Journal of Computational Physics **182,** 337–345 (2002)
doi:10.1006/jcph.2002.7172

A Simple Compact Fourth-Order Poisson Solver on Polar Geometry

Ming-Chih Lai

Department of Applied Mathematics, National Chiao Tung University, 1001 Ta Hsueh Road, Hsinchu 30050, Taiwan, Republic of China
E-mail: mclai@math.nctu.edu.tw

Received October 22, 2001; revised July 23, 2002

We present a simple and efficient compact fourth-order Poisson solver in polar coordinates. This solver relies on the truncated Fourier series expansion, where the differential equations of the Fourier coefficients are solved by the compact fourth-order finite difference scheme. By shifting a grid a half mesh away from the origin and incorporating the symmetry constraint of Fourier coefficients, we can easily handle coordinate singularities without pole conditions. The numerical evidence confirms fourth-order accuracy for the problem on an annulus and third-order accuracy for the problem on a disk. In addition, a simple and comparably accurate approximation for the derivatives of the solution is also presented.

Key Words: fast Poisson solver; polar coordinates; compact scheme; FFT.

1. INTRODUCTION

A fourth-order compact (nine point) formula for the Poisson equation on a two-dimensional rectangular domain has been known for almost 40 years. This popular finite difference approximation was developed by Collatz [3] and implemented by Houstis and Papatheodorou (FFT9) for public use in Netlib. A comparison of its accuracy and computational efficiency with that of the second-order (five point) scheme was done by Gupta [4]. In [4], a new compact approximation preserving fourth-order accuracy for the gradient of the solution was also presented. More recently, Zhuang and Sun [12] have modified Collatz's scheme to the singular Neumann problem. The authors also list a detailed survey along with other high-order difference discretizations.

In many physical problems, one often needs to solve the Poisson equation on a non-Cartesian domain, such as polar or cylindrical domains. It is more convenient to rewrite the equation in those coordinates. There are few papers in the literature that discuss the fourth-order finite difference schemes for the Poisson equation in polar coordinates (see [5, 6, 10], and references therein). Although the methods presented in [5, 6, 10] are compact

0021-9991/02 $35.00

(nine-point stencil), the solutions at the origin require being known or approximated in some complicated way. Thus, the resulting linear systems might not have efficient direct solvers.

In this paper, a simple and efficient compact fourth-order direct solver for the Poisson equation in polar coordinates is presented. This solver relies on the truncated Fourier series expansion, where the differential equations of the Fourier coefficients are solved by the compact fourth-order finite difference discretizations. By shifting a grid a half mesh away from the origin [8, 9] and incorporating the symmetry constraint of Fourier coefficients, we can easily handle coordinate singularities without pole conditions. In addition, a simple and accurate approximation for the derivatives of the solution is also presented. The numerical evidence confirms fourth-order accuracy for the problem on an annulus and third-order accuracy for the problem on a disk.

2. COMPACT FOURTH-ORDER SCHEME

Consider a Dirichlet problem for the Poisson equation on a unit disk:

$$\frac{\partial^2 u}{\partial r^2} + \frac{1}{r}\frac{\partial u}{\partial r} + \frac{1}{r^2}\frac{\partial^2 u}{\partial \theta^2} = f(r, \theta), \quad 0 < r < 1,\ 0 \le \theta < 2\pi, \tag{1}$$

$$u(1, \theta) = g(\theta). \tag{2}$$

Since a function on the disk is periodic in θ, we can approximate the solution using the truncated Fourier series

$$u(r, \theta) = \sum_{n=-N/2}^{N/2-1} \widehat{u_n}(r)\, e^{in\theta}, \tag{3}$$

where $\widehat{u_n}(r)$ is the complex Fourier coefficient given by

$$\widehat{u_n}(r) = \frac{1}{N}\sum_{j=0}^{N-1} u(r, \theta_j)\, e^{-in\theta_j}, \tag{4}$$

and where $\theta_j = j\Delta\theta$ with $\Delta\theta = 2\pi/N$ and N is the number of grid points along the circle. The above transformation between the physical space and Fourier space can be efficiently performed using the fast Fourier transform (FFT) with $O(N \log_2 N)$ arithmetic operations.

Substituting those expansions into Eq. (1) and equating the Fourier coefficients, the nth Fourier mode $\widehat{u_n}(r)$ satisfies the ordinary differential equation

$$\frac{d^2 \widehat{u_n}}{dr^2} + \frac{1}{r}\frac{d\,\widehat{u_n}}{dr} - \frac{n^2}{r^2}\,\widehat{u_n} = \widehat{f_n}, \quad 0 < r < 1, \tag{5}$$

$$\widehat{u_n}(1) = \widehat{g_n}, \tag{6}$$

where the complex Fourier coefficients $\widehat{f_n}(r)$ and $\widehat{g_n}$ are defined in a manner similar to that of (3) and (4). As we know, the above equation has a singular point at $r = 0$. Recently, the author and Wang [8] presented a second- and fourth-order scheme for Eq. (5) which shifts a grid a half mesh away from the origin and incorporates the symmetry constraint of Fourier coefficients to handle the singularity. In order to achieve fourth-order accuracy, a

five-point-long stencil (not compact) with one-sided derivative formulas must be applied to discretize the above equation. In the following, we will derive a three-point compact approximation for the solution of Eq. (5). This new approach does not need to use one-sided difference approximations, and the resulting linear system is much simpler. For simplicity of notation, we drop the subscript and hat and denote the Fourier coefficients $U(r) = \widehat{u_n}(r)$ and $F(r) = \widehat{f_n}(r)$, respectively.

Our goal is to derive a fourth-order finite difference approximation to Eq. (5). Obviously, the first and second derivatives, U' and U'', must be approximated to fourth order accurately. First, let us write down two difference formulas for the first and second derivatives with the truncation error $O(\Delta r^4)$:

$$U' = \delta_0 U - \frac{U'''}{6}\Delta r^2 + O(\Delta r^4), \tag{7}$$

$$U'' = \delta^2 U - \frac{U''''}{12}\Delta r^2 + O(\Delta r^4). \tag{8}$$

Here δ_0 and δ^2 are the centered difference operators for the first and second derivatives, defined as

$$\delta_0 U_i = \frac{U_{i+1} - U_{i-1}}{2\Delta r}, \quad \delta^2 U_i = \frac{U_{i+1} - 2\,U_i + U_{i-1}}{\Delta r^2}, \tag{9}$$

where U_i are the discrete values defined at the grid points r_i. A special choice of r_i to avoid the polar singularity [8, 9] is given by

$$r_i = (i - 1/2)\,\Delta r, \quad i = 0, 1, \ldots M + 1, \tag{10}$$

with the mesh width $\Delta r = 2/(2M + 1)$.

In order to have fourth-order approximations for U' and U'', we need to approximate the higher order derivatives U''' and U'''' in Eqs. (7) and (8) to be second-order accurate. Moreover, the approximations should involve only three grid points. To accomplish this, we differentiate Eq. (5) once and twice, respectively, to obtain the higher order derivatives of U:

$$U''' = F' - \frac{U''}{r} + \frac{1+n^2}{r^2}\,U' - \frac{2n^2}{r^3}\,U, \tag{11}$$

$$U'''' = F'' - \frac{F'}{r} + \frac{3+n^2}{r^2}\,U'' - \frac{3+5n^2}{r^3}\,U' + \frac{8n^2}{r^4}\,U. \tag{12}$$

Now the derivatives U''', U'''' are written in terms of U', U'', F', F'', and U. Those lower order derivatives can be approximated further by the three-point centered difference formulas, δ^2 and δ_0, to achieve second-order accuracy. Substituting those approximations into Eqs. (7) and (8) and applying to Eq. (5), we obtain the finite difference scheme as follows. For $1 \le i \le M$, we need to solve

$$\begin{aligned}\delta^2 U_i - \frac{\Delta r^2}{12}\left(\delta^2 F_i - \frac{1}{r_i}\,\delta_0 F_i + \frac{3+n^2}{r_i^2}\,\delta^2 U_i - \frac{3+5n^2}{r_i^3}\,\delta_0 U_i + \frac{8n^2}{r_i^4}\,U_i\right)\\ + \frac{1}{r_i}\,\delta_0 U_i - \frac{\Delta r^2}{6r_i}\left(\delta_0 F_i - \frac{1}{r_i}\,\delta^2 U_i + \frac{1+n^2}{r_i^2}\,\delta_0 U_i - \frac{2n^2}{r_i^3}\,U_i\right) - \frac{n^2}{r_i^2}\,U_i = F_i.\end{aligned} \tag{13}$$

The above difference equation uses the difference operators δ^2 and δ_0; thus only a three-point (compact) stencil is involved. This results in a tridiagonal linear system of equations for U_i, $i = 1, 2, \ldots M$, which can be solved by $O(M)$ arithmetic operations. It is worthwhile pointing out that if those terms inside the parentheses are ignored, then the scheme becomes second-order accurate.

In order to close the linear system, the numerical boundary values U_0 and U_{M+1} should be supplied. Choosing of r_i as described in (10), we have $r_{M+1} = 1$; thus, the outer numerical boundary value U_{M+1} is simply given by the boundary value $\widehat{g_n}$. The inner numerical boundary value U_0 can be obtained by the symmetry constraint of Fourier coefficients, which is derived as follows. The transformation between Cartesian and polar coordinates can be written as $x = r\cos\theta$, $y = r\sin\theta$. When we replace r with $-r$, and θ with $\theta + \pi$, the Cartesian coordinates of a point remain the same. Therefore, any scalar function $u(r, \theta)$ satisfies $u(-r, \theta) = u(r, \theta + \pi)$. Using this equality, we have

$$u(-r, \theta) = \sum_{n=-\infty}^{\infty} \widehat{u_n}(-r)\, e^{in\theta} = \sum_{n=-\infty}^{\infty} \widehat{u_n}(r)\, e^{in(\theta+\pi)} = \sum_{n=-\infty}^{\infty} (-1)^n\, \widehat{u_n}(r)\, e^{in\theta}. \quad (14)$$

Thus, when the domain of a function is extended to a negative value of r, the nth Fourier coefficient of this function satisfies $\widehat{u_n}(-r) = (-1)^n\, \widehat{u_n}(r)$. Using the above condition, we thus have

$$U_0 = U(r_0) = U(-\Delta r/2) = (-1)^n\, U(\Delta r/2) = (-1)^n\, U(r_1) = (-1)^n\, U_1. \quad (15)$$

Therefore, the numerical boundary value U_0 has been supplied.

In a sequential implementation, we summarize our scheme and the operation counts in the following three steps.

1. Compute the Fourier coefficients for the right-hand side function as in (4) by FFT, which requires $O(MN\ \log_2 N)$ arithmetic operations.
2. Solve the tridiagonal linear system for each Fourier mode. This requires $O(MN)$ operations.
3. Convert the Fourier coefficients as in (3) by FFT to obtain the solution, which requires $O(MN\ \log_2 N)$ operations.

The overall operation count is thus $O(MN\ \log_2 N)$ for $M \times N$ grid points. The method can be easily extended to the Helmholtz-type equation in a straightforward manner.

The main issue of this paper is to show the simplicity and the accuracy of the method. However, it is also worth noting that the scheme is highly parallelizable. A fast Fourier transform is performed in the azimuthal direction on each concentric grid circle. These transformations can be processed in parallel. The resulting tridiagonal linear system associated with different Fourier modes are decoupled from each other and thus can be inverted in parallel as well. The present method is also very easy to implement since many FFT subroutines are available.

Recently, Borges and Daripa [2] introduced a fast parallel algorithm for the Poisson equation on a disk. Their method differs from ours only in Step 2. Instead of discretizing the Fourier mode equation, they write the Fourier coefficient $\widehat{u_n}(r)$ as one-dimensional integrals in the radial direction and then use a quadrature rule such as the trapezoidal or Simpson's rule to evaluate the integrals. Thus, the two methods in [2] are second- and third-order accurate, respectively. Although the asymptotic operation count for their method is

roughly the same as ours ($O(MN \log_2 N)$), our numerical results show better accuracy than the results obtained in [2] (see the discussion in Section 4).

3. APPROXIMATIONS OF THE FIRST-ORDER DERIVATIVES

Once the numerical solution of the Poisson equation has been obtained, we may want to compute the first-order derivatives. One straightforward application is to find the velocity based on given vorticity in Navier–Stokes computations on a disk. The relation between the vorticity (ω) and stream function (ψ) is governed by

$$\Delta\psi = -\omega, \quad \psi(1, \theta) = 0 \text{ (no-slip condition)}, \tag{16}$$

where Δ is the Laplace operator. Equation (16) is a Poisson-type equation which can be solved by our fast compact fourth-order solver in the previous section. Once the stream function is computed, we want to recover the radial (u) and azimuthal (v) velocity components by using

$$u = \frac{1}{r}\frac{\partial\psi}{\partial\theta}, \quad v = -\frac{\partial\psi}{\partial r}. \tag{17}$$

If we use the regular three-point centered difference formula to approximate the first derivatives, we only have second-order accuracy for u and v. This indicates that even the fourth-order accuracy of the stream function does not lead to comparably accurate velocity components. Instead, we can use the well-known fourth-order compact difference formula [11] for the first derivative to compute the velocity. That is,

$$\frac{1}{6}(u_{i,j+1} + 4\,u_{ij} + u_{i,j-1}) = \frac{\psi_{i,j+1} - \psi_{i,j-1}}{2\,r_i\,\Delta\theta}, \tag{18}$$

$$\frac{1}{6}(v_{i+1,j} + 4\,v_{ij} + v_{i-1,j}) = -\frac{\psi_{i+1,j} - \psi_{i-1,j}}{2\,\Delta r}. \tag{19}$$

The above scheme is an implicit type, meaning that the velocity is not explicitly represented by the stream function. This increase in accuracy can be simply achieved at the cost of inverting a symmetric tridiagonal matrix, which can be done very efficiently.

4. NUMERICAL RESULTS

In this section, numerical results for the scheme presented in the previous sections are given. All numerical experiments were carried out on a PC workstation with 512 RAM using double-precision arithmetics. Our code is written in Fortran and is available to the public upon request.

Throughout this paper, we are more interested in the accuracy of the method for different resolution points used in the radial direction while the number of Fourier coefficients used is fixed. For a sufficiently smooth solution, only a few Fourier modes are needed to represent the function accurately in a finite Fourier space. In other words, for the smooth solution, the dominant error of our numerical scheme comes from the radial discretization, regardless of the number of Fourier coefficients in use. On the other hand, for a function that presents rapid variations, the influences of high-frequency Fourier modes cannot be ignored. In that

TABLE I
The Relative Errors for Example 1 on a Unit Disk Using Distinct Values of M and N

$N\backslash M$	16	32	64	128
16	8.0695E-06	8.2862E-06	8.2947E-06	8.2936E-06
32	1.9514E-06	1.3018E-07	8.3945E-09	5.3254E-10
64	1.9514E-06	1.3018E-07	8.3945E-09	5.3254E-10
128	1.9514E-06	1.3018E-07	8.3945E-09	5.3254E-10

case, using the distinct number of Fourier modes in the discretization will make significant differences in accuracy. Therefore, we have to make sure that enough Fourier modes are included in our approximation. Those behaviors have been confirmed numerically by solving the Dirichlet problem for the first example below with distinct values of M and N.

As in [2], we have run a series of tests by fixing the value M but varying the number N, or vise versa. The range of M and N are from 16 to 128. Table I shows the relative errors in the norm $\|\cdot\|_\infty$ for distinct values of M and N. One can see that when the number $N = 16$, the accuracy does not improve even though we refine the radial mesh. This is not surprising because using too few Fourier modes causes the dominant error coming from the azimuthal discretization. When the number N is greater than or equal to 32, the dominant error comes from the radial discretization, and the errors occur irrespective of the number of Fourier modes in use. It is important to note that the magnitude of the errors obtained by our scheme is significantly smaller than those obtained in [2]. For instance, in order to obtain the error of order 10^{-10}, as shown in Table I, we only need to use 128 grid points in the radial direction, but in [2] they need 2048 grid points to achieve the same accuracy. The other test examples show similar behaviors as well.

In the following numerical experiments, we simply fix the number $N = 64$ as suggested in [2] and vary the number M from 16 to 128. We have run four different examples to test the accuracy of our numerical scheme. Those problems are simply described by the solution u in Cartesian coordinates so that the right-hand-side function f and the boundary conditions can easily be obtained from u. We state them as follows:

1. $u(x, y) = 3e^{x+y}(x - x^2)(y - y^2) + 5$.
2. $u(x, y) = \frac{e^x + e^y}{1 + xy}$.
3. $u(x, y) = x^3 e^x (y + 1)\cos(x + y^3)$.
4. $u(x, y) = ((x + 1)^{5/2} - (x + 1))(y + 1)^{5/2} + ((x + 1) - (x + 1)^{5/2})(y + 1)$.

Notice that these test examples are used in [2] and are designed with some interesting properties. For instance, the second example possesses the symmetry for all four quadrants while the third one does not have the symmetry. The last example even has a discontinuity in the "2.5" derivative. The original references of those examples can be found in [2]. One should also note that those functions can be easily written in polar coordinates by using the transformation $x = r\ \cos\theta$, $y = r\ \sin\theta$.

Table II shows the relative errors in $\|\cdot\|_\infty$ norm of the above Dirichlet problems. The second and third columns are the maximum errors and the rate of convergence, respectively, for the different solutions on a unit disk $\{0 < r \le 1\}$. The fourth column is the maximum errors for the r-derivative of the solutions, which are computed by the formula (19). The θ-derivative of the solution can be computed in a similar manner by the formula (18), so we omit it here.

TABLE II
The Maximum Errors of Different Solutions to the Poisson Equation with Dirichlet Boundary Using $N = 64$

	$0 < r \le 1$			$0.5 \le r \le 1$			
M	$\|u\|_\infty$	Rate	$\left\|\frac{\partial u}{\partial \gamma}\right\|_\infty$	$\|u\|_\infty$	Rate	$\left\|\frac{\partial u}{\partial \gamma}\right\|_\infty$	Rate
$u(x, y) = 3e^{x+y}(x - x^2)(y - y^2) + 5$							
16	1.9514E-06		7.2205E-05	4.3797E-08		1.8240E-06	
32	1.3018E-07	3.91	9.2879E-06	3.0802E-09	3.83	1.2854E-07	3.83
64	8.3945E-09	3.95	1.1149E-06	2.0436E-10	3.91	8.3423E-09	3.95
128	5.3254E-10	3.98	1.3270E-07	1.3157E-11	3.96	5.3138E-10	3.97
$u(x, y) = \frac{e^x + e^y}{1 + xy}$							
16	2.2269E-05		1.2520E-04	1.0732E-06		8.1055E-06	
32	1.6070E-06	3.79	1.4423E-05	7.3759E-08	3.86	5.8481E-07	3.79
64	1.6828E-07	3.26	2.7107E-06	4.8259E-09	3.93	3.9624E-08	3.88
128	1.9083E-08	3.14	6.0413E-07	3.0665E-10	3.98	3.1239E-09	3.66
$u(x, y) = x^3 e^x (y + 1)\cos(x + y^3)$							
16	7.6278E-05		2.5728E-04	4.6496E-07		2.0526E-06	
32	8.3632E-06	3.19	5.6937E-05	3.1535E-08	3.88	1.5001E-07	3.77
64	9.7001E-07	3.11	1.3235E-05	2.0540E-09	3.94	1.0207E-08	3.88
128	1.1619E-07	3.06	3.2090E-06	1.3106E-10	3.97	6.6653E-10	3.94
$u(x, y) = ((x + 1)^{5/2} - (x + 1))(y + 1)^{5/2} + ((x + 1) - (x + 1)^{5/2})(y + 1)$							
16	4.6848E-05		1.3695E-04	8.7158E-08		3.5723E-07	
32	5.3996E-06	3.12	3.2406E-05	9.6606E-09	3.17	5.0802E-08	2.81
64	6.4462E-07	3.07	7.8751E-06	8.8599E-10	3.45	6.0617E-09	3.07
128	7.8506E-08	3.04	1.9396E-06	8.9131E-11	3.31	1.5570E-09	1.96

Note. The left half shows the errors for the disk case and the right half that for the annulus case.

One can see that the errors of the solutions show fourth-order convergence for the first example but third-order convergence for the other three solutions. The loss of one order of accuracy seems to come from the discretization near the origin. This can be seen from the following truncation error analysis. In the Fourier mode equation (5), the $U'(=\frac{d\widehat{u_n}}{dr})$ term is divided by r. So the second-order approximation of U''' in (7) is divided by an $O(\Delta r)$ term near the origin, which makes the approximation of U'''/r first-order accurate. This has the consequence that the overall truncation error of the U'/r term in the vicinity of the origin is $O(\Delta r^3)$ and thus so is the Fourier mode equation (5). However, this loss of accuracy does not appear when solving the problem on an annulus. Let us explain why that is the case next.

The present scheme can be easily applied to solve the Poisson equation on an annulus $\{a \le r \le b\}$, where $a > 0$. As the whole disk case, we need to solve Eqs. (5) and (6) with an additional boundary condition imposed at $r = a$. Instead of setting a grid as in (10), we choose a regular grid,

$$r_i = a + i\,\Delta r, \quad i = 1, 2, \ldots M, M + 1, \tag{20}$$

with the mesh width $\Delta r = (b - a)/(M + 1)$. Now the second-order approximation of U''' in (7) is divided by an $O(a + \Delta r)$ term instead of an $O(\Delta r)$ term, so the truncation error of the U'''/r term is still $O(\Delta r^2)$. Therefore, the overall truncation error of Eq. (5) is $O(\Delta r^4)$.

TABLE III
Comparison of the Numerical Results Obtained by Our Compact Fourth-Order Solver and the Fishpack Solver SEPX4.f [1]

(M, N)	Present	Fishpack	(M, N)	Fishpack
$u(x, y) = 3e^{x+y}(x - x^2)(y - y^2) + 5$				
(16,64)	4.3797E-08	4.6689E-06	(16,256)	6.4809E-08
(32,64)	3.0802E-09	4.4177E-06	(32,512)	4.3461E-09
(64,64)	2.0436E-10	4.3568E-06	(64,1024)	2.7674E-10
(128,64)	1.3157E-11	4.3433E-06	(128,2048)	1.7190E-11
$u(x, y) = \dfrac{e^x + e^y}{1 + xy}$				
(16,64)	1.0732E-06	1.6547E-05	(16,256)	3.8164E-06
(32,64)	7.3759E-08	1.4413E-05	(32,512)	9.0794E-08
(64,64)	4.8259E-09	1.4317E-05	(64,1024)	5.2635E-09
(128,64)	3.0665E-10	1.4293E-05	(128,2048)	3.8846E-10
$u(x, y) = x^3e^x(y + 1)\cos(x + y^3)$				
(16,64)	4.6496E-07	2.6202E-05	(16,256)	5.9412E-07
(32,64)	3.1535E-08	2.4533E-05	(32,512)	5.0378E-08
(64,64)	2.0540E-09	2.4139E-05	(64,1024)	3.3497E-09
(128,64)	1.3106E-10	2.4055E-05	(128,2048)	2.1246E-10
$u(x, y) = ((x + 1)^{5/2} - (x + 1))(y + 1)^{5/2} + ((x + 1) - (x + 1)^{5/2})(y + 1)$				
(16,64)	8.7158E-08	3.4903E-05	(16,256)	7.9015E-07
(32,64)	9.6606E-09	3.2557E-05	(32,512)	9.5403E-08
(64,64)	8.8599E-10	3.2009E-05	(64,1024)	1.1850E-08
(128,64)	8.9131E-11	3.1866E-05	(128,2048)	1.4876E-09

The fifth and sixth columns of Table II show the errors and the rate of convergence for the solutions on an annulus $\{0.5 \le r \le 1\}$. We can see that indeed the fourth-order convergence can be achieved for all examples except the last one, which has a discontinuity in the "2.5" derivative. Despite the less regularity for the last example, the solution still shows better than third-order accuracy. One can also see from the last two columns that the computation of r-derivative also preserves the desired accuracy and is comparable to the accuracy of the solution itself.

Table III shows a comparison of our present numerical results for the annulus case with the results obtained by the popular software package Fishpack [1]. This package has a subroutine called SEPX4.f, which is developed to solve the separable elliptic equation using either a second- or a fourth-order finite difference approximations. Note that by multiplying r^2 on the both sides of Eq. (1), the resulting equation is separable in $r - \theta$ domain. The fourth-order difference scheme is discretized using the deferred corrections approach described in [7], and then the resulting linear system is solved by the generalized cyclic reduction algorithm.

When we fix the number of N and vary the number M, we can see from Table III that the errors of the present scheme converge in the desired fourth-order rate while the errors of Fishpack do not decrease. This is not surprising since the dominant error of the Fishpack solver comes from the discretization in the θ direction. Thus, in order to obtain the same accuracy, many more grid points must be used for Fishpack. For instance, to obtain the error of magnitude 10^{-11} for Example 1, the present scheme needs only 128×64 grid points for the discretization, but the Fishpack solver needs 32 times more grid points (128×2048)

to achieve the same accuracy. The other test examples show the same behavior. So indeed, we can conclude that our compact fourth-order solver outperforms the Fishpack solver in accuracy and efficiency.

5. CONCLUSIONS

We present a simple and efficient compact fourth-order finite difference scheme for the Poisson equation in polar coordinates. We first use the truncated Fourier series expansion to derive a set of singular ordinary differential equations, then solve those singular equations using a compact fourth-order discretization. By shifting a grid a half mesh away from the origin and incorporating the symmetry constraint of Fourier coefficients, we can easily handle the coordinate singularities without pole conditions. In addition, a simple and accurate approximation to the derivatives of the solution is also presented. The numerical evidence confirms the fourth-order accuracy for the problem on an annulus and third-order accuracy for the problem on a disk. Furthermore, the present method shows better accuracy than some of numerical schemes in the literature. The total computational cost of the method is $O(MN \log_2 N)$ arithmetic operations for $M \times N$ grid points.

ACKNOWLEDGMENTS

The author thanks the referees for helpful comments and suggestions. The work was supported by the National Science Council of Taiwan under Research Grant NSC-90-2115-M-194-018.

REFERENCES

1. J. Adams, P. Swarztrauber, and R. Sweet, *Fishpack—A Package of Fortran Subprograms for the Solution of Separable Elliptic Partial Differential Equations* (http://www.netlib.org/fishpack, 1980).
2. L. Borges and P. Daripa, A fast parallel algorithm for the Poisson equation on a disk, *J. Comput. Phys.* **169**, 151 (2001).
3. L. Collatz, *The Numerical Treatment of Differential Equation* (Springer-Verlag, New York, 1960).
4. M. M. Gupta, A fourth-order Poisson solver, *J. Comput. Phys.* **55**, 166 (1984).
5. S. R. K. Iyengar and R. Manohar, High order difference methods for heat equation in polar cylindrical coordinates, *J. Comput. Phys.* **77**, 425 (1988).
6. M. K. Jain, R. K. Jain, and M. Krishna, A fourth-order difference scheme for quasilinear Poisson equation in polar co-ordinates, *Commun. Numer. Methods Eng.* **10**, 791 (1994).
7. H. B. Keller, *Numerical Solution of Two Point Boundary Value Problems* (Soc. for Indr. & Appl. Math., Philadelphia, 1976).
8. M.-C. Lai and W.-C. Wang, Fast direct solvers for Poisson equation on 2D polar and spherical geometries, *Numer. Methods Partial Differential Eq.* **18**, 56 (2002).
9. K. Mohseni and T. Colonius, Numerical treatment of polar coordinate singularities, *J. Comput. Phys.* **157**, 787 (2000).
10. R. C. Mittal and S. Gahlaut, High-order finite-difference schemes to solve Poisson's equation in polar coordinates, *IMA J. Numer. Anal.* **11**, 261 (1991).
11. J. C. Strikwerda, *Finite Difference Schemes and Partial Differential Equations* (Wadsworth & Brooks/Cole, Belmont, CA, 1989).
12. Y. Zhuang and X.-H. Sun, A high-order fast direct solver for singular Poisson equations, *J. Comput. Phys.* **171**, 79 (2001).