JOURNAL OF COMPUTATIONAL PHYSICS 104, 93–98 (1993)

On the Direct Solution of Poisson's Equation on a Non-uniform Grid

V. BABU AND SEPPO A. KORPELA

Department of Mechanical Engineering, The Ohio State University, Columbus, Ohio 43210

Received March 13, 1991; revised March 12, 1992

A direct method for the numerical solution of two- and three-dimensional Poisson equations on non-staggered and non-uniform grids is presented. The method is an extension of the method of matrix decomposition proposed by Buzbee *et al.* (*SIAM J. Numer. Anal.* 7 (1970)). Attention is focussed on the solution of the Poisson equation subject to Neumann boundary condition and details of the performance of the algorithm including operation counts are presented.

1. INTRODUCTION

The need to solve a Poisson equation for pressure with Neumann boundary conditions arises among other places in the numerical solution of the primitive equation form of the incompressible Navier–Stokes equations [1]. Although the Neumann–Poisson problem for pressure can be avoided in some formulations (the vorticity–streamfunction form being usual in two-dimensional studies) it seems that in three-dimensional flows other formulations are neither as simple, nor as direct, as those based on primitive variables. If one is interested in the unsteady development of the flow, pressure needs to be determined from a Poisson equation at each time step and thus it is desirable to do this as efficiently as possible. This is particularly true today when fast computers have made direct simulations of three-dimensional oscillatory or chaotic flows feasible and long calculations are needed in order to obtain statistical information on the flow.

With these notions in mind, in this paper we discuss how to extend the direct solution method based on matrix decomposition that appeared in the work of Buzbee *et al.* [2], to the Poisson equation with Neumann boundary conditions. Since accurate resolution often calls for the use of non-uniform grids, our aim was to formulate the method on such a mesh. The method is closely related to separation of variables and Fourier analysis. The latter, together with cyclic reduction, has been reviewed by Dorr [3] and Swarztrauber [4]. Indeed, the method of matrix decomposition can be considered to be a natural extension of these methods when coordinate transformation destroys those properties of the equations that make Fourier analysis and cyclic reduction efficient, and our contribution in this note is to show how to carry out this extension to the solution of the Neumann–Poisson problem for grids that are non-uniform in each of the coordinate directions.

It is sufficient here only to discuss the two-dimensional problem, for the extension to three dimensions follows directly from it. We have carried it out and are in the process of calculating solutions to some problems of three-dimensional time-dependent thermal convection in a cavity by methods based on further development of the ideas we have discussed in [1] and of those discussed below.

We are interested in solving on a non-staggered, non-uniform grid the equation

$$\frac{\partial^2 p}{\partial x^2}+\frac{\partial^2 p}{\partial y^2}=c, \tag{1}$$

subject to given values of the normal derivative, $\partial p/\partial n$ on the boundaries. It is well known that a solution to this problem is unique only to an arbitrary constant and it is subject to an integral constraint of the form

$$\iint c\, dA=\int \frac{\partial p}{\partial n}\, dl, \tag{2}$$

where the right-hand side involves the normal derivative of p along the boundary.

To put Eq. (1) into a proper form when using a non-uniform grid we first transform this grid in the physical domain into a uniform grid in a computational domain using one-dimensional transformations of the form $\xi=\xi(x)$ and $\eta=\eta(y)$. The transformations $\xi(x)$ and $\eta(y)$ have been chosen to have odd symmetry about the vertical

0021-9991/93 $5.00

and horizontal centerlines, respectively. The transformed equation (1) is then

$$\left(\frac{\partial \xi}{\partial x}\right)^2 \frac{\partial^2 p}{\partial \xi^2}+\frac{\partial^2 \xi}{\partial x^2}\frac{\partial p}{\partial \xi}+\left(\frac{\partial \eta}{\partial y}\right)^2 \frac{\partial^2 p}{\partial \eta^2}+\frac{\partial^2 \eta}{\partial y^2}\frac{\partial p}{\partial \eta}=c \tag{3}$$

and the boundary conditions are transformed similarly. Alternatively, one could discretize Eq. (1) in the physical domain itself and solve the resulting finite-difference equations using matrix decomposition methods (Farnell [5]).

2. MATRIX DECOMPOSITION ON NON-UNIFORM GRIDS

The derivatives of p appearing in Eq. (3) are discretized using second-order accurate central differences at all *interior* grid points and the transformation derivatives are evaluated either analytically or to second-order accuracy using finite differences. Thus, Eq. (3) can be written in a finite difference form as

$$\begin{aligned}
&\left(\frac{1}{\Delta\xi^2}\left(\frac{\partial \xi}{\partial x}\right)^2+\frac{1}{2\Delta\xi}\left(\frac{\partial^2 \xi}{\partial x^2}\right)\right)_i p_{i+1,j}\\
&\quad+\left(\frac{1}{\Delta\xi^2}\left(\frac{\partial \xi}{\partial x}\right)^2-\frac{1}{2\Delta\xi}\left(\frac{\partial^2 \xi}{\partial x^2}\right)\right)_i p_{i-1,j}\\
&\quad+\left(\frac{1}{\Delta\eta^2}\left(\frac{\partial \eta}{\partial y}\right)^2+\frac{1}{2\Delta\eta}\left(\frac{\partial^2 \eta}{\partial y^2}\right)\right)_j p_{i,j+1}\\
&\quad+\left(\frac{1}{\Delta\eta^2}\left(\frac{\partial \eta}{\partial y}\right)^2-\frac{1}{2\Delta\eta}\left(\frac{\partial^2 \eta}{\partial y^2}\right)\right)_j p_{i,j-1}\\
&\quad-\left(\frac{2}{\Delta\xi^2}\left(\frac{\partial \xi}{\partial x}\right)_i^2+\frac{2}{\Delta\eta^2}\left(\frac{\partial \eta}{\partial y}\right)_j^2\right)p_{i,j}=c_{i,j}.
\end{aligned} \tag{4}$$

As usual Eq. (4) can also be applied at the boundaries by introducing a row of fictitious grid points outside the domain. These fictitious grid points can then be eliminated from the finite difference equations by using central differences for discretizing the boundary conditions. Alternatively, one can apply Eq. (4) at the interior points alone. The boundary points that occur in these finite difference equations can be eliminated by using second-order accurate one-sided differences for discretizing the boundary conditions. We follow the latter approach and discretize the boundary condition, for example, at $i=0$, as

$$\left(\frac{\partial \xi}{\partial x}\right)_{i=0}\frac{-p_{2,j}+4p_{1,j}-3p_{0,j}}{2\Delta\xi}=\left[\frac{\partial p}{\partial x}\right]_{i=0}.$$

In either case, the solution methodology that we present can be applied for solving the resulting system of linear algebraic equations. This system can be written in a block tridiagonal form as

$$\begin{pmatrix} H_1 & A_1 & 0 & \cdot & \cdot \\ B_2 & H_2 & A_2 & 0 & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & 0 & B_{N-1} & H_{N-1} & A_{N-1} \\ \cdot & \cdot & 0 & B_N & H_N \end{pmatrix}\begin{pmatrix} q_1 \\ q_2 \\ \vdots \\ q_N \end{pmatrix}=\begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_N \end{pmatrix}, \tag{5}$$

where

$$q_j=\begin{pmatrix} p_{1,j} \\ p_{2,j} \\ \vdots \\ p_{M,j} \end{pmatrix}, \qquad d_j=\begin{pmatrix} c_{1,j} \\ c_{2,j} \\ \vdots \\ c_{M,j} \end{pmatrix},$$

and

$$H_j=\begin{pmatrix} h_{1,j} & g_{1,j} & 0 & \cdot & \cdot \\ f_{2,j} & h_{2,j} & g_{2,j} & 0 & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & 0 & f_{M-1,j} & h_{M-1,j} & g_{M-1,j} \\ \cdot & \cdot & 0 & f_{M,j} & h_{M,j} \end{pmatrix},$$

$$B_j=\begin{pmatrix} b_{1,j} & 0 & \cdot & \cdot & \cdot \\ 0 & b_{2,j} & 0 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & 0 & b_{M-1,j} & 0 \\ \cdot & \cdot & \cdot & 0 & b_{M,j} \end{pmatrix},$$

$$A_j=\begin{pmatrix} a_{1,j} & 0 & \cdot & \cdot & \cdot \\ 0 & a_{2,j} & 0 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & 0 & a_{M-1,j} & 0 \\ \cdot & \cdot & \cdot & 0 & a_{M,j} \end{pmatrix}.$$

We take M and N to be odd and note in passing that, as usual, those elements $c_{i,j}$ on the grid lines next to the boundaries, namely, $i=1$ and M, $j=1$ and N, now include terms from the boundary conditions also.

To solve the matrix Eq. (5) we first put the matrix **S**, given by

$$\mathbf{S}=\begin{pmatrix} H_1 & A_1 & 0 & \cdot & \cdot \\ B_2 & H_2 & A_2 & 0 & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & 0 & B_{N-1} & H_{N-1} & A_{N-1} \\ \cdot & \cdot & 0 & B_N & H_N \end{pmatrix},$$

in a symmetric form, for the reason that we prefer to remain in the *terra firma* of the symmetric eigenvalue problem that

it leads to, rather than trek into the *terra incognita* of the unsymmetrical case. In fact, the potential instability of the method is only linked to the stability of the calculation of the eigenvalues, for the rest of the algorithm requires only matrix multiplications.

The process of bringing S into a symmetric form is best carried out by first examining the center row corresponding to $j = N/2 + 1$. Since the transformation $\eta = \eta(y)$ is an odd function about the horizontal centerline, the second derivative of η with respect to y is zero at all points in this row and Eq. (4) shows that $B_{N/2+1} = A_{N/2+1}$. By similar arguments for $\xi(x)$ it can be seen that $f_{M/2+1,j} = g_{M/2+1,j}$, for $1 \leqslant j \leqslant N$. The matrix $H_{N/2+1}$ can now be made symmetric by starting with the middle row and working outwards while multiplying each row by the appropriate factors. We denote these multiplicative factors as $\varepsilon_{i,N/2+1}$. As a result of making the matrix $H_{N/2+1}$ symmetric, the elements of the off-diagonal matrices $B_{N/2+1}$ and $A_{N/2+1}$ and the right-hand side vector $d_{N/2+1}$ will also be altered. Proceeding away from the central block to bring the matrix S into a symmetric form, those operations that make the off-diagonal blocks symmetric also make the diagonal matrices, the H_j's, symmetric. In addition, the matrices are such that after the symmetry operations are completed, the relations

$$H_j = \alpha_j H_{N/2+1} + \beta_j A_{N/2+1}, \qquad 1 \leqslant j \leqslant N, \tag{6}$$

and

$$A_j = B_{j+1} = \gamma_j A_{N/2+1}, \qquad 1 \leqslant j \leqslant N-1, \tag{7}$$

hold. The coefficients α_j, β_j, and γ_j can easily be determined and accumulated into vectors. Indeed, since the matrix $H_{N/2+1}$ is tridiagonal and the matrix $A_{N/2+1}$ is diagonal, the factors α_j can be determined by comparing the off-diagonal terms on either side of the equation. After this the factor β_j is adjusted so that the diagonal terms are equal (see Eq. (11) below what these relations reduce to when the grids are uniform). Also, there is no loss of generality involved in assuming M and N to be odd since the cases where M and N are not odd can be handled similarly. For example, when L is even and M is odd, we have $B_{M/2+1} = A_{M/2+1}$ and $g_{L/2,j} = f_{L/2+1,j}$, for $1 \leqslant j \leqslant N$. In this case, one can leave the middle two rows of $G_{M/2+1}$ and work outwards as before.

As a result of the Neumann boundary conditions the matrix S is singular. This causes some small difficulties. As discussed in Mitchell [6], if now all the rows of S were added together the resulting row would consist entirely of zeroes. Thus for this system to be consistent, the sum of all the $c_{i,j}$'s should be equal to zero as well. This procedure can be identified as the application of the integral constraint in a discretized form. As a result of discretization errors, the above sum usually differs from zero. To assure that it will be zero, we follow Briley [7] and distribute this difference equally amongst the $c_{i,j}$'s so that now their sum is equal to zero to machine accuracy. This procedure is particularly important when working with transformed equations, since the truncation errors introduced as a result of transforming to the computational domain tend to be quite large if the stretching of the grids in the physical domain is severe [8]. After the $c_{i,j}$'s have been corrected the system of equations given by (5) is linearly dependent to machine accuracy.

To apply the method of matrix decomposition to solve this system of equations the matrices $H_{N/2+1}$, $A_{N/2+1}$ are diagonalized simultaneously so that

$$E^{\mathrm{T}} H_{N/2+1} E = \Lambda, \qquad E^{\mathrm{T}} A_{N/2+1} E = I, \tag{8}$$

where I is the identity matrix. The matrix E is the matrix of eigenvectors and Λ is the diagonal matrix of eigenvalues of the generalized eigenvalue problem $H_{N/2+1} e_i = \lambda A_{N/2+1} e_i$. Since the matrices H_j, A_j, and B_j involve a linear combination of matrices $H_{N/2+1}$ and $A_{N/2+1}$, the same eigenbasis can be used for diagonalizing all the other blocks of S. This diagonalization results in the separation of the i index and yields a scalar tridiagonal system on each $j =$ constant line. As a result the system of equations given by Eq. (5) can be written for $1 \leqslant i \leqslant M$ as

$$\begin{aligned} &(\alpha_1 \lambda_i + \beta_1)\, \bar{p}_{i,1} + \gamma_1 \bar{p}_{i,2} = \bar{c}_{i,1}, \\ &\gamma_{j-1} \bar{p}_{i,j-1} + (\alpha_j \lambda_i + \beta_j)\, \bar{p}_{i,j} + \gamma_j \bar{p}_{i,j+1} \\ &\qquad = \bar{c}_{i,j}, \qquad 2 \leqslant j \leqslant N-1, \\ &\gamma_{N-1} \bar{p}_{i,N-1} + (\alpha_N \lambda_i + \beta_N)\, \bar{p}_{i,N} = \bar{c}_{i,N}, \end{aligned} \tag{9}$$

where $(\bar{p}_{1,j}, \bar{p}_{2,j}, ..., \bar{p}_{M,j})^{\mathrm{T}} = E^{-1}(p_{1,j}, p_{2,j}, ..., p_{M,j})^{\mathrm{T}}$ and $(\bar{c}_{1,j}, \bar{c}_{2,j}, ..., \bar{c}_{M,j})^{\mathrm{T}} = E^{\mathrm{T}}(c_{1,j}, c_{2,j}, ..., c_{M,j})^{\mathrm{T}}$. These tridiagonal systems can be easily solved. However, the tridiagonal matrix corresponding to $i = M$, $j = N$ is singular owing to the system of equations being overdetermined, as already mentioned. The procedure, then, is to set $\bar{p}_{M,N}$ arbitrarily to zero and solve for the remaining $\bar{p}_{M,j}$'s [9]. This is allowed since the solution to Eq. (4) can be determined only up to an arbitrary constant. Once all the $\bar{p}_{i,j}$'s have been obtained, we can obtain p from

$$\begin{pmatrix} p_{1,j} \\ p_{2,j} \\ \vdots \\ p_{M,j} \end{pmatrix} = E \begin{pmatrix} \bar{p}_{1,j} \\ \bar{p}_{2,j} \\ \vdots \\ \bar{p}_{M,j} \end{pmatrix}, \qquad 1 \leqslant j \leqslant N. \tag{10}$$

In summary, for a given right-hand side c the algorithm consists of the following steps:

(i) Form the matrix S and put it in a symmetric form. This is accomplished by starting with the center block

and working outwards with each row multiplied by the appropriate factor ($\varepsilon_{i,j}$). Determine the coefficients α_j, β_j, and γ_j.

(ii) Compute the matrix of eigenvalues and eigenvectors by solving the generalized eigenvalue problem $H_{N/2+1}e_i = \lambda A_{N/2+1}e_i$.

(iii) Modify the right-hand side by multiplying the $c_{i,j}$'s (which now include the boundary conditions also) by the corresponding $\varepsilon_{i,j}$'s. If, the sum of the $c_{i,j}$'s deviates from zero, then distribute the deviation uniformly among these elements, so that their sum is zero to machine accuracy.

(iv) Obtain $\bar{c}$ by performing the matrix–vector multiplication $(\bar{c}_{1,j}, \bar{c}_{2,j}, ..., \bar{c}_{M,j})^{\mathrm{T}} = E^{\mathrm{T}}(c_{1,j}, c_{2,j}, ..., c_{M,j})^{\mathrm{T}}$ for $1 \leqslant j \leqslant N$.

(v) Solve the tridiagonal systems for $\bar{p}$.

(vi) Obtain p from $\bar{p}$ by performing the matrix–vector multiplication in Eq. (10).

3. MATRIX DECOMPOSITION ON UNIFORM GRIDS

The algorithm that we have outlined above solves the discretized Poisson's equation on a non-uniform grid to machine accuracy and the code that we have written is completely vectorizable. It is well known that the method of matrix decomposition is related to Fourier transform methods for solving partial differential equations [9, 10]. This relationship can be easily seen for uniform grids, since setting $\xi = x$ and $\eta = y$ in Eq. (4) yields, after the symmetry operations have been carried out, the equations

$$\begin{pmatrix} 3H/2+2A & A & 0 & \cdot & \cdot \\ A & H & A & 0 & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & 0 & A & H & A \\ \cdot & \cdot & 0 & A & 3H/2+2A \end{pmatrix} \begin{pmatrix} q_1 \\ q_2 \\ \vdots \\ q_N \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_N \end{pmatrix}, \tag{11}$$

where

$$H = \begin{pmatrix} -4 & 1 & 0 & \cdot & \cdot \\ 1 & -4 & 1 & 0 & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & 0 & 1 & -4 & 1 \\ \cdot & \cdot & 0 & 1 & -4 \end{pmatrix},$$

$$A = \begin{pmatrix} 3/2 & 0 & \cdot & \cdot & \cdot \\ 0 & 1 & 0 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & 0 & 1 & 0 \\ \cdot & \cdot & \cdot & 0 & 3/2 \end{pmatrix}.$$

Here we have used the same second-order accurate one-sided differences as above for discretizing the boundary conditions. The generalized eigenvalue problem that needs to be solved is given by $He_i = \lambda Ae_i$. This can be solved analytically and the eigenvectors can be expressed in terms of linear combinations of sines as shown in the Appendix. If central differences are used for discretizing the boundary conditions, the eigenvectors would be cosines as shown, for example, in Pickering [9], making the correspondence to Fourier transforms transparent.

4. PERFORMANCE ANALYSIS

An examination of the sequence of operations given in Section 2 shows that steps (i) and (ii) need to be performed only once and the results can be stored and used for any given right-hand side vector. Thus, these two steps need not be included in the estimate of the operation count for the algorithm. Step (iii) of the algorithm requires at most MN multiplications and $2MN$ additions/subtractions. The solution of the tridiagonal systems in step (v) requires only $3MN$ multiplications/divisions and $2MN$ additions/subtractions, after one recognizes that the forward elimination of the tridiagonal systems need be done only once and the multipliers can be stored and used for any right-hand side. The matrix multiplications in steps (iv) and (vi) require $2N M^2$ multiplications and $2N M^2$ additions. Hence, the algorithm requires $2N M^2 + 4MN$ additions/subtractions and an equal number of multiplications/divisions. For large values of M and N, the matrix multiplications can be expected to dominate the operation count. Numerical experiments do in fact show that the actual operation count asymptotes to $2N M^2$ for large values of M and N (Table I).

In order to evaluate both the performance and the accuracy of the algorithm, we solved the model Neumann–Poisson problem (with the exact solution $p = x^2 + y^2$),

$$\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} = 4,$$

TABLE I

Grid ($M \times N$)	Actual operations: Multiplications	Actual operations: Additions	CPU time (seconds)	Average error
63×63	$2.15M^3$	$2.18M^3$	0.019	5.8×10^{-5}
127×127	$2.07M^3$	$2.08M^3$	0.049	1.3×10^{-5}
255×255	$2.03M^3$	$2.03M^3$	0.254	3.0×10^{-6}
511×511	$2.01M^3$	$2.01M^3$	1.802	7.4×10^{-7}
1023×1023	$2.00M^3$	$2.00M^3$	13.925	1.8×10^{-7}
63×63[a]	$2.13M^3$	$2.23M^3$	0.013	3.2×10^{-12}

[a] Uniform grid.

in the unit square $-\frac{1}{2} \leqslant x, y \leqslant \frac{1}{2}$, subject to

$$\frac{\partial p}{\partial x} = \pm 1, \qquad x = \pm \tfrac{1}{2},$$

$$\frac{\partial p}{\partial y} = \pm 1, \qquad y = \pm \tfrac{1}{2}.$$

Solutions to this problem were carried out on several grids ranging from $(0:64 \times 0:64)$ to $(0:1024 \times 0:1024)$ and the results were compared to the exact solution. The non-uniform grids were generated by using the Roberts transformation [11] which was written as

$$x = \beta \tanh\left[\frac{1}{2} \log\left(\frac{\beta+1}{\beta-1}\right)\xi\right],$$

for the x-coordinate and similarly for the y-coordinate. This allowed us to evaluate the transformation derivatives analytically. The parameter β was set equal to 1.5 for generating all the non-uniform grids. The equation and boundary conditions were then transformed to the computational domain and discretized as discussed in Section 2. The discretized equations were solved on the Cray Y-MP8/864 at the Ohio Supercomputer Center in uniprocessor mode. There is an added benefit to solving these equations on the Cray, for the matrix multiplications in steps (iv) and (vi) can be performed efficiently using Cray Assembly Language routines.

The average error on each grid is shown in Table I. By using the performance tools available on the Cray, the actual operations performed during execution of the code were obtained and are also shown in Table I. As mentioned before, the extension of the above algorithm to three dimensions is straightforward and the details are discussed in [12]. We solved a three-dimensional Neumann–Poisson problem that is similar to the two-dimensional problem above, except that the exact solution has a z^2 term now added to it. It is easy to show that the operation count for the three-dimensional case on a $M \times N \times K$ grid should tend to $2(K N M^2 + K M N^2)$, for large values of M, N, and K. Numerical solutions to this problem were obtained on three grids and the results of these calculations are shown in Table II. The actual operation count is seen to asymptote to the theoretical value given above.

TABLE II

Grid ($M \times N \times K$)	Actual operations: Multiplications	Actual operations: Additions	CPU time (seconds)	Average error
$31 \times 31 \times 31$	$4.4M^4$	$4.5M^4$	0.076	3.4×10^{-4}
$63 \times 63 \times 63$	$4.2M^4$	$4.2M^4$	0.625	6.4×10^{-5}
$127 \times 127 \times 127$	$4.1M^4$	$4.1M^4$	7.259	1.3×10^{-5}

5. CONCLUSION

We have given above an extension of the matrix decomposition method to solve the two-dimensional Poisson equation subject to Neumann boundary conditions on a non-uniform grid. The three-dimensional version follows directly from this. We would like to stress how the symmetrization procedure above brings out clearly the integral constraint on the Neumann problem as a consistency condition on the finite difference equations for both uniform and non-uniform grids. Thus it illustrates how the system of equations can be forced to satisfy the consistency condition. This is of importance in cases where the equations and boundary conditions are differenced in such a way that they do not satisfy such a condition, as is often true in the case of the Poisson equation for pressure that arises during the solution of the Navier–Stokes equation in primitive variables. The other point that we would like to stress is that finding the eigenvectors of the Laplacian operator at the preprocessing stage leads to an increase in the computational speed in this part of the solution of the Navier–Stokes equations, for the pressure can now be efficiently determined by multiplying the vector representing the right-hand side by a previously stored matrix.

APPENDIX

The generalized eigenvalue problem $He_l = \lambda A e_l$ can be written as

$$\begin{pmatrix} -4 & 1 & 0 & \cdot & \cdot \\ 1 & -4 & 1 & 0 & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & 0 & 1 & -4 & 1 \\ \cdot & \cdot & 0 & 1 & -4 \end{pmatrix} e_l$$

$$= \lambda_l \begin{pmatrix} 3/2 & 0 & \cdot & \cdot & \cdot \\ 0 & 1 & 0 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & 0 & 1 & 0 \\ \cdot & \cdot & \cdot & 0 & 3/2 \end{pmatrix} e_l, \qquad \text{(A1)}$$

where, $e_l = (\phi_{1,l}, \phi_{2,l}, \ldots, \phi_{M,l})^{\mathrm{T}}$. By multiplying both sides of Eq. (A1) by A^{-1}, we obtain

$$\begin{pmatrix} -8/3 & 2/3 & 0 & \cdot & \cdot \\ 1 & -4 & 1 & 0 & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & 0 & 1 & -4 & 1 \\ \cdot & \cdot & 0 & 2/3 & -8/3 \end{pmatrix} e_l = \lambda_l e_l. \qquad \text{(A2)}$$

The system (A2) can also be written as

$$\phi_{i-1,l} - (4+\lambda_l)\,\phi_{i,l} + \phi_{i+1,l} = 0, \qquad 1 \leqslant i \leqslant M, \tag{A3}$$

subject to the boundary conditions

$$3\phi_{0,l} - 4\phi_{1,l} + \phi_{2,l} = 0, \tag{A4}$$

$$3\phi_{M+1,l} - 4\phi_{M,l} + \phi_{M-1,l} = 0. \tag{A5}$$

The solution to the above system is, after substituting $\lambda_l = -4 + 2\cos\theta_l$,

$$\phi_{i,l} = a\cos i\theta_l + b\sin i\theta_l, \tag{A6}$$

where a and b are arbitrary constants. Elimination of b by using Eq. (A4) leads to the equation for the eigenvectors,

$$\phi_{i,l} = c(-3\sin i\theta_l + 4\sin(i-1)\,\theta_l - \sin(i-2)\,\theta_l), \tag{A7}$$

where the constant c can be fixed by proper normalization that does not need to concern us here. Substituting the eigenvectors of Eq. (A7) into Eq. (A5) leads to the equation

$$9\sin(M+1)\,\theta_l - 24\sin M\theta_l + 22\sin(M-1)\,\theta_l - 8\sin(M-2)\,\theta_l + \sin(M-3)\,\theta_l = 0, \tag{A8}$$

for the eigenvalues λ_l (or θ_l).

ACKNOWLEDGMENTS

We thank the Ohio Supercomputer Center for granting us the computing time necessary to carry out this work. We are also grateful to Ted Scheick for the beneficial discussions we had with him during the course of this work.

REFERENCES

1. V. Babu and S. A. Korpela, *Comput. Fluids* **18** (2), 229 (1990).
2. B. L. Buzbee, G. H. Golub, and C. W. Nielson, *SIAM J. Numer. Anal.* **7** (4), 627 (1970).
3. F. Dorr, *SIAM Rev.* **12**, 248 (1970).
4. P. Swarztrauber, *SIAM Rev.* **19**, 490 (1977).
5. L. Farnell, *J. Comput. Phys.* **35**, 408 (1980).
6. A. R. Mitchell, *Computational Methods in Partial Differential Equations* (Wiley, New York, 1969).
7. R. Briley, *J. Comput. Phys.* **14**, 8 (1977).
8. M. Vinokur, *J. Comput. Phys.* **50**, 215 (1983).
9. M. Pickering, *An Introduction to Fast Fourier Transform Methods for Partial Differential Equations, with Applications* (Research Studies Press Ltd. Wiley, New York, 1986).
10. R. C. Le Bail, *J. Comput. Phys.* **9**, 44 (1972).
11. D. A. Anderson, J. C. Tannehill, and R. H. Pletcher, *Computational Fluid Mechanics and Heat Transfer* (Hemisphere, Washington, DC, 1984).
12. V. Babu, Ph.D. thesis, Ohio State University, Columbus, OH.