TIME PARALLEL ALGORITHMS FOR SOLUTION OF TIME-DEPENDENT PDES

AMIR FIJANY, JACOB BARHEN, and NIKZAD TOOMARIAN
Center for Space Microelectronics Technology
Jet Propulsion Laboratory, California Institute of Technology
Pasadena, CA 91109, USA

ABSTRACT

Parallel algorithms for solution of time-dependent partial differential equations (PDEs) are developed. It is shown that, for a wide class of such PDEs, the seemingly strictly sequential time-stepping procedures can be fully parallelized in time. This results in algorithms that offer a massive degree of coarse grain temporal parallelism in the computation, and have a highly decoupled structure with simple communication and synchronization requirements. Such algorithms are particularly efficient for implementation on emerging massively parallel MIMD architectures.

Keywords: **Time-parallel algorithms, time-dependent PDEs, time-stepping procedures, Crank-Nicolson method, time- and space-parallel computation, MIMD architectures.**

1. Introduction

The solution of time-dependent PDEs is at the heart of many scientific and engineering applications. The corresponding development of fast and accurate algorithms has been extensively studied in the literature. The advent of massively parallel architectures offers a new opportunity for faster solution of evermore complex problems. However, in order to fully exploit the computing power of these new architectures, existing algorithms must be reexamined, with emphasis on their efficiency for parallel implementation. Eventually, new algorithms may have to be developed that, from the onset, take a greater advantage of the available massive parallelism.

In this paper, we present a novel technique for solution of time-dependent PDEs which is particularly efficient for implementation on emerging massively parallel MIMD architectures. By departing from conventional approaches, our proposed technique allows the solution of such PDEs to be fully parallelized in time, resulting in algorithms which offer a massive degree of coarse grain temporal parallelism

with a minimum of communication and synchronization requirement. We have already applied this technique to the solution of a simple model problem, i.e., the heat equation [1]. The subsequent practical implementation of our time-parallel algorithm on a massively parallel MIMD architecture, the Intel's Touchstone Delta, has shown that a significant speedup can be achieved even by using a very large number of processors, e.g., on the order of hundreds [2].

In the sequel, we present the general concepts underlying our approach to time-parallel computation, and discuss its domain of efficient applicability. We also analyze key issues in extending our technique to more complex problems, which highlights the framework underlying our current research efforts. The paper is organized as follows. Time-stepping procedures for solution of time-dependent PDEs and current parallel computation approaches are reviewed in §2. The novel time-parallel algorithms are presented in §3, and their efficiency and applicability are discussed in §4. In §5, time-parallel algorithms for two model problems are presented and their performance is analyzed. Finally, some concluding remarks are made in §6.

2. Background

2.1. Time-Stepping Procedures

The algorithms discussed in this paper can be applied to the solution of a wide class of time-dependent PDEs. However, in order to provide a simple illustration for our discussion, let us consider a parabolic equation on a 2D regular domain Ω, with boundary W, i.e.,

$$\frac{\partial U}{\partial t} = \mathcal{L}U + f(x,y,t) \qquad \text{in } \Omega \text{ for } T \geq t > 0 \tag{1}$$

with initial and Dirichlet boundary conditions given by

$$U_0 = h(x,y), \text{ in } \Omega \text{ for } t = 0, \text{ and } U = g(x,y), \text{ on W for } T \geq t > 0$$

In Eq. (1), $\mathcal{L}$ is a linear elliptic operator and $f(x, y, t)$ is a source vector.

The discretization of Eq. (1) in both time and space, by superimposing a uniform grid of size $\gamma = \Delta x = \Delta y$ on Ω, and using a standard finite-difference scheme, leads to a family of time-stepping methods formalized as

$$(I + 2\alpha\beta\mathcal{M}_{\mathcal{L}})U^{(m)} = (I - 2\alpha(1 - \beta)\mathcal{M}_{\mathcal{L}})U^{(m-1)} + F^{(m)} \quad m = 1 \text{ to } M \tag{2}$$

where I and $\mathcal{M}_{\mathcal{L}} \in \Re^{N^2 \times N^2}$ denote the identity matrix and the matrix arising from the discretization of $\mathcal{L}$, respectively. Here, $\alpha = \tau/2\gamma^2$, $\tau = \Delta t$ is the time step size, $M = T/\tau$, and N is the grid size. Furthermore, $U^{(m)}$ and $F^{(m)} \in \Re^{N^2}$ denote the approximate solution vector at time step m and the vector resulting from the discretization of $f(x,y,t)$ and $g(x, y)$, respectively.

Three methods are usually invoked for the solution of Eq. (1). They are characterized in terms of the parameter β as: Explicit method ($\beta = 0$); Implicit method ($\beta = 1$); and Crank-Nicolson (C-N) method ($\beta = 1/2$). The explicit method is conditionally stable, while both the implicit and C-N methods are unconditionally stable. Furthermore, the C-N method is second-order accurate in time, while the implicit method is only first-order accurate.

From a computational point of view, tile solution of Eq. (2) is both time (number of time steps, M) *and* space (size of grid, N) dependent. In the sequel, the term space-prrrallcl is used for algorithms that only exploit parallelism in solving Eq. (2) at each time step, while the term *time-parallel* refers to those algorithms that exploit parallelism in the computation of all vectors $U^{(m)}$, $m = 1$ to M .

2.2. Current Trends in Developing Parallel Algorithms

Recent directions in developing fast parallel algorithms for solution of time-dependent P D Es seem to have emerged from two widely acknowledged observations (see, for example, [3,4]) regarding the efficiency of time-stepping methods for parallel computation:

(i) The explicit methods, while limited in their range of stability, are highly efficient for parallel/vector processing, sine.c the computation at each time step mainly **involves** a matrix-vector multiplication (hIVM);

(ii) The implicit and C-N methods, despite their superior numerical properties, are not efficient for parallel/vector processing, since the solution of a linear system is required at each time step.

The first observation has motivated the development of new explicit methods which, while preserving the efficiency for parallel/vector computation, offer better numerical properties [5,6]. Note, however, that the parallelism in these methods is rather fine grain and, even with improved stability, a very large number of time steps is till required. The second observation has led to approaches emphasizing an increase in the efficiency of implicit methods for p arallel computatio n[4,7]. All these approaches result in space-parallel algorithms, since they attempt to parallelize the computation at each time step while the overall computation remains strictly sequential in time.

The computation of time-stepping procedures in Eq. (2) is generally assumed to be strictly sequential in time. This has motivated the development of new paradigms, mostly in the form of iterative techniques, to achieve some level of temporal parallelism [8]-[13]. However, all these approaches seem to have achieved a rather limited parallelism in time. Indeed, Womble [10] supports the assessment of [14], wherein a simultaneous solution for all time steps is not considered practical.

In the sequel, we present a technique which, for a wide class of problems, allows the time-stepping computation to be fully parallelized in time. Interestingly, for this class of problems, our approach establishes a rather surprising result: *the C-N method is significan tly more efficient than the explicit and implicit n ietlzods for fi?ttc-parallel computation.*

3. Time Parallel Algorithms

To motivate the idea of time-par allelism , notice that Eq. (2) can be interpreted in terms of First-Order Inhomogeneous Linear Recurrences (F OIL R). The solution of such recurrences can be computed in $O(\text{Log A4})$ by using, for example, the algorithms in [15]. Thi s indicates that th e cor nputation is fully parallelizable in time, wherein the time lower bound of $O(Log\ M)$ can be achieved.

From an efficiency stand point, however, suet] an approach is not practical, since it involves the computation of explicit inverse and powers of matrices in Eq. (2). Starting with highly sparse matrices, this would lead to the generation of increasingly dense and ultimately full matrices. For 2D and 3D problems, the Complexity of each step of such a time-parallel algorithm would then be of $O(N^6)$ and $O(N^9)$, respectively, with the overall complexity of the algorithm being of $O(N^6 \text{Log } M)$ and $O(N^9 \text{Log } M)$. Thus, although the complexity of the computation could be reduced in time, it would be signifi c antly increased in the spatial domain. As a result, the computational cost of such time-parallel algorithms would be orders of magnitude greater than that of the best serial algorithm for most practical cases. *Nevertheless, this observation clearly suggests that, insofar as the data dependency* in *the* computation *is concerned, the time-stepping procedures can, in* principle, *be fully parallelized in time.*

3.1. Introduction of Temporal Parallelism in Computation

Motivated by the above observation, we propose a technique which, for a wide class of time-dependent PDEs, allows the time-stepping procedures to be *efficiently* parallelized in time, leading to highly practical algorithms for massively parallel computation. The fundamental idea of our time-parallel computation is to introduce a diagonalization process in the time-stel)ping schemes. 1'0 describe our approach, let us consider the C-N method given by Eq. (2). Note that the matrices on the left- and right-hand sides of the equation are simultaneously diagonalizable, i.e., they have the same set of eigenvectors. Let the Eigenvalue-Eigenvector Decomposition (EED) of the nonsingular matrix $\mathcal{M}_{\mathcal{L}}$ be given by:

$$\mathcal{M}_{\mathcal{L}} = \Theta \Lambda \Theta^{-1}$$

Substituting the above decomposition into Eq. (2) results in

$$(I + \alpha\Theta\Lambda\Theta^{-1})U^{(m)} = (I - \alpha\Theta\Lambda\Theta^{-1})U^{(m-1)} + F^{(m)} \qquad m = 1 \text{ to } M$$

$$\Theta(I + \alpha\Lambda)\Theta^{-1}U^{(m)} = \Theta(I - \alpha\Lambda)\Theta^{-1}U^{(m-1)} + F^{(m)} \qquad m = 1 \text{ to } M$$

Multiplying both sides by the nonsingular matrix Θ^{-1}, we obtain

$$(I + \alpha\Lambda)\Theta^{-1}U^{(m)} = (I - \alpha\Lambda)\Theta^{-1}U^{(m-1)} + \Theta^{-1}F^{(m)} \qquad m = 1 \text{ to } M$$

Let $\tilde{U}^{(m)} = \Theta^{-1}U^{(m)}, m = 0 \text{ to } M$, and $\tilde{F}^{(m)} = \Theta^{-1}F^{(m)}, m = 1 \text{ to } M$. It follows that

$$(I + \alpha\Lambda)\tilde{U}^{(m)} = (I - \alpha\Lambda)\tilde{U}^{(m-1)} + \tilde{F}^{(m)} \qquad m = 1 \text{ to } M$$

Defining the diagonal matrices

$$D_1 = (I + \alpha\Lambda)^{-1}(I - \alpha\Lambda) \text{ and } D_2 = (I + \alpha\Lambda)^{-1} \tag{3}$$

we get

$$\tilde{U}^{(m)} = D_1\tilde{U}^{(m-1)} + D_2\tilde{F}^{(m)} \qquad m = 1 \text{ to } M \tag{4}$$

which is a transformation of Eq. (2) into a diagonal form. In contrast to Eq. (2), however, Eq. (4) is highly efficient for parallel computation. For 2D and 3D problems, the complexity of sequential implementation of Eq. (4) is of $0(MN^2)$ and $0(MN^3)$, respectively. With $O(M)$ processors, the computation can be performed in $O(N^2 \text{Log} M)$ and $O(N^3 \text{Log} M)$ by adapting the algorithms of [15].

3.2. *Structure of the Time-Parallel Algorithms*

From our discussion regarding the diagonalization process, a high level description of the time-parallel algorithms can be given in terms of the following steps:

1. Compute the eigenpairs of $\mathcal{M}_\mathcal{L}$
2. Compute the diagonal matrices D_1 and D_2 from Eq. (3)
3. Compute $\tilde{U}^{(0)}$ and $\tilde{F}^{(m)}$

$$\tilde{U}^{(0)} = \Theta^{-1} U^{(0)} \tag{5}$$

$$\tilde{F}^{(m)} = \Theta^{-1} F^{(m)} \qquad m = 1 \text{ to } M \tag{6}$$

4. Compute $\tilde{U}^{(m)}$ by solving in parallel the FOILR in Eq. (4).
5. Compute $U^{(m)}$

$$U^{(m)} = \Theta \tilde{U}^{(m)} \qquad m = 1 \text{ to } M \tag{7}$$

If $M_\mathcal{L}$ is symmetric, then $\Theta^{-1} = \Theta^T$ where T denotes transpose. Thus, in Eqs. (5)-(6), MVMS rather than linear system solutions are required. For nonsymmetric $M_\mathcal{L}$, the linear system solutions can also be avoided by noting that Θ^{-1} is the matrix of the eigenvectors of $\mathcal{M}_\mathcal{L}^T$. Thus, if in Step 1 the eigenvectors of $\mathcal{M}_\mathcal{L}^T$ are also computed, then the operations in Eqs. (5)-(6) can be reduced to MVMS. in a parallel environment. with a sufficient number of processors, this does not degrade the computational efficiency, since the computations for $M_\mathcal{L}$ and $\mathcal{M}_\mathcal{L}^T$ can be performed in parallel (see also §5.2).

The computations in Steps 1 and 2 are space dependent, and need to be performed once (see below). Steps 3 and 5 highlight the full temporal decoupling. With $O(M)$ processors, the corresponding computations can he performed with a complexity of 0(1) in time anti of $O(k)$ in space, with no communication among processors, where k stands for the cost of an MVM. The only communication occurs in the parallel solution of the FOILR in Eq. (4). As can he seen, our proposed time-parallel algorithms have a highly decoupled structure. For many applications of interest, M is very large (of the order of hundreds or thousands). Thus, such a time-parallel computing approach will lead to a massive degree of coarse grain temporal parallelism with simple communication and synchronization requirements.

Besides Step I, the most computation-intensive part of the time-parallel algorithms is the MVM in Steps 3 and 5. In general, for 2D and 3D problems, tile corresponding cost is of $O(N^4)$ and $O(N^6)$. This represents a major improvement over the direct parallel computation of Eq. ('2). However, as will be shown below, for a wide class of problems one can exploit the structure of matrix $\mathcal{M}_\mathcal{L}$, so that both the computation of Step 1 and, the MVMs in Steps 3 and 5 can be performed with a greater efficiency.

3.3. *Time-Parallel Computation of the Explicit and Implicit Methods*

our tilne-parallel computing approach can also be directly applied to the explicit and implicit methods, given by Eq. (2), by using the same diagonalization technique. The resulting time-parallel algorithms will have a similar computational structure, the only difference being in the definition of the diagonal matrices D_1 and D_2. For the explicit method, we would obtain $D_1 = I - 2\alpha\Lambda$ and $D_2 = I$ while for the implicit method, we would have $D_1 = D_2 = (I + 2\alpha\Lambda)^{-1}$.

However, in order to satisfy the stability constraints, and achieve the same level of accuracy as the C-N method, both the implicit and explicit methods require a much larger number of time-steps. This implies that, for a given accuracy and overall computation time, the time-parallel computation of the implicit and explicit methods would require a significantly larger number of processors. In other words, for a given number of processors, the time-parallel computation of the C-N method will result in a significantly more efficient overall implementation, with a much better accuracy, than either the implicit or explicit methods.

4. Efficiency and Applicability of Time-Parallel Algorithms

The basic idea of time-parallel computation is rather simple. It is also general in the sense that it can directly be applied to other types of time-dependent PDEs, such as first order and second order hyperbolic equations [16]. however, what makes it a highly practical and efficient approach for massively parallel environments is the fact that, for a wide class of applications, the computation of eigenpairs of the matrix $\mathcal{M}_\mathcal{L}$ can be performed very efficiently. Furthermore, tile multiplication of the matrix of eigenvectors by a vector can also be carried out with a great efficiency.

Motivated by the simple structure and massive coarse grain parallelism underlying our approach, we have attempted to identify those problems for which it can be applied most efficiently. The two main factors under consideration are:

(i) Fast and accurate computation of the eigenpairs of $\mathcal{M}_\mathcal{L}$;
(ii) Fast multiplication of the matrix of eigenvectors by a vector.

Although our results arc preliminary, rather surprisingly they indicate that a wide class of problems are *readily* well suited for such an approach. These problems can be characterized as those for which the structure of matrix $\mathcal{M}_\mathcal{L}$ can be exploited to achieve a high computational efficiency for the above two factors. In the following, a brief discussion of these factors is presented.

4. 1. *Fast Computation of Eigenpairs of* $\mathcal{M}_\mathcal{L}$

Depending on tile structure of $\mathcal{M}_\mathcal{L}$, which results from the type of elliptic. operator, the boundary conditions, and the considered domain, we can identify three main techniques for fast computation of its eigenpairs. The first and second techniques arc more special, in tile sense that they take a greater advantage of the specific structure of $\mathcal{M}_\mathcal{L}$, while the third technique is more general, wherein only the sparsity of $\mathcal{M}_\mathcal{L}$ is exploited.

4.1.1. Analytical expression of eigenpairs of $\mathcal{M}_\mathcal{L}$

For a few simple elliptic operators on a regular domain, the analytical expression of the eigenpairs of $\mathcal{M}_\mathcal{L}$ is known a priori. A typical example is the Laplace operator on 2D and 3D regular domains. However, our analysis indicates that it is possible to develop analytical formulae for some cases for which no such expressions were previously known. The first example is the matrix arising from the application of the C-N method to the solution of a first-order hyperbolic equation. The second example is the matrix arising from the discretization of convection-diffusion type operators with constant coefficients. The corresponding derivations are discussed in [16].

Even though the class of problems for which the analytical expression of eigenpairs can be derived is rather limited, it needs to be further explored since they are well suited for time-parallel computation. Interestingly, however, as we shall see in the second example of §5, the availability of analytical expressions dots not necessarily imply optimal efficiency for our time-parallel computation approach.

4.12. A divide and conquer method

The second method can be considered as a divide and conquer approach) which exploits the specific structure of $\mathcal{M}_\mathcal{L}$. It reduces the computation of the eigenpairs of $\mathcal{M}_\mathcal{L}$ to the computation of eigenpairs of a set of simpler matrices, e.g., symmetric and nonsymmetric tridiagonal matrices. This method can be applied to a wide class of linear operators on regular domains. Our analysis [16] indicates that c.lasses of such operators include the Laplace operator in polar and spherical coordinates, separable elliptic. operators, some cases of nonseparable elliptic. operators, etc...

The advantage of this method is its high efficiency, yielding an optimal computational cost for most cases. It also offers a high degree of coarse grain parallelism, since the computation of eigenpairs of the set of tridiagonal matrices can be performed in parallel. Furthermore, the matrix of the eigenvectors can be obtained as a factor of sparse matrices, which allows its fast multiplication by any vector. This method is further illustrated in the second example treated in §5.

4.1.3. General sparse matrix techniques

The third method is more genera], and can be used for problems for which the first and the second methods cannot be applied. Regardless of the type of operator and boundary condition, $\mathcal{M}_\mathcal{L}$ has a highly sparse structure. '1'bus, a genera] approach for efficient computation of its eigenpairs would be based on the use of well known techniques' for Sparse eigenproblems [1'i]. *The key point, however, is the fact that we are not interested in explicit computation of the matrix of eigenvectors, but rather in its multiplication with some vector.* Specifically, it is more efficient to obtain the matrix of eigenvectors in a factored form, since its multiplication by a vector can then be performed with a greater efficiency (see below). This is a clear departure from most conventional sparse eigenproblem techniques, wherein the explicit, computation of the matrix of eigenvectors is sought.

4.2. Fast multiplication of a matrix of eigenvectors by a vector

in general, the matrix of eigenvectors, Θ, is dense, For a 2D problem, the cost of multiplication of Θ by a vector is of $O(N^4)$. Our emphasis on efficient techniques for this MVM is based on the following facts. Computing the matrix Θ in a factored form allows, in most cases, a much faster MVM, e.g., in $O(N^3)$ or even in $O(N^2 \mathrm{Log}\, N)$. For a typical 2D problem with N of the order of hundreds, this would result in *more than two orders of magnitude speedup*, regardless of the degree to which the computation is parallelized in time. Such a significant improvement provides a strong rationale for techniques that compute the matrix of eigenvectors in a factored form and thus, by avoiding its explicit evaluation, also achieve a greater efficiency in computing the eigenpairs of matrix $\mathcal{M}_{\mathcal{L}}$.

5. Two Examples of Time-Parallel Algorithms

In order to better illustrate the concept of time-paral]cl computation, we briefly discuss its application to two model problems.

5.1. 2D Heat equation

As a first example, we consider the solution of a 2D heat equation or: a square domain. Note that this model problem has also been considered in [4,5,9, 10,12]. For the heat equation, the elliptic operator in Eq. (1) is tile Laplace operator. Thus, in terms of the C-N method, Eq. (2) represents a sequence of Poisson equations. By using *Fast Poisson Solvers* [18] each solution of Eq. (2) can be obtained in 0(N^2Log N), leading to an overall complexity of $O(M$ N^{2}Log N) in the framework of the C-N formalism.

On a square domain, the analytical expressions of eigenpairs of the Laplace operator for different boundary conditions are known. For the Dirichlet boundary condition, the matrix Θ is the operator of the 2D Discrete Sine Transform (DST) [1], and is given as $\Theta = QPQ$ where $Q = \mathrm{Diag}\{Q, Q, \ldots, Q\} \varepsilon \Re^{N^2 \times N^2}$ and $Q \varepsilon \Re^{N \times N}$ is the operator of the 1 D D. ST. Furthermore, $P \epsilon \Re^{N^2 \times N'^2}$ is a permutation matrix which arises in application of 2D discrete transforms. Note that, $P^T = \mathrm{P}^{-1}$, since P is a permutation matrix and hence is orthogonal.

The multiplication of matrix Θ hy any vector is tantamount to performing a 2DDST, which can he computed in $O(N^2$ Log N). 'J'bus, in a parallel implementation using $O(M)$ processors, the computational complexity of the time-parallel algorithm, C_{TP}, is given by

$$C_{TP} = a_1 N^2 \mathrm{Log}\ N + a_2 N^2 \mathrm{Log}\ M$$

where a_1 and a_2 are constants and the lower degree terms are neglected. The speedup, S_{TP}, is then obtained as

$$S_{TP} = \frac{M N^2 \mathrm{Log}\ N}{a_1 N^2 \mathrm{Log\ N} + a_2 N^2 \mathrm{Log}\ M} = O((M\, Log\, N)/(Log\ N\ +\ Log\, M)) \qquad (8)$$

A comparison of our theoretic.al result, given by Eq. (8), and the practical results of implementation of the time-parallel algorithm on the Intel Delta, presented in [2], with those reported in [4,5,9, 10,12] for this same model problem clearly indicates the efficiency of our time-parallel computing approach. The results in [2] also support the assessment regarding the efficiency of the time-parallel algorithms for implementation on massively parallel MIMD architectures.

5.2. *Laplace operator in polar coordinates*

As a second example, we consider the solution of problems involving the Laplace operator in polar coordinates. Note that, for this example, the analytical expression of eigenpairs of $\mathcal{M}_{\mathcal{L}}$ is not known. However, as shown below, these eigenpairs can be efficiently computed by using the Divide-and-Conquer method of *34.1.2*.

The Laplace operator in polar coordinates is given by

$$\nabla^2 U = \frac{\partial^2 U}{\partial \rho^2} + \frac{1}{\rho}\frac{\partial U}{\partial \rho} + \frac{1}{\rho^2}\frac{\partial^2 U}{\partial \phi^2}$$

The domain Ω is considered to be an annulus between two circles with radii ρ_0 and ρ_1. The mesh points in the ρ-ϕ plane are defined by the points of intersection of the circles $\rho = i\Delta\rho, i = q-1$ to $p+1$, and the straight lines $\phi = j\Delta\phi$, j = 1 to N, where $\Delta\phi = 2\pi/N$. For simplicity, it is assumed that $\rho_0 = (q-1)\Delta\rho$, $\rho_1 = (p+1)\Delta\rho$ and $K = p - q + 1$. With Dirichlet boundary conditions, and using the five-point finite-difference scheme, $\mathcal{M}_{\mathcal{L}}$ is a block tridiagonal matrix given by (see also [19])

$$M_c = \text{Tridiag}[(1 - 1/2i)I, B_i, (1 + 1/2i)I] \in \Re^{KN \times KN}, \; i = q \text{ to } p$$

where I is the N x N identity matrix, $B_i = \beta_i B - 2I$ with $\beta_i = (1/i\Delta\phi)^2$, and B is an N x N symmetric tridiagonal matrix with a periodic structure given by $B = \text{Tridiag}[1, -2, 1]$ with $B_{1N} = B_{N1} = 1$. Note that this periodic structure of B results from the periodicity in ϕ. The following theorem is used in deriving the EED of M_c.

Theorem 1. The EED of the matrix B is given by

$$B = FDF^{-1}$$

where F is the matrix of eigenvectors and $D = \text{Diag}\{d_j\}$, j = 1 to N, is the diagonal matrix of eigenvalues, with $d_j = -4\sin^2[(j-1)\pi/N]$.

Proof. See for example [20, p.253]. C1

The matrices F and F^* are the direct and inverse (1/N) Discrete Fourier Transform (DFT) operators. Thus, their multiplication by any vector can be performed in $O(N \, Log \, N)$. From the definition of B_i, it follows that B_i and B share the same set of eigenvectors. The EED of B_i is then given by $B_i = F\lambda_i F^{-1}$ where $\lambda_i = \text{Diag}\{\lambda_{ij}\} = \beta_i D - 2I$, $j = 1$ to N, and $\lambda_{ij} = \beta_i d_j - 2$.

Theorem 2. The EED of $\mathcal{M}_{\mathcal{L}}$ is given by

$$\mathcal{M}_{\mathcal{L}} = \mathcal{F}\mathcal{P}\mathcal{Q}\Lambda\mathcal{Q}^{-1}\mathcal{P}^{T}\mathcal{F}^{-1} \tag{9}$$

where $\mathcal{F}$ = Diag$\{F, F, \ldots . F\}$ and A is the diagonal matrix of eigenvalues which, along with the matrix Q,"is defined below.

Proof. The matrix $Mr.$ can be written as

$$\mathcal{M}_{\mathcal{L}} = \text{Tridiag}[(1 - 1/2i)I, F\lambda_i F^{-1}, (1 + 1/2i)I] = \mathcal{F}\mathcal{R}\mathcal{F}^{-1} \tag{lo}$$

where $\mathcal{R}$ = Tridiag[(1 - 1/2i)I, λ_i, (1 -t 1/2i)I]. The block elements of $\mathcal{R}$ are diagonal. l'bus, it can be reduced to a block diagonal matrix T using

$$\mathcal{R} = \mathcal{P}\mathcal{P}^{T}\mathcal{R}\mathcal{P}\mathcal{P}^{T} = \mathcal{P}(\mathcal{P}^{T}\mathcal{R}\mathcal{P})\mathcal{P}^{T} = \mathcal{P}\mathcal{T}\mathcal{P}^{T} \tag{11}$$

where T = Diag$\{T_j\} \varepsilon \Re^{KN \times KN}$ and T_j = Tridiag $[(1 -- 1/2i), \lambda_{ij}, (1+1/2i)] \varepsilon \Re^{K \times K}$, j = 1 to N and $i = q$ to p. No w, let tile EED of T_j be given by $T_j = Q_j \Lambda_j Q_j^{-1}$. Defining Q = Diag$\{Q_j\}$ and A = Diag$\{\Lambda_j\}$, j = 1 to N, it follows that

$$\mathcal{T} = Q\Lambda Q^{-1} \tag{12}$$

The EED of *ML*, given by Eq. (9), is obtained by substituting, Eqs. (1 1)-(12) into Eq. (lo). **D**

The matrix T_j is sign *symmetric*, since the products of pairs of the corresponding off-diagonal elements are all nonzero and positive, and hence has real eigenvalues. Due to this property, all eigenpairs of T_j can be efficiently computed by using, for example, the subroutine RT provided by EISPACK [21]. Multiplication of a vector by Q_j^{-1} corresponds to the solution of a linear system involving a computational cost of $0(1<^{3})$. However, following our discussion in §3.2., this operation can be reduced to an MVM with a cost of $O(K^2)$ by noting that Q_j^{-1} is the matrix of eigenvectors of matrix T_j^T. Therefore, if the EED of T_j^T is also computed, then the matrix Q_j^{-1} is explicitly obtained and its multiplication by a vector represents a simple MVM. For typical values of K of the order of hundreds, this scheme results in two orders of magnitude improvement in the computational efficiency.

Now, let us consider a parallel implementation hy using M processors. Assuming $M \geq 2N$, the EED of $2N$ sign symmetric tridiagonal matrices T_j and T_j^T can be performed in parallel, leading to a cost of $O(K^2)$ for Step 1. The diagonal matrices D_1 and D_2 can be computed in $O(KN)$ and the cost of parallel solution of the FOILR in Eq. *(4) is* of $O(K\,\mathrm{N}\,Log\,M)$. Note that, the matrix of eigenvectors, Θ, is obtained in a factored form as $\Theta = \mathcal{F}\mathcal{P}Q$. '1'bus, the multiplication of Θ and Θ^{-1} by a vector in Steps 3 and 5 can be performed at a cost of $a_1K^2N + a_2KN\,Log\,\mathrm{N}$, where a_1 and a_2 are constants, and lower degree terms are neglected. The cost of such an implementation of time-parallel algorithm, C_{TP}, is then given by

$$C_{TP} = a_1K^2N + a_2KN\,Log\,N + a_3KN\,Log\,M$$

where a_3 is a constant. and lower degree terms are neglected.

The fast Poisson solvers of [18] can also be extended to solution of the problem in polar coordinates. This implies that, for the C-N method, the cost of the best serial algorithm for each solution of Eq.(2) is of $O(K N Log N)$, leading to an overall complexity of O(A4 K N Log N). The speedup of the time-parallel algorithm, S_{TP}, is then given by

$$S_{TP} = \frac{M K N Log\ N}{a_1 K^2 N + a_2 K N Log\ N + a_3 K N Log\ M} = O((M Log\ N)/K)$$

At first glance, it might have seemed that the time-parallel algorithms are most efficient for those problems for which the analytical expression of eigenpairs of $\mathcal{M}_L$ are known, and hence no computation is needed. However, our analysis of the performance of the time-parallel algorithm for the above example, in which additional computations are required for derivation of eigenpairs, appears to clearly indicate the contrary. This result seems to be rather general and shows that, for most cases, *the performance of the time-parallel algorithms will not be degraded due to the need of computing the eigenpairs, if the latter is performed efficiently.*

6. Conclusion

We have presented a novel technique for massively parallel solution of time-dependent PDEs. Our discussion on the efficient applicability of this technique clearly indicates that it is relevant to a wide class of PDEs. It was shown that, for time-parallel computation, the C-N method is significantly more efficient than either the explicit or implicit methods. Therefore, our approach enables the use of fully implicit methods with superior numerical properties and, at same time, with optimal efficiency for parallel computation.

By using the first and particularly the second technique of § 4.1. for fast computation of eigenpairs of matrix M_c, we have shown that our approach can *readily* be applied to a rather large class of problems of practical interest [16]. However, the solution of problems involving more complex elliptic. operators and/or operators on irregular domains, requires further research work with focus on analyzing and designing appropriate sparse eigenproblems techniques in the framework of the discussion in §4.1.3.

Acknowledgements

This research was performed at the Center for Space Microelectronics Technology, Jet Propulsion Laboratory, California Institute of '1'ethnology. It was jointly sponsored by innovative Science and '1'ethnology Office of the Ballistic Missile Defense Organization, and by the National Aeronautics and Space Administration, Office of Advanced Concepts and Technology. The support and encouragement of Dr. Paul Messina, Director of the Concurrent Supercomputing Consortium, is greatly acknowledged.

References

1. A. Fijany, Time-Parallel Algorithms for Solution of Linear Parabolic PDEs, *Proc. Int. Conf. Parallel Processing (ICPP)*, Aug. 1993, Vol. 11 I, 51-55.
2. N. Toomarian, A. Fijany, and J. Barhen, Time Parallel Solution of Linear Partial Differential Equations on the Intel Touchstone Delta Supercomputer, *Concurrency: Practice and Experience*, (in press 1994).
3. J.M. Ortega and R.G. Voigt, **Solution** *of Partial Differential Equations on Vector and Parallel Computers* **(SIAM Pub., 1984).**
4. E. Gallopoulos and Y. Saad, On the Parallel Solution of Parabolic Equations, *Proc. ACM Int. Conf. on Supercomputing*, June **1989, 17-28.**
5. G. **Rodriguez and D.** Wolitzer, Preconditioned Time-Differencing for the Parallel Solution **of the Heat** Equation, *Proc. 4th SIAM Conf.* **on Parallel** Processing, **1990,** 268-272.
6. D. J. Evans, Alternating Group Explicit Methods for the Diffusion Equation, *Appl. Math. Modelling*, **9** *(1985), 201-206.*
7. S. **M.** Serbia, A Scheme for Parallelizing Certain Algorithms for the Linear Inhomogeneous Heat equation, *SIAM J. Sci. Stat. Comput.* 13 *(1992) 449-458.*
8. *E. Lelarasmee et al, The Waveform Relaxation Method for the Time Domain Analysis of Large Scale Integrated* Circuits, *IEEE* **Trans.** *Computer-Aided* **Design, 1** *(1982) 131-145.*
9. *J. H. Saltz and V. K.* Nail, *Towards Developing Robust A lgorithms for Solving* Partial *Differential Equations on MIMD Machines,* Parallel Computing G (1988) 19-44.
10. *D. E. Womble, A Time-Stepping A lgorithm for Parallel Computers, SIAM J. Sci. Stat. Comput.* **11** (1990) *824-837.*
11. *W. Hackbusch, Fast* Numerical Solution *of Time* Periodic *Parabolic Problems by a Multigrid Method, SIAM J. Sci. Stat. Comput. 2 (1981) 198-206'.*
12. *G. Horton and R. Knirsch, A Time-Parallel Multigrid- Extrapolation Method jor* Parabolic Partial *Differential* Equations, Parallel *Computing* 18 *(1992) 21-29.*
13. *S. Vandewalle and R. Piessens, Efficient Parallel Algorithms for Solving Initial-Boundary Value and Time-Periodic Parabolic Differential Equations, SIAM J. Sci. Stat. Comput. 13 (1992) 1330-1346.*
14. *G. Strang and G. J. Fix, An Analysis of the Finite Element Method (Prentice-Hall, 1973).*
15. *R. Hockney and C. Jesshope, Parallel Computers (Adam Hilger, 1981).*
16. *A. Fijany, J. Barhen, N. Toomarian, On the Structure of Tifflc-Parallel Algorithms jor Solution of Time-dependent PDEs, In* **preparation.**
17. **S.** *Piss anetzky, Sparse Matrix t echnology (Academic Press, 1984).*
18. *B. Buzbee, G. Golub, and C.* **Nielson,** *On Direct Methods jor* Solving *Poisson Equations, SIAM J. Numer. A nal. 7 (1970) 627-656.*
19. *G. D. Smith, Numerical Solution of Partial Differential Equat* ions *(Clarendon* Press, *1985).*
20. *C. Van Loan, Computational Frameworks for Fast Fourier Transform (.71AM Pub., 1992).*
21. *B.T. Smith et al, Matrix Eigensystem Routines- Eispack Guide* **(Springer** *Verlag, 1976).*