 621-0390

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*
The parallel diagonal dominant (PDD) algorithm is an efficient tridiagonal solver. This paper presents for study a variation of the PDD algorithm, the reduced PDD algorithm. The new algorithm maintains the minimum communication provided by the PDD algorithm, but has a reduced operation count. The PDD algorithm also has a smaller operation count than the conventional sequential algorithm for many applications. Accuracy analysis is provided for the reduced PDD algorithm for symmetric Toeplitz tridiagonal (STT) systems. Implementation results on Langley's Intel Paragon and IBM SP2 show that both the PDD and reduced PDD algorithms are efficient and scalable.

14. SUBJECT TERMS
Parallel algorithm; Parallel tridiagonal solver; CFD application; Symmetric Toeplitz tridiagonal solver

15. NUMBER OF PAGES
15

16. PRICE CODE
A03

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102