Journal of Computational Physics **179,** 170–179 (2002)
doi:10.1006/jcph.2002.7049

Multigrid Method and Fourth-Order Compact Scheme for 2D Poisson Equation with Unequal Mesh-Size Discretization[1]

Jun Zhang[2]

Laboratory for High Performance Scientific Computing and Computer Simulation, Department of Computer Science, University of Kentucky, 773 Anderson Hall, Lexington, Kentucky 40506-0046
E-mail: jzhang@cs.uky.edu

Received March 9, 2001; revised January 25, 2002

A fourth-order compact difference scheme with unequal mesh sizes in different coordinate directions is employed to discretize a two-dimensional Poisson equation in a rectangular domain. Multigrid methods using a partial semicoarsening strategy and line Gauss–Seidel relaxation are designed to solve the resulting sparse linear systems. Numerical experiments are conducted to test the accuracy of the fourth-order compact difference scheme and to compare it with the standard second-order difference scheme. Convergence behavior of the partial semicoarsening and line Gauss–Seidel relaxation multigrid methods is examined experimentally.

Key Words: Poisson equation; fourth-order compact scheme; unequal mesh size; multigrid method.

1. INTRODUCTION

We seek high-accuracy numerical solution of the two-dimensional (2D) Poisson equation

$$u_{xx}(x, y) + u_{yy}(x, y) = f(x, y), \quad (x, y) \in \Omega, \tag{1}$$

where Ω is a rectangular domain, or a union of rectangular domains, with suitable boundary conditions defined on $\partial\Omega$. The solution $u(x, y)$ and the forcing function $f(x, y)$ are assumed to be sufficiently smooth and to have the required continuous partial derivatives. For convenience, we consider a rectangular domain $\Omega = [0, L_x] \times [0, L_y]$. Here the subscripts are not derivatives. We discretize Ω with uniform mesh sizes $\triangle x$ and $\triangle y$ in the x

[1] This research was supported by the U.S. National Science Foundation under Grants CCR-9902022, CCR-9988165, and CCR-0092532.

[2] URL: http://www.cs.uky.edu/~jzhang.

0021-9991/02 $35.00

and y coordinate directions. $N_x = L_x/\triangle x$ and $N_y = L_y/\triangle y$ are the numbers of uniform intervals along the x and y coordinate directions, respectively. The mesh points are (x_i, y_j), with $x_i = i\triangle x$ and $y_j = j\triangle y$, $0 \leq i \leq N_x, 0 \leq j \leq N_y$. In the sequel, we may also use the index pair (i, j) to represent the mesh point (x_i, y_j).

The standard second-order central-difference operators defined at (i, j) are

$$\delta_x^2 u_{i,j} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\triangle x^2}, \quad \delta_y^2 u_{i,j} = \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\triangle y^2}.$$

Equation (1) can be discretized at a given grid point (x_i, y_j) as

$$\delta_x^2 u_{i,j} + \delta_y^2 u_{i,j} = f_{i,j} + O(\triangle^2), \tag{2}$$

where $O(\triangle^2)$ denotes the truncated terms on the order of $O(\triangle x^2 + \triangle y^2)$.

Although uniform finite-difference discretization using equal mesh size in both the x and y directions is simple to implement, there are certain situations in which the use of unequal mesh sizes in different coordinate directions is more cost effective. In those situations, the physical quantity modeled may have uneven distribution in different directions. A typical example is to model the temperature distribution in a thin rod, in which the physical domain in one direction is much longer than that in the other direction. Another example is in microscale heat-transfer modeling problems, in which the thickness of one dimension is at the microscale, but those of the other dimensions are not [3, 13].

The efforts to compute a more accurate solution using limited grid sizes have directed researchers' attention to developing high-order compact finite-difference discretization schemes. In the context of fourth-order compact finite-difference discretizations, much research and many applications have been focused on equal-mesh-size discretizations [8, 11]. This article discusses the use of fourth-order compact unequal-mesh-size discretizations for solving 2D Poisson equations and the design of specialized multigrid methods for such applications.

2. FOURTH-ORDER COMPACT APPROXIMATIONS

The fourth-order compact approximation of a 1D Poisson equation can be written as [8]

$$\delta_x^2 u_i = \left(1 + \frac{\triangle x^2}{12}\delta_x^2\right) f_i + O(\triangle x^4). \tag{3}$$

This fourth-order compact approximation is different from the second-order approximation (2) only in the approximation of the right-hand-side function f. Equation (3) can be formulated symbolically as [9]

$$\left(1 + \frac{\triangle x^2}{12}\delta_x^2\right)^{-1} \delta_x^2 u_i = f_i + O(\triangle x^4). \tag{4}$$

Here the operator $(1 + \frac{\triangle x^2}{12}\delta_x^2)^{-1}$ has symbolic meaning only. In application, the fourth-order compact difference scheme is given by Eq. (3), not by Eq. (4). An analogous symbolic fourth-order compact approximation operator can be obtained for the y variable.

We can apply the symbolic fourth-order compact approximation operators to the second derivatives u_{xx} and u_{yy} in Eq. (1), respectively. This yields symbolically [9]

$$\left(1+\frac{\triangle x^2}{12}\delta_x^2\right)^{-1}\delta_x^2 u_{i,j}+\left(1+\frac{\triangle y^2}{12}\delta_y^2\right)^{-1}\delta_y^2 u_{i,j}=f_{i,j}+O(\triangle^4),\tag{5}$$

where $O(\triangle^4)$ denotes the truncated terms on the order of $O(\triangle x^4+\triangle y^4)$. Applying the symbolic operators, we have

$$\begin{aligned}\left(1+\frac{\triangle y^2}{12}\delta_y^2\right)\delta_x^2 u_{i,j}+\left(1+\frac{\triangle x^2}{12}\delta_x^2\right)\delta_y^2 u_{i,j}&=\left(1+\frac{\triangle x^2}{12}\delta_x^2\right)\left(1+\frac{\triangle y^2}{12}\delta_y^2\right)f_{i,j}+O(\triangle^4)\\&=\left[1+\frac{1}{12}\left(\triangle x^2\delta_x^2+\triangle y^2\delta_y^2\right)\right]f_{i,j}+O(\triangle^4).\end{aligned}$$

Here we absorbed the $O(\triangle x^2\cdot\triangle y^2)$ term into the $O(\triangle^4)$ term. After some rearrangement and dropping the $O(\triangle^4)$ term, the general fourth-order compact approximation scheme for a 2D Poisson equation is given by

$$\left(\delta_x^2+\delta_y^2\right)u_{i,j}+\frac{1}{12}(\triangle x^2+\triangle y^2)\delta_x^2\delta_y^2 u_{i,j}=f_{i,j}+\frac{1}{12}\left(\triangle x^2\delta_x^2+\triangle y^2\delta_y^2\right)f_{i,j}+O(\triangle^4).\tag{6}$$

In the special case with $\triangle x=\triangle y$, the approximation formula (6) is the same as for *Mehrstellen* [2]. Various multigrid implementation strategies with *Mehrstellen* are discussed and compared with the standard second-order central-difference scheme in [6].

In the general case, let us denote the mesh aspect ratio $\gamma=\triangle x/\triangle y$. We have from Eq. (6) the general fourth-order compact discretization scheme, with a truncation error on the order of $O(\triangle^4)$:

$$\begin{aligned}(1+\gamma^2)&(u_{i+1,j+1}+u_{i+1,j-1}+u_{i-1,j+1}+u_{i-1,j-1})/2+(5\gamma^2-1)(u_{i,j+1}+u_{i,j-1})\\&+(5-\gamma^2)(u_{i+1,j}+u_{i-1,j})-10(1+\gamma^2)u_{i,j}\\&=\triangle x^2(8f_{i,j}+f_{i+1,j}+f_{i-1,j}+f_{i,j+1}+f_{i,j-1})/2.\end{aligned}\tag{7}$$

The sparse linear system formed by equations of the form (7) at all interior grid points has nine nonzero diagonals.

We remark that exactly the same discrete equation can be obtained by using equal-mesh-size discretization on a transformed equation. The mathematical formulation, multigrid implementations, and numerical experiments with fourth-order compact difference schemes for more general nonuniform grids are discussed in [5]. A mapping $\bar{y}=\gamma y$ changes the y direction domain from $[0, L_y]$ to $[0, \gamma L_y]$ and Eq. (1) to an anisotropic Poisson equation $u_{xx}(x,\bar{y})+\gamma^2 u_{\bar{y}\bar{y}}(x,\bar{y})=f(x,\bar{y})$, on which the uniform equal-mesh-size fourth-order compact discretization scheme can be applied. However, direct discretization using unequal mesh sizes may be more convenient in certain situations to maintain the shape of the physical domain. A typical situation is in local mesh refinement with patched grids.

3. SPECIALIZED MULTIGRID METHODS

Both the fourth-order compact difference scheme and the second-order central-difference scheme result in sparse linear systems which can be solved efficiently by multigrid methods [10]. The multigrid method utilizes some relaxation method to remove high-frequency errors and makes use of coarse grid correction to remove smooth errors. For solving 2D and 3D Poisson equations discretized by the standard fourth-order compact difference schemes (with equal mesh size), efficient multigrid methods are implemented in [6, 12]. It is shown that the fourth-order compact difference schemes are more cost effective than the corresponding second-order central-difference scheme with the multigrid methods.

However, for solving an anisotropic Poisson equation, or, equivalently, a Poisson equation discretized with unequal mesh sizes, a standard multigrid method with a point Gauss–Seidel-type relaxation and standard mesh coarsening strategy (the coarse grid mesh sizes double that of the fine grid) does not work very well [10]. There are at least two strategies to treat linear systems arising from discretized anisotropic equations. The first strategy is to use line relaxation to replace point relaxation; e.g., line Gauss–Seidel relaxation can be shown to be very effective in removing high-frequency errors in the dominant direction with large coefficients [10]. Each line relaxation requires a forward solution followed by a back substitution. The second strategy to deal with anisotropy is to use semicoarsening; i.e., mesh coarsening is only performed along the dominant direction. The mesh size along the other direction is not coarsened [7].

For the particular problem considered in this paper, the dominant direction is always either the x or the y direction, but not both. So either a single-line relaxation or a semicoarsening along one direction will suffice. To simplify our discussion without loss of generality, we assume that $\triangle x \leq \triangle y$ with $\gamma \leq 1$. We also assume $\Omega = [0, 1] \times [0, 1]$. For efficient implementation of multigrid methods, we further assume that $N_x = 2^{n_x}$, $N_y = 2^{n_y}$ for some positive integers $n_x > 1$ and $n_y > 1$. By our assumptions, it is easy to see that $n_x \geq n_y$ and $\gamma = 2^{n_y - n_x}$.

3.1. *Line Relaxation*

Since x is the dominant direction, we only perform line relaxation along the x direction on each successive grid. The successive grids are coarsened in both the x and y directions. We point out that with our assumption, $N_x > N_y$ holds for all grids. On the coarsest grid, $N_y = 2$ and there is a 1D linear system with $(N_x - 1)$ unknowns along the x direction. The line relaxation is a direct solver on this coarsest grid. Since the standard coarsening strategy maintains the mesh aspect ratio γ on all grids, the discrete equations on all grids are the same.

Let us assume that the grid points are ordered lexicographically, i.e., first from left to right along the x direction, then from bottom to top along the y direction. The coefficient matrix of the fourth-order compact difference scheme with this ordering can be written as a block tri-diagonal matrix of block order $(N_y - 1)$ (the order of the coefficient matrix A is $(N_x - 1) \times (N_y - 1)$, where $A = \text{diag}[A_1, A_0, A_1]$, and where $A_0 = \text{diag}[5 - \gamma^2, -10(1 + \gamma^2), 5 - \gamma^2]$, $A_1 = \text{diag}[(1 + \gamma^2)/2, 5\gamma^2 - 1, (1 + \gamma^2)/2]$ are symmetric tridiagonal submatrices of the order $(N_x - 1)$). They represent the submatrix of each grid line along the x direction. Thus a line Gauss–Seidel relaxation is carried out by solving $A_0 \mathbf{u}_j = \mathbf{f}_j - A_1(\mathbf{u}_{j-1} + \mathbf{u}_{j+1})$ for each line $j = 1, 2, \ldots, (N_y - 1)$. Here $\mathbf{u}_j$ is part of the solution vector representing the grid points on the jth line, and $\mathbf{f}_j$ is the corresponding part of the right-hand-side vector. A_0 needs only to be factored once on each level, since the matrix A has constant blocks which do not change from one grid line to another. Hence the factorization cost of A_0 is negligible.

In multigrid method with line Gauss–Seidel relaxations, we use standard bilinear interpolation to transfer corrections from a coarse grid to a fine grid, and we use a full-weighting scheme to project residual from a fine grid to a coarse grid [6].

3.2. *Partial Semicoarsening*

In the implementation of semicoarsening strategy, every other grid line along the x direction is eliminated from a fine grid to a coarse grid. Since the mesh aspect ratio γ is related to the relative values of $\triangle x$ and $\triangle y$, we have different discrete equations on different grids with different γ (see Eq. (7)). Thus, inevitably, there will be a coarse grid on which $N_x = N_y$. On this coarse grid, the discrete equation is actually the standard Poisson equation (with $\gamma = 1$). Starting from this grid, the following coarsening strategy will be standard full coarsening; i.e., every other grid lines in both the x and y directions is eliminated. So we call this specialized coarsening strategy *partial semicoarsening strategy*. As in standard multigrid method, the coarsest grid will have only one unknown.

For relaxation methods (smoothers), we use a point Gauss–Seidel relaxation in lexicographical ordering, in red–black ordering, and in four-color ordering. We point out that the red–black ordering with the fourth-order compact difference scheme does not decouple the grid points completely, so there is no inherent parallelism drawn from this implementation. However, red–black Gauss–Seidel relaxation is shown to have a better smoothing effect than lexicographical Gauss–Seidel relaxation [6]. The four-color Gauss–Seidel relaxation does decouple the grid points into four subgroups; each subgroup with a different color can be updated independently. This implementation embodies inherent parallelism to Gauss–Seidel relaxation [1]. Experimental results also show that four-color Gauss–Seidel relaxation has a better smoothing effect than the lexicographical Gauss–Seidel relaxation [6].

Mulder [7] argues that since point relaxation has no effect on the weak direction (here the y direction), the residual (error) components along the y direction may not be smoothed. If the initial residual contains high-frequency components in the y direction, these high-frequency components will remain. Standard residual projection operators such as the full-weighting scheme will cause cancellation. As a result, these residual components will not show up on the coarse grid and cannot be removed by the coarse grid correction strategy.

For residual projection, we use a one-way average operator. Assume that x is the dominant direction. The residual at the grid points corresponding to the coarse grid points is averaged with respect to its two neighboring grid points in the x direction only. No averaging is performed with respect to the y direction. Letting $r_{i,j}$ be the residual at fine grid point (i, j), and $\bar{r}_{\bar{i},j}$ the corresponding coarse grid residual, we have $i = 2\bar{i}$, and

$$\bar{r}_{\bar{i},j} = 4 \times \frac{1}{4}(r_{i-1,j} + 2r_{i,j} + r_{i+1,j}) = r_{i-1,j} + 2r_{i,j} + r_{i+1,j}.$$

The factor 4 in the middle of the above formula represents the right-hand-side difference in $\triangle x^2$ between the fine and the coarse grids.

For interpolation operator, we use a similar strategy. Corrections for the approximate solution at fine grid points corresponding to the coarse grid points are transfered directly.

Corrections for other fine grid points take the average of the neighboring two grid points in the x direction only. Again, assuming $i = 2\bar{i}$, we write

$$r_{i-1,j} = \frac{1}{2}(\bar{r}_{\bar{i},j} + \bar{r}_{\bar{i}+1,j}), \quad r_{i,j} = \bar{r}_{\bar{i},j}.$$

In our implementation, the multigrid methods with the partial semicoarsening strategy use these one-way residual restriction and correction interpolation operators, regardless of the particular relaxation method used. However, once the grid is reduced to have equal mesh size in both directions, the standard full-coarsening strategy is used with the standard bilinear interpolation and full-weighting operators [6].

4. NUMERICAL EXPERIMENTS

Numerical experiments are conducted to solve a 2D Poisson equation (1) on the unit square domain [0, 1] × [0, 1]. The right-hand-side function and the Dirichlet boundary condition are prescribed to satisfy the exact solution $u(x, y) = \sin(100\pi x)\ \cos(2\pi y)$. This function is constructed so that it changes more rapidly in the x direction than in the y direction. Hence, we use 2D mesh with $N_x \geq N_y$. All numerical experiments and comparisons should be equally valid if the solution changes more rapidly in the y direction, with the multigrid methods being modified for the changed dominant direction.

The fourth-order compact difference scheme is compared against the standard central difference scheme, in terms of solution accuracy, multigrid convergence rate, and CPU timing. All multigrid methods use V cycle algorithm; the coarsest grid is the one with at least one dimension being the coarsest possible. One presmoothing and one postsmoothing are applied at each level. The initial guess is the zero vector. The iteration stops when the Euclidean norm (2-norm) of the residual vector is reduced by 10^{-10}. The maximum absolute error reported is the maximum absolute error between the computed solution at convergence and the exact solution over the entire fine grid points. The code is written in Fortran 77 programming language in double precision.

4.1. *Comparison of Second- and Fourth-Order Difference Schemes*

We first compare the fourth-order compact difference scheme with the standard second-order central difference scheme using a multigrid method with a line Gauss–Seidel relaxation. Table I compares the number of multigrid iterations (V cycles, I) and the corresponding CPU time (T) in seconds required for convergence. We see that the line Gauss–Seidel relaxation multigrid methods with these two discretization schemes have approximately the same convergence rate. With fine mesh discretizations $N_x = N_y \geq 512$; the multigrid method with the fourth-order compact difference scheme converges slightly faster than that with the second-order difference scheme. The fourth-order compact difference scheme is more expensive than the second-order difference scheme with the same discretization parameters, N_x and N_y. This is no surprise, as the number of arithmetic operations in the fourth-order compact difference scheme is more than that in the second-order difference scheme. However, the test results in Table II show that the solution computed from the fourth-order compact difference scheme is much more accurate than that computed from the second-order difference scheme.

TABLE I

Comparison of Iteration Counts (I) and CPU Seconds (T) for a Multigrid Method with Different Discretization Schemes and with Line Gauss–Seidel Relaxation

		$N_y = 64$		$N_y = 128$		$N_y = 256$		$N_y = 512$		$N_y = 1024$	
Scheme	N_x	I	T	I	T	I	T	I	T	I	T
Second order	64	11	0.09								
Fourth order		10	0.10								
Second order	128	9	0.14	11	0.34						
Fourth order		10	0.19	11	0.41						
Second order	256	10	0.30	11	0.66	11	1.41				
Fourth order		10	0.37	11	0.83	11	1.72				
Second order	512	10	0.61	11	1.40	11	2.87	12	6.30		
Fourth order		10	0.76	11	1.74	11	3.49	11	7.10		
Second order	1024	10	1.37	11	3.13	11	6.40	11	12.41	12	28.03
Fourth order		10	1.79	11	3.94	11	7.97	11	15.97	11	32.95

In order to justify the fourth-order compact difference scheme and the use of unequal mesh sizes in two different coordinate directions, we investigate the computational cost (CPU time) required for computing an approximate solution with a given accuracy; e.g., in Table II, we find that the computed solution with a maximum absolute error around 8.31×10^{-5} from the second-order scheme uses $N_x = 1024$, $N_y = 128$. The data in Table I shows that the cost is 3.13 CPU seconds. For the fourth-order compact difference scheme, a more accurate approximate solution, can be computed with $N_x = 128$, $N_y = 64$ (see row 3, Table II). The corresponding data in Table I shows that the computational cost of the fourth-order compact difference scheme is 0.19 CPU seconds. Thus the fourth-order compact difference scheme is about 16 times faster than the second-order difference scheme. Similar comparisons can be made with other data to reach similar conclusions.

The data in Tables I and II also indicate the advantage of using unequal-mesh-size discretization for both the second- and the fourth-order compact difference schemes. The data

TABLE II

Comparison of Maximum Absolute Errors of the Computed Approximate Solutions from Different Discretization Schemes

Scheme	N_x	$N_y = 64$	$N_y = 128$	$N_y = 256$	$N_y = 512$	$N_y = 1024$
Second order	64	1.95 (−2)				
Fourth order		4.87 (−4)				
Second order	128	4.87 (−3)	4.85 (−3)			
Fourth order		1.06 (−5)	1.36 (−6)			
Second order	256	1.24 (−3)	1.22 (−3)	1.21 (−3)		
Fourth order		8.39 (−8)	6.58 (−7)	8.46 (−7)		
Second order	512	3.33 (−4)	3.09 (−4)	3.04 (−4)	3.02 (−4)	
Fourth order		1.80 (−7)	5.27 (−9)	4.11 (−8)	5.29 (−8)	
Second order	1024	1.06 (−4)	8.31 (−5)	7.74 (−5)	7.59 (−5)	7.55 (−5)
Fourth order		4.46 (−8)	1.13 (−8)	3.30 (−10)	2.57 (−9)	3.30 (−9)

Note. 1.95 (−2) = 1.95×10^{-2}.

in Table I show that the CPU seconds are more than doubled when either N_x or N_y is doubled. However, the results in Table II show that increasing N_y does not always lead to a reasonable increase in accuracy (small error) in the computed solution; e.g., the data in Table II show that for the fourth-order compact difference scheme, the least-error 3.30×10^{-10} is achieved with $N_x = 1024$, $N_y = 256$ in 7.97 CPU seconds (Table I). Use of equal-mesh $N_x = N_y = 1024$ does not compute more accurate solutions. The cost, however, is increased to 32.95 CPU seconds.

4.2. *Comparison of Different Multigrid Methods*

We further compare the convergence histories of a full-coarsening multigrid method with four-color Gauss–Seidel relaxation, a semicoarsening multigrid method with four-color Gauss–Seidel relaxation, and a multigrid method with full coarsening and line Gauss–Seidel relaxation. We first choose $N_x = 512$ and $N_y = 256$ so that the anisotropy is modest. The left panel of Fig. 1 shows that both semicoarsening and line Gauss–Seidel relaxation multigrid methods converge quickly, but the standard full-coarsening multigrid method takes more iterations to converge.

In the right panel of Fig. 1, we increase the anisotropy by choosing $N_x = 512$ and $N_y = 128$. We can see that the convergence rates of both the semicoarsening and line Gauss–Seidel relaxation multigrid methods are barely affected by the change of mesh size in the y direction. However, the number of full-coarsening multigrid iterations increases substantially. It takes 81 iterations for the standard full-coarsening multigrid method to converge. In comparison, the number of semicoarsening multigrid iterations is 11, and that of the line Gauss–Seidel relaxation multigrid method is 12. These comparisons justify our efforts to develop specialized multigrid methods for this application.

4.3. *Comparison of Different Multigrid Relaxation Schemes*

For the fourth-order compact difference scheme, we compare multigrid methods with different Gauss–Seidel relaxation schemes. In Table III, the multigrid methods with line Gauss–Seidel relaxation (line GS), the point Gauss–Seidel relaxation in lexicographical

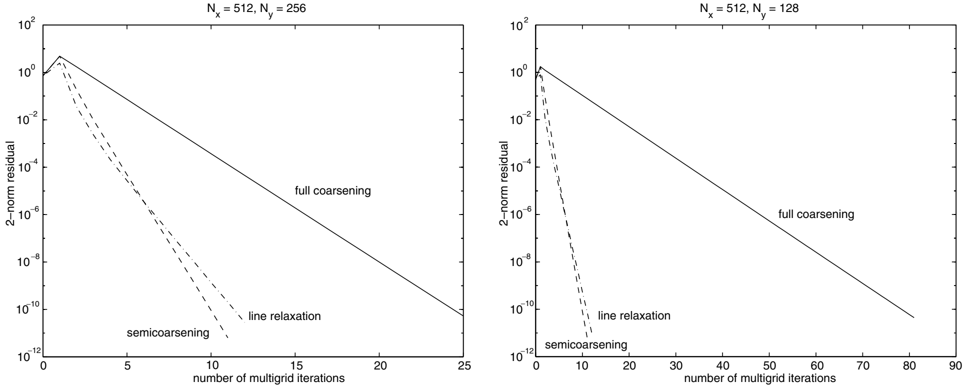

FIG. 1. Comparison of convergence rates of a full-coarsening multigrid method (solid line), a semicoarsening multigrid method (dashed line), and a multigrid method with line Gauss–Seidel relaxation (dash-dot line). On the finest grid, $N_x = 512$; $N_y = 256$ for the left panel and $N_y = 128$ for the right.

TABLE III
Comparison of Iteration Counts (I) and CPU Seconds (T) for Different Multigrid Smoothers with the Fourth-Order Compact Difference Scheme

		$N_y = 64$		$N_y = 128$		$N_y = 256$		$N_y = 512$		$N_y = 1024$	
Smoother	N_x	I	T	I	T	I	T	I	T	I	T
Line GS	64	10	0.09								
Natural GS		11	0.08								
Red–black GS		8	0.06								
Four-color GS		9	0.08								
Line GS	128	11	0.19	11	0.41						
Natural GS		12	0.12	11	0.32						
Red–black GS		8	0.15	9	0.29						
Four-color GS		9	0.18	9	0.30						
Line GS	256	10	0.36	11	0.84	11	1.73				
Natural GS		13	0.50	12	0.88	11	1.34				
Red–black GS		10	0.40	9	0.70	9	1.23				
Four-color GS		10	0.43	9	0.74	10	1.50				
Line GS	512	10	0.77	11	1.74	11	3.50	11	7.13		
Natural GS		14	1.11	13	2.08	12	3.60	12	5.90		
Red–black GS		11	0.94	11	1.95	9	3.03	9	4.97		
Four-color GS		10	0.93	10	1.96	10	3.80	10	6.22		
Line GS	1024	10	1.78	11	3.94	11	7.94	11	15.97	11	32.83
Natural GS		14	2.73	14	5.51	13	10.03	12	17.18	12	30.51
Red–black GS		11	2.37	12	5.20	11	9.43	9	14.46	10	27.93
Four-color GS		10	2.38	10	4.90	10	9.73	10	18.21	10	31.09

ordering (natural GS), the point Gauss–Seidel relaxation in red–black ordering (red–black GS), and the point Gauss–Seidel relaxation in four-color ordering (four-color GS) are compared in terms of the number of iterations and CPU time in seconds. All point relaxations are implemented with semicoarsening if $N_x \neq N_y$.

The test results demonstrate the superb robustness of both line GS and four-color GS relaxation schemes. The numbers of multigrid iterations with these two relaxation schemes are least affected by the variation of N_x or N_y. The red–black GS is seen to be more robust than the lexicographical GS. In a few cases, the red–black GS relaxation achieves the fewest number of iterations among the four relaxation schemes compared. This is the case, e.g., when $N_x = 1024$, $N_y = 512$.

However, the data in Table III indicate that the relaxation scheme with a small number of iterations is not necessarily the one that takes a small number of CPU seconds. In fact, if the numbers of iterations are the same, line GS and natural GS are faster than both red–black GS and four-color GS. The difference is caused by the cache effect of computer memory due to data locality. In both line and lexicographical relaxations, all grid points are visited and the corresponding data are brought into cache just once in each iteration. However, in our current implementation, red–black GS visits the entire grid twice and four-color GS goes over the entire grid four times in each iteration. The same data need to be brought into the computer cache multiple times in the colored relaxation cases. This causes more cache misses and delay in computation. Special code writing can alleviate such a cache effect and avoid multiple data movements in colored relaxations [4]. These cache-aware multigrid methods are beyond the scope of this paper.

5. CONCLUDING REMARKS

We have studied a fourth-order compact difference scheme with unequal mesh sizes for discretizing a 2D Poisson equation. Special multigrid methods are developed to solve the resulting sparse linear systems efficiently. The multigrid methods with line Gauss–Seidel relaxation or partial semicoarsening are found to work very well in solving the fourth-order compact-scheme-discretized 2D Poisson equation. In particular, partial semicoarsening with red–black and four-color Gauss–Seidel relaxations is shown to be robust with respect to the variation in the mesh aspect ratio. The four-color Gauss–Seidel relaxation is particularly attractive due to its improved convergence rate and inherent parallelism, compared to the standard natural Gauss–Seidel relaxation.

The proposed fourth-order compact discretization methodology may be generalized to three dimensions straightforwardly. However, the analogous generalizations of specialized multigrid methods to 3D are not straightforward; at least their implementations are nontrivial. Line Gauss–Seidel relaxation in 2D will be generalized to plane relaxation in 3D, with each planewise solution being obtained by using line relaxations. Partial semicoarsening is also complicated, since there are two directions in which the mesh size may change.

REFERENCES

1. L. M. Adams and J. Ortega, A multicolor SOR method for parallel computers, in *Proceedings of the 1982 International Conference on Parallel Processing* (IEEE, Bellaire, MI, 1982), p. 53.
2. L. Collatz, *The Numerical Treatment of Differential Equations* (Springer-Verlag, Berlin, 1960).
3. W. Dai and R. Nassar, A finite difference method for solving 3-D heat transport equations in a double-layered thin film with microscale thickness and nonlinear interfacial conditions, *Numer. Heat Transfer A* **39**, 21 (2001).
4. C. C. Douglas, Caching in with multigrid algorithms: Problems in two dimensions, *Parallel Algorithms Appl.* **9**, 195 (1996).
5. L. Ge and J. Zhang, Accuracy, robustness, and efficiency comparison in iterative computation of convection diffusion equation with boundary layers, *Numer. Methods Partial Differential Equations* **16**(4), 379 (2000).
6. M. M. Gupta, J. Kouatchou, and J. Zhang, Comparison of second and fourth order discretizations for multigrid Poisson solver, *J. Comput. Phys.* **132**, 226 (1997).
7. W. A. Mulder, A new multigrid approach to convection problems, *J. Comput. Phys.* **83**(2), 303 (1989).
8. W. F. Spotz and G. F. Carey, High-order compact scheme for the steady streamfunction vorticity equations, *Int. J. Numer. Methods Eng.* **38**, 3497 (1995).
9. J. C. Strikwerda, *Finite Difference Schemes and Partial Differential Equations* (Champman & Hall, New York, 1989).
10. P. Wesseling, *An Introduction to Multigrid Methods* (Wiley, Chichester, UK, 1992).
11. J. Zhang, An explicit fourth-order compact finite difference scheme for three dimensional convection-diffusion equation, *Commun. Numer. Methods Eng.* **14**, 209 (1998).
12. J. Zhang, Fast and high accuracy multigrid solution of the three dimensional Poisson equation, *J. Comput. Phys.* **143**, 449 (1998).
13. J. Zhang and J. J. Zhao, Unconditionally stable finite difference scheme and iterative solution of 2D microscale heat transport equation, *J. Comput. Phys.* **170**, 261 (2001).