JOURNAL OF COMPUTATIONAL PHYSICS **122**, 39–50 (1995)

Parallel Block Preconditioning Techniques for the Numerical Simulation of the Shallow Water Flow Using Finite Element Methods

Y. CAI

Department of Chemical Engineering, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139

AND

I. M. NAVON

Department of Mathematics and Supercomputer Computations Research Institute, Florida State University, Tallahassee, Florida 32306-4052

Received February 17, 1995; revised June 1995

In this paper, we report our work on applying Krylov iterative methods, accelerated by parallelizable domain-decomposed (DD) preconditioners, to the solution of nonsymmetric linear algebraic equations arising from implicit time discretization of a finite element model of the shallow water equations on a limited-area domain. Two types of previously proposed DD preconditioners are employed and a novel one is advocated to accelerate, with post-preconditioning, the convergence of three popular and competitive Krylov iterative linear solvers. Performance sensitivities of these preconditioners to inexact subdomain solvers are also reported. Autotasking, the parallel processing capability representing the third phase of multitasking libraries on CRAY Y-MP, has been exploited and successfully applied to both loop and subroutine level parallelization. Satisfactory speedup results were obtained. On the other hand, automatic loop-level parallelization, made possible by the autotasking preprocessor, attained only a speedup smaller than a factor of two. © 1995 Academic Press, Inc.

1. INTRODUCTION

Parallelism was introduced to be one of the novel architectural features of today's computers for further increasing the computing speed and making possible the numerical solution of even larger scientific and engineering problems.

One of the research focuses in the area of parallel computing has centered on the issue of how to cost-effectively introduce parallelism into strongly coupled problems, such as the parallel solution of large linear or nonlinear systems of algebraic equations, which arise from the finite difference or finite element discretization of PDEs in many areas of industrial applications.

Despite the fact that numerous approaches have already been developed and implemented on different types of parallel computers (see, for example, [13, 31, 32]) for the parallel solution of linear or nonlinear algebraic equations, more recently proposed parallel numerical algorithms to this end are largely based on, or closely related to, the principles of domain decomposition.

For the past eight years, many domain decomposition methods have been proposed and developed (see [7] and references therein). These general approaches, with a modification of one or more of their ingredients, provide almost infinitely many variants of the so-called iterative domain decomposition algorithms. Another dimension should be added to this variety if implementation details are taken into account. A very recent comprehensive review of this subject was furnished in [11].

The Schur domain decomposition method, as discussed in [30], reduces the solution of a system of linear equations defined on the original computational domain to a dense, but much smaller, Schur complement linear system on the interfaces only. This reduced linear system is solved by a "divide and feedback" iterative procedure. The solution of the Schur linear system then serves as the boundary conditions for parallel solution of the subdomain problems.

However, the aforementioned "divide and feedback" process usually requires repeatedly exact subdomain solutions which are not cheaply implementable for nonseparable elliptic operators or for other more complicated cases such as the shallow water equations.

To improve efficiency of the parallel solution of partial differential equations, for which no fast subdomain solvers are available, at least two other approaches have been proposed. One is the domain-decomposed (DD) preconditioner approach (also called full matrix domain decomposition in [22]) first advocated in [6]; the other is the recently proposed modified interface matrix (MIM) approach (see [30]). Both approaches abandon the idea of Schur domain decomposition, that decoupled subdomain problems are independently solved *only after* the interface solution is obtained.

The DD preconditioner approach consists essentially of the construction of a preconditioner in a domain-decomposed way such that approximate solutions in the subdomains and on the interfaces can be successively updated at the cost of only inexact subdomain solvers. On the other hand, the MIM approach successively improves the subdomain and interface approximate solutions with iterative improvements of initial guesses in both the subdomains and interfaces being made. Thus it mitigates the disadvantage due to the absence of fast subdomain solvers.

In this paper, which is an extended and revised version of our earlier contribution in [8], we concentrate our attention only on the application of DD preconditioners to the finite element shallow water flow simulation. A detailed comparison between DD preconditioned Krylov methods and MIM approach for the problem at hand will be addressed elsewhere in a separate research work.

We test all proposed algorithms, to be presented shortly, on a shallow water equations model using the Grammeltvedt's initial conditions [19] on a limited-area rectangular region—a channel on the rotating earth. All ideas and test results obtained on this model will then serve as references for extension of these domain decomposition ideas to irregular domains with nonuniform and unstructured grids, in order to study real weather-prediction-related problems.

The plan of the current paper is as follows. In Section 2, we briefly describe the shallow water equations system and the finite element discretization. Section 3 is concerned with block preconditioning of the block-bordered matrix introduced by substructure numbering of the global nodes. Two types of widely used DD preconditioners are employed and a novel one is proposed. Carefully selected numerical results on a single processor of the CRAY Y-MP are reported and discussed in section 4. Section 5 will discuss parallelization issues on the CRAY Y-MP/432 high performance vector-parallel supercomputer via domain decomposition methods and multicolor numbering techniques for the finite element assembly. Summary and conclusions are provided in Section 6.

2. THE SHALLOW WATER EQUATIONS

The shallow water equations model under our consideration, in its primitive variables, assumes the form

$$\frac{\partial U}{\partial t} + \begin{bmatrix} u & \varphi & 0 \\ 1 & u & 0 \\ 0 & 0 & u \end{bmatrix} \frac{\partial U}{\partial x} + \begin{bmatrix} v & 0 & \varphi \\ 0 & v & 0 \\ 1 & 0 & v \end{bmatrix} \frac{\partial U}{\partial y} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -f \\ 0 & f & 0 \end{bmatrix} U = 0 \tag{1}$$

or

$$\frac{\partial U}{\partial t} + A\frac{\partial U}{\partial x} + B\frac{\partial U}{\partial y} + CU = 0, \tag{2}$$

where $U = (\varphi, u, v)^T$. $\varphi = gh$ is the geopotential of the fluid, g is the acceleration of gravity, and h is the height of the free surface. f is the Coriolis parameter given by a β plane approximation, while u and v are the horizontal velocity components in x and y directions, respectively.

The model assumes that the fluid is homogeneous, inviscid, barotropic, and incompressible. These pure convection equations are defined on a limited-area rectangular domain which corresponds to a channel on the rotating earth. The southern and northern boundaries are assumed to be rigid, i.e., $v = 0$ and the flow is assumed periodic in the west–east direction. This is a standard shallow water equations model which was used to test various finite difference schemes in [19].

To scale the problem, we need to have a good control of the magnitude of the geopotential field φ, which is dimensionally the dominant variable. We use the following set of dimensionless variables, which are different from those introduced in [19], to nondimensionalize the equations and auxiliary conditions (e.g., initial conditions, geostrophic relationship, etc.), where φ_0 is a preselected reference geopotential

$$\begin{aligned} x' &= x/L, & y' &= y/L, & t' &= t\sqrt{\varphi_0}/L, \\ \varphi' &= \varphi/\varphi_0, & u' &= u/\sqrt{\varphi_0}, & v' &= v/\sqrt{\varphi_0}, \\ h' &= h/L, & H_0' &= H_0/L, & H_1' &= H_1/L, \\ H_2' &= H_2/L, & g' &= gL/\varphi_0, & f' &= fL\sqrt{\varphi_0}, \end{aligned} \tag{3}$$

where H_0, H_1, and H_2 are three constants related to the Grammeltvedt's initial height field.

By using this set of dimensionless variables, the governing equations assume the same form after dropping the primes. However, the Coriolis parameter and the geostrophic relationship, amongst others, require minor changes (see [30] for details).

Upon using the Galerkin finite element discretization procedure with triangular piecewise linear elements and an extrapolated Crank–Nicolson scheme for the time discretization, in which the nonlinear advective terms are quasilinearized by the following second-order approximation in time

$$u^{n+1/2} = \tfrac{3}{2}u^n - \tfrac{1}{2}u^{n-1} \tag{4}$$

$$v^{n+1/2} = \tfrac{3}{2}v^n - \tfrac{1}{2}v^{n-1}, \tag{5}$$

we obtain, omitting details, the three linear systems which need to be inverted at each time step,

$$A^n\Delta\varphi^n = f^n_\varphi, \quad B^n\Delta u^n = f^n_u, \quad C^n\Delta v^n = f^n_v, \tag{6}$$

where A^n, B^n, and C^n are nonsymmetric matrices due to the presence of advective terms and $\Delta\varphi^n = \varphi^{n+1} - \varphi^n$, $\Delta u^n = u^{n+1} - u^n$, $\Delta v^n = v^{n+1} - v^n$.

It is worth pointing out that the above discretization procedure transforms the originally coupled PDEs into a set of decoupled discrete algebraic equations at the $(n + 1)$th level, which corresponds to reduced storage requirements and improved computational efficiency compared to methods which generate coupled algebraic equations. The explicit appearance of $\Delta\varphi$, Δu, and Δv instead of φ, u, and v minimizes the buildup of roundoff errors in the computation (see [18]) and offers direct indications as to the choice of initial guesses for iterative methods.

For a high order two-stage Numerov–Galerkin treatment of quadratically nonlinear advective terms, we refer to [27] for theory and to [28] for implementation issues. Some essential components of the two-stage Numerov–Galerkin procedure are concisely reviewed in [30].

3. DOMAIN DECOMPOSED PRECONDITIONERS

In this section, three types of DD preconditioners are presented, along with a heuristic analysis of these preconditioners. Through this analysis, we show that for preconditioners of the third type to be proposed in this section, the matrix A_{ss}, which corresponds to the discretization of the differential operator restricted to the interfaces, is the optimal interface preconditioner in some sense.

This optimal interface preconditioner $G = A_{ss}$, which is part of a domain decomposed preconditioner of the third type, is thus explicitly obtained without additional computational work associated with the construction of an approximate Schur complement matrix, which is nevertheless required for DD preconditioners of the first two types to be presented in what follows.

3.1. *Problem Setup*

Several DD preconditioners are presented in this section using the notation adopted in [30]. We consider solving linear systems (6) resulting from finite element discretization in space with an implicit temporal finite difference scheme, as described in Section 2.

Since we are interested in implementing the algorithms on parallel computers with powerful vector processors, the original domain is divided into strips along the west–east directions in order to yield longer vectors (see [23]).

In order to explore the inherent parallelism at the subroutine level, following [33], we number the global nodes in a substructuring way, such that the coefficient matrices of geopotential and velocities at each time step assume block-bordered structures,

$$A = \begin{bmatrix} A_{dd} & A_{ds} \\ A_{sd} & A_{ss} \end{bmatrix}, \tag{7}$$

where

$$A_{dd} = \operatorname{diag}[A_{11}, A_{22}, \ldots, A_{nn}] \tag{8}$$

is a block diagonal matrix with each block A_{ii}, for $i = 1, 2, \ldots, n$ being the discrete analog of the restriction of the original differential operator on each subdomain.

A_{ds} and A_{sd} represent connections between subdomains to interfaces. They assume block bi-diagonal forms,

$$A_{ds} = \begin{bmatrix} E_1 & & & \\ F_2 & E_2 & & \\ & F_3 & \ddots & \\ & & \ddots & E_{n-1} \\ & & & F_n \end{bmatrix} \tag{9}$$

and

$$A_{sd} = \begin{bmatrix} G_1 & H_2 & & & \\ & G_2 & H_3 & & \\ & & \ddots & \ddots & \\ & & & G_{n-1} & H_n \end{bmatrix}, \tag{10}$$

where

$$E_i = (\underbrace{0, 0, \ldots, 0}_{m_i-1 \text{ blocks}}, E_{i,m_i})^{\mathrm{T}} \tag{11}$$

$$F_i = (F_{i,1}, \underbrace{0, 0, \ldots, 0}_{m_i-1 \text{ blocks}})^{\mathrm{T}} \tag{12}$$

$$G_i = (\underbrace{0, 0, \ldots, 0}_{m_i-1 \text{ blocks}}, G_{i,m_i}) \tag{13}$$

$$H_i = (H_{i,1}, \underbrace{0, \ldots, 0, 0}_{m_i-1 \text{ blocks}}), \tag{14}$$

m_i being the number of horizontal grid lines in the ith subdomain. The blocks E_{i,m_i}, $F_{i,1}$, G_{i,m_i} and $H_{i,1}$ in matrices E_i, F_i, G_i, and H_i, respectively, are either diagonal or bi-diagonal point matrices, depending on whether a five-point finite difference scheme or a seven-point stencil, resulting from a linear triangular finite element method, is used.

A_{ss} corresponds to the discretization of the original differential operator restricted to the interfaces. Since there are $n - 1$ numbers of internal boundaries for an n subdomain case, A_{ss} assumes the block diagonal form, in which each block is associated with one of the interfaces,

$$A_{ss} = \operatorname{diag}[T_{11}, T_{22}, \ldots, T_{n-1,n-1}], \tag{15}$$

where the nonzero entries of T_{ii}, $i = 1, 2, \ldots, n-1$, amount to the following cyclic tridiagonal structure due to presence of periodic boundary conditions:

$$T_{ii} = \begin{bmatrix} b_1^{(i)} & c_1^{(i)} & & & a_1^{(i)} \\ a_2^{(i)} & b_2^{(i)} & c_2^{(i)} & & \\ & \ddots & \ddots & \ddots & \\ & & a_{n-1}^{(i)} & b_{n-1}^{(i)} & c_{n-1}^{(i)} \\ c_n^{(i)} & & & a_n^{(i)} & b_n^{(i)} \end{bmatrix}. \tag{16}$$

The numerical solution of $Ax = f$ is equivalent to solving

$$Cx_s = g \quad \text{on } \Gamma \tag{17}$$

$$A_{ii}x_i = f_i - A_{is}x_s \quad \text{in } \Omega_i, \tag{18}$$

where Γ and Ω_i, $i = 1, 2, \ldots, n$, stand for the interfaces and subdomains, respectively, and

$$C = A_{ss} - \sum_{i=1}^{n} A_{si}A_{ii}^{-1}A_{is} \tag{19}$$

and

$$g = f_s - \sum_{i=1}^{n} A_{si}A_{ii}^{-}f_i, \tag{20}$$

where C is the well-known Schur complement matrix.

The Schur domain decomposition method starts by first determining x_s on the interfaces between artificially divided subdomains by solving (17). Upon obtaining x_s, the subdomain problems (18) trivially decouple and may be solved in parallel. The main computational cost for the iterative solution of (17) depends on the number of iterations required to achieve convergence to a preset accuracy criterion and the computational costs of subdomain solvers. This approach has already been applied to the finite element simulation of the shallow water flow (see [30]).

3.2. *Three Types of DD Preconditioners*

For problems where a fast subdomain solver is not available, the DD preconditioned Krylov method may turn out to be more efficient. Instead of solving for the interface unknowns first, this approach successively updates, at the cost of only inexact subdomain solves, the approximate solutions in both the subdomains and on the interfaces. The idea here is to solve linear system $Ax = f$ directly with an appropriate preconditioner having the same block-bordered structure as that in (7). Two types of such preconditioners were reviewed in the literature (see [21], for instance).

One of the preconditioners employed for our problem is of the structurally symmetric form

$$B = \begin{bmatrix} B_{dd} & A_{ds} \\ A_{sd} & B_{ss} \end{bmatrix}, \tag{21}$$

where A_{ds} and A_{sd} are given by (9) and (10). The matrix B_{dd} has the same structure as that given by (8), except that each B_{ii}, $i = 1, 2, \ldots, n$, is now an approximation of A_{ii}. For example, B_{ii} might be the relaxed incomplete LU factorization (RILU) [4] of A_{ii}. B_{ss} is given by

$$B_{ss} = G + \sum_{i=1}^{n} A_{si}B_{ii}^{-1}A_{is}, \tag{22}$$

where G is an appropriate preconditioner to the Schur complement C. Historically, the preconditioner of the type presented in (21) was motivated by a theorem of Eisenstat in 1985 in the context of the preconditioned conjugate gradient method for symmetric linear systems (see [21]).

It may be verified that the solution of the preconditioning linear system $Bp = q$ is equivalent to solving the linear systems

$$Gp_s = q_s - \sum_{i=1}^{n} A_{si}B_{ii}^{-1}q_i \tag{23}$$

$$B_{ii}p_i = q_i - A_{is}p_s \quad \text{for } i = 1, 2, \ldots, n, \tag{24}$$

where the meaning of p_i and p_s is self-evident.

Instead of solving a linear system with a coefficient matrix B_{ii} exactly, we may equivalently solve the original linear system with coefficient matrix A_{ii} approximately. Therefore, the preconditioning system $Bp = q$ may be solved in the following fashion:

1. Solve approximately in each subdomain

$$A_{ii}p_i^{(1)} = q_i \tag{25}$$

for $i = 1, \ldots, n$ in parallel;

2. Solve the interface preconditioning system

$$Gp_s = q_s - \sum_{i=1}^{n} A_{si}p_i^{(1)}; \tag{26}$$

3. Solve approximately in each subdomain

$$A_{ii}p_i^{(2)} = -A_{is}p_s \tag{27}$$

for $i = 1, \ldots, n$ in parallel;

4. Form

$$p_i = p_i^{(1)} + p_i^{(2)} \quad \text{for } i = 1, \ldots, n. \tag{28}$$

The other type of DD preconditioner, applicable only to a nonsymmetric linear system, assumes the following block upper triangular form:

$$B = \begin{bmatrix} B_{dd} & A_{ds} \\ 0 & G \end{bmatrix}. \tag{29}$$

Now the solution of the preconditioning system $Bp = q$ requires only one inexact subdomain solve in each subdomain, compared with two such solves in the previous case with B given by (21). Obviously, the solution p of $Bp = q$ can be obtained by solving the preconditioning system on the interfaces

$$Gp_s = q_s \tag{30}$$

and approximately solving in each subdomain

$$A_{ii}p_i = q_i - A_{is}p_s \tag{31}$$

for $i = 1, ..., n$ in parallel.

A third type of DD preconditioner which we would like to propose and focus on in this paper assumes the following block lower triangular form:

$$B = \begin{bmatrix} B_{dd} & 0 \\ A_{sd} & G \end{bmatrix}. \tag{32}$$

We need to solve

$$A_{ii}p_i = q_i \tag{33}$$

approximately for $i = 1, ..., n$ in parallel and the preconditioning system on the interfaces

$$Gp_s = q_s - \sum_{i=1}^{n} A_{si}p_i. \tag{34}$$

Similar to the second type of DD preconditioners, it applies only to nonsymmetric systems of algebraic equations due, typically, to the discretization of the convection terms.

We note that, in addition to the structurally symmetric, upper triangular, and lower triangular block preconditioning matrices presented in (21), (29), and (32), respectively, a block diagonal matrix was used to precondition the GMRES [16]. This may result in better parallelization results, however, at the cost of requiring a greater number of iterations to attain convergence.

It is easy to see that the computational work involved here for the third type of DD preconditioner is only part of that required for the first type—compare Eqs. (33) and (34) with Eqs. (25) and (26). However, a greatly improved computational efficiency results (see Section 5).

A question arises as to what constitutes an appropriate algebraic form for the interface preconditioner G for each of the three types of domain-decomposed preconditioners. For preconditioners of the first two types, G may be constructed as a preconditioner to the Schur complement matrix. This construction is, however, not appropriate for preconditioners of the third type and it leads to a deterioration in performance. In order to see this, we provide formulas for AB^{-1} below. Due to the importance of these formulas, we derive them in the form of a lemma. The following lemma may be readily verified following the procedure demonstrated in [5, pp. 71–72].

LEMMA 1. *Let*

$$A = \begin{bmatrix} P & Q \\ R & S \end{bmatrix}. \tag{35}$$

If A, P, and S are nonsingular, then

$$A^{-1} = \begin{bmatrix} X & -P^{-1}QW \\ -WRP^{-1} & W \end{bmatrix} = \begin{bmatrix} X & -XQS^{-1} \\ -S^{-1}RX & W \end{bmatrix}, \tag{36}$$

where

$$W = (S - RP^{-1}Q)^{-1} = S^{-1} + S^{-1}RXQS^{-1} \tag{37}$$

$$X = P^{-1} + P^{-1}QWRP^{-1} = (P - QS^{-1}R)^{-1}. \tag{38}$$

On applying the lemma to (21), (29), and (32) to obtain the inverses and by post-multiplying (7) by these inverses, we obtain

$$AB^{-1} = \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix}, \tag{39}$$

where

$$P_{11} = \begin{cases} A_{dd}B_{dd}^{-1} + (A_{dd}B_{dd}^{-1} - I_{dd})A_{ds}G^{-1}A_{sd}B_{dd}^{-1} & \text{for (21)} \\ A_{dd}B_{dd}^{-1} & \text{for (29)} \\ (A_{dd} - A_{ds}G^{-1}A_{sd})B_{dd}^{-1} & \text{for (32)} \end{cases} \tag{40}$$

$$P_{12} = \begin{cases} (I_{dd} - A_{dd}B_{dd}^{-1})A_{ds}G^{-1} & \text{for (21)} \\ (I_{dd} - A_{dd}B_{dd}^{-1})A_{ds}G^{-1} & \text{for (29)} \\ A_{ds}G^{-1} & \text{for (32)} \end{cases} \tag{41}$$

$$P_{21} = \begin{cases} [I_{ss} - (A_{ss} - A_{sd}B_{dd}^{-1}A_{ds})G^{-1}]A_{sd}B_{dd}^{-1} & \text{for (21)} \\ A_{sd}B_{dd}^{-1} & \text{for (29)} \\ (I_{ss} - A_{ss}G^{-1})A_{sd}B_{dd}^{-1} & \text{for (32)} \end{cases} \tag{42}$$

$$P_{22} = \begin{cases} (A_{ss} - A_{sd}B_{dd}^{-1}A_{ds})G^{-1} & \text{for (21)} \\ (A_{ss} - A_{sd}B_{dd}^{-1}A_{ds})G^{-1} & \text{for (29)} \\ A_{ss}G^{-1} & \text{for (32).} \end{cases} \tag{43}$$

By examining (43), one may see that, for the first two types of preconditioners, the optimal interface preconditioner G (in the sense that $P_{22} = I_{ss}$) should be constructed such that

$$G = A_{ss} - A_{sd}B_{dd}^{-1}A_{ds} = A_{ss} - \sum_{i=1}^{n} A_{si}B_{ii}^{-1}A_{is}. \tag{44}$$

Hence, G consists essentially of the approximate Schur complement matrix of the problem. For the preconditioner of the third type, however, $G = A_{ss}$ is an optimal choice. Under these choices, $P_{21} = 0$ and $P_{22} = I_{ss}$ for the first type of preconditioners, $P_{22} = I_{ss}$ for the second type, $P_{21} = 0$ and $P_{22} = I_{ss}$ for preconditioners of the third type.

Although the matrix G is explicitly available for preconditioners of the third type, for the first two types of DD preconditioners, nn_s inexact subdomain solves must be carried out for the construction of G so that $P_{22} = I_{ss}$, where n is the number of subdomains and n_s is the number of nodes on the interfaces. To reduce the number of subdomain solves and to improve the computational efficiency, we should be satisfied with an approximate construction of G.

In fact, the particular structure of the matrix G, although dense, indicates a strong coupling between neighboring nodes and very weak dependence between nodes which are a mesh-size distance apart on the interfaces (see [8]) for details). The property allows itself to be appropriately approximated by a very low-bandwidth sparse matrix. This suggests that the ideas of interface probing techniques [12], which were mainly developed for elliptic PDEs, may be extended for use in the current problem (see [10] for a recent review of interface probe preconditioning).

The primary idea involved in the interface probe construction is to capture the strong dependence between grid points on the interfaces and to ignore the weak dependencies. For this approach, the number of subdomain solves reduces to kn, where k is the number of probing vectors and is quite often just 1 or 3. Numerical experience with interface probe preconditioners was reported in, for example, [9, 21, 22, 30].

Following notations used in [9], IP(0) and IP(1) are the most commonly used among many choices available within this class. However, as in [30], we construct G by retaining the cyclic tridiagonal structure of each T_{ii} in A_{ss} and then replace each entry in the main diagonal of A_{ss} by the corresponding row-sum of G. We denote this interface probe approach by MIP(0). Only one probing vector of the form $(1, 1, ..., 1)^T$ is required for this construction, which requires n inexact subdomain solves.

Numerical experiments confirm that this construction yields better results than those obtained by using IP(0) for the shallow water equations case. Since IP(1) is not found to be superior to IP(0) or MIP(0) in terms of the CPU time, we focus our interests in the MIP(0) interface probing.

With this particular interface probing construction MIP(0) of the matrix G, we can readily observe that the number of subdomain solves involved in using each of these preconditioners for the solution of a linear system at each time step is $(2k_1 + 3)n$, $(k_2 + 2)n$, and $(k_3 + 1)n$, respectively, where k_1, k_2, and k_3 are numbers of iterations required to achieve convergence for an iterative method (where only one matrix–vector multiplication is needed in each iteration, like GMRES) using the first, second, and third types of DD preconditioners, respectively. Note that, in the above counts of subdomain solves, the work required for recovering the final solution $x = B^{-1}\tilde{x}$ is also included.

For these three types of preconditioners, the interface preconditioners G preserve the block diagonal structures of A_{ss}; see (15). We denote G in a unified way by

$$G = \mathrm{diag}[\tilde{T}_{11}, \tilde{T}_{22}, ..., \tilde{T}_{n-1,n-1}]. \tag{45}$$

Then, the linear system $Gu = v$ may be split up into $n - 1$ smaller systems $\tilde{T}_{ii}u_i = v_i$, $i = 1, 2, ..., n - 1$, which may be solved in parallel. Since each of these $\tilde{T}_{ii}$ is cyclic tridiagonal, we use the so-called (see [36]) Ahlberg–Nielson–Walsh algorithm [2], which is an extension of the well-known double-sweep algorithm of Thomas [38]. For an efficient numerical algorithm for solving cyclic pentadiagonal systems, see [29].

As to inexact subdomain solvers, we employ the so-called relaxed incomplete LU factorization (RILU) [3, 4] to approximate the subdomain stiffness matrices A_{ii}, $i = 1, 2, ..., n$, namely, $B_{ii} = L_iU_i = A_{ii} + R_{ii}$, where R_{ii} is an error matrix. In short, given a matrix $A_{n\times n}$, the RILU factorization of A can be obtained in the following $n - 1$ steps of transformation,

$$A = A^1 \rightarrow A^2 \rightarrow \cdots \rightarrow A^n = U,$$

where for $k = 1, 2, ..., n - 1$ the calculations

$$l_{ik} = a_{ik}^{(k)}/a_{kk}^{(k)}$$

$$a_{ij}^{(k+1)} = \begin{cases} a_{ij}^{(k)} - l_{ik}a_{kj}^{(k)}, \\ \quad \text{if } (k + 1 \leq j \leq n) \cap ((i,j) \in J) \cap i \neq j; \\ 0, \\ \quad \text{if } (k + 1 \leq j \leq n) \cap ((i,j) \notin J); \\ a_{ii}^{(k)} - l_{ik}a_{ki}^{(k)} + \omega \sum_{(p=k+1,(i,p)\notin J)}^{n} (a_{ip} - l_{ik}a_{kp}^{(k)}) \\ \quad \text{if } j = i \end{cases}$$

are carried out. Here J is a subset of ordered integer pairs $J \subset \mathcal{J} = \{(i, j) \mid i, j = 1, 2, ..., n\}$ which defines the allowed fillins, where n is the size of the matrix under consideration and $0 \le \omega \le 1$.

To conclude this section, we provide some results corresponding to three types of domain-decomposed preconditioners applied from the left. We may verify that if

$$B^{-1}A = \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix} \tag{46}$$

then entries Q_{ij}'s, $i, j = 1, 2$, of the matrix B^{-1}A are given by

- For the first type of DD preconditioners

$$Q_{11} = B_{dd}^{-1}A_{dd} + B_{dd}^{-1}A_{ds}G^{-1}A_{sd}(B_{dd}^{-1}A_{dd} - I_{dd}) \tag{47}$$

$$Q_{12} = B_{dd}^{-1}A_{ds}[I_{ss} + G^{-1}(A_{sd}B_{dd}^{-1}A_{ds} - A_{ss})] \tag{48}$$

$$Q_{21} = G^{-1}A_{sd}(I_{dd} - B_{dd}^{-1}A_{dd}) \tag{49}$$

$$Q_{22} = G^{-1}(A_{ss} - A_{sd}B_{dd}^{-1}A_{ds}) \tag{50}$$

- For the second type of DD preconditioners

$$Q_{11} = B_{dd}^{-1}(A_{dd} - A_{ds}G^{-1}A_{sd}) \tag{51}$$

$$Q_{12} = B_{dd}^{-1}A_{ds}(I_{ss} - G^{-1}A_{ss}) \tag{52}$$

$$Q_{21} = G^{-1}A_{sd} \tag{53}$$

$$Q_{22} = G^{-1}A_{ss} \tag{54}$$

- For the third type of DD preconditioners

$$Q_{11} = B_{dd}^{-1}A_{dd} \tag{55}$$

$$Q_{12} = B_{dd}^{-1}A_{ds} \tag{56}$$

$$Q_{21} = G^{-1}A_{sd}(I_{dd} - B_{dd}^{-1}A_{dd}) \tag{57}$$

$$Q_{22} = G^{-1}(A_{ss} - A_{sd}B_{dd}^{-1}A_{ds}). \tag{58}$$

We notice that, for left preconditioning, the approximate construction of an interface preconditioner is required for the first and third types of DD preconditioners, but not for the second type.

4. NUMERICAL RESULTS AND DISCUSSIONS

In this section, we present selected numerical results along with some discussions. All numerical experiments are carried out on the CRAY Y-MP/432 vector-parallel supercomputer for the shallow water equations discussed in Section 2. The numerical values of some constants are summarized here

$$L = 6000\,\text{km}, \quad D = 4400\,\text{km}$$
$$g = 10\,\text{m/s}, \quad H_0 = 2000\,\text{m}$$

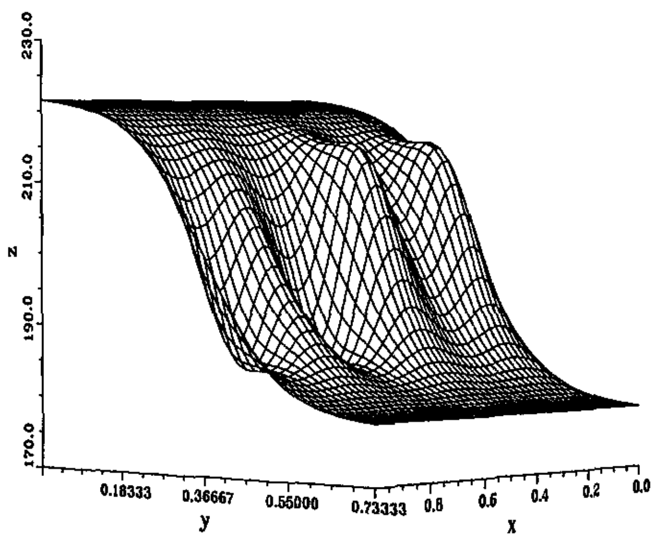

FIG. 1. A 3D view of the initial non-dimensionalized geopotential field.

$$H_1 = 220\,\text{m}, \quad H_2 = 133\,\text{m}$$
$$\hat{f} = 10^{-4}\,\text{s}^{-1}, \quad \beta = 1.5 \times 10^{-11}\,\text{s}^{-1}\,\text{m}^{-1},$$

where L and D are the length and width of the channel, g is the acceleration of gravity, $\hat{f}$ and β are used to determine the Coriolis parameter $f = \hat{f} + \beta(y - D/2)$, and, finally, three constants H_0, H_1, and H_2 are used to analytically define the initial geopotential and velocity fields by the geostrophic relationship,

$$\varphi = gh, \quad u = -(g/f)\frac{\partial h}{\partial y}, \quad v = (g/f)\frac{\partial h}{\partial x}, \tag{59}$$

where the initial height field is given by

$$\begin{aligned} h(x, y) = H_0 &+ H_1 \tanh[9(D/2 - y)/2D] \\ &+ H_2 \operatorname{sech}^2[9(D/2 - y)/2D] \sin(2\pi x/L). \end{aligned} \tag{60}$$

We scale the problem by choosing a reference geopotential $\varphi_0 = 10^2\,\text{m}^2/\text{s}^2$. The corresponding dimensionless constants are

$$L' = 1, \quad D' = 0.7333333$$
$$g' = 6 \times 10^5, \quad H_0' = 0.33333 \times 10^{-3}$$
$$H_1' = 0.366666 \times 10^{-4}, \quad H_2' = 0.221666 \times 10^{-4}$$
$$\hat{f}' = 60, \quad \beta' = 54.$$

With this choice of dimensionless constants, a 3D presentation of the initial condition is seen in Fig. 1.

Although the conjugate gradient method is almost invariably used for the solution of a positive definite linear system, it is

TABLE I

A Comparison of CPU Time (Number of Iterations) Required for Solving the Geopotential Linear System at the End of One Hour with a Half Hour Time Step

Preconditioner types		First type	Second type	Third type	Third type with MIP/(0)
40 × 35	GMRES	0.178 (12)	0.102 (12)	0.088 (11)	0.101 (12)
mesh	CGS	0.199 (7)	0.099 (6)	0.106 (7)	0.111 (7)
resolution	Bi-CGSTAB	0.219 (7)	0.099 (6)	0.092 (6)	0.111 (7)
80 × 75	GMRES	0.89 (14)	0.53 (15)	0.47 (14)	0.52 (15)
mesh	CGS	1.08 (9)	0.54 (8)	0.50 (8)	0.59 (9)
resolution	Bi-CGSTAB	0.97 (8)	0.54 (8)	0.50 (8)	0.58 (9)
120 × 115	GMRES	2.56 (18)	1.49 (19)	1.41 (19)	1.48 (19)
mesh	CGS	3.24 (12)	1.64 (11)	1.56 (11)	1.62 (11)
resolution	Bi-CGSTAB	2.74 (10)	1.63 (11)	1.55 (11)	1.60 (11)

Note. Using GMRES, CGS, and Bi-CGSTAB algorithms accelerated by three types of DD preconditioners.

not apparent which Krylov solver is to be preferred for solving nonsymmetric linear systems.

In this work, we choose to use, among many others, three popular solvers, namely, GMRES [34], CGS [35], and Bi-CGSTAB [37] for solving the geopotential and velocity nonsymmetric linear algebraic systems, accelerated by three types of DD preconditioners discussed above. We choose to use preconditioning from the right only, since it does not yield results very different from those obtained by preconditioning from the left and, moreover, a linear system preconditioned from the right preserves the residual of the original linear system.

4.1. *The Convergence Behavior*

In the first set of numerical experiments, we test relative efficiencies of the three types of DD preconditioners discussed in Section 3, using MIP(0) interface probe construction for the first two types of DD preconditioners and RILU-based subdomain solvers. In addition, we also present computational results corresponding to preconditioners of the third type, however, with G given by (44) instead of the optimal A_{ss}.

For RILU-based subdomain solvers, we compute and compare different values of ω and find that the optimal ω in terms of the number of iterations is in the interval (0, 0.5), depending on which type of DD preconditioners and which iterative method (GMRES, CGS, or Bi-CGSTAB) are used. This result is in contrast to 0.95 found in a recent work [39] for some other computations in the original whole domain. On the other hand, in terms of CPU time, we find that RILU with $\omega = 0$ yields the best performance. Hence, unless otherwise stated, we apply RILU with $\omega = 0$, i.e., the ILU factorization to our problem.

The convergence behavior and CPU time required, respectively, are summarized in Table I for three mesh resolutions of 40×35, 80×75, and 120×115. The stopping criterion is that the final Euclidean residual norm, namely, $\|f - Ax^{(k)}\|_2$, be smaller than 0.1×10^{-10}. A time-step size of $\Delta t = 1800$ s is used throughout unless otherwise stated. Note that timings in Table I do not include those for RILU factorizations.

It should be pointed out that CPU time recorded here is for solving the geopotential linear system only and on a specific time level ($t = 3600$ s). Quite often, however, we need to integrate the shallow water equations set to the end of 10 days. If the time step is a half hour, it would require solving each of the three linear systems, which govern the geopotential and velocity components, 480 times. The geopotential fields corresponding to the end of one day, two days, ..., 10 days are illustrated in [7].

From these results, we see that the three nonsymmetric iterative methods, GMRES, CGS, and Bi-CGSTAB, are competitive with each other. GMRES requires the largest number of iterations to attain convergence. However, this does not mean that GMRES is the most expensive algorithm to use since only one matrix–vector multiplication is required for each GMRES iteration, while two such operations are needed for each CGS or Bi-CGSTAB iteration. However, GMRES imposes a higher demand for storage, which may be alleviated by restarting the procedure, but often at the cost of requiring more iterations for convergence.

We observe that no sizable difference exists in terms of the number of iterations required to attain convergence between these three types of preconditioners. Preconditioners of the first type, first proposed for a symmetric linear system arising from the discretization of some self-adjoint elliptic PDEs, cannot reduce the number of iterations to such an extent as to offset the disadvantage of two inexact subdomain solves in each subdomain for solving the preconditioning linear system $Bp = q$. As a result, they turn out to be the most expensive for the current application. Preconditioners of the second and third types behave much better in terms of CPU time due to only one inexact subdomain solve being required in each subdomain for the solution of $Bp = q$. Moreover, an approximate construction of G is not required on the interfaces for the third type of DD preconditioners.

4.2. *Sensitivities of the Three Types of DD Preconditioners to Inexact Subdomain Solvers*

In the above set of numerical experiments, we obtain that the number of iterations required for convergence is almost the same for three types of preconditioners with ILU subdomain solvers and thus preconditioners of the third type are computationally least expensive. A question that naturally arises is if similar conclusions hold for the same types of preconditioners but with more or less accurate subdomain solvers compared to ILU. This requires us to test the sensitivities of these preconditioners to inexact subdomain solvers.

To test the sensitivities of the preconditioners given in (21), (29), and (32) to inexact subdomain solvers, we use the idea of m-step preconditioning [1]. For this preconditioning approach, we consider a splitting of the matrix $A_{ii} = P_i - Q_i$ and define $G_i = P_i^{-1}Q_i$. As an approximation of the subdomain matrix A_{ii}, we take $B_{ii} = P_i(\sum_{k=0}^{m-1} G_i^k)^{-1}$ for $i = 1, 2, \dots, n$. The solution of $B_{ii}p_i = q_i$ may be easily verified to be equivalent to m iterations the linear stationary iterative scheme $P_i p_i^{k+1} = Q_i p_i^k + q_i$ with the initial solution $p_i^0 = 0$.

The P_i may be taken to be any easily invertible simple matrix as long as the spectral radius $\rho(G_i) < 1$. For example, P_i may be chosen to be the ILU factorization of $P_i = L_iU_i$ of A_{ii}, or simply the diagonal part $P_i = D_i$ of the matrix A_{ii}. Once P_i has been chosen, B_{ii} often becomes a better inexact subdomain solver as m increases, in the sense that the number of iterations required to reach a prescribed convergence criterion for solution of the original linear system $Ax = f$ decreases. By gradually increasing m, we are able to observe the relative performance behavior of these DD preconditioners corresponding to increasingly accurate subdomain solvers.

For the current experiment, we decided to take P_i as the lower triangular part (including the diagonal part) of the matrix A_{ii}. This corresponds to m Gauss–Seidel linear stationary iterations in each subdomain. However, this choice of P_i does not guarantee proper subdomain solvers for all mesh resolutions in our case. For example, we found that SOR iterations with $\omega = 0.5$ are appropriate for an 80×75 mesh resolution. In Tables II–IV we report results obtained with a 40×35 resolution, of GMRES, CGS, and Bi-CGSTAB iterations corresponding to three types of preconditioners, respectively

From these three tables, we see that preconditioners of the third type can accelerate the convergence of three iterative methods at about the same rate as the other two types of preconditioners, except for cases when subdomain solvers may be considered to be exact or close to exact. Similar observations are obtained for the case of mesh resolutions higher than 40×35. Since, in practice, the inexact subdomain solvers are far from exact, we consider preconditioners of the third type to be the best.

TABLE III

Iteration Counts as a Function of m Using CGS Accelerated by Three Types of DD Preconditioners

$m = 1$	$m = 2$	$m = 3$	$m = 4$	$m = 5$	$m = 6$	$m = 7$
15	8	7	6	6	5	5
14	8	7	6	6	5	5
16	9	6	7	6	7	6

4.3. *Extensions to the Cases of More Than Four Subdomains*

For implementation on a large number of processors or, ambitiously, for massively parallel processing implementation, it is very desirable for domain decomposition algorithms to possess convergence rates that do not deteriorate as the number of subdomains increases. Unfortunately, it is often the case, rather than the exception, that the number of iterations will increase as the number of subdomains increases, even though the discrete problem size is kept fixed and the stopping criterion remains the same, for both overlapping and nonoverlapping domain decomposition cases.

In a few cases, the optimal preconditioners, in the sense that the convergence rate is independent of both the mesh size h and the typical subdomain size H, are known. In other cases, nearly optimal preconditioners have been constructed with the property that the condition number of the preconditioned matrix is proportional to $(1 + \log(H/h))^m$, $m = 2$ or 3 (see [14, 15]

TABLE II

Iteration Counts as a Function of m Using GMRES Accelerated by Three Types of DD Preconditioners

$m = 1$	$m = 2$	$m = 3$	$m = 4$	$m = 5$	$m = 6$	$m = 10$	$m = 20$
26	15	12	11	10	10	9	8
27	15	12	11	10	10	9	8
28	15	12	11	10	10	10	10

TABLE IV

Iteration Counts as a Function of m Using BI-CGSTAB Accelerated by Three Types of DD Preconditioners

$m = 1$	$m = 2$	$m = 3$	$m = 4$	$m = 5$	$m = 6$
14	8	7	6	6	5
14	8	7	6	6	5
14	8	6	6	6	6

TABLE V

Iteration Counts of the GMRES Algorithm Accelerated by Three Types of DD Preconditioners for Various Mesh Resolutions and Numbers of Subdomains

Mesh resolutions	Types of preconditioners	$n = 2$	$n = 4$	$n = 8$	$n = 16$
36×31	First	12	12	12	13
	Second	12	12	12	14
	Third	11	11	11	14
85×79	First	15	15	15	15
	Second	15	15	16	17
	Third	15	15	16	18
120×111	First	18	18	18	19
	Second	19	19	20	22
	Third	19	19	20	23
150×143	First	21	21	21	23
	Second	23	23	24	27
	Third	24	24	25	27

and references cited therein). One of the most important reasons for the success of most of these preconditioners is the introduction, aimed at enhancing communications amongst subdomains, of a much smaller global problem corresponding to the discretization on a coarse grid. However, we notice that these optimal, or nearly optimal, domain decomposition algorithms may not provide computationally the cheapest way to obtain solutions to specific problems. The convergence rate alone does not tell the whole story of the computational complexity.

In the following, we provide some numerical results corresponding to the convergence rates of the GMRES algorithm with each of the three types of DD preconditioners for $n = 2$, 4, 8, and 16 subdomains and for various mesh resolutions (see Table V).

We observe that the number of iterations increases only very mildly for each fixed-size problem as the number of subdomains increases from two to 16. Similar numerical results were reported in [23–25]. Thus, it is a worthwhile effort to implement these domain decomposition algorithms on such parallel computers as the CRAY C90 which has a total maximum number of 16 processors able to perform in parallel.

5. PARALLEL IMPLEMENTATION RESULTS

Being the most computationally expensive stage, solutions of systems of algebraic equations resulting from finite element discretization can be sought in parallel by employing domain-decomposition algorithms presented in this paper. However, as is well known, the parallel performance result will be seriously degraded when even a small percentage of the total work (measured in CPU time) is not processed in parallel. This implies that a good speed-up due to parallelism may not be achieved unless computations related to the finite element discretization are also efficiently parallelized

In view of this, we will exploit not only the parallelism corresponding to subdomain by subdomain computations via domain decomposition techniques, but also the parallelism inherent in element by element calculations. Subroutine level parallelism may be sought for subdomain by subdomain computations, e.g., subdomain preconditionings and inexact solves. Loop level parallelism may be exploited for element by element calculations, e.g., the formation of element stiffness matrices and the assembly into a global stiffness matrix.

For the finite element discretization, the setup of local stiffness matrices and their assembly into a single global stiffness matrix is the only part of the calculations which requires being repeatedly carried out as the computation marches over time. The efficient parallelization of this part is thus highly critical to the overall parallel performance.

As was pointed out in [17] (see also some references therein), the critical region in the assembly process is not inherent in the element-by-element calculations and may be bypassed by assembling the element matrices in a particular order. The basic observation is that the necessity of introducing a critical region into the assembly process is due to the possible simultaneous contributions to a common node by more than one of its surrounding elements. The critical region may be removed if the assembly process can be carried out in groups such that, within each group, no two or more elements connected to a common node are able to make contributions to that node. This idea may be realized by a multicolor numbering of the elements to be assembled.

For the triangular linear element mesh used in the current problem [27], six colors are required to guarantee that any node in the physical domain is surrounded by elements of different colors (see Fig. 2), where each integer represents a unique color. The elements in the mesh may now be divided into six groups. Elements of the same color comprise one group and different groups have different colors. The assembly is carried

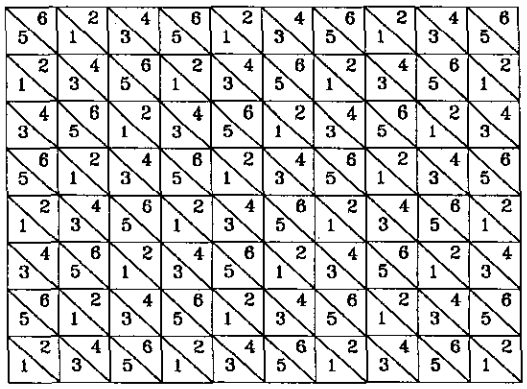

FIG. 2. Multicolor numbering of elements for a triangular element mesh. Each integer stands for a unique chosen color. A node in the mesh is surrounded by elements of different colors.

out one group after the other. Within each group, the elements are internally disjoint and so the parallel assembly process is made asynchronously. Notice that the multicoloring scheme achieves not only the removal of critical regions, but also a sharp reduction in the number of synchronization points.

The domain decomposition code corresponding to the use of the third type of DD preconditioners was carefully tuned and tested on the four-processor CRAY Y-MP/432 for a number of mesh resolutions. The main computational work for the three nonsymmetric iterative methods employed in this paper is associated with matrix–vector products, which are obtained through approximately solving the problem defined on each subdomain. The parallelization of these tasks is carried out at subroutine levels. Since we do not find a sizable difference in speedup results obtained by using GMRES, CGS or Bi-CGSTAB, the parallel performance results to be reported in what follows correspond to the Bi-CGSTAB iterative method.

Prior to presenting the results, we need to clarify what we mean by speedup. Speedup may be defined to be the ratio of the wall clock time elapsed in a dedicated system, for computing the same problem, using the best serial algorithm and the parallel algorithm. However, the optimal serial algorithm is usually unknown, especially for modeling a meaningful physical process. In our problem, use of GMRES, CGS, or Bi-CGSTAB preconditioned by ILU in the original domain consumes more CPU time than the use of the same iterative method and preconditioner treated in a domain-decomposed way. In other words, apart from parallelization issues, consideration of computational complexity alone justifies the use of domain decomposition.

Since the best serial treatment of the present problem cannot be determined, the speedup results reported here refer to measurements of wall clock time in a dedicated system relative to the uni- and multi-processor implementation of the same domain decomposition algorithm (see also, among others, [20, 22, 26]). This definition properly incorporates communication overhead and synchronization delays and shows how well the domain-to-processor mappings are done. However, this definition presents a serious drawback. Following this definition, a parallel algorithm achieving a perfect speedup may actually take longer time to execute than a serial algorithm for solving the same problem.

We integrated the finite element model of the shallow water equations for four different mesh resolutions, namely, 19×15, 34×27, 49×43, and 64×55 for a period of 5 h with corresponding time-step sizes of $\Delta t = 1800$ s, 1000 s, 600 s, and 400 s, respectively. The experimental results are summarized in Table VI. It should be pointed out that the automatic do-loop level parallelization as detected and exploited by the autotasking preprocessor does not yield a speedup larger than two. The reason is that autotasking is unable to detect parallelism across subroutine boundaries. To explore subroutine level parallelism as offered by domain decomposition, one has to manually insert appropriate autotasking directives into the code.

TABLE VI

Parallel Performance Results for Four Different Mesh Resolutions Using Bi-CGSTAB Preconditioned by the Third Type of DD Preconditioners on the Four-Processor CRAY Y-MP/432 Vector Parallel Supercomputer

Mesh resolutions	19×15	34×27	49×43	64×55
Serial seconds	1.03	6.26	35.19	108.07
Parallel seconds	0.38	2.03	10.29	29.77
Speedup ratios	2.7	3.1	3.4	3.6

6. SUMMARY AND CONCLUSIONS

In this paper, two types of existing DD preconditioners and a novel one, proposed here, were used to precondition three well-known Krylov iterative methods, GMRES, CGS, and Bi-CGSTAB, for the efficient numerical solution of systems of nonsymmetric linear algebraic equations resulting from the implicit time discretization of a finite element model of the shallow-water equations. For all test cases, preconditioners of the first type are roughly twice as expensive to use as those of the second and third types. Preconditioners of the third type turned out to be computationally the least expensive.

We observe that, for all cases, GMRES requires roughly twice as many iterations as required by the CGS or Bi-CGSTAB methods to satisfy the same convergence criterion. However, all three algorithms are nearly equally efficient in terms of CPU time for the current application.

Autotasking is able to exploit both small or large granularity parallelism efficiently. However, by relying on the automatic detection and exploitation of do-loop level parallelism offered by autotasking, the speedup of domain decomposition code turns out to be very low. Large-granularity parallelism inherent to domain decomposition approach cannot be detected automatically by the autotasking preprocessor and must be exploited by building case/end case structures into parallel regions.

Parallelization issues and numerical results on the shared memory CRAY C-90 and distributed memory IBM SP-2 parallel computers in the context of shallow-water flow modeling will be reported and discussed in detail in another paper. We are also looking into more challenging issues on domain-decomposition algorithms designed to efficiently handle an irregular domain discretized with an unstructured mesh. The results obtained from the current work will partly serve as a benchmark to those more complicated issues we are now investigating.

ACKNOWLEDGMENTS

The authors thank Thomas Oppe for helpful discussions and David Keyes for pointing out an error in the original manuscript of the paper. We greatly appreciate the invaluable and constructive suggestions provided by two anonymous referees. The research of the first author was supported by U.S. DOE Contracts 120054323 and DE-FC05-85ER250000, through the Supercomputer Computations Research Institute of Florida State University.

REFERENCES

1. L. Adams, *SIAM J. Sci. Stat. Comput.* **6**(2), 452 (1985).
2. H. H. Ahlberg, E. N. Nielson, and J. L. Walsh, *The Theory of Splines and Their Application,* Mathematics in Science and Engineering, Vol. 38 (Academic Press, New York, 1967).
3. O. Axelsson, *J. Comput. Appl. Math.* **12/13,** 3 (1985).
4. O. Axelsson and G. Lindskog, *Numer. Math.* **48,** 479 (1986).
5. S. Barnett, *Matrices, Methods and Applications* (Clarendon Press, Oxford, 1990).
6. J. H. Bramble, J. E. Pasciak, and A. H. Schatz, *Math. Comput.* **47,** 103 (1986).
7. Y. Cai, Technical Report FSU-SCRI-94T-54 (Ph.D. dissertation), Department of Mathematics and Supercomputer Computations Research Institute, Florida State University, 1994.
8. Y. Cai and I. M. Navon, ''Parallel Domain Decomposed Preconditioners for the Finite Element Shallow Water Flow Modeling,'' in *Seventh International Conference on Domain Decomposition Methods in Scientific and Engineering Computing*, edited by D. E. Keyes and J. Xu, AMS Series of Contemporary Mathematics (Amer. Math. Soc., 1995), p. 471.
9. T. F. Chan and D. E. Keyes, ''Interface Preconditioner for Domain-Decomposed Convection–Diffusion Operators,'' in *Third International Symposium on Domain Decomposition Methods for Partial Differential Equations*, edited by T. F. Chan, R. Glowinski, J. Périaux, and O. B. Widlund. SIAM, Philadelphia, 1990), p. 245.
10. T. F. Chan and T. P. Mathew, *SIAM J. Matrix Anal. Appl.* **13**(1), 212 (1992).
11. T. F. Chan and T. P. Mathew, *Acta Numer.* 61 (1994)
12. T. F. Chan and D. Resasco, Technical Report 414, Computer Science Dept., Yale University, 1985.
13. J. J. Dongarra, I. S. Duff, D. C. Sorensen, and H. A. Van der Vorst, *Solving Linear Systems on Vector and Shared Memory Computers* (SIAM, Philadelphia, 1991).
14. M. Dryja and O. B. Widlund, ''Some Domain Decomposition Algorithms for Elliptic Problems, in *Iterative Methods for Large Linear Systems,* edited by D. R. Kincaid and L. J. Hayes, (Academic Press, San Diego, 1990) p. 273.
15. M. Dryja and O. B. Widlund, ''Towards a Unified Theory of Domain Decomposition Algorithms for Elliptic Problems,'' in *Third International Symposium on Domain Decomposition Methods for Partial Differential Equations,* edited by T. F. Chan, R. Glowinski, J. Périaux, and O. B. Widlund (SIAM, Philadelphia, 1990), p. 3.
16. L. C. Dutto, W. G. Habashi, and M. Fortin, *Comput. Methods Appl. Mech. Eng.* **117,** 15 (1994).
17. C. Farhat and L. Crivelli, *Comput. Methods Appl. Mech. Eng.* **72,** 153 (1989).
18. C. A. J. Fletcher, *Computational Techniques for Fluid Dynamics I* (Springer-Verlag, Berlin/Heidelberg, 1988).
19. A. Grammeltvedt, *Mon. Weather Rev.* **97**(5), 384 (1969).
20. W. D. Gropp and D. E. Keyes, *SIAM J. Sci. Stat. Comput.* **9**(2), 312 (1988).
21. D. E. Keyes and W. D. Gropp, *SIAM J. Sci. Stat. Comput.* **8**(2), 166 (1987).
22. D. E. Keyes and W. D. Gropp, ''Domain Decomposition for Nonsymmetric Systems of Equations: Examples from Computational Fluid Dynamics,'' in *Second International Symposium on Domain Decomposition Methods for Partial Differential Equations,* edited by T. F. Chan, R. Glowinski, J. Périaux, and O. B. Widlund, (SIAM, Philadelphia, 1989), p. 321.
23. G. Meurant, *Int. J. Supercomputer Appl.* **2**(4), 5 (1988).
24. G. Meurant, ''Domain Decomposition versus Block Preconditioning,'' in *First International Symposium on Domain Decomposition Methods for Partial Differential Equations,* edited by R. Glowinski, G. H. Golub, G. A. Meurant, and J. Périaux (SIAM, Philadelphia, 1988), p. 231.
25. G. Meurant, ''Incomplete Domain Decomposition Preconditioners for Nonsymmetric Problems,'' in *Second International Symposium on Domain Decomposition Methods for Partial Differential Equations,* edited by T. F. Chan, R. Glowinski, J. Périaux, and O. B. Widlund (SIAM, Philadelphia, 1989), p. 219.
26. R. Natarajan, *J. Comput. Phys.* **94,** 352 (1991).
27. I. M. Navon, *J. Comput. Phys.* **52,** 313 (1983).
28. I. M. Navon, *Comput. Geosci.* **13,** 255 (1987).
29. I. M. Navon, *Commun. Numer. Methods* **3,** 63 (1987).
30. I. M. Navon and Y. Cai, *Comput. Methods Appl. Mech. Eng.* **106**(1-2), 179 (1993).
31. J. M. Ortega, *Introduction to Parallel and Vector Solution of Linear Systems* (Plenum, New York, 1988).
32. J. M. Ortega and R. G. Voigt, *SIAM Rev.* **27,** 147 (1985).
33. J. S. Przemieniecki, *AIAA J.* **1,** 138 (1963).
34. Y. Saad and M. H. Schultz, *SIAM J. Sci. Stat. Comput.* **7**(3), 856 (1986).
35. P. Sonneveld, *SIAM J. Sci. Stat. Comput.* **10**(1), 36 (1989).
36. C. Temperton, *J. Comput. Phys.* **19,** 317 (1975).
37. H. A. van der Vorst, *SIAM J. Sci. Stat. Comput.* **13**(3), 631 (1992).
38. R. S. Varga, *Matrix Iterative Analysis* (Prentice–Hall, Englewood Cliffs, NJ, 1962).
39. C. Vuik, *Int. J. Numer. Methods Fluids* **16,** 507 (1993).