18th World IMACS / MODSIM Congress, Cairns, Australia 13-17 July 2009
http://mssanz.org.au/modsim09

Scalability studies of two-level domain decomposition algorithms for systems of nonlinear PDEs

Liu, S.[1] **and X.-C. Cai**[2]

[1] *Department of Applied Mathematics, University of Colorado at Boulder, Boulder, CO 80309, USA*
Email: si.liu@colorado.edu
[2] *Department of Computer Science, University of Colorado at Boulder, Boulder, CO 80309, USA*
Email: cai@cs.colorado.edu

Abstract: Multilevel domain decomposition methods have attracted significant attention for solving nonlinear equations. In this paper we study several two-level domain decomposition algorithms for solving nonlinear algebraic systems arising from the discretization of coupled systems of partial differential equations. The class of Newton-Krylov-Schwarz algorithms (NKS) is often used for such problems, and the robustness and scalability of NKS depends heavily on its inner most component; i.e., the Jacobian preconditioner. In this paper, we investigate four two-level overlapping domain decomposition preconditioners, all designed as general purpose preconditioners for systems of PDEs. To show the parallel efficiency of the algorithms, we apply them to some rather difficult nonlinear systems obtained from the one-shot discretization of inverse problems. Linear and superlinear scalability results are obtained on parallel computers with hundreds of processors.

Keywords: *Multilevel domain decomposition, parallel computing, inexact Newton, inverse problems*

1. INTRODUCTION

Computer simulations of many interesting physical phenomena require the numerical solution of systems of partial differential equations. This includes, for example, the modeling of the global climate, fluid flows around airplanes, or blood flows in the human body (e.g. Barker and Cai, 2009, Cai et al., 1998, and Yang et al., 2009). One of the key elements of simulation technologies is the design and implementation of algorithms and software that are scalable for massively parallel computers. In this paper, we discuss some domain decomposition methods, which are a class of divide-and-conquer methods for solving mathematical problems defined on a physical domain for large-scale simulations on parallel, distributed-memory computers. The focus is on several two-level domain decomposition algorithms that exhibit linear, and sometimes superlinear, scalability for solving complex coupled problems. To show the applicability of the techniques, we present some results obtained from solving some difficult systems arising from the one-shot method for solving inverse problems, such as

$$\begin{cases} -\beta\Delta\rho + \nabla u \cdot \nabla\lambda = 0 \\ -\nabla\cdot(\rho\nabla\lambda) + (u - z) = 0 \\ -\nabla\cdot(\rho\nabla u) - f = 0, \end{cases} \tag{1}$$

in which $\rho(x, y)$ is the coefficient to be determined and it satisfies the forward problem

$$\begin{cases} -\nabla\cdot(\rho\nabla u) &= f(x,y), \quad (x,y) \in \Omega \subset R^2 \\ u(x,y) &= g(x,y), \quad (x,y) \in \partial\Omega, \end{cases} \tag{2}$$

whose solution $u(x, y)$ is supposed to be available through measurement $z(x, y)$. Here $\lambda(x, y)$ is the Lagrange multiplier and β is a regularization parameter. More details about (1) will be discussed in Section 2. Using the standard finite difference method, (1) is transformed into a large nonlinear system of equations

$$F(X) = 0, \tag{3}$$

where X represents the unknowns ρ, u, and λ defined at the mesh points.

2. MULTILEVEL DOMAIN DECOMPOSITION METHODS

The rest of the paper is focused on the class of Newton-Krylov-Schwarz methods (NKS)(Cai et al., 1998) for solving (3). Let X_0 be an initial guess. The inexact Newton iteration is carried out in two steps. First, a Newton direction ΔX_k is computed approximately by solving

$$\|F(X_k) + J(X_k)\Delta X_k\| \le \eta\|F(X_k)\|, \tag{4}$$

and then the current solution X_k is updated to obtain a new solution

$$X_{k+1} = X_k + \xi_k\Delta X_k. \tag{5}$$

Here $J(X_k) = F'(X_k)$ is the Jacobian matrix at X_k, ξ_k is the steplength determined by a linesearch procedure (Dennis and Schnabel, 1996), and η is the linear stopping condition. The Jacobian system is often large, sparse, and highly ill-conditioned. A high quality preconditioner is essential for the success of NKS. Three requirements for the preconditioner are: (1) capable of decreasing the condition number of J substantially, (2) highly parallel in order to solve the problem on computers with hundreds or even thousands of processors, and (3) not sensitive to certain noise in the problem. We next consider the class of two-level overlapping Schwarz preconditioners.

Following the convectional notations (Toselli and Widlund, 2005), we rewrite the Jacobian system as $AX = b$, and we describe the Schwarz method as a right preconditioner

$$(AM^{-1})X' = b, \tag{6}$$

and $X = M^{-1}X'$ is the actual solution. To formally define the one-level additive Schwarz preconditioning matrix M^{-1}, we first partition the computational domain Ω into N non-overlapping subdomains Ω_l, $l = 1, \cdots, N$, where N equals the number of processors (np). Each subdomain Ω_l is extended to a larger region

$\Omega_l^{'}$, such that $\Omega_l \subset \Omega_l^{'} \subset \Omega$. On each subdomain, we define A_l as a restriction of A in $\Omega_l^{'}$. Let B_l^{-1} be the inverse of A_l, or an approximation, the one-level restricted additive Schwarz preconditioning matrix can then be written as

$$M_{AS}^{-1} = \sum_{l=1}^{N} (R_l^0)^T B_l^{-1} R_l, \tag{7}$$

where R_l restricts a vector from Ω to $\Omega_l^{'}$, and R_l^0 is defined similarly but only works for the non-overlapping part Ω_l. When the number of processors is small, the one-level preconditioner works well for many difficult problems. However, when the number of processors is large, the communication of information between distant subdomains becomes slow, and as a result, NKS converges very slowly and sometime doesn't converge at all. To improve the communication, we introduce a coarse-level solver, which provides global communication for distant subdomains. With different combinations of coarse and local subdomain matrices, four types of two-level preconditioners can be defined.

To describe the algorithms, we assume that there are two meshes available on Ω. On the fine mesh the Jacobian system is given as A, and on the coarse mesh the Jacobian matrix is given as A_c. We also assume there is an interpolation matrix I_c^f which maps a coarse mesh vector to a fine mesh vector, and a restriction matrix R_f^c which maps a fine mesh vector to a coarse mesh vector. There are many choices of coarse meshes, as well as interpolation and restriction matrices, we don't discuss the details here because of the limited available space. Interested readers should consult with related literatures (e.g. Toselli and Widlund, 2005).

Let $M_c^{-1} = I_c^f A_c^{-1} R_f^c$. Corresponding to the idea of multigrid V-cycle (Briggs et al., 2000), a two-level multiplicative Schwarz preconditioner consists of three basic steps: a one-level additive Schwarz pre-smoothing, a coarse grid correction, and a one-level additive Schwarz post-smoothing. The preconditioner matrix is defined as

$$M_{mult}^{-1} = A^{-1}[I - (I - AM_{AS}^{-1})(I - AM_c^{-1})(I - AM_{AS}^{-1})]. \tag{8}$$

If the pre-smoothing step is omitted, the two-level multiplicative Schwarz preconditioner is changed to the so-called kaskade (cascade) Schwarz preconditioner

$$\begin{aligned} M_{kask}^{-1} &= A^{-1}[I - (I - AM_{AS}^{-1})(I - AM_c^{-1})] \\ &= M_c^{-1} + M_{AS}^{-1} - M_{AS}^{-1} A M_c^{-1}. \end{aligned} \tag{9}$$

The last term in the kaskade Schwarz preconditioner consists of matrix-matrix multiplications, which may be hard to parallelize. If we choose to drop that term, we arrive at the classical two-level additive Schwarz preconditioner

$$M_{addi}^{-1} = M_c^{-1} + M_{AS}^{-1}. \tag{10}$$

Borrowing an idea from grid sequencing, which uses an interpolated coarse grid solution as the initial guess for the fine grid iteration, we can define another multiplicative Schwarz type preconditioner, which is very similar to the full multigrid V-cycle,

$$\begin{aligned} M_{full}^{-1} &= M_c^{-1} \\ &+ A^{-1}[I - (I - AM_{AS}^{-1})(I - AM_c^{-1})(I - AM_{AS}^{-1})] \\ &- A^{-1}[I - (I - AM_{AS}^{-1})(I - AM_c^{-1})(I - AM_{AS}^{-1})] A M_c^{-1}. \end{aligned} \tag{11}$$

3. INVERSE ELLIPTIC PROBLEMS

To understand the efficiency of the algorithms, we consider the system of nonlinear equations arising from the one-shot finite difference discretization of inverse elliptic problems in two-dimensional space. Specifically, we want to determine the coefficient function $\rho(x, y)$ in (2) assuming some measurement of $u(x, y)$ is available as $z(x, y)$. We use the output least-squares Tikhonov regularization method to convert the inverse problem into a least-squares minimization problem:

$$\text{minimize } \; J(\rho, u) = \frac{1}{2} \int_{\Omega} (u - z)^2 dx + \frac{\beta}{2} \int_{\Omega} |\nabla \rho|^2 dx, \tag{12}$$

with the constraint (1) satisfied by the pair (ρ, u). The β-term is called the regularization term, in which the constant β controls the strength of regularization.

For the above constraint optimization problems, we define the Lagrangian functional as

$$\mathcal{L}(\rho, u, \lambda) = \frac{1}{2}\int_{\Omega}(u-z)^2 dx - \int_{\Omega}(\nabla \cdot \rho \nabla u + f)\lambda dx + \frac{\beta}{2}\int_{\Omega}|\nabla \rho|^2 dx. \tag{13}$$

Under certain assumptions, the solution of (12) is the solution of the following saddle-point problem: Find (ρ, u, λ) such that

$$\begin{cases} (\nabla_\rho \mathcal{L})q = 0 \\ (\nabla_u \mathcal{L})\omega = 0 \\ (\nabla_\lambda \mathcal{L})\mu = 0 \end{cases} \tag{14}$$

for any (q, ω, μ). This system leads to (1), which is solved with the boundary conditions $\dfrac{\partial \rho}{\partial n} = 0$, $\lambda = 0$, and $u = g$ on $\partial\Omega$.

For simplicity, we only consider the case that Ω is a rectangular domain covered by a uniform mesh of size h, and we discretize (1) with the standard five-point central finite difference. The ordering of the unknowns and the ordering of the equations are crucial in our algorithms. They are ordered mesh point by mesh point in a "fully coupled" manner (Cai et al., 2009). The ordering of variable u and variable λ is switched in order to avoid the zero pivot problem in the LU factorization based subdomain solver.

4. NUMERICAL RESULTS AND DISCUSSION

We study the performance of the proposed algorithms for two problems defined on different computational domains. We assume that the observed function takes the form $z(x, y) = \sin(\pi x)\sin(\pi y)$ or $z(x, y) = \cos(\pi x)\cos(\pi y)$. The Newton iteration is stopped if the following condition is satisfied:

$$\|F(X_k)\| \le \max\left\{10^{-6}\|F(X_0)\|, 10^{-10}\right\}. \tag{15}$$

A restarted GMRES is employed to solve the Jacobian system, and the GMRES iteration is stopped if

$$\|F(X_k) + J(X_k)\Delta X_k\| \le \max\left\{10^{-6}\|F(X_k)\|, 10^{-10}\right\}. \tag{16}$$

To test the robustness of the algorithms, random noise is added to the observation data, and δ represents the magnitude of the noise level. To measure the accuracy of the numerical solution, we compute $error_u$ and $error_\rho$, which are the normalized discrete L^2 norms of the errors. In the case of high noise level, (i.e., $\delta = 10\%$), we smooth the measured data z before the Newton iteration (Cai et al., 2009). In addition to the accuracy issues, we pay close attention to the scalability of the algorithms with respect to the size of the problems and the number of processors. We implement our algorithms using the Portable Extensible Toolkit for Scientific computation (Baley et al., 2009). All programs are run on an IBM Blue/Gene L supercomputer with 1024 nodes.

Test 1. This problem is defined on the domain $\Omega = (0, l_x) \times (0, l_y)$, and the right-hand side f is constructed so that the elliptic coefficient to be identified is

$$\rho = 1 + (-1)^{i+j}100[(x-i)(y-j)(1-(x-i))(1-(y-j))]^2,$$

when $(x, y) \in [i, i+1) \times [j, j+1)$. i and j are integers less than l_x and l_y, respectively.

Test 2. This is a more complicated problem defined on a larger domain $(-4.5, 4.5) \times (-4.5, 4.5)$. The right-hand side f is chosen so that the elliptic coefficient to be identified is

$$\rho = 8 + 3(1-x)^2 e^{-x^2-(y+1)^2} - 10\left(\frac{1}{5}x - x^3 - y^5\right)e^{-x^2-y^2} - \frac{1}{3}e^{-(x+1)^2-y^2}.$$

In Test 1, we first fix the domain to be the unit square and test the problem with different mesh sizes, $1/40$, $1/80$, $1/160$, and $1/320$ to check the accuracy of the algorithms. The error and the number of Newton iterations are shown in Table 1. It is clear that the algorithms converge to the solution for different mesh sizes with different noise levels. The total number of Newton iterations is stable and not sensitive to the noise level.

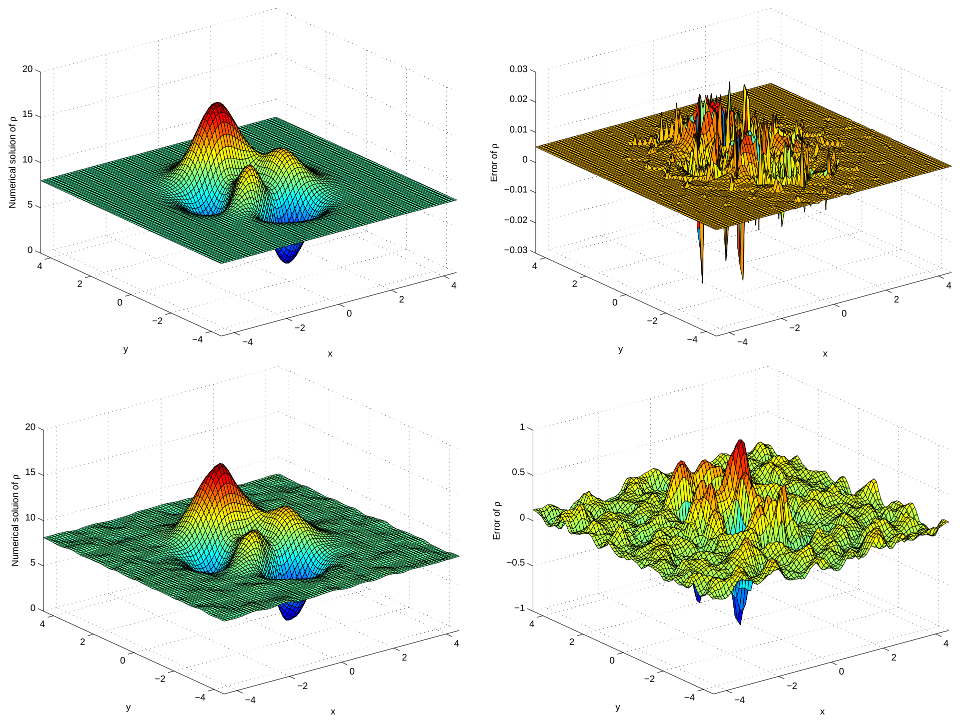

Figure 1. Numerical results of Test 2 on the computational domain $(-4.5, 4.5) \times (-4.5, 4.5)$. These four pictures show the numerical solutions (left two) and errors (right two) with $\delta = 0\%$ (top two) and $\delta = 10\%$ (bottom two).

When the noise level is high, larger β values are generally necessary for Newton to converge. For a given noise level, the results are usually more accurate when a finer mesh is used.

For Test 2, we first show the numerical solutions in Figure 1. The numerical results indicate that our algorithms are capable of solving complicated problems defined on a large domain.

To study the parallel scalability of the algorithms, we consider a rather fine mesh 1830×1830 and we use up to 900 processors in the experiments. The inter subdomain overlap is fixed to be 10.

In Table 2, we show the number of Newton iterations, the average number of GMRES iterations, and the total running time for Test 1 with $l_x = l_y = 2$. The one-level additive Schwarz algorithm and the two-level algorithms all require the same number of Newton iterations to converge. As the number of processors increases, the GMRES iteration of the one-level algorithm increases dramatically. This also results in the divergence of the one-level algorithm in several tests using 400 or 900 processors. However, the GMRES iteration of all two-level algorithms stays at near a constant, which results in great improvements in both the running time and the scalability.

Among the four two-level algorithms, additive Schwarz requires more iterations and running time. The full algorithm takes the smallest number of iterations, and the Kaskade version is the fastest in terms of running time. All two-level algorithms have achieved linear or even superlinear scalability in this test case with $np = 100, 144, 225, 400,$ and 900. The kaskade-type and full-type algorithms usually have better scalability than the multiplicative and additive algorithms. These results are shown in Figure 2.

5. CONCLUSIONS

Several two-level domain decomposition methods were studied for very large nonlinear systems on computers with hundreds of processors. Specifically, we investigated four preconditioners for solving the coupled system arising from the one-shot discretization of inverse elliptic problems. We showed numerically that the two-level approaches outperform the previously introduced one-level algorithm in terms of both iteration numbers and the computing time. Linear and even superlinear scalability were observed for the two-level methods. In addition, we found that the methods are stable even with high-level noise in the data.

Table 1. This table shows the errors and the number of Newton iterations for Test 1 when $l_x = l_y = 1$.

mesh	β and δ	$error_u$	$error_\rho$	$Newton$
41×41	$\beta = 10^{-7}, \delta = 0$	0.000014	0.001261	3
	$\beta = 10^{-5}, \delta = 1\%$	0.000560	0.009130	3
	$\beta = 10^{-4}, \delta = 10\%$	0.005382	0.030639	4
81×81	$\beta = 10^{-7}, \delta = 0$	0.000011	0.001021	3
	$\beta = 10^{-6}, \delta = 1\%$	0.000264	0.008025	3
	$\beta = 10^{-5}, \delta = 10\%$	0.002372	0.022016	4
161×161	$\beta = 10^{-7}, \delta = 0$	0.000011	0.001029	3
	$\beta = 10^{-6}, \delta = 1\%$	0.000167	0.004910	3
	$\beta = 10^{-5}, \delta = 10\%$	0.001184	0.015266	4
321×321	$\beta = 10^{-7}, \delta = 0$	0.000011	0.001034	3
	$\beta = 10^{-6}, \delta = 1\%$	0.000113	0.004465	3
	$\beta = 10^{-5}, \delta = 10\%$	0.000687	0.013618	4
641×641	$\beta = 10^{-7}, \delta = 0$	0.000011	0.001035	3
	$\beta = 10^{-6}, \delta = 1\%$	0.000086	0.003530	3
	$\beta = 10^{-6}, \delta = 10\%$	0.000392	0.010869	4

Table 2. This table shows the running time and average GMRES iteration per Newton iteration of Test 2 in both one-level and two-level algorithms for $\delta = 0\%, 1\%,$ and 10%, with $l_x = l_y = 2$. $\beta = 10^{-6}, 10^{-5}$, and 10^{-5} are employed to solve these three problems. $3, 4,$ and 4 Newton steps are required respectively. "/" means the GMRES iteration is over 500 for that test.

		$\delta = 0\%$		$\delta = 1\%$		$\delta = 10\%$	
np	Type	GMRES	Time(s)	GMRES	Time(s)	GMRES	Time(s)
The results of the one-level method							
100	One-level	109.7	416.2	70.5	441.3	114.0	563.2
144	One-level	153.9	354.4	82.8	318.2	144.5	491.3
225	One-level	183.0	232.7	99.0	203.7	265.8	413.1
400	One-level	283.0	187.7	/	/	/	/
900	One-level	/	/	/	/	/	/
The results of the two-level method							
100	Full	8.0	240.6	5.5	299.0	6.0	302.7
144	Full	8.0	158.5	6.0	198.7	6.0	198.9
225	Full	9.0	91.4	6.5	111.7	6.0	109.7
400	Full	8.7	50.6	6.3	61.5	6.0	60.9
900	Full	9.3	25.3	6.8	30.7	6.0	30.0
100	Kaskade	10.0	219.5	7.3	279.9	7.8	282.0
144	Kaskade	9.7	143.0	7.0	181.4	8.0	184.5
225	Kaskade	10.0	79.1	7.3	99.4	8.0	100.7
400	Kaskade	9.7	43.5	7.3	54.0	8.0	55.4
900	Kaskade	10.3	21.7	7.3	26.5	7.0	27.0
100	Additive	47.7	339.4	36.5	403.7	35.3	398.7
144	Additive	46.3	225.8	35.8	268.5	34.3	264.0
225	Additive	49.3	136.0	37.0	156.6	34.5	152.2
400	Additive	48.3	77.1	34.5	82.6	32.8	84.3
900	Additive	46.3	37.4	33.5	42.0	32.3	40.9
100	Multiplicative	14.3	277.4	10.0	333.8	10.5	337.5
144	Multiplicative	14.7	184.5	9.8	218.9	10.3	221.6
225	Multiplicative	15.7	108.6	10.8	126.1	10.8	126.1
400	Multiplicative	15.7	61.1	10.3	69.2	10.5	69.7
900	Multiplicative	14.7	29.1	9.8	33.3	9.0	32.4

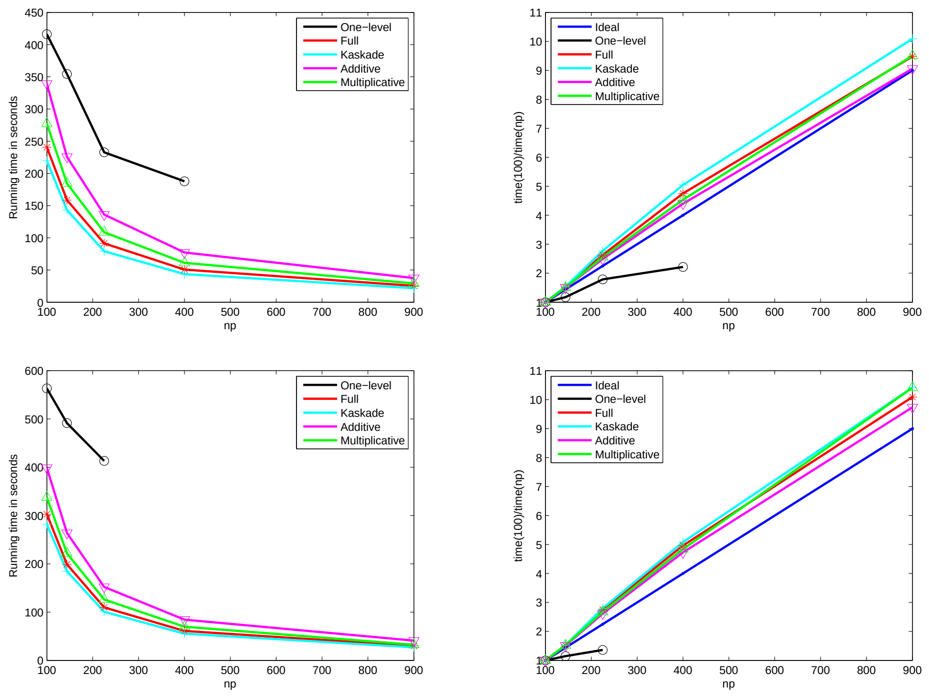

Figure 2. Running time (left two) and speedup curve (right two) as compared to the ideal speedup for Test 1 for $\delta = 0\%$ (top two) and 10% (bottom two). $\circ$, $*$, $+$, Δ, and ∇ in the pictures represent the one-level case, the full type two-level case, the kaskade type two-level case, and the multiplicative type two-level case respectively.

REFERENCES

Barker, A. and Cai, X.-C. (2009), Scalable parallel methods for monolithic coupling in fluid-structure interaction with application to blood flow modeling. *J. Comput. Phys.*, to appear.

Baley, S., Buschelman, K., Eijkhout, V., Gropp, W., Kaushik, D., Knepley, M., McInners, L., Smith, B., and Zhang H. (2009), PETSc Users Manual, *Argonne National Laboratory*.

Briggs, W., Henson, V., and McCormick, S. (2000), A Multigrid Tutorial, *SIAM*, Philadelphia.

Cai, X.-C., Gropp, W., Keyes, D., Melvin, R., and Young, D. (1998), Parallel Newton-Krylov-Schwarz algorithms for the transonic full potential equation, *SIAM J. Sci. Comput.*, 19, 246-265.

Cai, X.-C., Liu, S., and Zou, J. (2009), Parallel fully coupled algorithms for inverse elliptic problems, *Communications in Applied Mathematics and Computational Science*, to appear.

Dennis, J. and Schnabel, R. (1996), Numerical Methods for Unconstrained Optimization and Nonlinear Equations, *SIAM*, Philadelphia.

Saad, Y. (1996), Iterative Methods for Sparse Linear Systems, *PWS Publishing Company*, Boston.

Toselli, A. and Widlund, O. (2005), Domain Decomposition Methods – Algorithms and Theory, *Springer*, Berlin.

Yang, C., Cao, J., and Cai, X.-C. (2009), A fully implicit domain decomposition algorithm for shallow water equations on the cubed-sphere, *SIAM J. Sci. Comput.*, to appear.