Contemporary Mathematics
Volume 180, 1994

Parallel Domain-Decomposed Preconditioners in Finite Element Shallow Water Flow Modeling

Y. CAI AND I.M. NAVON

ABSTRACT. This paper is a highly condensed account of applying a domain-decomposed (DD) preconditioner approach to the parallel numerical solution of the shallow water equations by the finite element method. Three types of DD preconditioners are employed to accelerate, with right preconditioning, the convergence of several competitive non-symmetric linear iterative solvers of current interest. Analysis and comparisons are provided in this contribution. The resulting algorithms are parallelized at both the subroutine level to accommodate the subdomain by subdomain computation and at the loop level to exploit the parallelism of the finite element discretization. The implementations were carried out on the parallel vector supercomputer CRAY Y-MP/432.

1. Introduction

Although domain decomposition ideas are traceable to the work of Schwarz [17] in 1869 and that of engineers beginning from the 1960s [8,14,15], an efficient way to handle the coupling between artificially divided non-overlapping substructures was first proposed in [7]. In essence, this is a divide-and-feedback process which continues until a prescribed convergence criterion on the interfaces is satisfied.

However, the method mentioned above can be expensive in the absence of fast subdomain solvers. To remedy this disadvantage and, at the same time, retain the nice parallelization property, at least two other approaches have been

1991 *Mathematics Subject Classification.* Primary 65M55, 65Y05; Secondary 65N30, 65Y20.
The first author was supported in part by US DOE Grant #120054323 and DE-FC05-85ER250000, through the Supercomputer Computations Research Institute of the Florida State University. The authors are thankful to David Keyes for his valuable suggestions on the first draft of the paper.
The final version of this paper will be submitted for publication elsewhere.

proposed. One is the domain-decomposed preconditioner approach (DDPA) (also called full matrix domain decomposition in [**11**]) advocated in [**3**], the other being the recently proposed modified interface matrix approach (MIMA) (see [**13**]). Both approaches abandon the idea of the Schur complement matrix approach that decoupled subdomain problems are independently solved *only after* the solution on the interfaces is obtained.

The DDPA approach consists of the construction of a preconditioner in a domain-decomposed way such that approximate solutions in the subdomains and on the interfaces can be simultaneously updated at the cost of only inexact subdomain solvers. On the other hand, the MIMA approach successively improves the subdomain and interface approximate solutions with improved initial solutions. Thus it mitigates the disadvantage due to the absence of fast subdomain solvers.

In this paper, we shall concentrate our attention only on the application of DDPA to finite element shallow water flow simulation. A detailed comparison between DDPA and MIMA for the problem at hand will be addressed elsewhere in a separate research work.

We test all the proposed algorithms, to be presented shortly, on a shallow water equations model using the Grammeltvedt initial conditions [**9**] on a limited-area rectangular region — a channel on the rotating earth. The model is essentially the same as the one earlier used by Houghton, et al. [**10**].

Non-symmetric linear iterative solvers are important kernels to the current domain decomposition approach. Among many available algorithms, we are especially interested in three, namely, GMRES [**16**], CGS [**18**] and Bi-CGSTAB [**19**].

In section 2, we give three types of DD preconditioners under consideration. Carefully selected numerical results are given in section 3 and main conclusions are drawn.

2. Domain-decomposed preconditioners

The DD preconditioner approach is presented in this section using the notation adopted in [**13**]. We consider solving linear systems resulting from finite element discretization in space with an implicit temporal difference scheme [**4**].

Since we are interested in implementing the algorithms on parallel computers with powerful vector processors, the original domain is divided into strips along the west-east directions in order to yield longer vectors (see [**12**]). Thus, cross points are eliminated from consideration.

In order to explore the inherent parallelism at the subdomain level, following [**14**], we number the global nodes in a substructured way, such that the coefficient matrices of geopotential and velocities at each time step present the following block-bordered structures

$$A = \begin{bmatrix} A_{dd} & A_{ds} \\ A_{sd} & A_{ss} \end{bmatrix} \tag{2.1}$$

The meaning of each element in the matrix is explained in [**13**]

Three types of DD preconditioners employed for our problem assume, respectively, the structurally symmetric form, the block upper and the lower triangular forms

(1)

$$B = \begin{bmatrix} B_{dd} & A_{ds} \\ A_{sd} & B_{ss} \end{bmatrix} \tag{2.2}$$

(2)

$$B = \begin{bmatrix} B_{dd} & A_{ds} \\ 0 & G \end{bmatrix} \tag{2.3}$$

(3)

$$B = \begin{bmatrix} B_{dd} & 0 \\ A_{sd} & G \end{bmatrix} \tag{2.4}$$

where B_{dd} approximates A_{dd} in some sense. For example, B_{ii} might be the relaxed incomplete LU factorization (RILU) [**2**] of A_{ii}. B_{ss} is given, for the first type of DD preconditioners, by

$$B_{ss} = G + \sum_{i=1}^{n} A_{si} B_{ii}^{-1} A_{is} \tag{2.5}$$

For all three cases, the matrix G is to be determined.

Let

$$AB^{-1} = \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix} \tag{2.6}$$

Then we may show that (P_{11}, P_{12} and P_{21} are given in [**4**])

$$P_{22} = \begin{cases} (A_{ss} - A_{sd} B_{dd}^{-1} A_{ds}) G^{-1} & \text{for (2.2)} \\ (A_{ss} - A_{sd} B_{dd}^{-1} A_{ds}) G^{-1} & \text{for (2.3)} \\ A_{ss} G^{-1} & \text{for (2.4)} \end{cases} \tag{2.7}$$

The matrix G is chosen such that P_{22} is an identity matrix. Thus we have that $G = A_{ss}$ for the third type of DD preconditioners and

$$G = A_{ss} - A_{sd} B_{dd}^{-1} A_{ds} = A_{ss} - \sum_{i=1}^{n} A_{si} B_{ii}^{-1} A_{is}. \tag{2.8}$$

for the first two types. For the latter case, nn_s number of inexact subdomain solves must be carried out for the construction of G such that P_{22} is an identity matrix, where n and n_s are, respectively, the number of subdomains and number of nodes on the interfaces. After one has constructed the preconditioner B, the major computational work required for solving the preconditioning linear system

TABLE 1. Amount of work for solving $Bp = q$

First type	$2n$ inexact subdomain solves
Second type	n inexact subdomain solves
Third type	n inexact subdomain solves

of the form $Bp = q$ at each time step corresponding to each of the three types of DD preconditioners is summarized in Table 1.

Similar to many other applications, it turns out that action of the matrix operator G is, in fact, predominantly local and the matrix G given by (2.8), although dense, allows itself to be reasonably approximated by a very low-bandwidth sparse matrix. The modified rowsum preserving interface probing ideas [4] (see also [5,6]) are employed for the approximate formation of G, for which only one inexact solve is needed in each subdomain.

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

3. Numerical results and main conclusions

In this section, we present selected numerical results along with some discussions. All numerical experiments are carried out on the four-processor CRAY Y-MP/432 vector-parallel supercomputer. The numerical values of some parameters associated with the original PDEs are summarized in [13]. The problem is non-dimensionalized by choosing $\varphi_0 = 10^2\ m^2/s^2$ [4].

In this set of numerical experiments, we test the relative efficiencies of the three types of DD preconditioners presented in section 2. As inexact solvers in the subdomains, we use RILU (relaxed incomplete LU) factorization along with forward and back substitutions. On the interfaces, the MIP(0) interface probe construction is used for the approximation of G given in (2.11) for the first two types of DD preconditioners. Table 2 gives some numerical results.

We see that GMRES, CGS and Bi-CGSTAB are very competitive with each other. GMRES requires the largest number of iterations to attain prescribed convergence. However we note that only one matrix-vector multiplication per iteration will drive GMRES to convergence, compared with two such multiplications required for CGS or Bi-CGSTAB at each iteration. The non-smooth convergence behavior (not shown here) of the CGS algorithm is observed in the current set of experiments. Bi-CGSTAB and GMRES methods generate smoother convergence history.

TABLE 2. A comparison of CPU time (number of iterations)

Preconditioner types		First type	Second type	Third type
40 by 35	GMRES	0.178 (12)	0.102 (12)	0.088 (11)
mesh	CGS	0.199 (7)	0.099 (6)	0.106 (7)
resolution	Bi-CGSTAB	0.219 (7)	0.099 (6)	0.092 (6)
80 by 75	GMRES	0.89 (14)	0.53 (15)	0.47 (14)
mesh	CGS	1.08 (9)	0.54 (8)	0.50 (8)
resolution	Bi-CGSTAB	0.97 (8)	0.54 (8)	0.50 (8)

Using the idea of m-step preconditioning [1], we can test the sensitivities of these preconditioners to inexact subdomain solvers. We observe that preconditioners of the third type can accelerate the convergence of three iterative methods at about the same rate as the other two types of preconditioners, except for cases when subdomain solvers may be considered to be exact or nearly exact.

In practice, the inexact subdomain solvers are far from exact, we consider preconditioners of the third type to be the best, although the second type of DD preconditioners is very competitive. Even if exact or nearly exact subdomain solvers are used for the first two types of DD preconditioners, the gain in the number of iterations is far from offsetting the additional computational cost required for constructing these solvers.

For parallelization, subdomain by subdomain computations are carried out at the subroutine level, and loop level parallelism is exploited for calculations involved in finite element discretization using a multicoloring scheme [4].

We integrated the finite element model of the shallow water equations for four different mesh resolutions for a period of five hours with corresponding time step sizes of $\Delta t = 1800$ s, 1000 s, 600 s and 400 s, respectively, using Bi-CGSTAB preconditioned by the third type of DD preconditioners. The speed-up results are summarized in Table 3. It should be pointed out that the automatic do-loop level parallelization as detected and exploited by the autotasking preprocessor does not yield a speed-up larger than two. The reason is that autotasking is unable to detect parallelism across subroutine boundaries.

TABLE 3. Parallel performance results on the CRAY

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

The main conclusions of this research are

(1) Three types of DD preconditioners were found to work reasonably well

with GMRES, CGS and Bi-CGSTAB, with the third type being computationally the least expensive and the first type most expensive.

(2) For all cases, GMRES requires roughly twice as many iterations as required by the CGS or Bi-CGSATB. However, these three algorithms were found to be approximately equally efficient in terms of CPU time.

(3) To achieve better speed-up results, it is important to remove the critical regions in the assembly process by using multicoloring of the elements.

REFERENCES

1. L. Adams, *m-Step preconditioned conjugate gradient methods*, SIAM J. Sci. Stat. Comput. **6(2)** (1985), 452–463.
2. O. Axelsson and G. Lindskog, *On the eigenvalue distribution of a class of preconditioning methods*, Numer. Math. **48** (1986), 479–498.
3. 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.
4. Y. Cai and I.M. Navon, *Parallel block preconditioning techniques for the numerical simulation of the shallow water flow using finite element methods*, J. Comput. Phys. (to appear).
5. T.F. Chan and T.P. Mathew, *The interface probing technique in domain decomposition*, SIAM J. Matrix Anal. Appl. **13(1)** (1992), 212–238.
6. T.F. Chan and D. Resasco, *A survey of preconditioners for domain decomposition*, Technical Report 414, Computer Science Dept., Yale Univ., 1985.
7. M. Dryja, *A capacitance matrix method for Dirichlet problem on polygon regions*, Numer. Math. **39** (1982), 51–64.
8. T. Furnike, *Computerized multiple level substructuring analysis*, Computers and Structures **2** (1972), 1063–1073.
9. A. Grammeltvedt, *A survey of finite difference scheme for the primitive equations for a barotropic fluid*, Mon. Wea. Rev. **97(5)** (1969), 384–404.
10. D. Houghton, A. Kasahara and W. Washington, *Long-term integration of the barotropic equations by the lax-wendroff method*, Mon. Wea. Rev. **94(3)** (1966), 141–150.
11. 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. Stat. Comput. **8(2)** (1987), s166–s202.
12. G. Meurant, *Domain decomposition methods for partial differential equations on parallel computers*, Int. J. Supercomputer Appl. **2(4)** (1988), 5–12.
13. I.M. Navon and Y. Cai, *Domain decomposition and parallel processing of a finite element model of the shallow water equations*, Comput. Methods Appl. Mech. & Eng. **106(1-2)** (1993), 179–212.
14. J.S. Przemieniecki, *Matrix structural analysis of substructures*, AIAA J. **1** (1963), 138–147.
15. M.F. Rubinstein, *Combined analysis by substructures and recursion*, ASCE J. of the Structural Division **93(ST2)** (1967), 231–235.
16. Y. Saad and M.H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput. **7(3)** (1986), 856–869.
17. H.A. Schwarz, *Über einige Abbildungsaufgaben*, Ges. Math. Abh. **11** (1869), 65–83.
18. P. Sonneveld, *CGS, A fast Lanczos-solver for nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput. **10(1)** (1989), 36–52.
19. H.A. van der Vorst, *Bi-CGSTAB: a more smoothly converging variant of CG-S for the solution of nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput. **13(3)** (1992), 631–644.

DEPARTMENT OF MATHEMATICS AND SUPERCOMPUTER COMPUTATIONS RESEARCH INSTITUTE, FLORIDA STATE UNIVERSITY, TALLAHASSEE, FLORIDA 32306

E-mail address: cai@scri.fsu.edu