A RIGOROUS ANALYSIS OF TIME DOMAIN PARALLELISM*

A. DESHPANDE†, S. MALHOTRA† , C. C. DOUGLAS‡ AND M. H. SCHULTZ†

Abstract. Time dependent partial differential equations are often solved using algorithms which parallelize the solution process in the spatial domain. However, as the number of processors increases, the parallel efficiency is limited by the increasing communication/computation ratio. Recently, several researchers have proposed algorithms incorporating time domain parallelism in order to increase efficiency. In this paper we discuss a class of such algorithms and analyze it rigrously.

Key words. iterative methods, time domain parallelism, partial differential equations.

AMS(MOS) subject classifications. G.1.0. Parallel Algorithms [Numerical Analysis]; G.1.3. Linear Systems (Direct and Iterative methods); G.1.8. Parabolic Equations (Partial Differential Equations); D.1.3. Parallel Programming.

1. Introduction. We investigate a parallel algorithm for the numerical solution of linear, time-dependent partial differential equations of the form

$$\begin{cases} \frac{\partial u}{\partial t} + \mathcal{L}u = f, & x \in \Omega, \quad 0 < t < T, \\ \mathcal{B}(u) = u_b(t), & x \in \partial\Omega, \quad 0 < t < T, \\ u(x,0) = u_0(x), & x \in \Omega \end{cases}$$

where $\mathcal{L}$ is a linear elliptic spatial operator, $\mathcal{B}$ is the boundary operator, and Ω is a spatial domain with boundary $\partial\Omega$. For simplicity of presentation we choose the one dimensional problem $\Omega = (0,1)$ and Dirichlet boundary conditions $u(0,t) = u(1,t) = 0$. Our model problem in this paper will be the one dimensional heat equation.

Implicit time stepping schemes, coupled with finite difference approximations of the spatial derivatives, lead to linear systems at each time step t_k, of the form,

$$Au^k = Bu^{k-1} + b, \qquad k = 1, 2, \ldots n, \tag{1}$$

where A and B are $m \times m$ matrices (m is the number of grid points in the interior of the domain) whose elements depend on $\mathcal{L}$ and $\mathcal{B}$ and u^k is an m–vector containing function values u at all the grid points at the $k'th$ time step. Starting from u^0, which is known from the initial condition, we can use an iterative algorithm such as Jacobi, Gauss-Seidel, or SOR to solve (1) sequentially for each time step:

$$u_i^k = Tu_{i-1}^k + c, \qquad i = 1, 2, \ldots, \qquad k = 1, 2, \ldots n,$$

where T is the iteration matrix and c is a vector of known values.

Usually, the entire process is spatially parallelized by splitting the domain Ω into subdomains and distributing problems on the subdomains to multiple processors (see [3] and [5]). At each iteration, the processors need to exchange boundary information with processors holding adjacent subdomains. As the number of processors increases, the communication/computation ratio increases making the parallel efficiency decrease. In an effort to forestall this and to allow increasing numbers of processors to

* This work is supported in part by ONR Grant # N0014-91-J-1576 and by an IBM/Yale joint study.

† Yale Center for Parallel Supercomputing, Department of Computer Science, Yale University, New Haven, CT 06520-8285.

‡ IBM Research Division, T. J. Watson Research Center, Yorktown Heights, NY 10598-0218.

be used effectively, a number of researchers have suggested algorithms which introduce time domain parallelism as well as space domain parallelism [4, 7, 8, 9, 10].

In this paper, we investigate a time domain parallel algorithm for solving (1). Other authors have considered a similar algorithm ([4], [9] and [10]), but not from a rigorous theoretical point of view. We demonstrate that the parallel and serial algorithms do not converge in the same number of iterations. We also provide a convergence theory for the parallel algorithm and computational experiments, which show that the number of iterations required for convergence in the parallel scheme is just enough to negate the advantages of time domain parallelism in case of the model problem considered above. We note at this stage that the parallel approach has been successfully used to solve certain nonlinear problems [2] and time domain parallelism may indeed be a viable and useful approach in those application areas.

2. The Temporal Method. Consider an iterative scheme for time domain parallelism by solving the linear systems at different time steps simultaneously. The central idea is to assemble n steps of (1) into the form

$$(2) \qquad Gu \equiv \begin{pmatrix} A & & & & \\ -B & A & & & \\ & -B & \ddots & & \\ & & \ddots & A & \\ & & & -B & A \end{pmatrix} \begin{pmatrix} u^1 \\ u^2 \\ \vdots \\ u^{n-1} \\ u^n \end{pmatrix} = \begin{pmatrix} u^0 + b_1 \\ b_2 \\ \vdots \\ b_{n-1} \\ b_n \end{pmatrix} \equiv b.$$

Equation (2) can be solved in parallel using an iterative scheme such as Jacobi or Gauss-Seidel with the red-black block ordering. If $A = D - L - U$, where D, L and U are diagonal, lower triangular, and upper triangular, then the iteration matrix T_A for solving (1) using Gauss-Seidel is $T_A = (D - L)^{-1}U$.

The iteration matrix T_G for Gauss-Seidel for (2) is $T_G = (\bar{D} - \bar{L})^{-1}\bar{U}$, where $\bar{D}, \bar{L}$ and $\bar{U}$ are the diagonal, lower triangular, and upper triangular parts of G. As was shown in [9], T_G is a block lower triangular matrix with diagonal blocks equal to $(D - L)^{-1}U$. Hence, the eigenvalues of T_G are the collections of all eigenvalues of all the block matrices on the main diagonal. Since all blocks on the main diagonal are equal to T_A, T_G has the same eigenvalues as T_A but with higher multiplicity. This shows that

$$\varrho(T_A) = \varrho(T_G),$$

where $\varrho(T)$ is the spectral radius of T. Hence, both iterations appear to converge at the same rate. This would seem to imply that the algorithm using time domain parallelism is far superior to those employing only spatial domain parallelism. However, we demonstrate below that this is not valid.

Any stationary linear iteration scheme can be written in the form $u_{k+1} = Tu_k + c$ where T is the iteration matrix and c is a vector of known values. The error e_k in the k^{th} approximation to the solution is given by $e_k = T^k e_0$.

Hence, the sequence of iterates $u_1, u_2, \ldots, u_k, \ldots$ will converge to the true solution as $k \to \infty$ if and only if $\lim_{k\to\infty} T^k = 0$ since u_0, and hence e_0, is arbitrary. If the $m \times m$ matrix T has m linearly independent eigenvectors $v_s, s = 1, \ldots, m$, then it follows that

$$(3) \qquad e_k = \sum_{s=1}^{m} c_s \lambda_s^k v_s,$$

where λ_s is the eigenvalue corresponding to v_s. Thus e_k will tend to 0 for an arbitrary e_0 if and only if $\mid \lambda_s \mid < 1$ for all s, i.e., if and only if $\varrho(T) < 1$, and the convergence rate behaves like ϱ^k.

While the preceding result is true in general (see [6]), this argument holds only if the matrix T has m linearly independent eigenvectors. If T is defective, i.e., lacking eigenvectors, then we may not be able to express e_0 as a linear combination of the eigenvectors and (3) may not hold. In this case, we cannot claim that the convergence rate behaves like ϱ^k except in an asymptotic sense, which may not yield any useful information.

Now, consider solving (2) using a block Jacobi iteration scheme, where each block corresponds to one time step. The resulting iteration matrix is

$$T = \begin{pmatrix} 0 & & & \\ A^{-1} & 0 & & \\ & \ddots & \ddots & \\ & & A^{-1} & 0 \end{pmatrix}.$$

T is an $mn \times mn$ matrix, where m is the order of A and n is the number of time steps we are attempting to solve simultaneously. This matrix is defective for any matrix A and all its eigenvalues are 0. Hence, the spectral radius does not give a true picture of the convergence rate and one must actually look at the norm of the k^{th} power of the iteration matrix, T^k, to determine the number of iterations it takes for the iterative scheme to converge. Since $\|e_k\|_2 = \|T^k e_0\|_2 \leq \|T^k\|_2 \|e_0\|_2$ and there exists at least one initial error vector e_0 for which the equality holds, we can only assert convergence if the norm of the error is reduced in k iterations by a factor of $\epsilon < 1$, i.e. if

$$\|T^k\|_2 \leq \epsilon < 1, \tag{4}$$

we can determine lower and upper bounds on the number of iterations for convergence by determining k such that (4) holds. We will show that the number of iterations required to solve (2) is enough to negate any advantages of time domain parallelism.

3. Block Iterative Methods. In this section, we derive bounds on $\|T^k\|_2$ for block Jacobi and Gauss-Seidel methods.

In order to simplify the notation in this section, we restrict our attention to backward Euler and Crank-Nicolson difference schemes. First, consider the block Jacobi iterative method. In the case of the backward Euler scheme, the matrices in (1) are $B = I_m$ and

$$A_{m\times m} = \begin{pmatrix} 1+2r & -r & & \\ -r & 1+2r & \ddots & \\ & \ddots & \ddots & -r \\ & & -r & 1+2r \end{pmatrix},$$

where $r \equiv \frac{(m+1)^2}{n} \equiv \frac{\Delta t}{(\Delta x)^2}$. The number of timesteps being being solved for in parallel is n.

The k^{th} power of the iteration matrix is given by

$$T_{BJ}^k = \begin{pmatrix} 0 & & & & \\ \vdots & 0 & & & \\ A^{-k} & & \ddots & & \\ & \ddots & & 0 & \\ & & A^{-k} & \dots & 0 \end{pmatrix}.$$

Let t_p and t_s be the wall clock times for the time parallel and serial algorithms, respectively. In the serial case the time, t_s, for n time steps is ns, where s is the serial time to solve $Ax = b$.

THEOREM 3.1. *The total wall clock time using n processors and ignoring communication time is*

$$t_p \geq \frac{1}{\pi^2} \ln\left(\frac{1}{\epsilon}\right) t_s$$

for the block Jacobi method applied to the backward Euler and Crank-Nicolson discretization of the heat equation.

Proof. We prove the above statement for the backward Euler method. A proof for the Crank-Nicolson case can be found in [1]. Note that

$$\|T_{BJ}^k\|_2 = \|A^{-k}\|_2 = \frac{1}{\lambda_{min}^k(A)}.$$

A simple analysis shows that

$$\lambda_{min}(A) = 1 + 4r\sin^2\left(\frac{\pi}{2(m+1)}\right) \leq 1 + \frac{\pi^2}{n}.$$

Hence,

$$\|T_{BJ}^k\|_2 \geq [1 + \frac{\pi^2}{n}]^{-k}.$$

To be convergent, we need that $\|T_{BJ}^k\|_2 \leq \epsilon < 1$. This holds if

$$k \geq n\frac{\ln\left(\frac{1}{\epsilon}\right)}{\pi^2} = O(n).$$

The total work in aggregate in the parallel scheme is

$$W_p = k \cdot work/iteration = kns \geq \frac{sn^2}{\pi^2} \ln\left(\frac{1}{\epsilon}\right).$$

Since $t_s = sn$, the total wall clock time using n processors, *assuming no communication overhead*, is

$$t_p = \frac{W_p}{n} \geq \frac{ns}{\pi^2} \ln\left(\frac{1}{\epsilon}\right) = \frac{\ln\left(\frac{1}{\epsilon}\right)}{\pi^2} t_s. \qquad \blacksquare$$

□

m = 60			m = 90		
n	k		n	k	
	Jacobi	Gauss-Seidel		Jacobi	Gauss-Seidel
16	16	8	16	16	8
22	22	11	22	22	11
28	28	14	28	28	14
34	34	17	34	34	17
40	40	20	40	40	20
46	43	23	46	46	23
52	45	24	52	52	26
58	46	24	58	58	29

TABLE 1
Iterations (k) required for convergence of block methods.

Corollary: Even if we ignore the cost of communication, the total wall clock time to solve the problem using a time parallel approach is at best asymptotically the same as the serial time in spite of the larger number of processors used. In particular the time parallel algorithm will actually take more wall-clock time than the serial version if $\epsilon \leq 5.17 \times 10^{-5}$.

We now analyze block Gauss-Seidel methods. We state the formal result showing that the time parallel methods never provide an asymptotic speedup and may be slower for sufficiently small time steps.

THEOREM 3.2. *Let t_s^{BE}, t_p^{BE}, t_s^{CN} and t_p^{CN} be the total wall clock times for block Gauss-Seidel for the backward Euler and Crank-Nicolson (serial and parallel) variants. The total wall clock times, assuming no communication overhead, satisfy $t_p^{BE} \geq \frac{1}{2\pi^2} \ln\left(\frac{1}{\epsilon}\right) t_s^{BE}$ and $t_p^{CN} \geq \frac{1}{2\pi^2} \ln\left(\frac{\sqrt{2}}{\epsilon}\right) t_s^{CN}$.*

Proof. See [1]. □

Corollary: The parallel algorithm is slower than the serial one when

$$\epsilon \leq \begin{cases} 2.68 \times 10^{-9} & backward\ Euler \\ 3.78 \times 10^{-9} & Crank-Nicolson \end{cases}$$

Since communication time has not been factored into the above calculation, the parallel versions will in practice be slower for much larger values of ϵ.

We have implemented the above block schemes in order to illustrate and validate our claims. In the results to follow, we counted the number of iterations required for convergence of block Jacobi and Gauss-Seidel schemes for the backward Euler method with $r \equiv \frac{\Delta t}{(\Delta x)^2} = 100$. Convergence is asserted when the norm of the residual is less than 10^{-7}.

Table 1 contains numerical results for the block Jacobi and Gauss Seidel methods applied to the backward Euler equations. The number of iterations required to converge to the desired solution increases linearly with the number of timesteps in parallel as predicted by the theory. This negates any potential benefits from parallelization.

4. Point Iterative Methods. In this section we state some results (see [1] for proofs) for point Jacobi and Gauss-Seidel methods for solving the conventional

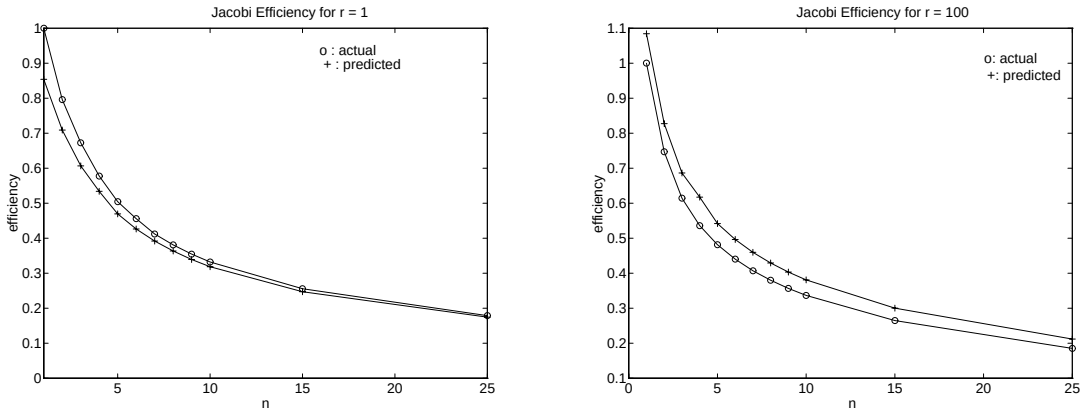

FIG. 1. Efficiencies for m = 60

backward Euler scheme whose iteration matrix is given by

$$T_{m\times m} = \frac{1}{(1+2r)} \begin{pmatrix} 0 & -r & & \\ -r & 0 & \ddots & \\ & \ddots & \ddots & -r \\ & & -r & 0 \end{pmatrix}.$$

First, we consider the Jacobi method.

THEOREM 4.1. *The Jacobi iterations take approximately k iterations to converge where k is given by*

$$k = \frac{(\ln \epsilon) + \ln(1 + 1/(4r))}{\ln(2r) - \ln(1+2r)}.$$

THEOREM 4.2. *In the time parallel case, the point Jacobi method takes at least k iterations to converge where k is the solution to the equation*

$$\frac{C_n^k}{\lambda^n} \frac{\lambda^k}{(1+2r)^k} = \epsilon \quad with \quad \lambda = 2r\cos\left(\frac{\pi}{m+1}\right).$$

Since the above expression for the number of iterations cannot be solved for k in a simple manner we present the numerically determined values for k. We also present the number of iterations that were observed experimentally to validate our claim. In Figure 1 we present data for the computed and experimentally observed efficiency of the parallel scheme for two different values of r. As can be easily observed, the efficiency of the parallel scheme decreases rapidly with increasing n.

This qualitative behavior is largely independent of m as can be seen from the graph in Figure 2, in which we show the efficiency of the parallel scheme for different values of m. Clearly, the efficiency does not stay constant with n as claimed in [9].

Now consider the point Gauss-Seidel scheme. Unfortunately, we are unable to derive a lower bound for the 2-norm of k^{th} power of the Gauss-Seidel iteration matrix. However, we present numerical results to demonstrate that the efficiency of the parallel

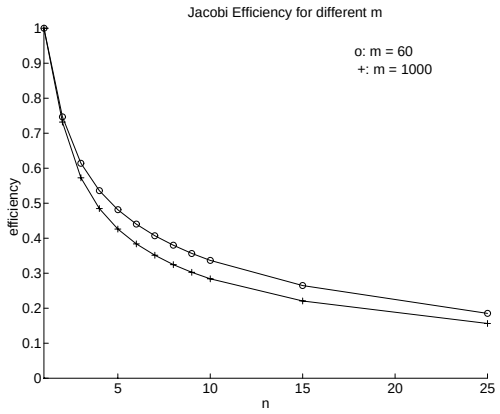

FIG. 2. Efficiency for m = 60 and m = 1000

scheme goes down with increasing the number of parallel time steps contrary to the claims made in [9] that it remains constant.

In Figure 3 we show the efficiency of the parallel scheme for two different values of r. As the graph shows, the efficiency decreases with increasing n.

In Figure 4, we show the behavior of the parallel scheme for two different values of m. As the graph shows, the efficiency decreases rapidly with increasing n irrespective of the value of m and does not stay constant.

5. Conclusions. We have analyzed the numerical solution of a linear, one dimensional, time dependent partial differential equation using time domain parallelism. We have shown that the increase in the number of iterations required for the convergence of block and point Jacobi and Gauss-Seidel methods negates any advantages of time domain parallelism. As noted by others [2, 4, 7, 8, 9, 10] these methods can be useful for some classes of problems. Hence, care must be employed when using them.

REFERENCES

[1] A. DESHPANDE, S. MALHOTRA, C. C. DOUGLAS, AND M. H. SCHULTZ, *Temporal domain parallelism: Does it work?*, Tech. Rep. YALEU/DCS/TR-996, Department of Computer Science, Yale University, New Haven, CT, 1993.

[2] D. E. KEYES, 1994. Private communication.

[3] K. MILLER, *Numerical analogs to the Schwarz alternating procedure*, Numer. Math., 7 (1965), pp. 91–103.

[4] J. H. SALTZ, *Parallel and Adaptive Algorithms for Problems in Scientific and Medical Computing*, PhD thesis, Department of Computer Science, Duke University, Durham, NC, 1985.

[5] H. A. SCHWARZ, *Uber einige abbildungsaufgaben*, Ges. Math. Abh., 11 (1869), pp. 65–83.

[6] R. S. VARGA, *Matrix Iterative Analysis*, Prentice-Hall Inc., Englewood Cliffs, NJ, 1962.

[7] D. WOMBLE, *A time stepping algorithm for parallel computers*, SIAM J. Sci. Stat. Comput., 11 (1990), pp. 824–837.

[8] P. WORLEY, *Parallelizing across time when solving time-dependent partial differential equations*, in Proc. 5th SIAM Conf. on Parallel Processing for Scientific Computing, D. Sorensen, ed., SIAM, 1991.

[9] J. ZHU, *A new parallel algorithm for the numerical solutions of time dependent partial differential equations*, tech. rep., Mississippi State University, Mississippi State, MS, 1991.

[10] ———, *Solving Partial Differential Equations on Parallel Computers*, World Scientific Publishing, Singapore, 1994.

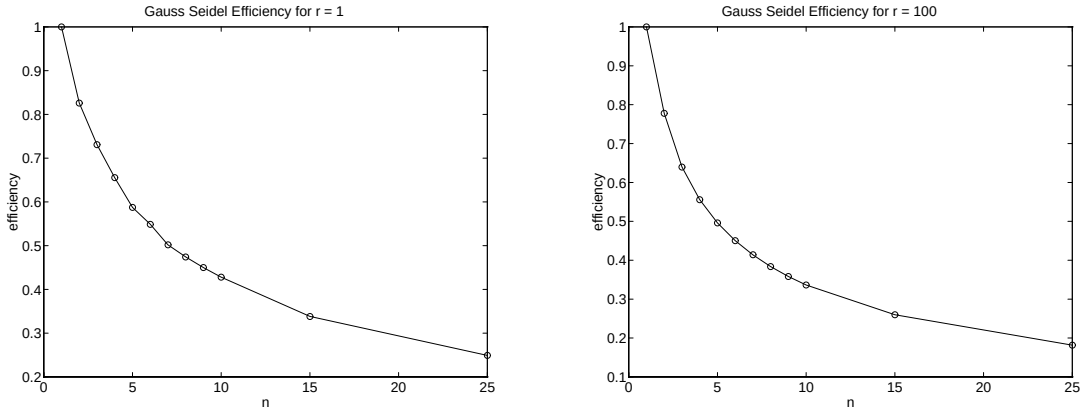

FIG. 3. Efficiencies for m = 1000

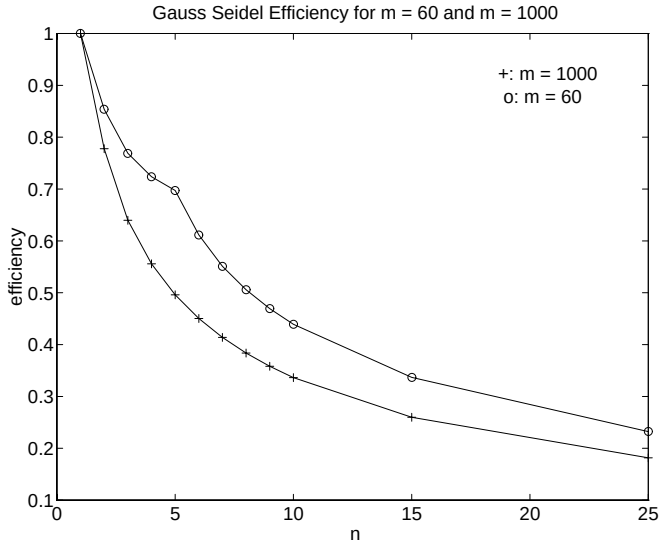

FIG. 4. Efficiency for m = 60 and m = 1000