Kangweon-Kyungki Math. Jour. **13** (2005), No. 1, pp. 113–122

ITERATIVE ALGORITHMS AND DOMAIN DECOMPOSITION METHODS IN PARTIAL DIFFERENTIAL EQUATIONS

Jun Yull Lee

Abstract. We consider the iterative schemes for the large sparse linear system to solve partial differential equations. Using spectral radius of iteration matrices, the optimal relaxation parameters and good parameters can be obtained. With those parameters we compare the effectiveness of the SOR and SSOR algorithms. Applying Crank-Nicolson approximation, we observe the error distribution according to domain decomposition. The number of processors due to domain decomposition affects time and error. Numerical experiments show that effectiveness of SOR and SSOR can be reversed as time size varies, which is not the usual case. Finally, these phenomena suggest conjectures about equilibrium time grid for SOR and SSOR.

1. Introduction

When one solves the system of linear equations $Ax = b$, where A is a nonsingular matrix and b is a given vector, it seems natural to take multiples of b as the approximation to the solution:

$$x_1 \in \text{Span}\{b\},\ x_2 \in \text{Span}\{b, Ab\}, \cdots,\ x_k \in \text{Span}\{b, Ab, \cdots, A^{(k-1)}b\}$$

where $k = 1, 2, 3, \cdots$. If it turns out that the space does not contain a good approximate solution or if such an approximate solution cannot be computed easily, then one might use a preconditioner M and effectively solve the modified problem $M^{-1}Ax = M^{-1}b$, say $x \approx M^{-1}b$, by

Received January 30, 2005.

2000 Mathematics Subject Classification: 65F10, 65M06, 65D25.

Key words and phrases: SOR, SSOR, preconditioning, Crank-Nicolson, domain decomposition.

This research is accomplished with Research Fund provided by Kangwon National University, Support for 2003 Faculty Research Abroad.

generating approximate solutions $x_1, x_2, x_3, \cdots$ satisfying

$$x_k \in \text{Span}\{M^{-1}b, (M^{-1}A)M^{-1}b, \cdots, (M^{-1}A)^{(k-1)}M^{-1}b\}.$$

Here the preconditioner M must be chosen so that such linear systems are much easier to solve than the original problem. If different parts of a system problem could be solved independently and then the results somehow pieced together to the entire problem, then a loosely coupled array of parallel processors could be used for the task. This may provide a faster and more parallel solution method than applying a standard iterative method directly to the large problem. This solution approach is equivalent to using a preconditioner that involves solving on subdomains. Domain decomposition methods fall roughly into two classes - those using overlapping domains and those using non-overlapping domains. In this research, we describe the preconditioners M arising from SOR and SSOR methods, and compare the effectiveness according to run-time and numbers of iterations. Then domain decompositions will be treated and applied to model problems for error analysis. Then we analyze its convergence performance according as grid sizes vary.

2. Basic Iterative Algorithms

Let us consider the following partial differential equation:

$$\frac{\partial u}{\partial t} = a(x,y)\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + b(x,y)\frac{\partial u}{\partial x} + c(x,y)\frac{\partial u}{\partial y} + d(x,y)u + f(x,y,t)$$

where $u(x,y,t)$ is defined on $[0,1]\times[0,1]\times[0,1]$. Using Crank-Nicolson approximation and five point formula for U_C^n, the center approximate at n time level, the above equation is written as follows.

$$\begin{aligned}\frac{U_C^{n+1} - U_C^n}{dt} &= a_C \frac{U_E^{n+1/2} - 2U_C^{n+1/2} + U_W^{n+1/2}}{dx^2} \\ &+ a_C \frac{U_N^{n+1/2} - 2U_C^{n+1/2} + U_S^{n+1/2}}{dy^2} \\ &+ b_C \frac{U_E^{n+1/2} - U_W^{n+1/2}}{2dx} + c_C \frac{U_N^{n+1/2} - U_S^{n+1/2}}{2dy} \\ &+ d_C U_C^{n+1/2} + f_C^{n+1/2}.\end{aligned}$$

Here a_C means $a(x, y)$ at the centered point C and the other notations are as such. Then we simplify the linear system:

$$\begin{aligned} &\lambda_C U_C^{n+1} + \lambda_E U_E^{n+1} + \lambda_N U_N^{n+1} + \lambda_W U_W^{n+1} + \lambda_S U_S^{n+1} \\ = \quad &(2 - \lambda_C) U_C^n - \lambda_E U_E^n - \lambda_N U_N^n - \lambda_W U_W^n - \lambda_S U_S^n + \tfrac{1}{2}(f_C^{n+1} + f_C^n) dt. \end{aligned}$$

Letting $\alpha = a_C(\frac{dt}{dx^2} + \frac{dt}{dy^2})$, the λ's are as follows:

$$\begin{aligned} \lambda_C &= 1 + \alpha - \frac{1}{2} d_C dt, \\ \lambda_E &= -\frac{1}{2} a_C \frac{dt}{dx^2} - \frac{1}{4} b_C \frac{dt}{dx}, \\ \lambda_N &= -\frac{1}{2} a_C \frac{dt}{dy^2} - \frac{1}{4} c_C \frac{dt}{dy}, \\ \lambda_W &= -\frac{1}{2} a_C \frac{dt}{dx^2} + \frac{1}{4} b_C \frac{dt}{dx}, \\ \lambda_S &= -\frac{1}{2} a_C \frac{dt}{dy^2} + \frac{1}{4} c_C \frac{dt}{dy}. \end{aligned}$$

Therefore, we have a linear system

$$AU^{(n+1)} = BU^{(n)} + f^{(n+1/2)} + \text{boundary condition},$$

which is of the form $Au = b$, where $U^{(n)}$, $f^{(n+1/2)}$ and the boundary condition are known, and $U^{(n+1)}$ is unknown.

Now let us consider how to solve the linear system of the form $Au = b$. We employ an iterative scheme for the system:

$$\begin{aligned} u^{(n+1)} &= Gu^{(n)} + k \\ &= (I - Q^{-1}A)u^{(n)} + Q^{-1}b \\ &= u^{(n)} + Q^{-1}(b - Au^{(n)}). \end{aligned}$$

Here, $G = I - Q^{-1}A$, $k = Q^{-1}b$ for some nonsingular preconditioning matrix Q. Note that the last expression should be used to implement the iterative algorithms. It helps coding and computation. Preconditioning matrix Q is chosen to be a simply easily invertible matrix, such as a diagonal, tridiagonal, lower triangular, upper triangular, or a product of such matrices, and is usually chosen so that $Q^{-1}A$ has a better condition number than A. Now the matrix A in the system will be symmetric and positive definite(SPD) in this paper. Let A be written as

$$A = D - C_L - C_U,$$

where D or D_A is a diagonal matrix with the same diagonal elements as A. C_L and C_U are strictly lower and strictly upper tridiagonal matrices, respectively. Let us denote the preconditioning matrices Q_J, Q_{GS}, Q_{SOR}, Q_{SSOR} for the Jacobi, Gauss-Seidel, successive over-relaxation (SOR), and symmetric SOR(SSOR) method, respectively. Then they are defined as follows:

$$\begin{aligned} Q_J^{-1} &= D^{-1}, \\ Q_{GS}^{-1} &= (D - C_L)^{-1}, \\ Q_{SOR}^{-1} &= (\frac{1}{\omega}D - C_L)^{-1}, \\ Q_{SSOR}^{-1} &= \left(\frac{\omega}{2-\omega}(\frac{1}{\omega}D - C_L)D^{-1}(\frac{1}{\omega}D - C_U)\right)^{-1}. \end{aligned}$$

The iterative matrices for such preconditioning matrices are given below.

$$\begin{aligned} G_J &= I - D^{-1}A, \\ G_{GS} &= I - (D - C_L)^{-1}A, \\ \mathscr{L}_\omega = G_{SOR} &= I - (\frac{1}{\omega}D - C_L)^{-1}A, \\ \mathscr{S}_\omega = G_{SSOR} &= I - \left((\frac{1}{\omega}D - C_L)(\frac{2-\omega}{\omega}D)^{-1}(\frac{1}{\omega}D - C_U)\right)^{-1}A. \end{aligned}$$

In the above, parameter ω is a relaxation factor.

3. Crank-Nicolson Approximation Method

Let us consider the elliptic equation

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0.$$

When Crank-Nicolson approximation is adapted, the above five point formula produces coefficient matrix H and the right$-$hand side matrix B_H, say $Hu = B_H$. For this $B_H = I - D_H^{-1}H$, if Jacobi method is applied, then the spectral radius $\rho(G_J^{B_H})$ is $\cos(\pi h), h = dx = dy$, without domain decomposition. If we decompose the domain into two stripe partitions, then the spectral radius $\rho(G_J^{B_H})$ is $1/2(\cos(\pi h) + \cos(2\pi h)[\mathbf{?}]$.

THEOREM 3.1. *For the SOR iterative matrix $\mathscr{L}_\omega$ of the linear system in solving the elliptic equation, the optimum value ω_{opt} of parameter ω is given by*

$$\omega_{opt} = \frac{2}{1+\sqrt{1-\rho(G_J^B)^2}} = \frac{2}{1+\sin\pi h}.$$

Then

$$\rho(\mathscr{L}_\omega) = \frac{1-\sqrt{1-\rho(G_J^B)^2}}{1+\sqrt{1-\rho(G_J^B)^2}} = \frac{1-\sin\pi h}{1+\sin\pi\omega}.$$

Proof. See [**?**]. □

LEMMA 3.2. *If λ is an eigenvalue of a matrix M, then $\frac{2\alpha}{(1+2\alpha)}\lambda$ is an eigenvalue of $\frac{2\alpha}{(1+2\alpha)}M$.*

Proof. Since $Mx = \lambda x$, $\frac{2\alpha}{(1+2\alpha)}Mx = \frac{2\alpha}{(1+2\alpha)}\lambda x$. □

THEOREM 3.3. *If we have the linear system $Au = B_A$ without a decomposition of the domain from the parabolic equation $\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$, then the spectral radius $\rho(B_A)$ of B_A is*

$$\rho(G_J^A) = \frac{2\alpha\cos(\pi h)}{1+2\alpha}$$

where $\alpha = dt/h^2$, $h = dx = dt/dy$.

Proof. From the equation $\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$, we have the following approximation formula.

$$\begin{aligned}\frac{U_C^{n+1}-U_C^n}{dt} &= \frac{U_E^{n+1/2}-2U_C^{n+1/2}+U_W^{n+1/2}}{dx^2} + \frac{U_N^{n+1/2}-2U_C^{n+1/2}+U_S^{n+1/2}}{dy^2} \\ &= \frac{1}{2}\left[\frac{U_E^{n+1}-2U_C^{n+1}+U_W^{n+1}}{dx^2} + \frac{U_E^n-2U_C^n+U_W^n}{dx^2}\right] \\ &\quad + \frac{1}{2}\left[\frac{U_N^{n+1}-2U_C^{n+1}+U_S^{n+1}}{dy^2} + \frac{U_N^n-2U_C^n+U_S^n}{dy^2}\right].\end{aligned}$$

Now this formula gives a linear system below.

$$\begin{aligned}&(1+2\alpha)U_C^{n+1} - \tfrac{1}{2}\alpha U_E^{n+1} - \tfrac{1}{2}\alpha U_N^{n+1} - \tfrac{1}{2}\alpha U_W^{n+1} - \tfrac{1}{2}\alpha U_S^{n+1} \\ &\quad = (1-2\alpha)U_C^n + \tfrac{1}{2}\alpha U_E^n + \tfrac{1}{2}\alpha U_N^n + \tfrac{1}{2}\alpha U_W^n + \tfrac{1}{2}\alpha U_S^n.\end{aligned}$$

Thus we have the coefficient matrix

$$A = \begin{pmatrix} 1+2\alpha & -\frac{1}{2}\alpha & 0 & -\frac{1}{2}\alpha & 0 & 0 & \dots & 0 & 0 \\ -\frac{1}{2}\alpha & 1+2\alpha & -\frac{1}{2}\alpha & 0 & -\frac{1}{2}\alpha & 0 & \dots & 0 & 0 \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ -\frac{1}{2}\alpha & 0 & 0 & 1+2\alpha & -\frac{1}{2}\alpha & 0 & -\frac{1}{2}\alpha & 0 & 0 \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ 0 & 0 & 0 & \dots & \dots & -\frac{1}{2}\alpha & 0 & -\frac{1}{2}\alpha & 1+2\alpha \end{pmatrix}$$

$$= I + \frac{1}{2}\alpha \begin{pmatrix} 4 & -1 & 0 & -1 & 0 \\ -1 & 4 & \dots & 0 & -1 \\ \ddots & \ddots & \ddots & \ddots & \ddots \\ -1 & \dots & -1 & \dots & -1 \\ \ddots & \ddots & \ddots & \ddots & \ddots \\ 0 & \dots & \dots & -1 & 4 \end{pmatrix}.$$

We let $A = I + \frac{1}{2}\alpha B$. For the system $Bu = b'$, the iterative matrix for the Jacobi Method is

$$G_J^B = I - D_B^{-1}B = I - \frac{1}{4}B.$$

The Jacobi iterative matrix for the system $Ax = b$ is, then,

$$\begin{aligned} G_J^A &= I - D_A^{-1}A = I - \frac{1}{1+2\alpha}A = I - \frac{1}{1+2\alpha}(I + 1/2\alpha B) \\ &= (1 - \frac{1}{1+2\alpha})I - \frac{\alpha}{2(1+2\alpha)}B = \frac{2\alpha}{1+2\alpha}I - \frac{2\alpha}{(1+2\alpha)}(I - G_J^B) \\ &= \frac{2\alpha}{1+2\alpha}G_J^B. \end{aligned}$$

Therefore,

$$\rho(G_J^A) = \frac{2\alpha}{1+2\alpha}\cos(\pi h).$$

□

From the $SSOR$ iterative matrix G_{SSOR}, the eigenvalue of G_{SSOR} are real and nonnegative. It is known that the rate of convergence of the SSOR method is relatively insensitive to the exact choice of ω so that a precise optimum value of ω is not crucial. If the spectral radius of the matrix $C_L C_U$ satisfies $S(C_L C_U) \leq \frac{1}{4}$, which is true in our model case,

then a good value of ω is given by $\omega = \frac{2}{1+\sqrt{2(1-\rho(G_J^B))}}$. Also with the choice of ω, the spectral radius of $\rho(\mathscr{S}_\omega)$ satisfies

$$\rho(\mathscr{S}_\omega) \leq \left(1 - \sqrt{\frac{1-\rho(G_J^B)}{2}}\right) / \left(1 + \sqrt{\frac{1-\rho(G_J^B)}{2}}\right).$$

4. Domain Decomposition

Now we describe domain decomposition scheme. Stripwise partitions of the domain yield rather unsophisticated as well as efficient data structure and coding. Explicit interface approximation is applied. Then the system we considered has the following interface points formula. Here M,j, n mean the x-interface level, the y-interface level, and the time level, respectively.

$$\begin{aligned} U_{M,j}^{n+1} = a(x_M, y_j)\Big[&\frac{U_{M+1,j}^n - 2U_{M,j}^n + U_{M-1,j}^n}{dx^2} \\ &+ \frac{U_{M,j+1}^n - 2U_{M,j}^n + U_{M,j-1}^n}{dy^2}\Big]dt \\ &+ b(x_M, y_j)\frac{U_{M+1,j}^n - U_{M-1,j}^n}{2dx}dt \\ &+ c(x_M, y_j)\frac{U_{M,j+1}^n - U_{M,j-1}^n}{2dy}dt \\ &+ (1 + d(x_M, y_j)dt)U_{M,j}^n + f(x_M, y_j)dt. \end{aligned}$$

Then, using Crank-Nicolson approximation, we predict interior points using iterative methods discussed in section 2. After that, we solve the linear system

$$\lambda_C^{'} U_{M,j}^{n+1} + \lambda_E^{'} U_{M,j-1}^{n+1} + \lambda_W^{'} U_{M,j+1}^{n+1} = U_{M,j}^n - \lambda_S^{'} U_{M-1,j}^n - \lambda_N^{'} U_{M+1,j}^n$$

to correct interface values implicitly. For example, we can use Crout's method to correct interface values. Usually, Crank-Nicolson approximation gives more accurate result than forward difference formula in interface predicting, interior approximating, and correcting phases.

5. Numerical Experiments

In the stripwise decomposition of domain, the plane is decomposed in regular strips which are then assigned to individual processors. Again our model problem is of the form

$$\frac{\partial u}{\partial t} = \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + b(x,y)\frac{\partial u}{\partial x} + c(x,y)\frac{\partial u}{\partial y} + d(x,y)u,$$

where initial values and boundary values take exact values. We know that the exact solution is $u = e^{-t}\sin x \cos y$ if $b(x,y) = \sin x \sin y$, $c(x,y) = \cos x \cos y$, and $d(x,y) = 1$.

Different methods were programmed to approximate the solution of the model problem. First, we explicitly predict interface point values. Then, we solve linear systems using iterative methods. Later implicit correction were employed for those interface points. Crank-Nicolson approximation is very useful in finite difference formula.

5.1. Domain Decomposition and Error Distribution. The following table shows the performance of convergence and stability of the SOR algorithm when several processors were considered in the elliptic model problem, $b(x,y) = c(x,y) = d(x,y) = 0$.

# of processors	1	2	4	10	20
Time size	cpu time	cpu time	cpu time	cpu time	cpu time
0.1	55.72012	24.03456	4.917070	3314766	2.894161
0.05	72.37407	35.83152	8.452153	6.379173	5.588035
0.02	86.91498	51.02337	16.60388	15.26195	13.30914
0.01	93.51447	58.37394	32.11618	29.47238	25.60682

We can see the error distributions at last time level for 2-, 4-, 10-processors upon domain decomposition, respectively. SOR iteration methods were used to solve the five-diagonal linear system of equations for the case, $b(x,y) = \sin x \sin y$, $c(x,y) = \cos x \cos y$, and $d(x,y) = 1$. It shows the errors at $y = 0.5$ and at the last time level.

5.2. Speedup and Efficiency. We can speedup using more processors as before. Also we can adapt more efficient algorithms such as SOR and SSOR with appropriate parameters. Next table shows the optimal parameters for SOR and good parameters for SSOR with their spectral

radii with the number of iterations to converge when $b(x,y) = c(x,y) = d(x,y) = 0$.

dt=0.01, $h = dx = dy$					
h		0.25	0.1	0.05	0.01
α		0.16	1	4	100
$\rho(G_A^J)$		0.17142	0.634038	0.877945	0.994534
SOR	$\rho(\mathscr{L}_\omega)$	0.079605	0.365	0.16379	0.904
	ω_{opt}	1.00746(7)	1.12784(18)	1.35248(36)	1.81092(184)
SSOR	$\rho(\mathscr{S}_\omega)$	0.04879	0.172455	0.38253	0.81996
	ω_b	0.874391(8)	1.07786(14)	1.33862(24)	1.81068(116)

The number in ( ) means the number of iterations to converge.

5.3. Time mesh size and Effectiveness of SOR and SSOR. As time grid size varies, for fixed $h = dx = dy$, the run-time effectiveness for SOR and SSOR methods were tested. In general, it is believed that SSOR is slower than SOR method. But the following table with $b(x,y) = \sin x \sin y$, $c(x,y) = \cos x \cos y$, and $d(x,y) = 1$ shows that the effectiveness changes depend on time size.

h=1/50			h=1/200			h=1/500		
dt	SOR	SSOR	dt	SOR	SSOR	dt	SOR	SSOR
.0297	.1803	.2103	.0340	15.9630	16.5438	.0400	323.7255	368.2896
.0298	.1803	.1903	.0320	15.9630	16.0030	.0350	323.7255	368.2896
.0299	.1702	.1702	.0315	15.7727	15.7727	.0340	312.6295	312.4993
.0300	.1803	.1702	.0310	15.7727	15.6124	.0330	314.7826	308.5236
.0301	.1803	.1702	.0305	15.7727	15.4823	.0310	314.1417	297.5479

From the above tables we can conjecture as follows:

CONJECTURE 5.1. *For a fixed h, there is a time size dt_b such that (i) if $dt > dt_b$, then the work on SOR is less than the work on SSOR, and (ii) if $dt < dt_b$, then the work on SOR is greater than the work on SSOR*

CONJECTURE 5.2. *For a fixed h, there is a time size dt_b such that if $dt < dt_b$, then we can have $\rho(\mathscr{L}_\omega) > \rho(\mathscr{S}_\omega)$.*

References

[1] J. L. Buchanan and P. R. Turner, *Numerical Methods and Analysis*, McGraw–Hill, Singapore, (1992).

[2] A. Greenbaum, *Iterative Methods for Solving Linear Systems*, SIAM, Philadelphia, (1997).

[3] L. A. Hageman and D. M. Young, *Applied Iterative Methods*, Academic Press, N.Y., (1981).
[4] T. Mai and X. Chen, *Empirical Study of Domain Decomposition Algorithm for Solving Parabolic Partial Differential Equations*, Preprint
[5] T. Mai, X. Chen and D. Harpern, *SJSOR Additive Iterative Methods for Solving Linear Systems*, Proceedings of the 4th IMACS International Symposium on Iterative Methods, (1999), 77–84.
[6] D. E. Womble, *A Time–Stepping Algorithm for Parallel Computers*, SIAM J. Sci. Stat. Comput., Vol **11** (5) (1990), 824–837.
[7] D. M. Young, *Iterative Solution of Large Linear Systems*, Acamedic Press, N.Y., (1971).
[8] D. M. Young and T. Mai, *Iterative Algoritms and Software for Solving Large Sparse Linear Systems*, Communications in Applied Numerical Methods,**4** (1988), 435–456.
[9] Y. Saad, *Iterative Methods for Sparse linear systems*, PWS publishing Co., Boston, (1996).

Department of Mathematics Education
Kangwon National University
Chunchon 200-701, Korea
E-mail: jylee@kangwon.ac.kr