Computer Methods in Applied Mechanics and Engineering 106 (1993) 179–212
North-Holland

CMA 369

Domain decomposition and parallel processing of a finite element model of the shallow water equations

I.M. Navon and Y. Cai

Department of Mathematics and Supercomputer Computations Research Institute, Florida State University, Tallahassee, FL 32306, USA

Received 19 June 1992

We present nonoverlapping domain decomposition techniques applied to a two-stage Numerov–Galerkin finite element model of the shallow water equations over a limited-area domain. The Schur complement matrix formulation is employed and a modified interface matrix approach is proposed to handle the coupling between subdomains. The resulting nonsymmetric Schur complement matrices, modified interface matrices as well as the subdomain coefficient matrices are solved using Preconditioned Conjugate Gradient Squared (PCGS) non-symmetric iterative solvers. Various stages of the finite element solution are parallelized and the code is implemented on a four processor CRAY Y-MP supercomputer applying multitasking techniques in a dedicated environment.

1. Introduction and motivation

Recently there has been an increase in research activities in the area of parallel computing due to the advent and growth of various parallel processing architectures. The domain decomposition approach achieves the highest level of parallelism in the numerical solution of partial differential equations. Specifically, the problem defined on the original domain of interest is explicitly split up into several smaller subproblems defined on subdomain regions. Each of these subproblems is associated with a task or software process which is scheduled onto a different processor for parallel processing.

The idea of domain decomposition goes back to Schwarz [1] or the work of structural engineers in the early 1960s [2]. However, it was only during the last 10 years that researchers developed and extended these ideas for use with parallel architectures. While taking advantage of available modern powerful parallel computers is the main reason for carrying out this research, there are additional reasons for developing this technique. Some of these are listed below:

(1) The domain decomposition technique can be employed for the solution of problems defined on irregular domains when one is applying finite difference or spectral methods and thus rendering these numerical schemes more flexible and versatile geometrically (see [3, 4]).

Correspondence to: Dr. I.M. Navon, Department of Mathematics and Supercomputer Computations Research Institute, Florida State University, Tallahassee, FL 32306, USA.

(2) The domain decomposition techniques also allow us to use different numerical schemes and different resolutions (or different types and orders of elements for the finite element method) for different subdomains (see, for example, [5]). Thus they allow us to combine the advantages of finite element, spectral and multigrid methods and provide opportunities for devising more efficient and accurate algorithms applicable to parallel architectures (see [6] and references therein).
(3) It also provides us with an easy means for isolating regions which require special treatment. Thus the computational effort can be focused upon regions where large gradients, boundary layers, shocks or even singularities occur (see [7, 8]).
(4) The technique is particularly useful for problems which require different mathematical models for different subdomains, for instance, in fluid dynamics, using a viscous model near the boundary and an inviscid model in the far field. Two good examples are in [9, 10].

There are two main approaches for the domain decomposition, namely overlapping and nonoverlapping, characterized by the way the subdomains are constructed. For the nonoverlapping approach, most of the research, up to now, has been almost exclusively focused on the interface(s), or more specifically, on finding good preconditioners of the Conjugate Gradient (CG) algorithm [11] for the symmetric capacitance linear systems arising from finite difference or finite element discretizations of elliptic partial differential equations in two- or three-dimensional domains (see [12–18] for more details).

The primary reason for this focus is that the iterative solution of the interface Schur complement matrix system involves repeated solutions of all the subdomain problems and the interface solver itself is a potential bottleneck for the coarse-grained parallelism.

In this paper, we extend the applicability of domain decomposition methods to a set of coupled nonlinear hyperbolic shallow water partial differential equations defined on a 2-D limited-area domain, using finite element discretization in space and an implicit integration scheme in time.

Due to the paramount importance of the shallow water equations in meteorology and oceanography, where they serve as test models for the development of new algorithms, the efficient finite element solution of the shallow water equations has attracted the interest of many researchers. A tremendous amount of work has been done in this direction, see, for example, [19–28], to cite but a few references. These algorithms were not, however, designed to run efficiently on various parallel processing architectures.

In the present paper, we report our work on the Schur complement matrix and modified interface matrix approaches applied to a two-stage Numerov–Galerkin finite element model of the shallow water equations [24, 25] over a limited-area domain. The main points in each of the following sections are summarized below.

In Section 2, we give a mathematical presentation of the 2-D shallow water equations under consideration along with a nondimensionalized version of the problem, followed by a brief presentation in Section 3 of the essential components of the two stage Numerov–Galerkin finite element method. In Section 4, the Schur complement and relevant formulas for n ($n \geq 2$) nonoverlapping subdomains are briefly reviewed.

One of the key steps for carrying out the Schur complement domain decomposition is to obtain the so-called arrow-head matrix, from which information can be derived regarding the Schur complement matrix or the modified interface matrix and the systems governing the

various subdomain problems. In Section 5 we present a scheme which transforms the global finite element automatically into this arrow-head matrix.

In Section 6, we describe a computational algorithm employed for our numerical experiments of a relatively new nonsymmetric iterative solver [29] of the Lanczos type, PCGS (preconditioned conjugate gradient squared) along with other relevant issues. Section 7 deals with the Schur complement matrix approach to our problem. A class of relatively robust and efficient boundary probe preconditioners [30] was employed to accelerate the convergence on the interfaces.

The theory and algorithms for a modified interface matrix approach of the nonoverlapping domain decomposition are presented in Section 8. The modified interface matrix is constructed by using a Neumann series expansion and the MILU (modified incomplete LU) factorization in each subdomain.

For this approach, it is observed that iterative improvements of the initial guesses are possible for the iterative solutions in the subdomains as well as on the interfaces. The reduction in the number of iterations, due to the improved initial guesses, for the solution of the subdomain problems may be regarded as a remedy for the disadvantage that no fast solvers are available in the subdomains. The iterative improvement of the initial guesses made on the interfaces greatly reduces the computational cost and thus the parallel processing overhead for the solution of the modified interface matrix linear system.

In Section 9, we present the results of various numerical experiments along with the parallel implementation of our algorithms on the four-processor Cray Y-MP/432 supercomputer in a dedicated environment by applying multitasking techniques. Finally, summary and conclusions are presented in Section 10.

To conclude the introduction, we point out the existence of another good approach which constructs a domain-decomposed preconditioner for the simultaneous iterative procedure on the whole domain. One of the advantages of this approach is that only approximate subdomain solvers are required. This alternative approach was first proposed in [31, 32], and further studied in [33, 34]. Our work based on this approach along with the application and comparison of various nonsymmetric iterative solvers for the shallow water equations will be reported in another paper [35].

2 Problem presentation

The shallow water equations are a set of first order nonlinear hyperbolic partial differential equations having many important applications in meteorology and oceanography. These equations can be used in studies of tides and surface water run-off. They may also be used to study large-scale waves in the atmosphere and ocean if terms representing the effects of the earth's rotation (Coriolis terms) are included.

Indeed, it has become customary in developing new numerical methods for numerical weather prediction or oceanography, to study first the simpler nonlinear shallow water equation system, which possesses the same mixture of slow- and fast-moving waves as the more complex baroclinic three-dimensional primitive equations of motion.

Here we are concerned with the domain decomposition solution of the 2-D shallow water

equations on a limited-area domain discretized by finite element approximations. The 2-D shallow water equations in Cartesian coordinates assume the following form:

$$\frac{\partial \varphi}{\partial t} + u\frac{\partial \varphi}{\partial x} + v\frac{\partial \varphi}{\partial y} + \varphi\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) = 0, \tag{2.1}$$

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + \frac{\partial \varphi}{\partial x} - fv = 0, \tag{2.2}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + \frac{\partial \varphi}{\partial y} + fu = 0, \tag{2.3}$$

$$0 \leqslant x \leqslant L, \quad 0 \leqslant y \leqslant D, \quad t > 0.$$

The last two equations are termed momentum equations, while the first equation is the continuity equation. The Coriolis parameter is given by the β-plane approximation

$$f = \hat{f} + \beta(y - D/2), \tag{2.4}$$

where L and D are the dimensions of a rectangular domain of area $A = LD$, u and v are the velocity components in the x and y directions, respectively, h is the depth of the fluid, $\varphi = gh$ denotes geopotential, g is the acceleration of gravity, f is the Coriolis parameter, and $\hat{f}$ and β are two constants.

Periodic boundary conditions are assumed in the x direction for all the field variables. Let $w = (u, v, \varphi)^t$. Then

$$w(x, y, t) = w(x + L, y, t). \tag{2.5}$$

In the y direction, rigid boundary conditions are specified only for the v-velocity component, that is,

$$v(x, 0, t) = v(x, D, t) = 0. \tag{2.6}$$

Initial conditions need to be specified for the u, v and φ fields, namely

$$w(x, y, 0) = w_0(x, y). \tag{2.7}$$

Under these boundary and initial conditions, the total energy

$$E = \frac{1}{2}\int_0^L \int_0^D (u^2 + v^2 + \varphi)\frac{\varphi}{g}\,dx\,dy \tag{2.8}$$

is independent of time, i.e., it is an integral invariant.

The test problem used here is one proposed in [36] for the nonlinear shallow water equations in a channel on the rotating earth, the initial conditions being determined by the following initial height field:

$$h(x, y) = H_0 + H_1 \tanh\left(\frac{9(D/2 - y)}{2D}\right) + H_2 \operatorname{sech}^2\left(\frac{9(D/2 - y)}{2D}\right) \sin\left(\frac{2\pi x}{L}\right). \tag{2.9}$$

The initial geopotential φ and velocity fields u and v are derived from the initial height field using the geostrophic relationship

$$\varphi = gh\,, \qquad u = -(g/f)\frac{\partial h}{\partial y}\,, \qquad v = (g/f)\frac{\partial h}{\partial x}\,. \tag{2.10}$$

The dimensionless form of any physical problems is not unique. A possible nondimensionalized version of the above set of the shallow water equations is obtained by introducing the following dimensionless variables, where φ_0 is a pre-selected reference geopotential:

$$\begin{aligned} x' &= x/L\,, & y' &= y/L\,, \\ t' &= t\sqrt{\varphi_0}/L\,, & \varphi' &= \varphi/\varphi_0\,, \\ u' &= u/\sqrt{\varphi_0}\,, & v' &= v/\sqrt{\varphi_0}\,, \\ h' &= h/L\,, & H_0' &= H_0/L\,, \\ H_1' &= H_1/L\,, & H_2' &= H_2/L\,, \\ g' &= gL/\varphi_0\,, & f' &= fL/\sqrt{\varphi_0}\,. \end{aligned} \tag{2.11}$$

By using (2.11) and dropping the primes, our problem can be shown to be governed by (2.1)–(2.7), (2.9) and (2.10) with L being 1, D replaced by D/L, $\hat{f}$ by $L\hat{f}/\sqrt{\varphi_0}$ and β by $L^2\beta/\sqrt{\varphi_0}$.

3. The two-stage Numerov–Galerkin finite element method

Using linear piecewise polynomials on triangular elements and a time extrapolated Crank–Nicolson time differencing scheme [20, 21], the usual Galerkin form of the shallow-water equations yields

$$M(\phi_i^{n+1} - \phi_j^n) - \tfrac{1}{2}\Delta t\, K_1(\phi_j^{n+1} + \phi_j^n) = 0\,, \tag{3.1}$$

$$M(u_i^{n+1} - u_j^n) + \tfrac{1}{2}\Delta t\, K_2^*(u_j^{n+1} + u_j^n) + \tfrac{1}{2}\Delta t\, K_2^*(K_{21}^{n+1} + K_{21}^n) + \Delta t\, P_2 = 0\,, \tag{3.2}$$

$$M(v_i^{n+1} - v_j^n) + \tfrac{1}{2}\Delta t\, K_3^*(v_j^{n+1} + v_j^n) + \tfrac{1}{2}\Delta t\, K_2^*(K_{31}^{n+1} + K_{31}^n) + \Delta t\, P_3 = 0\,, \tag{3.3}$$

for the continuity and the u- and v-momentum equations, respectively.

Here n is the time level ($t_n = n\,\Delta t$), Δt is the time-step, M is the mass matrix given by

$$M = \iint_A V_j V_i \,\mathrm{d}A\,, \tag{3.4}$$

where V_j is the basis function. $\phi_j(t)$, $u_j(t)$ and $v_j(t)$ are the discrete scalar nodal values for the geopotential φ and velocity fields u and v, respectively. The following matrix definitions have been used:

$$K_1 = \iint_A V_i V_k u_k^* \frac{\partial V_j}{\partial x} \, dA + \iint_A V_i V_k v_k^* \frac{\partial V_j}{\partial y} \, dA , \tag{3.5}$$

$$K_2^* = \iint_A u_K^* V_K V_i \frac{\partial V_j}{\partial x} \, dA + \iint_A v_K^* V_K V_i \frac{\partial V_j}{\partial y} \, dA , \tag{3.6}$$

$$K_{21}^{n+1} = \iint_A \phi_K^{n+1} \frac{\partial V_K}{\partial x} V_i \, dA , \tag{3.7}$$

$$P_2 = \iint_A f v_K^* V_k V_i \, dA , \tag{3.8}$$

$$K_3^* = \iint_A u_K^{n+1} V_K \frac{\partial V_j}{\partial x} \, dA + \iint_A v_K^* V_K \frac{\partial V_j}{\partial x} \, dA , \tag{3.9}$$

$$K_{31}^{n+1} = \iint_A \phi_K^{n+1} \frac{\partial V_K}{\partial y} V_i \, dA , \tag{3.10}$$

$$P_3 = \iint_A f u_K^{n+1} V_k V_i \, dA , \tag{3.11}$$

and similar definitions for K_{31}^n and K_{21}^n, respectively.

3.1. *Truncation error for the single-stage Galerkin method*

The single-stage Galerkin method [19] was applied to the nonlinear advective terms of form $\boldsymbol{v}\nabla\boldsymbol{v}$. If we consider the advective operator

$$L(u, v) = u \frac{\partial v}{\partial x} , \tag{3.12}$$

then, as shown in [19], we can consider a direct Galerkin approximation using two functions:

$$u = e^{ikx} , \qquad v = e^{ilx} , \tag{3.13a}$$

and with

$$\xi = kh , \qquad \eta = lh , \tag{3.13b}$$

where h is a positive mesh length. One can show the asymptotic truncation error of $u\,\partial v/\partial x$ is (by assuming Fourier modes)

$$|\text{T.E.}| \sim \frac{[4\eta^4 + 8\eta^2\xi + 7\eta^2\xi^2 - 2\eta\xi^3]}{720} . \tag{3.14}$$

For $\xi = \eta$, we obtain

$$|\text{T.E.}| \sim \frac{17}{720}\eta^4 . \tag{3.15}$$

3.2. Truncation error for the two-stage Numerov–Galerkin method

In this approach one calculates the Galerkin approximation to $\partial v/\partial x$ which we denote by Z:

$$\frac{1}{6}Z_{j-1} + \frac{2}{3}Z_j + \frac{1}{6}Z_{j-1} = \frac{1}{2}h^{-1}(V_{j+1} - V_{j-1}) , \tag{3.16}$$

then we calculate the product

$$W = u\frac{\partial v}{\partial x} , \tag{3.17}$$

$$\frac{1}{6}W_{j+1} + \frac{2}{3}W_j + \frac{1}{6}W_{j-1} = \frac{1}{12}(U_{j-1}Z_{j-1} + U_{j-1}Z_j + U_jZ_{j-1} + U_jZ_{j+1} + U_{j+1}Z_j + U_{j+1}Z_{j+1}) + \frac{1}{2}U_jZ_j . \tag{3.18}$$

It can be shown that this algorithm has an asymptotic truncation error of

$$\frac{|\text{T.E.}|}{\text{Two-stage N.G.}} \sim \frac{[2\xi^3\eta + 3\xi^2\eta^2 + 2\xi\eta^3 - 4\eta^4]}{720} , \tag{3.19}$$

and if $\xi = \eta$

$$\frac{|\text{T.E.}|}{\text{Two-stage N.G.}} \sim \frac{3}{720}\eta^4 , \tag{3.20}$$

i.e., an error at least six times smaller than (3.15) (see [24]).

3.3. Numerical implementation of the Numerov–Galerkin method

In our approach, we combine the two-stage Galerkin method with a high-order compact implicit difference approximation to the first derivative.

This approximation has a truncation-error of $o(h^{4l})$ and uses a finite difference stencil of $2l + 1$ grid points, at the price of solving a $2l + 1$ banded matrix (see [37, 23]). The compact Numerov $O(h^8)$ approximation to $\partial v/\partial x$ is given by

$$\frac{1}{70}\left[\left(\frac{\partial v}{\partial x}\right)_{i+2} + 16\left(\frac{\partial v}{\partial x}\right)_{i+1} + 36\left(\frac{\partial v}{\partial x}\right)_i + 16\left(\frac{\partial v}{\partial x}\right)_{i-1} + \left(\frac{\partial v}{\partial x}\right)_{i-2}\right] = \frac{1}{84h}[-5v_{i-2} - 32v_{i-1} + 32v_{i+1} + 5v_{i+2}] , \tag{3.21}$$

where $h = \Delta x = \Delta y$.

The estimation of $\partial v/\partial x$ necessitates solving a pentadiagonal system of the form

$$\frac{1}{70}\begin{bmatrix} 36 & 16 & 1 & & & & \\ 16 & 36 & 16 & 1 & & & \\ 1 & 16 & 36 & 16 & 1 & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & 1 & 16 & 36 & 16 & 1 \\ & & & 1 & 16 & 36 & 16 \\ & & & & 1 & 16 & 36 \end{bmatrix}\begin{bmatrix} \frac{\partial v}{\partial x} \end{bmatrix}$$

$$= \frac{1}{84h}\begin{bmatrix} 0 \\ -5v_0 - 32v_1 + 32v_3 + 5v_4 \\ -5v_1 - 32v_2 + 32v_4 + 5v_5 \\ \vdots \\ -5v_{N_y-4} - 32v_{N_y-2} + 32v_{N_y} + 5v_{N_y+1} \\ -5v_{N_y-3} - 32v_{N_y-2} + 32v_{N_y} + 5v_{N_y+1} \end{bmatrix} \tag{3.22}$$

for $j = 1, 2, \ldots, N_y$.

Here we interpolate for the boundary values of v_0 and v_{N_y+1} using

$$\begin{aligned} v_0 &= 4v_1 - 6v_2 + 4v_3 - v_4 , \\ v_{n_y+1} &= 4v_{N_y} - 6v_{N_y-1} + 4v_{N_y-2} - v_{N_y-3} , \end{aligned} \tag{3.23}$$

while for the intermediate expression Z, we have

$$Z_1 = \left(\frac{\partial v}{\partial y}\right)_1 = \frac{-25v_1 + 48v_2 - 36v_3 - 16v_4 - 3v_5}{12h} + o(h^4) , \tag{3.24a}$$

$$Z_{n_y} = \left(\frac{\partial v}{\partial y}\right)_{N_y} = \frac{3v_{N_y-4} - 16v_{N_y-3} + 36v_{N_y-2} - 48v_{N_y-1} + 25v_{N_y}}{12h} + o(h^4) . \tag{3.24b}$$

For the second stage of the finite element Numerov–Galerkin, we solve a tridiagonal system of the form

$$\begin{bmatrix} 4 & 1 & & & \\ 1 & 4 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 4 & 1 \\ & & & 1 & 4 \end{bmatrix}[w_j] = \frac{1}{2}\begin{bmatrix} v_{j-1}Z_{j-1} + v_jZ_{j-1} + v_{j-1}Z_j + \\ v_{j+1}Z_j + v_jZ_{j+1} + v_{j+1}Z_{j+1} + 6v_jZ_j \end{bmatrix} . \tag{3.25}$$

In the second stage, we interpolate the values of Z_0 and Z_{N_y+1} in a way similar to (3.24). A pentadiagonal and a cyclic pentadiagonal matrix solver (induced by the periodic boundary conditions) were developed following [38] and generalizing [39], respectively.

4. Substructuring and the Schur complement

It is well known in the finite element method that the internal degrees of freedom can be condensed out at the element level before the assembly process (see, for example, [40]). When this idea is applied to a group of elements, i.e., a substructure or a subdomain, it leads to what is known among engineers as the substructuring techniques.

The idea is that the whole structure or domain is considered to be an assembly of substructures or subdomains (see, for example, [2, 41, 42]). Each substructure or subdomain, in turn, is idealized as an assembly of finite elements, and all internal degrees of freedom are statically condensed out.

To fix ideas, two classes of variables are usually identified, namely the internal variables relevant to nodes within subdomains, and the interface variables relevant to nodes belonging to two or more subdomains. The internal variables may be numbered either before or after the interface ones.

Here we are particularly interested in the computational speedup resulting when using the domain decomposition technique. We will consider subdividing the domain Ω under consideration into n nonoverlapping parallel strips Ω_i, $i = 1, \ldots, n$, of equal or nearly equal sizes. The $n-1$ interfaces Γ_i, $i = 1, \ldots, n-1$, separating these n subdomains from each other are collectively denoted by Γ. We have the following relations:

$$\Omega = \Omega_1 \cup \Omega_2 \cup \cdots \cup \Omega_n \cup \Gamma , \tag{4.1}$$

$$\Omega_i \cap \Omega_j = \Phi , \quad \text{for } i \neq j , \tag{4.2}$$

$$\Gamma = \Gamma_1 \cup \Gamma_2 \cup \cdots \cup \Gamma_{n-1} . \tag{4.3}$$

The differential operator governing the problem on Ω can be split up into operators acting on the interfaces Γ and the n subdomains Ω_i, $i = 1, \ldots, n$, at each time step as can be realized by identifying two types of variables and renumbering. If we denote the matrix representations of these reduced operators as A_{ii}, $i = 1, \ldots, n$, on each of the subdomains and A_{ss} on the interfaces Γ, we obtain the following system of equations:

$$[A]\{x\} = \{f\} , \tag{4.4}$$

where A is an arrow-head matrix of the form

$$[A] = \begin{bmatrix} A_{dd} & A_{ds} \\ A_{sd} & A_{ss} \end{bmatrix} \tag{4.5a}$$

where

$$A_{dd} = \operatorname{diag}[A_{11}, A_{22}, \ldots, A_{nn}] \tag{4.5b}$$

is a block diagonal matrix. A_{ds} and A_{sd} are the block column and row vectors of the following forms:

$$A_{ds} = \{A_{1s}, A_{2s}, \ldots, A_{ns}\}^t, \tag{4.5c}$$

$$A_{sd} = \{A_{s1}, A_{s2}, \ldots, A_{sn}\}, \tag{4.5d}$$

and

$$\{x\} = \{x_1, x_2, \ldots, x_n, x_s\}^t, \tag{4.6}$$

$$\{f\} = \{f_1, f_2, \ldots, f_n, f_s\}^t. \tag{4.7}$$

Note that the matrix A consists essentially of the assembly of the subdomain stiffness matrices, see also [15]. If we let n_i, $i = 1, \ldots, n$, be the number of unknowns in each of the subdomains and n_s be the number of unknowns on the interfaces, then each of the matrices A_{ii}, A_{is} and A_{si} is of size $n_i \times n_i$, $n_i \times n_s$ and $n_s \times n_i$, respectively, for $i = 1, \ldots, n$. Likewise A_{ss} is of the size $n_s \times n_s$.

There is no direct coupling between any two different subdomains. Each of the matrices A_{ii}, $i = 1, \ldots, n$, has at most seven nonzero entries in each row for our seven-point stencil corresponding to the linear triangular finite element discretization. The coupling between the subdomains and interfaces is represented by the matrices A_{is} and A_{si}, $i = 1, \ldots, n$, which are very sparse matrices. Most rows of A_{is}, $i = 1, \ldots, n$, are zeroes (for a reasonably high mesh resolution) and there are at most two nonzero entries for any nonzero rows. The same arguments hold for the matrices A_{si}, $i = 1, \ldots, n$, if we refer to columns instead of rows.

The interfaces Γ include $n-1$ internal boundaries Γ_i, $i = 1, \ldots, n-1$, which satisfy (4.3), and no points from different internal boundaries are expected to appear in the same stencil. This accounts for the special structure of matrix A_{ss}. For instance, A_{ss} is diagonal for a three-point stencil 1-D problem; block diagonal

$$\text{diag}[B_{11}, B_{22}, \ldots, B_{n-1,n-1}], \tag{4.8}$$

with each of B_{ii}, $i = 1, \ldots, n-1$, being of the following form:

$$\begin{bmatrix} * & * & & & * \\ * & * & * & & \\ & \ddots & \ddots & \ddots & \\ & & * & * & * \\ * & & & * & * \end{bmatrix} \tag{4.9}$$

for a five-point finite difference stencil or a seven-point triangular finite element stencil discretization of the 2-D shallow-water equations problem presented in Section 2. The matrix structure in (4.8) will be used to explore the parallelism of the interface preconditioner presented in Section 7.

At each time step, we have to handle three systems of linear equations of the form (4.4) governing, respectively, the evolution of the geopotential field and two components of the velocity field distributions.

By using block Gaussian elimination, the system (4.4) can be reduced to the following systems of linear equations:

$$Cx_s = g \quad \text{on } \Gamma\,, \tag{4.10}$$

and for $i = 1, \ldots, n$

$$A_{ii}x_i = f_i - A_{is}x_s \quad \text{on } \Omega_i\,, \tag{4.11}$$

where

$$C = A_{ss} - \sum_{i=1}^{n} A_{si}A_{ii}^{-1}A_{is} \tag{4.12}$$

and

$$g = f_s - \sum_{i=1}^{n} A_{si}A_{ii}^{-1}f_i\,. \tag{4.13}$$

C is the so-called Schur complement, capacitance matrix or Gauss transform in different contexts (see [43–45]). It is clear that the subdomain problems (4.11) are essentially independent of each other and that the solutions can be sought in a highly parallel way once the interface Schur complement linear system (4.10) has been solved.

It is well known that although each of the matrices A_{ii}, $i = 1, \ldots, n$, and A_{ss} corresponds to a lower dimensional local differential operator and hence sparse, the Schur complement matrix C is generally not sparse. This fact is clearly seen in the finite element context by noticing that the Schur complement C consists essentially of the assembly matrix of the substructure stiffness matrices. Therefore the solutions of (4.10) are expensive to obtain using a direct solver, especially as the degrees of freedom on the interfaces increase when higher mesh resolutions are employed.

5. A node renumbering scheme

In a computer program designed for solving partial differential equations, we may introduce a modification of the code in order to accommodate various mesh resolutions aimed at obtaining higher orders of accuracy and testing the convergence of the method.

Let us denote the original nodal numbers (specific to each approximation method) as the old numbering, while the nodal numbers after renumbering (i.e. the substructuring numbering) will be denoted as the new numbering.

If we discretize the partial differential equation based on the new numbering scheme, the relations between the interface nodal numbers and those nodal numbers of the subdomains adjacent to the interfaces become difficult to predict for arbitrary high resolutions.

This becomes evident for the case of the finite element discretization in which the relationship amongst global nodes, local nodes and element numbering turns out to be very different if we try to formulate the problem using the new numbering systems for various mesh resolutions.

In view of this, other efficient ways for transforming the original system matrix into the arrow-head matrix like that in (4.5) need to be devised. For example, the following node renumbering scheme may be used for this purpose:

(1) Set up the relationship between the old and the new numbering system by defining a 1-D

array number(n), where n is the total number of nodes and the number(i) is the nodal number under the new numbering scheme corresponding to the old nodal number i.

(2) Renumber the nodes by putting the jth column of the original matrix into the jjth column of the transformed matrix, where jj = number(j).

(3) Change the positions of nodal actions by putting the ith row of the matrix obtained at step (2) into the iith row of the transformed matrix, where ii = number(i).

After these three steps have been implemented, we obtain the arrow-head matrix which has a similar structure to that of (4.5), corresponding exactly to the new nodal numbering defined by the array number(n). This idea also provides an easy way for implementing multicoloring techniques.

6. The CGS algorithm

As is well known, the conjugate gradient algorithm with suitable preconditioners is one of the best methods available today for the iterative solution of large sparse symmetric and positive definite linear systems. However, this algorithm is not applicable to the solution of nonsymmetric linear systems which may arise from the discretizations of nonself-adjoint elliptic partial differential equations.

A large number of generalizations were proposed in the literature (see [46, 47] and references therein) for solving large linear systems with nonsymmetric coefficient matrices. However, none of them may claim to be a clear winner. CGN (the conjugate gradient algorithm applied to the normal equation of the original nonsymmetric system), CGS (conjugate gradient squared) and GMRES (generalized minimal residual) seem to be the most often used algorithms. Recently, a fast squared Lanczos method for nonsymmetric linear systems was proposed in [48]. This new algorithm was found to be very fast and robust compared to GMRES.

Only limited numerical experience exists at present for most of these nonsymmetric iterative solvers, especially for real life applications. In their efforts to generate part of the production code for the modelling of weak plasma turbulence, Radicati et al. [49] studied the CGN, BCG (biconjugate gradient), CGS and GMRES algorithms as applied to the nonsymmetric time-dependent linear systems arising from the discretization of their problem. According to their report, the CGS and GMRES algorithms yield the best performances and are highly competitive with each other.

In this paper, we choose to apply CGS algorithm for the iterative solutions of linear systems both in the subdomains and on the interfaces and leave detailed comparisons between various nonsymmetric iterative solvers applied to our problem to a follow-up paper [35].

The CGS algorithm was derived from the BCG algorithm by squaring the residual and direction matrix polynomials. The BCG algorithm produces two sequences of residuals r_i and $\tilde{r}_i$ by simple relations similar to the CG algorithm in which $(r_i, \tilde{r}_j) = 0$, for $i \neq j$. It can be shown that $r_i = P_i(A)r_0$ and $\tilde{r}_i = P_i(A^t)\tilde{r}_0$ where $P_i(A)$ is a polynomial of degree i in the coefficient matrix A. The motivation for formulating CGS is through the observation that $\rho_i = (\tilde{r}_i, r_i)$ in BCG may be expressed as follows:

$$\rho_i = (\tilde{r}_i, r_i) = (P_i(A^t)\tilde{r}_0, P_i(A)r_0) = (\tilde{r}_0, P_i^2(A)r_0) \tag{6.1}$$

and a similar relation holds for the direction vectors p_i and $\tilde{p}_i$.

It is preconditioning that makes these iterative methods highly competitive. There are several possibilities for applying the preconditioning in the CGS algorithm. The PCGS algorithm used in our numerical experiments for solving $Ax = b$ is given in the following form ready for computer implementation:

$$r = b\,, \quad x = x_0\,, \qquad r = r - Ax\,.$$

Choose $\tilde{r}$ such that

$$\begin{aligned}
&(\tilde{r}, r) \neq 0\,, \qquad p = r\,, \qquad u = r\,,\\
&g = G^{-1}p\,, \qquad (*)\\
&p1 = Ag\,, \qquad \delta 0 = (\tilde{r}, p1)\,, \qquad \delta 1 = (\tilde{r}, r)\,, \qquad \alpha = \delta 1/\delta 0\,,\\
&q = u - \alpha p1\,, \qquad u = u + q\,, \qquad g = C^{-1}u\,, \qquad x = x + \alpha g\,.
\end{aligned} \tag{6.2}$$

If the convergence criterion is met then stop, otherwise continue

$$\begin{aligned}
&p1 = Ag\,, \qquad \delta 0 = (\tilde{r}, r)\,, \qquad r = r - \alpha p1\,, \qquad \delta 1 = (\tilde{r}, r)\,,\\
&\beta = \delta 1/\delta 0\,, \qquad u = r + \beta q\,, \qquad p = u + \beta(q + \beta p)\,, \qquad \text{Goto } (*)\,,
\end{aligned}$$

where the right-hand side b and the initial guess x_0 are input vectors and (,) denotes the usual Euclidean inner product. G is a preconditioning matrix. If $G = I$, then (6.2) reduces to a nonpreconditioned version of the CGS algorithm. For $G = A$, it is easy to show that (6.2) converges in one iteration. Between these two extremes, there are infinitely many choices of G.

From the algorithm (6.2), it is readily observed that seven one-dimensional arrays $\tilde{r}$, r, p, $p1$, q, u and x are required for implementing the nonpreconditioned version and an additional vector g is needed for the preconditioned version. By comparison, the conjugate gradient method requires the storage of only four vectors with or without preconditioning. The matrix A can be stored in a variety of ways due to its sparseness and the storage of the preconditioning matrix G is strongly case-dependent.

No theoretical convergence bound has yet been discovered for the CGS algorithm. However, the CGS algorithm is known to amplify the effects of the Lanczos method. In particular, we have

$$r_i^{\mathrm{CGS}} = P_i^2(A)r_0\,, \qquad r_i^{\mathrm{Lan}} = P_i(A)r_0 \tag{6.3}$$

for the residuals of the CGS and Lanczos method, respectively. If $P_i(A)$ defines a contraction, by (6.3) the expected convergence rate of CGS is thus roughly twice that of BCG, as was observed in practice [50].

7. The Schur complement matrix approach

Historically, it was common practice to apply the LU or Cholesky factorization in each subdomain and form the Schur complement matrix C and its right-hand side g. Thus the solution to (4.10) can be obtained, followed by that to each of the subsystems (4.11).

The coupling between subdomains is now handled more efficiently using preconditioned fast iterative solvers without constructing the Schur complement matrix explicitly. However, as has been pointed out in [51], the approach based on explicitly forming the Schur complement matrix still remains a useful procedure in some cases.

As formulated in [45], it requires $n(n_s + 1)$ forward and back substitutions to obtain C and g. A minor improvement can be made here to reduce this number to nn_s. The algorithm assumes the following form:

ALGORITHM 7.1

Step 1. Carry out the LU decomposition for each of the subdomain matrices $A_{ii} = L_i U_i$, $i = 1, \ldots, n$. This part is highly parallel.

Step 2. Solve $Y_{si} U_i = A_{si}$ and $X_{si} L_i = Y_{si}$ row by row for $i = 1, \ldots, n$. Form $C = A_{ss} - \Sigma_{i=1}^{n} X_{si} A_{is}$ and $g = f_s - \Sigma_{i=1}^{n} X_{si} f_i$. This part can also be calculated in parallel, where X_{si} and Y_{si} are two $n_s \times n_i$ matrices.

Step 3. Solve (4.10). This is a bottleneck for parallelism.

Step 4. Solve (4.11) in parallel by using the LU decompositions of A_{ii}, $i = 1, \ldots, n$.

The aforementioned algorithm requires the formation of the Schur complement matrix explicitly, a computationally expensive procedure for most problems. Like the Schur complement matrix approach formulated by the CG algorithm, the following algorithm may be formulated for any nonsymmetric solver in which only matrix–vector multiplications are required, although we only refer here to the CGS algorithm.

ALGORITHM 7.2

Step 1. Solve

$$A_{ii} b_i = f_i \tag{7.1a}$$

and form $A_{si} b_i$, $i = 1, 2, \ldots, n$, in parallel; form the right-hand side of the Schur complement matrix system

$$g = f_s - \sum_{i=1}^{n} A_{si} b_i \,. \tag{7.1b}$$

Step 2. Use the CGS algorithm to solve the Schur complement system (4.10). This is an iterative process carried out until a convergence criterion on the interfaces is met. The product of the Schur complement matrix with a vector w_s, Cw_s, may be evaluated as follows. Solve each subdomain problem

$$A_{ii} v_i = -A_{is} w_s \tag{7.2a}$$

once and form the product $A_{si}v_i$ for $i = 1, 2, \ldots, n$ in parallel; form the product

$$Cw_s = A_{ss}w_s + \sum_{i=1}^{n} A_{si}v_i \,. \tag{7.2b}$$

Step 3. Once the interface nodal values x_s are obtained, we may solve in parallel (4.11) for each subdomain.

Note that (7.1a) essentially requires us to solve the original set of shallow water equations (2.1), (2.2) and (2.3) on each subdomain with zero boundary conditions imposed on the interfaces Γ_i, $i = 1, \ldots, n-1$.

The above algorithm may be heuristically described as a divide-and-feedback process. The problem defined on the original domain is divided into smaller problems and solved in parallel within the subdomains. The information gathered from the solution of each of these subdomain problems is then fed back to the interface iterative solver. If the convergence criterion is not met on the interfaces, the domain is decomposed again and the subdomain problems solved. This divide-and-feedback process continues until the convergence criterion on the interfaces is fulfilled.

This divide-and-feedback process immediately indicates that the efficiency of this approach relies on the number of iterations required for obtaining convergence on the interfaces as well as on the computational cost for each subdomain solver. Preconditioning techniques may be used to reduce the number of iterations on the interfaces. In the subdomains, in the case where fast solvers do not exist, either direct solvers based on LU factorizations or iterative methods may be applied.

For the iterative subdomain solver, preconditioning based on either ILU [52] or MILU [53–55] factorizations may be applied. The MILU factorization preconditioning was proposed in order that the spectral condition number of the matrix $G^{-1}A$ satisfies the following asymptotic relation:

$$\kappa(G^{-1}A) \sim O(h^{-1}) \,, \quad \text{as } h \to 0 \,, \tag{7.3}$$

where h is the mesh size, instead of $O(h^{-2})$ as for ILU. Here $G = LU = A + D + R$, R is the so-called defect or error matrix whose rowsum is zero for each row of R. D is a diagonal matrix containing some preconditioning parameters.

Specifically, for a class of M matrices or a weakly diagonally dominant symmetric L matrices, it was shown that the MILU factorization could be constructed such that (7.3) holds. For some other more general coefficient matrices, as pointed out in [53], the same rate of convergence was observed.

It is readily observed that, at each time step and for each of the geopotential or velocity capacitance matrices, the incomplete factorization needs to be carried out only once for each subdomain during the entire divide-and-feedback process. The work required for the preconditioning of each subdomain consists mainly of the forward and back substitutions which may be computed in parallel.

There is no definite answer as to which subdomain solver is better, direct or iterative. The relative efficiency of these two approaches is problem-dependent. For our application,

numerical results indicate that CGS iterative subdomain solvers preconditioned by MILU factorizations are more efficient than direct solvers based on LU factorizations in the subdomains for higher mesh resolutions (see Section 9).

For the interface, a class of relatively robust preconditioners is the so-called boundary probe preconditioners (also called modified Schur complement preconditioners in [51]) which were first proposed in [30] and discussed in some detail in [56]. The basic idea behind this type of preconditioners is based on the empirical observation that the Schur complement matrix is close to a tridiagonal matrix for many elliptic operators. Thus it is reasonable to construct a tridiagonal or even a $2k+1$, for $k>1$, diagonal approximation to the Schur complement matrix C and use it as the preconditioning matrix for the interfaces.

A $2k+1$, for $k=0,1,\ldots,$ diagonal approximation to C may be constructed by evaluating the products of the Schur complement matrix C with $2k+1$ 'probing vectors' v_j, $j=1,\ldots,2k+1$. This would require $2k+1$ solves in each of the subdomains.

The success and efficiency of this class of preconditioners obviously depend on whether or not the elements in the Schur complement matrix die off rapidly enough away from the main diagonal. To show that our problem possesses this nice property, we produce in Fig. 5 (see Section 9) the surface plot of the elements in a typical geopotential Schur complement matrix as a function of its indices at the end of 1 h. The structures of the Schur complement matrices corresponding to velocity distributions are quite similar.

The special structure of the Schur complement matrix immediately suggests that we use the tridiagonal part of the Schur complement matrix as the preconditioning matrix. This tridiagonal part may be obtained efficiently by evaluating three matrix-vector products of C and v_i, for $i=1,2,3$, where

$$v_1 = \{1,0,0,1,0,0,\ldots\}^t, \tag{7.4a}$$

$$v_2 = \{0,1,0,0,1,0,\ldots\}^t, \tag{7.4b}$$

$$v_3 = \{0,0,1,0,0,1,\ldots\}^t. \tag{7.4c}$$

It may be easily verified that the elements of the tridiagonal part of the Schur complement matrix C are all contained in Cv_i, for $i=1,2,3$. Three solves are required in each subdomain for this construction.

However reasonable this approach may seem, it generally requires four or five iterations before the convergence criterion (the l_2 norm less than 10^{-10}) on the interfaces is met. It turns out that a more efficient interface preconditioner may be constructed for our case (two or three iterations) by retaining the block structure of A_{ss} (see (4.8) and (4.9)) and replace each entry in the main diagonal of A_{ss} by the corresponding row-sum of the Schur complement C.

Let us denote the preconditioning matrix constructed this way by

$$G = \mathrm{diag}[\tilde{B}_{11}, \tilde{B}_{22}, \ldots, \tilde{B}_{n-1,n-1}]. \tag{7.5}$$

The preconditioning matrix K will then be the same as A_{ss} except that the main diagonal is determined by the following:

$$\mathrm{diag}(G) = \mathrm{diag}(A_{ss}) + u, \tag{7.6}$$

where

$$u = -\sum_{i=1}^{n} A_{si} A_{ii}^{-1} A_{is} v \tag{7.7}$$

and $v = \{1, 1, 1, 1, 1, 1, \ldots\}^t$. Now only one solve is required in each subdomain for this construction.

The preconditioning linear system $Gg = p$ (see (6.2)) may be solved in parallel,

$$\tilde{B}_{ii} g_i = p_i \,, \tag{7.8}$$

for $i = 1, 2, \ldots, n-1$, where the definitions of g_i and p_i are obvious.

Various algorithms may be generated from Algorithm 7.2 depending on the types of subdomain solvers and interface preconditionings. To facilitate the comparison of various numerical experiments to be reported in Section 9, we use the following definitions:

(1) SCMA I, MILU-PCG in the subdomains and no interface preconditioning;
(2) SCMA II, MILU-PCG in the subdomains and interface preconditioning by the tridiagonal part of the Schur complement matrix;
(3) SCMA III, MILU-PCG in the subdomains and interface preconditioning based on (7.5)–(7.7);
(4) SCMA IV, direct subdomain solvers based on (complete) LU factorizations in the subdomains and interface preconditioning based on (7.5)–(7.7).

8. The modified interface matrix approach

For the Schur complement matrix approach, the subdomain problems (4.11) are solved only after the solution x_s on the interfaces is obtained. We propose, in this section, a new approach to handle the coupling between the different subdomains. It differs from the Schur complement matrix approach in that the approximations to the solutions on the interfaces and in the subdomains are successively improved.

8.1. The basic theory

Consider the following iterative procedure for solving (4.4): for $k = 0, 1, 2, \ldots,$

$$A_{ii} x_i^{(k+1)} = f_i - A_{is} x_s^{(k)} \,, \quad \text{for } i = 1, 2, \ldots, n \,, \tag{8.1a}$$

$$A_{ss} x_s^{(k+1)} = f_s - \sum_{i=1}^{n} A_{si} x_i^{(k+1)} \,. \tag{8.1b}$$

The iteration starts with an arbitrary initial guess $x_s^{(0)}$ on the interfaces $\Gamma = \Gamma_1 \cup \Gamma_2 \cup \cdots \cup \Gamma_n$, solves the subdomain problems (8.1a) in parallel, then updates the approximation on the interfaces by solving (8.1b). In this way, we solve the subdomain problems and the interface problem successively until the convergence criterion on the interfaces is met.

Let I_{ss} be an identity matrix of size $n_s \times n_s$, x_i^*, for $i = 1, 2, \ldots, n$, and x_s^* be the solutions for (4.11) and (4.10). We prove here the following result, where C is the Schur complement matrix defined by (4.12).

THEOREM 8.1. Sequences of approximation $x_i^{(k)}$, *for* $i=1,2,\ldots,n$, *and* $x_s^{(k)}$ *produced by (8.1) converge to* x_i^*, *for* $i=1,2,\ldots,n$, *and* x_s^* *for an arbitrary initial guess* $x_s^{(0)}$ *on the interfaces if and only if the spectral radius of the matrix* $I_{ss}-A_{ss}^{-1}C$ *satisfies*

$$\rho(I_{ss}-A_{ss}^{-1}C)<1\,, \tag{8.2}$$

PROOF. Define an error vector $e_s^{(k)}=x_s^{(k)}-x_s^*$ for the interface approximation. Since $x_i^{(k+1)}=A_{ii}^{-1}(f_i-A_{is}x_s^{(k)})$, for $i=1,2,\ldots,n$, it is straightforward to show that $A_{ss}x_s^{(k+1)}=g+(A_{ss}-C)x_s^{(k)}$, where g is defined by (4.13). It follows that $e_s^{(k+1)}=(I_{ss}-A_{ss}^{-1}C)e_s^{(k)}$. Hence we have $x_s^{(k)}\to x_s^*$, as $k\to\infty$ if and only if (8.2) holds. Now let $k\to\infty$ in (8.1a), it follows from (4.11) that $x_i^{(k)}\to x_i^*$, as $k\to\infty$, for $i=1,2,\ldots,n$. □

Compared with Algorithm 7.2, the algorithm based on (8.1) is quite straightforward and the condition (8.2) is satisfied in our case for various mesh resolutions. Specifically, our numerical experiments (see Section 9) indicate that for various mesh resolutions of the discretized domain, the spectral radii $\rho(I_{ss}-A_{ss}^{-1}C)$ are about 0.13. Hence the asymptotic rate of convergence $-\ln\rho$, (see, for example, [57]) is about 2.0402. Thus, in order to reduce the norm of the initial error vector on the interfaces by a factor of, say, 10^{-6}, roughly seven iterations will be required.

We now proceed to modify this iterative procedure. Since each iteration in (8.1) requires the solutions of all the subdomain problems once, it is important to reduce the number of iterations. It is well known that the smaller the spectral radius, the faster the convergence. In order to reduce the spectral radius, we construct a matrix K_{ss} such that $A_{ss}+K_{ss}$ is a good approximation of the Schur complement matrix C in the sense that the spectral radius $\rho[(A_{ss}+K_{ss})^{-1}C]$ is close to 1. The matrix $A_{ss}+K_{ss}$ is referred to as the modified interface matrix.

Now assume that the matrix K_{ss} has been chosen; the following modified iterative procedure can then be proposed: for $k=0,1,2,\ldots,$

$$A_{ii}x_i^{(k+1)}=f_i-A_{is}x_s^{(k)}\,,\quad \text{for } i=1,2,\ldots,n\,, \tag{8.3a}$$

$$(A_{ss}+K_{ss})\,\Delta x_s^{(k)}=f_s-A_{ss}x_s^{(k)}-\sum_{i=1}^{n}A_{si}x_i^{(k+1)}\,, \tag{8.3b}$$

where $\Delta x_s^{(k)}=x_s^{(k+1)}-x_s^{(k)}$. The iteration starts with an arbitrary initial guess $x_s^{(0)}$ on the interfaces and we successively solve the subdomain problems (8.3a) in parallel and problem (8.3b) on the interfaces until the norm given by (8.5) is small enough.

The following result may be proved in a similar way.

THEOREM 8.2. Sequences of approximation $x_i^{(k)}$, *for* $i=1,2,\ldots,n$, *and* $x_s^{(k)}$ *obtained by (8.3) converge to* x_i^*, *for* $i=1,2,\ldots,n$, *and* x_s^* *for an arbitrary initial guess* $x_s^{(0)}$ *on the interfaces if and only if the spectral radius of the matrix* $I_{ss}-(A_{ss}+K_{ss})^{-1}C$ *satisfies*

$$\rho(I_{ss}-(A_{ss}+K_{ss})^{-1}C)<1\,. \tag{8.4}$$

It is interesting to make an analogy here and think of each of the equations in (8.3a) as the

governing equation for the displacement distribution within a substructure subjected to a load f_i originally acting upon it and the boundary interaction forces $-A_{is}x_s^{(k)}$ due to interface connections between the substructures.

By taking $x_s^{(0)}$ to be some initial guess on the interfaces that differs from the true solution x_s^*, we are actually imposing some constraints in addition to the original constraints (the boundary conditions) to the structure and thus making it stiffer. However, as guaranteed by Theorem 8.2, the extra constraints introduced due to the incorrect initial guess will be continually relaxed by solving (8.3b) repeatedly.

In the sequel, our discussion will be related to the CGS nonsymmetric iterative method for each of the linear systems (8.3a) in the subdomains, preconditioned by the MILU factorization and for the system (8.3b) on the interfaces with a preconditioner similar to that given by (7.5)–(7.7) (see Section 8.3).

The stopping criterion for (8.3) may be based on the norm of the vector $(A_{ss} + K_{ss})\,\Delta x_s^{(k)}$, since the following relation can be readily verified:

$$\|(A_{ss} + K_{ss})\,\Delta x_s^{(k)}\| = \|g - Cx_s^{(k)}\|\,, \tag{8.5}$$

where C is the Schur complement matrix and g is the corresponding right-hand side. If this norm is small enough, we conclude that system (4.10) on the interfaces is approximately well solved.

8.2. *The construction of K_{ss}*

We now investigate the construction of the matrix K_{ss} such that the modified interface matrix $A_{ss} + K_{ss}$ constitutes a good approximation of the Schur complement matrix C. Since we are mainly interested in nonsymmetric iterative methods, for which only matrix–vector products are required, for the solution of the system (8.3b), it is more efficient to consider the algorithm of computing $k_{ss}w_s$ for a given vector w_s rather than first forming the matrix and then the product.

As a first approach, let us consider a splitting of each of the subdomain matrices A_{ii},

$$A_{ii} = P_{ii} - Q_{ii} = P_{ii}(I_{ii} - R_{ii})\,, \tag{8.6}$$

where P_{ii} is a nonsingular matrix and I_{ii} is the identity matrix of size $n_i \times n_i$. Assuming that the spectral radius of $R_{ii} = I_{ii} - P_{ii}^{-1}A_{ii}$ is less than 1, we may obtain the following Neumann series expansion:

$$A_{ii}^{-1} = \left[\sum_{k=0}^{\infty} (I_{ii} - P_{ii}^{-1}A_{ii})^k\right] P_{ii}^{-1}\,. \tag{8.7}$$

$C - A_{ss}$ can be approximated by using only a finite truncated expansion, i.e.,

$$C - A_{ss} \approx K_{ss} = -\sum_{i=1}^{n} A_{si}X_{is}^{(m+1)}\,, \tag{8.8a}$$

where

$$X_{is}^{(m+1)} = \left[\sum_{k=0}^{m} (I_{ii} - P_{ii}^{-1} A_{ii})^k \right] P_{ii}^{-1} A_{is} . \tag{8.8b}$$

The following algorithm can be shown to be valid for computing $K_{ss} w_s$:

$$K_{ss} w_s = - \sum_{i=1}^{n} A_{si} v_i^{(m+1)} , \tag{8.9a}$$

where the vector $v_i^{(m+1)}$ can be obtained by the following iterative procedure: for $k = 0, 1, \ldots, m$, starting with $v_i^{(0)} = 0$,

$$P_{ii}(v_i^{(k+1)} - v_i^k) = -A_{ii} v_i^{(k)} + u_i , \tag{8.9b}$$

for $i = 1, 2, \ldots, n$ and $u_i = A_{is} w_s$. Notice that (8.9b) can be implemented and $A_{si} v_i^{(m+1)}$ formed, for $i = 1, \ldots, n$, completely in parallel.

Another approach is to use MILU preconditioners in the subdomains $A_{ii} \approx L_i U_i$ to construct the matrix K_{ss}. Specifically, we take

$$K_{ss} = - \sum_{i=1}^{n} A_{si} (L_i U_i)^{-1} A_{is} . \tag{8.10}$$

Here the matrix–vector product may be evaluated as follows:

$$K_{ss} w_s = - \sum_{i=1}^{n} A_{si} v_i , \tag{8.11a}$$

where v_i can be determined by

$$L_i \hat{v}_i = A_{is} w_s \tag{8.11b}$$

and

$$U_i v_i = \hat{v}_i , \tag{8.11c}$$

for $i = 1, 2, \ldots, n$. The solutions of (8.11b) and (8.11c) can be carried out and $A_{si} v_i$ formed in parallel.

Numerical results indicate that the spectral radii $\rho(I_{ss} - (A_{ss} + K_{ss})^{-1} C)$ for various resolutions are around 0.04 by retaining just the first term in the Neumann series and 0.002 by using the MILU factorization. Thus, in order to reduce the norm of the initial error vector on the interfaces by a factor, say, 10^{-6}, roughly only four and two iterations, respectively, will be required.

8.3. The algorithm

Prior to presenting an algorithm of the modified interface matrix approach, we discuss here how to choose initial guesses for the solution of (8.3a) as well as that of (8.3b) and how to precondition (8.3b).

Instead of making a guess of the initial vector for the PCGS solver to start with, we take $x_i^{(k)}$, for $i = 1, 2, \ldots, n$, as the initial guess for the solution of each of the linear systems in

(8.3a), for $k = 1, 2, \ldots$, as well as a zero vector as the initial guess for the modified interface matrix linear system (8.3b), for $k = 0, 1, \ldots$.

It is clear from Theorem 8.2 that, as k increases, $x_i^{(k+1)} - x_i^{(k)} \to 0$, for $i = 1, 2, \ldots, n$, and $\Delta x_s^{(k)} \to 0$ on the interfaces. Consequently, the initial vectors selected this way become better approximations, as the iterative procedure (8.3) proceeds, to the subdomain solutions $x_i^{(k+1)}$, for $i = 1, 2, \ldots, n$, as well as to the solution on the interfaces $\Delta x_s^{(k)}$, and thus fewer iterations are required for the the iterative solutions in the subdomains (8.3a) and on the interfaces (8.3b). As a result, the computational cost of the modified interface matrix approach (8.3) decreases as k increases. The reduced cost for the subdomains mitigates the disadvantage that no fast subdomain solvers are available.

Here we use, for any fixed k, essentially the same preconditioner as that expressed by (7.5) and (7.6) to accelerate the convergence of the iterative solution of (8.3b), with u determined by $u = K_{ss}v$, where $v = \{1, 1, 1, 1, 1, 1, \ldots\}^t$. The product $K_{ss}v$ is evaluated by using either (8.9) or (8.11). If (8.11) is used, it is easy to see that just one inexact solve is required for the construction of this preconditioner in each subdomain.

Based on the above discussion, we present here an algorithm for the modified interface matrix approach:

Algorithm MIMA: (*modified interface matrix approach*)

Step 1. Carry out the MILU factorizations of A_{ii}, $A_{ii} \approx L_i U_i$, for $i = 1, 2, \ldots, n$, in parallel.

Step 2. Construct the preconditioner for (8.3b) of the form (7.3) and (7.4) with $u = K_{ss}v$. A large part of this calculation may be carried out in parallel by using either (8.9) or (8.11).

Step 3. Set $k = 0$. Specify $x_s^{(0)}$ on the interfaces and solve subdomain problems (8.3a) in parallel by using the MILU PCGS solver with a suitable initial guess.

Step 4. Solve (8.3b) by the PCGS solver with the preconditioner constructed in Step 2, and an initial guess taken to be the zero vector on the interfaces. The preconditioning system may be solved in parallel (see (7.8)), and the matrix–vector product $K_{ss}w_s$ may be computed by either (8.9) or (8.11) mostly in parallel, where w_s varies between iterations.

Step 5. Test for convergence on the interfaces (see (8.5)). If the convergence criterion is met, go to Step 6 and quit; Otherwise, go to Step 6 and then go back to Step 4.

Step 6. Set $k \leftarrow k + 1$. Solve subdomain problems (8.3a) in parallel by using the MILU PCGS solver with initial guesses $x_i^{(k)}$, for $i = 1, 2, \ldots, n$.

For the sake of numerical comparisons carried out in Section 9, Algorithm MIMA with $K_{ss}w_s$ computed by either (8.9) or (8.11) will be referred to as MIMA I or MIMA II, respectively.

9. Results

9.1. Numerical experiments and results

We carried out numerical experiments on the Cray Y-MP/432 which has a machine accuracy around 0.8×10^{-14} for a single precision. For our numerical experiments, we used the

following constants:

$$L = 6000\,\text{km}\,, \qquad D = 4400\,\text{km}\,, \tag{9.1a}$$

$$g = 10\,\text{m/s}\,, \qquad H_0 = 2000\,\text{m}\,, \tag{9.1b}$$

$$H_1 = 220\,\text{m}\,, \qquad H_2 = 133\,\text{m}\,, \tag{9.1c}$$

$$\hat{f} = 10^{-4}\,\text{s}^{-1}\,, \qquad \beta = 1.5 \times 10^{-11}\,\text{s}^{-1}\,\text{m}^{-1}\,. \tag{9.1d}$$

The geopotential field distribution in the present problem has the order of magnitude $10^4\ \text{m}^2/\text{s}^2$. The preselected reference geopotential has been chosen to be $\varphi_0 = 10^4\ \text{m}^2/\text{s}^2$ so that the nondimensional geopotential φ' as defined by (2.11) has the order of magnitude of O(1). Under this choice of reference geopotential, the dimensionless constants corresponding to (9.1) are

$$L' = 1\,, \qquad D' = 0.733333\,, \tag{9.2a}$$

$$g' = 6000\,, \qquad H_0' = 0.333333 \times 10^{-3}\,, \tag{9.2b}$$

$$H_1' = 0.366666 \times 10^{-4}\,, \qquad H_2' = 0.221666 \times 10^{-4}\,, \tag{9.2c}$$

$$\hat{f}' = 6\,, \qquad \beta' = 5.4\,. \tag{9.2d}$$

The two stage Numerov–Galerkin finite element discretization scheme has been employed in space to solve the set of nonlinear shallow water equations given in (2.1), (2.2) and (2.3) over a limited nondimensionalized rectangular domain $0 \leq x \leq L'$, $0 \leq y \leq D'$ and $t' > 0$. For simplicity, linear triangular elements have been used with a seven-point stencil structure at any point within the integration domain.

In order to allow for further flexibility of the code, a modification has been introduced which allows, by changing just two parameters corresponding to the number of grid points in the x and y directions, respectively, the introduction of arbitrary mesh resolutions of the model. The automatic transformation of the global finite element matrices to the arrow-head matrices has been rendered possible by the node renumbering scheme presented in Section 5.

In the actual implementation, the matrices are seldom stored in full due to their sparseness property. Hence, the general node renumbering scheme in Section 5 has been adapted for transforming the information corresponding to compact matrices. Here we essentially transform the arrays which record the nonzero elements of the sparse matrices.

The initial non-dimensional geopotential distribution for the 2-D shallow water equations under consideration is given by (2.9) and (2.10) and its contour lines as well as a 3-D view are presented in Figs. 1 and 2, respectively. The whole domain is divided into four subdomains of equal sizes with three internal boundaries which collectively constitute the interfaces of the domain as are shown by three dotted lines in Fig. 3.

We present, in Figs. 3 and 4, a typical finite element domain decomposition solution for the nondimensionalized geopotential distribution of the shallow water equations at the end of 5 h of numerical integration. A mesh resolution 45 by 39 was used to produce the solution. There

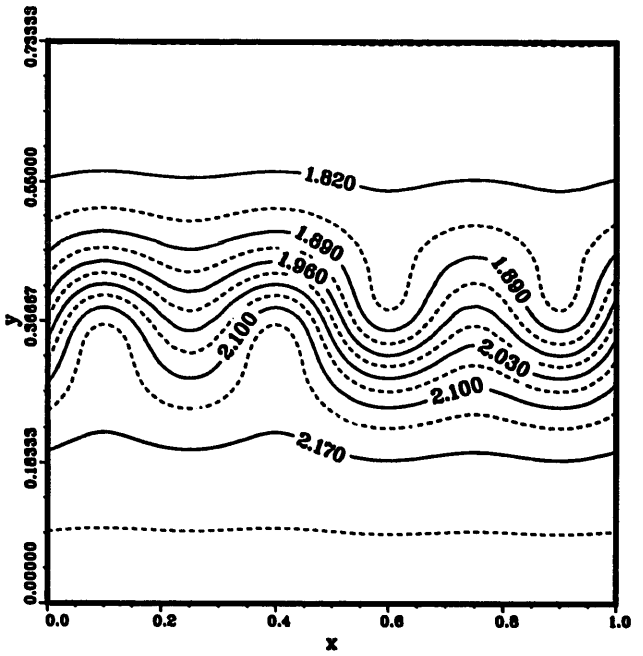

Fig. 1. Contour lines of the initial nondimensionalized geopotential field for the test problem run on the finite element shallow water equations model.

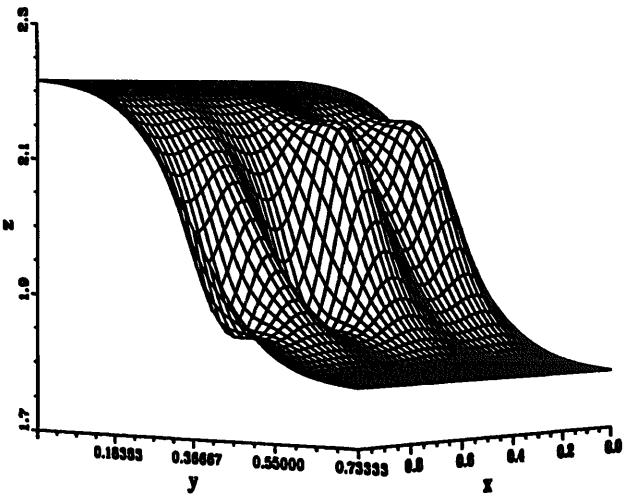

Fig. 2. A 3-D view of the initial nondimensionalized geopotential field using test problem as in Fig. 1.

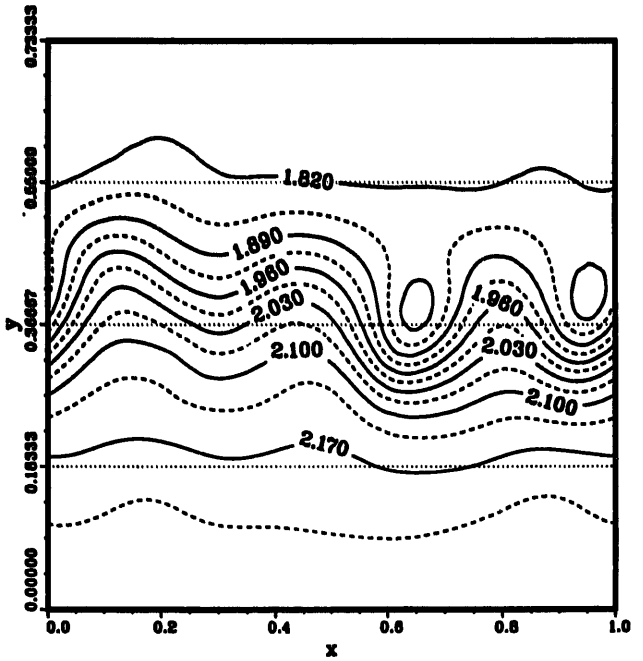

Fig. 3. Contour lines of the nondimensionalized geopotential field for the test problem with four subdomains domain decomposition after 5 h of numerical integration. The dotted lines illustrate the interfaces between subdomains.

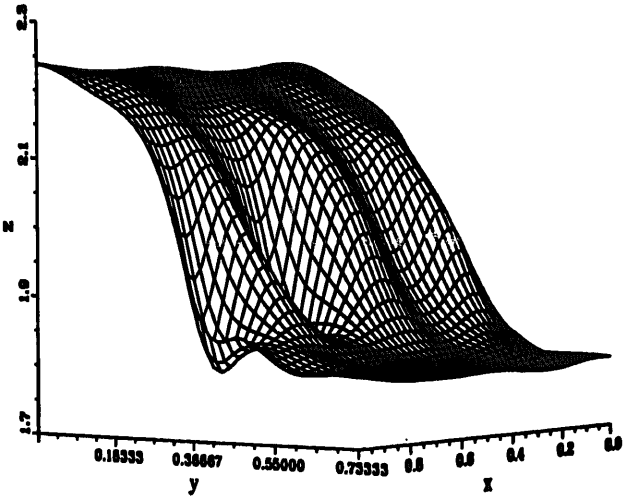

Fig. 4. A 3-D view of the nondimensionalized geopotential field for the test problem after 5 h of numerical integration using the finite element model with four subdomains domain decomposition.

are correspondingly 405 nodes in each of the four subdomains and 135 nodes on the interfaces. This run requires a total of 1755 nodes and 3420 elements in the whole domain. The time step is 0.5 h or 0.03 in nondimensionalized time.

It should be pointed out that although all of the algorithms presented in this paper may be applied to the finite element domain decomposition solution of the shallow water equations and yield similar numerical results, the respective computational costs corresponding to the implementation of different algorithms are vastly different.

In the sequel, we will compare the relative efficiencies of various algorithms. For convenience, the Schur complement matrix approach and the modified interface matrix approach will be abbreviated to SCMA and MIMA. The 2-norm is used throughout and the stopping criterion for the iterative algorithm is based on the 2-norm of the final residual vector being less than 10^{-10} unless indicated otherwise.

To justify our choosing to apply the boundary probe preconditioners to the iterative solution of the Schur complement matrix, we present in Fig. 5 the surface generated by the entries in the nondimensionalized geopotential Schur complement matrix as a function of its indices at the end of 1 h. The mesh resolution used for this calculation is 31 by 31 grid points. However, similar diagonally dominant structures were observed for other mesh resolutions.

It was also found that structures of the Schur complement matrices governing the velocity distributions on the interfaces and corresponding to different mesh resolutions are all similar to the tridiagonal structure displayed in Fig. 5. This typical structure of the Schur complement matrix was first observed in [14] for the case of the Laplace operator.

We tested the SCMA based on algorithm (6.2) corresponding to various mesh resolutions using SCMA II and SCMA III algorithms (see the end of Section 7). It was found that the SCMA III algorithm produced invariably better results in terms of both the number of iterations for convergence and the corresponding CPU time.

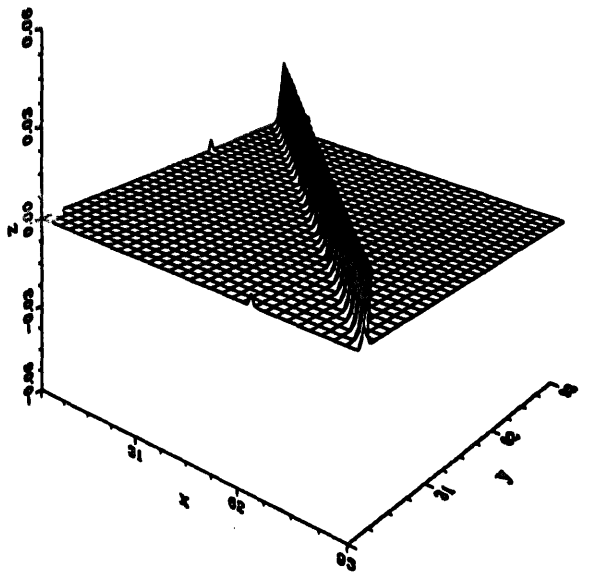

Fig. 5. The surface generated by the nondimensionalized geopotential Schur complement matrix at the end of 1 h. The mesh resolution is 31 by 31 for the original domain. For this choice, there are 93 nodes on the interfaces and 217 nodes in each of the four subdomains.

We present, in Fig. 6, the $\log_{10}$ norms of the residual vectors on the interfaces versus the number of iterations corresponding to each of these two cases. A mesh resolution of 51 by 51 was used for this calculation. For comparison purposes, the case corresponding to the SCMA I algorithm is also included. Both the number of iterations and the CPU time required for each case are recorded in Table 1 for the solution of the nondimensional geopotential Schur complement linear system at the end of 1 h.

Before confirming numerically that the MILU PCGS subdomain solvers are computationally more efficient than the LU direct subdomain solvers in our case, for higher resolutions we display the effect of the MILU preconditioner on the subdomain solution. A typical convergence behavior is plotted in Fig. 7, where the mesh resolution is 51 by 51 and the stopping criterion has been chosen to be that the 2-norm of the final residual vector is less that 10^{-15}. The results, displaying the numbers of iterations and the CPU time required to attain convergence, are presented in Table 2 for the solution of a typical subdomain.

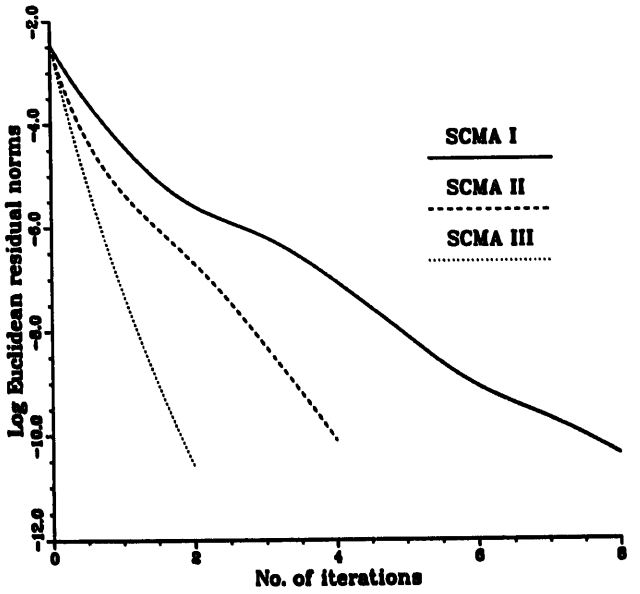

Fig. 6. The $\log_{10}$ residual norms for the Schur complement matrix linear system on the interfaces for the non-dimensionalized geopotential matrix system at the end of 1 h. The mesh resolution is 51 by 51 for the original domain. For this choice, there are 153 nodes on the interfaces and 612 nodes in each of the four subdomains.

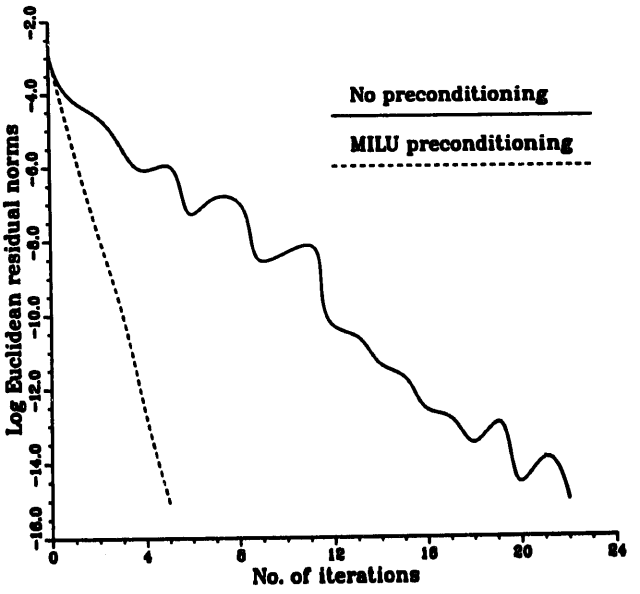

Fig. 7. The $\log_{10}$ residual norms in the subdomain for the nondimensionalized geopotential matrix system at the end of 1 h with and without preconditioning. The mesh resolution is 51 by 51 for the original domain and for this choice there are 612 nodes in each of the four subdomains.

Table 1
A comparison between interface preconditioners

SCMA I	SCMA II	SCMA III
8 iterations	4 iterations	2 iterations
4.08 s	2.12 s	1.02 s

Table 2
MILU precondition in a typical subdomain

No preconditioning	MILU preconditioning
22 iterations	5 iterations
0.2207 s	0.1480 s

In order to compare the relative computational costs of the SCMA III and SCMA IV algorithms, we computed the numerical solutions at the end of 1 h with a time step of 0.5 h various mesh resolutions for both the interfaces and the subdomains. We recorded the CPU time required to obtain the solution at the end of 1 h. See Figs. 8–10 for the graphical presentations of the results.

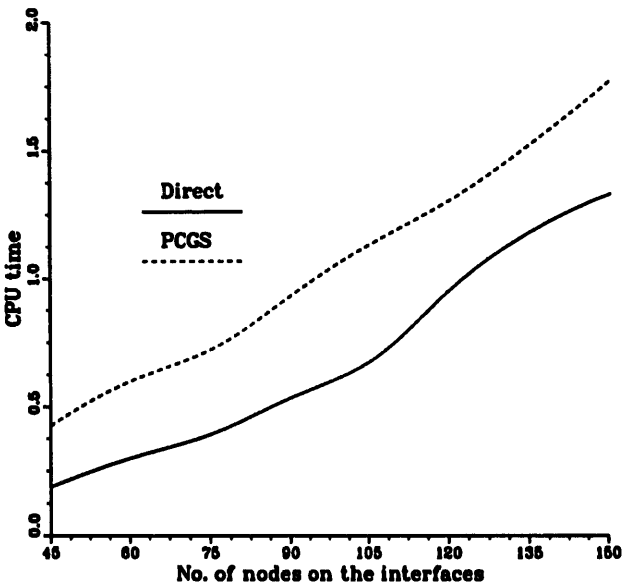

Fig. 8. A comparison of the LU direct and PCGS iterative subdomain solvers in terms of the CPU time. For this comparison, the number of each subdomain nodes is equal to the number of interface nodes.

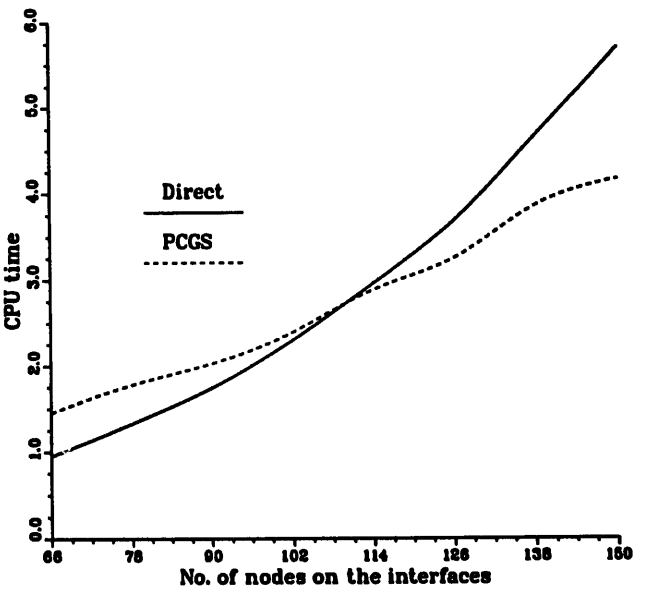

Fig. 9. A comparison of the LU direct and PCGS iterative subdomain solvers in terms of CPU time. For this comparison, the number of each subdomain nodes is equal to twice the number of interface nodes.

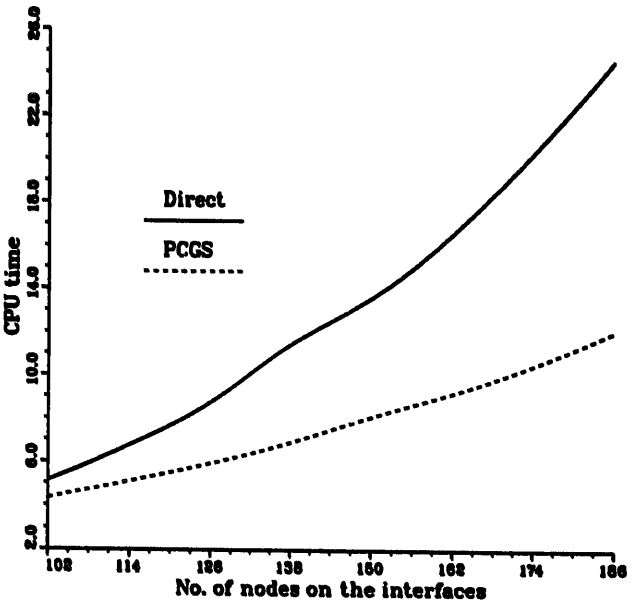

Fig. 10. A comparison of the LU direct and PCGS iterative subdomain solvers in terms of CPU time. For this comparison, the number of each subdomain nodes is equal to three times the number of interface nodes.

The computational results indicate that the SCMA IV algorithm is less expensive than the SCMA III algorithm when the mesh resolutions are relatively coarse and when the ratio between the size of each subdomain problem and that of the interface problem is less than two. However, as the mesh resolutions increase and the degrees of freedom on each subdomain get larger compared to those on the interfaces, the SCMA III algorithm turns out to be better. The results obtained here agree in some sense with those presented in [49], in which the Gauss direct solver was compared with the CGS solver.

We point out that, in most practical problems, the subdomain problems are much larger than the interface one. As a result the preconditioned iterative solution in each of the subdomains is to be preferred for our case, especially for large-scale problems.

We now turn our attention to the theory discussed in Section 8. The critical factor that affects the successful use of the theory and algorithms presented here is the spectral radii of the matrices $(I_{ss} - A_{ss}^{-1}C)$ and $(I_{ss} - (A_{ss} + K_{ss})^{-1}C)$, respectively. Our numerical experiments applied to various mesh resolutions indicate that condition (8.2) is satisfied and that the effect of modifying A_{ss} to $A_{ss} + K_{ss}$ is computationally significant.

We now present numerical results obtained using two mesh resolutions, one being 21 by 19, the other being 41 by 39. For these two cases, we obtained that $\rho(I_{ss} - A_{ss}^{-1}C) = 0.1349$ and 0.1340, respectively. For the construction of the matrix K_{ss}, both the MILU factorizations and Neumann series expansions in the subdomains are used. In the case of Neumann series expansion, P_{ii}, for $i = 1, 2, \ldots, n$, has been chosen to be the diagonal part of the matrix A_{ii}, see (8.6). The results for $(\rho(I_{ss} - (A_{ss} + K_{ss})^{-1}C)$ are summarized in Table 3, where the second row in the table corresponds to the spectral radii for the lower resolution, while the third row corresponds to the higher resolution.

We notice that the matrix K_{ss} constructed by MILU factorization (see (8.10)) in the subdomains yields the smallest spectral radius and therefore a faster convergence rate of the iterative procedure defined by (8.3) is to be expected. For $m = 3$, i.e. by taking four terms in the Neumann series expansion, the spectral radii for the lower and higher resolutions are, respectively, 1.508×10^{-2} and 1.199×10^{-2}. They are still not as small as those obtained using MILU factorizations. Moreover, the CPU time consumed by taking four or more terms in the Neumann series generally outweighs the gain obtained by the reduction of the number of iterations.

The modified interface matrix $A_{ss} + K_{ss}$ constructed according to (8.10) constitutes quite a good approximation to the Schur complement matrix. A plot of the surface formed by the elements in the modified interface matrix as a function of its indices can hardly be distinguished from Fig. 5. Instead, we present in Fig. 11 the surface generated by the entries in the matrix $C - (A_{ss} + K_{ss})$ corresponding to the nondimensional geopotential matrix at the end of 1 h. Similar structures are observed for other mesh resolutions.

Table 3
A comparison between spectral radii by using MILU factorization and Neumann series expansion

MILU	$m = 0$	$m = 1$	$m = 2$
2.127×10^{-3}	4.138×10^{-2}	2.750×10^{-2}	1.867×10^{-2}
1.952×10^{-3}	3.922×10^{-2}	2.571×10^{-2}	1.662×10^{-2}

To compare the performance behavior of the MIMA I and II algorithms, we present in Fig. 12, the $\log_{10}$ norms of the residual vectors (see (8.5)) on the interfaces for various cases, where the IMA (interface matrix approach), the approach based on (8.1), is also included for comparison. We see that the modified interface matrix formed by the MILU factorization in each subdomain proves to yield the best results. This is to be expected since $\rho(I_{ss} - (A_{ss} + K_{ss})^{-1}C)$ is of order 10^{-3} (see Table 3).

The number of iterations and the CPU time required for the solution of the geopotential distribution at the end of 1 h corresponding to each one of these cases are summarized in Table 4, where IMA is the case when $K_{ss} = 0$; $m = 0, 1, \ldots, 4$ correspond to K_{ss} constructed according to (8.8) by retaining one, two, ..., five terms in the Neumann series expansion, respectively.

To see how the initial guesses in the subdomains and on the interfaces improve as the iteration count k in algorithm MIMA increases, we carried out numerical experiments based on the algorithm MIMA II and kept a record of the initial residual norms for each of the four subdomain problems and the modified interface matrix linear system as functions of the number of iterations k (see Fig. 13 for details).

Table 4
A comparison between algorithms MIMA I and II

IMA	$m=0$	$m=1$	$m=2$	$m=3$	$m=4$	MILU
10	7	6	5	5	5	4
2.435	1.895	1.834	1.729	1.779	1.843	1.607

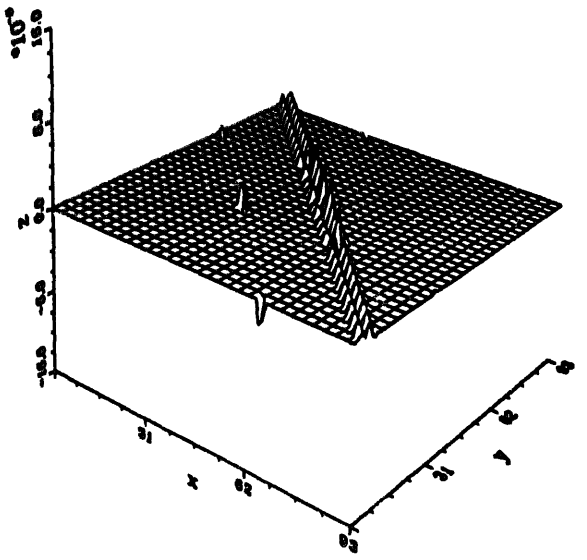

Fig. 11. The surface generated by the entries of the matrix $C - (A_{ss} + K_{ss})$ for the nondimensionalized geopotential system at the end of 1 h. The mesh resolution is 31 by 31 for the original domain. For this choice, there are 93 nodes on the interfaces and 217 nodes in each of the four subdomains.

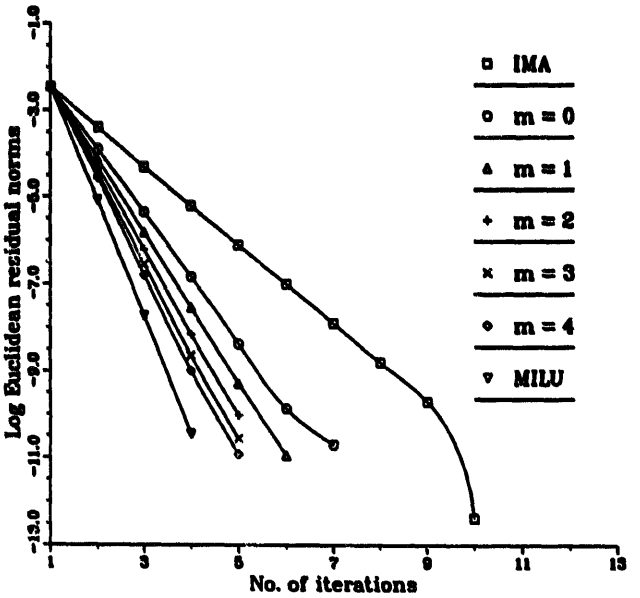

Fig. 12. The $\log_{10}$ residual norms for the modified interface matrix system on the interfaces for the nondimensionalized geopotential matrix system at the end of 1 h using the algorithm MIMA. The mesh resolution is 51 by 51. For this choice, there are 153 nodes on the interfaces and 612 nodes in each of the four subdomains.

Table 5 is presented here to provide further information. We integrated the shallow water equations by using algorithm MIMA II for 1 h for five different mesh resolutions with and without the improved initial guesses in the subdomains. These two cases are identified by MIMA II(a) and MIMA II(b). For larger problems, the subdomain solvers become more expensive and hence the differences of the computational cost between MIMA II(a) and II(b) become more pronounced as is confirmed by the results summarized in Table 5. For the case in which no improvements are made, the initial guesses are taken to be zero vectors for the subdomain solvers.

Finally we would like to compare the SCMA III with the MIMA II algorithms by integrating the shallow water equations for a period of 1 h with various mesh resolutions. It becomes evident that, as the mesh resolution increases, the computational costs of the solutions in the four subdomains and on the interfaces will play a prominent role for the efficiency of the entire integration process. Thus we expect the MIMA II algorithm to behave much better than the SCMA III algorithm for higher mesh resolutions, as confirmed by the numerical experiment results shown in Fig. 14.

Table 5
The effect of the improved initial guesses

Resolutions	CPU time for MIMA II(a)	CPU time for MIMA II(b)
15 × 15	0.418	0.525
27 × 27	1.515	1.886
39 × 39	3.854	4.936
51 × 51	8.879	11.205
63 × 63	17.537	23.349

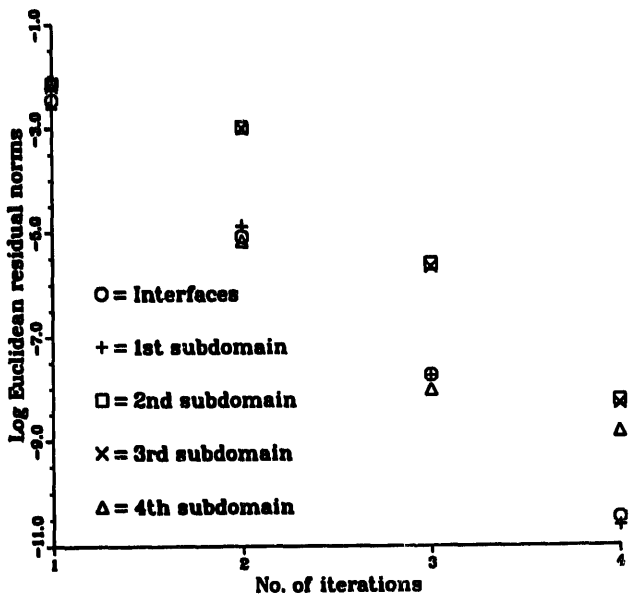

Fig. 13. The history of improved initial guesses in the subdomains and on the interfaces for the nondimensionalized geopotential matrix system at the end of 1 h using the MIMA II algorithm. The mesh resolution is 51 by 51 in the original domain.

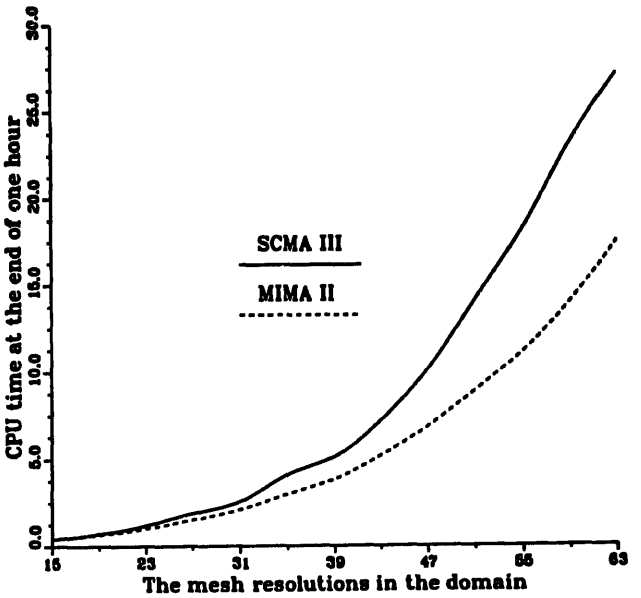

Fig. 14. A comparison in terms of CPU time between the SCMA III and MIMA II algorithms for various mesh resolutions, where the horizontal coordinates n, for $n = 15, 19, \ldots, 63$, stand for $n \times n$ mesh resolutions in the original domain.

9.2. Parallel implementation and results

We designed a multitasking code to implement the algorithm MIMA II on the CRAY YMP/432 supercomputer using four vector processors in a dedicated environment.

The domain decomposition approach formulated by using the SCMA or the MIMA approach essentially reduces the solution of a large linear system into that of solving several disjoint smaller subsystems. This is a typical example of the coarse-grained parallelism. This kind of parallelism may be efficiently implemented on the Cray YMP supercomputer by employing macrotasking techniques.

We exploited also microtasking techniques besides macrotasking for the following reasons: in a typical finite element solution of a time dependent partial differential equation problem, the following computational stages are required:

(1) input or preparation of the data required by the code, such as the global numbering of the nodes, the calculation of coordinates of these nodes, etc.,
(2) construction and representation of the triangulation if triangular elements are used;
(3) specification of the initial conditions;
(4) calculation of the element matrices and the load vectors for each time step;
(5) assembly of the element matrices and load vectors to obtain the global matrices and load vectors for each time step.
(6) the solution of the global matrix systems at each time step.

The domain decomposition based on the SCMA and the MIMA algorithms effectively parallelizes the last stage of a finite element code, i.e. the solution of global matrix systems. Although it is generally true that the main workload is concentrated at the final stage, the parallelization of the other stages turns out to be equally important. Moreover, the interface calculation is the overhead for coarse-grained parallelism for the subdomains, therefore the interface solver including the preconditioner should also be parallelized as much as possible.

The reason for this is given by Amdahl's law on the theoretical speed-up for parallel computing (see [58]). Amdahl's law points out that even a small portion of a serial code may drastically reduce the speed-up and the situation gets worse when more physical CPUs are involved. For example, suppose that 20% of computation is not multitasked, then Amdahl's law predicts that best speed-up is less than 2.5 for four processors, 3.33 for eight processors and 5 for an infinite number of processors.

It is not efficient to parallelize the other stages by using macrotasking techniques since the granularity is small and the overhead of macrotasking is high. Microtasking is preferable in this case, since microtasking permits multiple processors to work on the code at the do-loop level where the task size, or granularity is small and the microtasking overhead is generally much smaller than the macrotasking overhead.

Speed-up is often defined to be the ratio of the execution time for the best serial algorithm and the parallel algorithm. However, it is not a trivial task to determine an optimal serial algorithm for a particular application and computer architecture. Following [59, 60], the meaning of the speed-up reported here refers to measurements relative to the uni- and multi-processor implementation of the domain decomposition algorithm.

We applied the multitasking techniques to the algorithm MIMA II. Large granularity problems, such as solving the subdomain problems, are macrotasked. Small granularity problems in each subroutine are microtasked. Since microtasking is not allowed in the main program, the autotasking preprocessor has been invoked to split the loops which appear in the

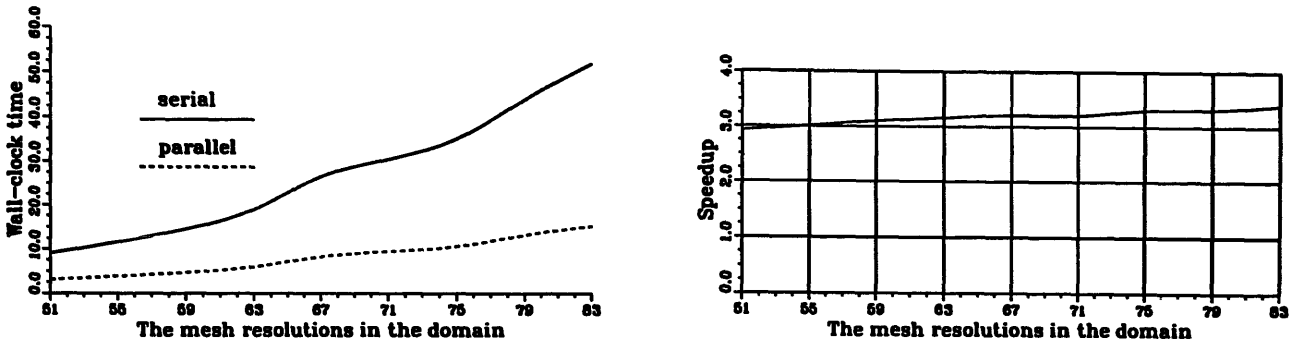

Fig. 15. A comparison of the wall-clock times required for a 1 h model integration for the serial and parallel computations by using the MIMA II algorithm and speed-up due to the parallelization of MIMA II versus the mesh resolutions in the entire domain, where the horizontal coordinates n, for $n = 51, 55, \ldots, 83$, stand for $n \times n$ mesh resolutions in the original domain.

main program among processors. The results of the parallel implementation for various mesh resolutions are presented graphically in Fig. 15.

We wish to point out here the existence of an inherent parallelism in most of the computational stages listed above. The assembly process, for example, is independent of the order in which individual element matrices are added into the global matrix and each element matrix makes its own independent contribution to the global stiffness matrix. The loop splitting (as has been applied here) or even the self-scheduling technique, see [61], may be used.

The important issue here is to prevent any particular entry in the global matrix from being modified simultaneously. An interprocess synchronization mechanism, the lock, must be invoked in order to guard the corresponding critical region in the code.

Many recurrence relations which appeared in the original single domain finite element code (see [25]) are not intrinsic and have been removed. For example, the global nodal numbers are fully determined by the local element nodal numbers, once a particular element has been selected. Hence the relations between the global numbering of the nodes, local numbering of the element vertices and element numbering can be calculated independent of the calculations made for the previous elements.

10. Conclusions

In this paper, we applied nonoverlapping domain decomposition techniques to the parallel processing of a two-stage Numerov–Galerkin finite element model of the 2-D shallow water equations on a limited-area domain, a channel on the rotating earth. The techniques based on the substructuring ideas, including both the SCMA and MIMA, prove to be applicable and efficient for the parallel processing of the finite element model. They provide logical structures for mapping a whole computational effort to a number of processors for concurrent processing.

SCMA is a divide-and-feedback process. One of the most important factors which determine the efficiency of this approach is the preconditioning on the interfaces for the iterative solution of the Schur complement matrix linear system. The boundary probe preconditioners,

proposed by Chan in [30], prove to be robust and applicable to more complicated operators like those of the shallow water equations.

The MIMA approach differs from the SCMA approach in that the former iteratively approximates the subdomain and the interface solutions successively. The efficiency of this approach relies on the modified interface matrix, which should be a good approximation of the Schur complement matrix in the sense that the spectral radius given in (8.4) is small. Two possibilities were given in Section 8.2 for the construction of the modified interface matrix. For the construction based on the MILU factorizations in the subdomains, an order of 10^{-3} of the spectral radii was obtained. We plan to explore, in the near future, the possibility of other efficient approximation techniques.

For more realistic applications where fast subdomain solvers are not available, it is necessary to solve the subdomain problems by either direct or by iterative solvers. If iterative subdomain solvers are used, the MIMA approach is capable of improving iteratively the initial guesses in both the subdomains and on the interfaces. Thus it is less expensive to obtain solutions in the subdomains as well as on the interfaces and the MIMA is preferred to the SCMA.

Acknowledgment

The research of the second author is supported by the Air Force grant AFOSR-89-0462 through the Geophysical Fluid Dynamics Institute of the Florida State University and the DOE grant 120053223 through the Supercomputer Computations Research Institute of the Florida State University.

References

[1] H.A. Schwarz, Über einige Abbildungsaufgaben, Ges. Math. Abh. 11 (1869) 65–83.
[2] J.S. Przemieniechi, Matrix structural analysis of substructures, AIAA J. 1 (1963) 138–147.
[3] D.A. Kopriva, Computation of hyperbolic equations on complicated domains with patched and overset Chebyshev grids, SIAM J. Sci. Statist. Comput. 10 (1989) 120–132.
[4] M.M. Rai, A conservative treatment of zonal boundaries for Euler equation calculations, J. Comput. Phys. 62 (1986) 472–503.
[5] D.A. Kopriva, Domain decomposition with both spectral and finite difference methods for the accurate computation of flows with shocks, Appl. Numer. Math. 6 (1989) 141–151.
[6] R. Temam, Survey of the status of finite element methods for partial differential equations, in: D.L. Dwoyer, M.Y. Hussaini, and R.G. Voigt, eds., Finite Elements, Theory and Application (Springer, New York, 1988) 1–33.
[7] D.A. Kopriva, Spectral solution of inviscid supersonic flows over wedges and axisymmetric cones, Comput. & Fluids 21 (2) (1992) 247–266.
[8] R. Glowinski, W. Kinton and M.F. Wheeler, Acceleration of domain decomposition algorithms for mixed finite elements by multi-level methods, in: T.F. Chan, R. Glowinski, J. Périaux and O.B. Widlund, eds., 3rd Internat. Symp. on Domain Decomposition Methods for Partial Differential Equations (SIAM, Philadelphia, PA, 1990) 263–289.
[9] Q.V. Dinh, R. Glowinski, J. Périaux and G. Terrasson, On the coupling of viscous and inviscid models for incompressible fluid flows via domain decomposition, in: R. Glowinski, G.H. Golub, G.A. Meurant and J. Périaux, eds., 1st Internat. Symp. on Domain Decomposition Methods for Partial Differential Equations (SIAM, Philadelphia, PA, 1988) 350–369.

[10] R. Glowinski, J. Périaux and G. Terrasson, On the coupling of viscous and inviscid models for compressible fluid flows via domain decomposition, in: T.F. Chan, R. Glowinsk, J. Périaux and O.B. Widlund, eds., 3rd Internat. Symp. on Domain Decomposition Methods for Partial Differential Equations (SIAM, Philadelphia, PA, 1990) 64–97.
[11] P. Concus, G.H. Golub and D.P. O'Leary, A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations, in: J.R. Bunch and D.J. Rose, eds., Sparse Matrix Computations (Academic Press, New York, 1976) 309–332.
[12] M. Dryja, A capacitance matrix method for Dirichlet problem on polygon regions, Numer. Math. 39 (1982) 51–64.
[13] M. Dryja, A finite element–capacitance method for elliptic problems on regions partitioned into subregions, Numer. Math. 44 (1984) 153–168.
[14] G.H. Golub and D. Mayers, The use of preconditioning over irregular regions, in: R. Glowinski and J.L. Lions, eds., Computing Methods in Applied Science and Engineering VI (North-Holland, Amsterdam, 1984) 3–14.
[15] P.E. Bjørstad and O.B. Widlund, Iterative methods for the solution of elliptic problems on regions partitioned into substructures, SIAM J. Numer. Anal. 23 (1986) 1097–1120.
[16] T.F. Chan, Analysis of preconditioners for domain decomposition, SIAM J. Numer. Anal. 24 (1987) 382–390.
[17] T.F. Chan, A domain-decomposed fast Poisson solver on a rectangle, SIAM J. Sci. Statist. Comput. 8 (1987) s14–s26.
[18] T.F. Chan and D.C. Resasco, A framework for the analysis and construction of domain decomposition preconditioners, in: R. Glowinski, G.H. Golub, G.A. Meurant and J. Périaux, eds., 1st Internat. Symp. on Domain Decomposition Methods for Partial Differential Equations (SIAM, Philadephia, PA, 1988) 217–230.
[19] M.J.P. Cullen and K.W. Morton, Analysis of evolutionary error in finite-element and other methods, J. Comput. Phys. 34 (1980) 245–267.
[20] I.M. Navon, A survey of finite-element methods in quasi-linear fluid flow problems, WISK Report 240, National Research Institute for Mathematical Sciences, Pretoria, South Africa (1977) 44 pp.
[21] I.M. Navon, Finite-element simulations of the shallow-water equations model on a limited area domain, Appl. Math. Modelling 3 (1979) 337–348.
[22] I.M. Navon and U. Muller, FESW – A finite-element Fortran IV program for solving the shallow-water equations, Adv. Engrg. Software 1 (1979) 77–86.
[23] I.M. Navon and H.A. Riphagen, An implicit compact fourth-order algorithm for solving the shallow-water equations in conservative law-form, Monthly Weather Rev. 107 (1979) 1107–1127.
[24] I.M. Navon, A Numerov–Galerkin technique applied to a finite-element shallow-water equations model with enforced conservation of integral invariants and selective lumping, J. Comput. Phys. 52 (1983) 313–339.
[25] I.M. Navon, FEUDX: A two-stage, high-accuracy finite-element FORTRAN program for solving shallow-water equations, Comput. Geosci. 13 (1987) 255–285.
[26] M. Kawahara, H. Hirano, K. Tsubota and K. Inagaki, Elective lumping finite-element method for shallow-water flow, Internat. J. Numer. Methods Fluids 2 (1982) 89–112.
[27] B. Neta and R.T. Williams, Stability and phase speed for various finite-element formulations of the advection equation, Comput. & Fluids 14 (1986) 393–410.
[28] J. Steppler, I.M. Navon and H.-I. Lu, Finite-element schemes for extended integrations of atmospheric models, J. Comput. Phys. 89 (1990) 95–124.
[29] P. Sonneveld, CGS, A fast Lanczos-solver for nonsymmetric linear systems, SIAM J. Sci. Statist. Comput. 10 (1989) 36–52.
[30] T.F. Chan and D. Resasco, A survey of preconditioners for domain decomposition, Technical Report 414, Computer Science Department, Yale University, 1985.
[31] J.H. Bramble, J.E. Pasciak and A.H. Schatz, An iterative method for elliptic problems on regions partitioned into substructures, Math. Comp. 46 (1986) 361–369.
[32] J.H. Bramble, J.E. Pasciak and A.H. Schatz, The construction of preconditioners for elliptic problems by substructures, Math. Comp. 47 (1986) 103–134.
[33] T.F. Chan and D. Goovaerts, A note on the efficiency of domain decomposed incomplete factorizations, SIAM J. Sci. Statist. Comput. 11 (1990) 794–803.
[34] C. Börgers, The Neumann–Dirichlet domain decomposition method with inexact solvers on the subdomains, Numer. Math. 55 (1989) 123–136.

[35] Y. Cai and I.M. Navon, Parallel processing of a finite element model of the shallow water equations via domain decomposition – A domain-decomposed preconditioner approach, unpublished.
[36] A. Grammeltvedt, A survey of finite difference scheme for the primitive equations for a barotropic fluid, Monthly Weather Rev. 97 (1969) 384–404.
[37] B.K. Schwarz and B. Wendroff, The relative efficiency of finite-difference and finite-element methods, I. Hyperbolic problems and splines, SIAM J. Numer. Anal. 11 (1974) 979–993.
[38] D.U. Von-Rosenberg, Methods for the Numerical Solution of Partial Differential Eqations (Elsevier, New York, 1969).
[39] H.H. Ahlberg, E.N. Nielson and J.L. Walsh, The Theory of Splines and their Application: Mathematics in Science and Engineering, 38 (Academic, New York, 1967).
[40] O.C. Zienkiewicz and K. Morgan, Finite Elements and Approximation (Wiley, New York, 1983).
[41] M.F. Rubinstein, Combined analysis by substructures and recursion, ASCE J. Structural Div. 93 (1967) 231–235.
[42] T. Furnike, Computerized multiple level substructuring analysis, Comput. & Structures 2 (1972) 1063–1073.
[43] R. Cottle, Manifestations of the Schur complement, Linear Algebra Appl. 8 (1974) 189–211.
[44] R.W. Hockney, The potential calculation and some applications, Methods Comput. Phys. 9 (1970) 135–211.
[45] J.M. Ortega, Introduction to Parallel and Vector Solution of Linear Systems (Plenum, New York, 1988).
[46] Y. Saad, Krylov subspace methods on supercomputers, SIAM J. Sci. Statist. Comput. 10 (1989) 1200–1232.
[47] J.J. Dongarra, L.S. Duff, D.C. Sorensen and H.A. Van der Vorst, Solving Linear Systems on Vector and Shared Memory Computers (SIAM, Philadelphia, PA, 1991).
[48] A.T. Chronopoulos, A fast squared Lanczos method for nonsymmetric linear systems, Research Report UMSI 91/310, University of Minnesota Supercomputer Institute, 1991.
[49] G. Radicati, Y. Robert and S. Succi, Iterative algorithms for the solution of nonsymmetric systems in the modelling of weak plasma turbulence, J. Comput. Phys. 80 (1989) 489–497.
[50] E.F. Kaasschieter, The solution of non-symmetric linear systems by bi-conjugate gradients or conjugate gradient squared, Research Report 86-21, Delft University of Technology, 1986.
[51] D.E. Keyes and W.D. Gropp, A comparison of domain decomposition techniques for elliptic partial differential equations and their parallel implementation, SIAM J. Sci. Statist. Comput. 8 (1987) s166–s202.
[52] J.A. Meijerink and H.A. van der Vorst, An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix, Math. Comp. 31 (1977) 148–162.
[53] I. Gustafsson, A class of first order factorization methods, BIT 18 (1978) 142–156.
[54] I. Gustafsson, Modified incomplete Cholesky (MIC) methods, in: D.J. Evans, eds., Preconditioning Methods: Analysis and Applications (Gordon and Breach, New York, 1983) 265–294.
[55] O. Axelsson and V.A. Barker, Finite Solutions of Boundary Value Problems, Theory and Computation (Academic Press, New York, 1984).
[56] D.E. Keyes and W.D. Gropp, Domain decomposition for nonsymmetric systems of equations: Examples from computational fluid dynamics, in: T.F. Chan, R. Glowinski, J. Périaux and O.B. Widlund, eds., 2nd Internat. Symp. on Domain Decomposition Methods for Partial Differential Equations (SIAM, Philadelphia, PA, 1989) 321–339.
[57] L.A. Hageman and D.M. Young, Applied Iterative Methods (Academic Press, New York, 1981).
[58] Cray Y-MP™, Cray X-MP EA™, and CRAY X-MP™ Multitasking Programmer's Manual, SR-0222 F-01, Cray Research, Inc., Distribution Center, 2360 Pilot Knob Road, Mendota Heights, MN 55120, 1989.
[59] R. Natarajan, Finite element applications on a shared-memory multiprocessor: Algorithms and experimental results, J. Comput. Phys. 94 (1991) 352–381.
[60] W.D. Gropp and D.E. Keyes, Complexity of parallel implementation of domain decomposition techniques for elliptic partial differential equations, SIAM J. Sci. Statist. Comput. 9 (1988) 312–326.
[61] S. Brawer, Introduction to Parallel Programming (Academic Press, New York, 1989).