SECOND INTERNATIONAL CONFERENCE ON COMPUTATIONAL MODELLING OF FREE AND MOVING BOUNDARY PROBLEMS 93

INTERNATIONAL SCIENTIFIC ADVISORY COMMITTEE

A. Bermudez
J.C. Bruch Jr.
G. Carey
J.L. Chenot
G. Comini
S.T. Grilli
I. Herrera
M.S. Ingber
D.B. Ingham
K. Mizumura
H. Power
E. Rank
V.R. Voller
A.S. Wood

Acknowledgement is made to N. Tosaka *et al.* for the use of figure 9(b) on page 205 which appears on the front cover of this book.

Computational Modelling of Free and Moving Boundary Problems II

Editors:
L.C. Wrobel
Wessex Institute of Technology
&
C.A. Brebbia
Wessex Institute of Technology

Computational Mechanics Publications
Southampton Boston

9. _____, "On the convergence of multi-grid iterations," *Beitrage zur Numer. Math.*, **9** (1981), 213–239.
10. P. W. Hemker, "On the structure of an adaptive multi-level algorithm," *BIT*, **20** (1980), 289–301.
11. A. Jameson, "Acceleration of transonic potential flow calculations on arbitrary meshes by the multiple grid method," *Proc. of AIAA 4th Computational Fluid Dynamics Conference* (1979), 122–146.
12. S. McCormick, "An algebraic interpretation of multigrid methods," *SIAM J. Numerical Analysis*, **19** (1982), 548–560.
13. W. J. A. Mol, "Numerical solution of the Navier-Stokes equations by means of a multigrid method and Newton-iteration," *Seventh International Conference on Numerical Methods in Fluid Dynamics*, ed. W. C. Reynolds and R. W. MacCormack, *Lecture Notes in Physics*, **141** (1981), Springer-Verlag.
14. P. Swarztrauber, "The Methods of cyclic reduction, Fourier analysis, and the FACR algorithm for the discrete solution of Poisson's equation in a rectangle," *SIAM Review*, **19** (1977), 490–501.
15. P. Wesseling, "The Rate of convergence of a multiple grid method," *Numerical Analysis Proceedings, Dundee 1979*, ed. G. A. Watson, *Lecture Notes in Mathematics*, **773** (1980), Springer-Verlag.
16. P. Wesseling and P. Sonneveld, "Numerical experiments with a multiple grid and a preconditioned Lanczos method," *Approximation Problems for Navier-Stokes Equations*, ed. R. Rautmann, *Lecture Notes in Mathematics*, **771** (1980), Springer-Verlag.

FAST POISSON SOLVERS

Paul N. Swarztrauber

1. INTRODUCTION

The purpose of this chapter is to introduce the reader to a particular class of efficient computer-aided methods for solving Poisson's equation (1.1). Given a function $f(x, y)$ defined on a rectangle $a < x < b$; $c < y < d$, then we wish to find a function $u(x, y)$ such that

$$u_{xx}(x, y) + u_{yy}(x, y) = f(x, y). \tag{1.1}$$

In general there will be many functions that satisfy (1.1) and one must specify boundary conditions to obtain a unique solution. For example, if $u(x, y)$ is specified on the boundary of the rectangle and satisfies (1.1) on the rectangle, then the solution is unique. In this chapter we will solve Poisson's equation subject to several standard boundary conditions.

There are two main reasons for interest in the efficient solution of Poisson's equation. First, the equation arises in many areas of science and engineering and, second, its solution can require a substantial amount of computing time. As we will show, a single

solution can be obtained with only a modest amount of computing time; however, in many applications, repeated solutions are necessary, which can require a significant amount of computing time.

For example, in models of atmospheric dynamics, Poisson's equation is imbedded in a system of time-dependent partial differential equations. This system is solved computationally by numerical integration in which the solution is tabulated at a sequence of time levels, t_n. A single solution of such a system may require many hours of computer time in which Poisson's equation is solved, possibly several thousand times. In addition, the system of equations will be solved many times in order to determine the effect of changing model parameters, i.e., to simulate different atmospheric conditions.

Similar calculations are performed in many other disciplines including oceanography, stellar physics, electrostatics and magnetics, reactor design and oil reservoir simulation, to name just a few. Indicative of the great diversity of applications is the work of Baum and Rehm [14]. Using model calculations much like those discussed above, they have been able to analyze the propagation of a fire in a room.

With the advent of the fast Poisson solvers, the time required to compute solutions has dropped by at least an order of magnitude and up to a factor of 50. This has resulted in a marked improvement in the overall performance of the large model calculations. Prior to the fast Poisson solvers, at least 50% of the computing time for a model was used in the repeated solution of Poisson's equation using an iterative method such as successive overrelaxation (SOR). In the same models, but with a fast Poisson solver, only about 10% of the total computing time is spent solving Poisson's equation.

When a problem of this type is encountered, usually the first goal is to find a closed-form solution, i.e., a formula into which we can plug the coordinates of any point in the rectangle and out of which comes the exact solution $u(x, y)$. More often than not, such closed-form solutions cannot be found, but even when they are, their use for tabulating the solution may be inefficient when compared to the methods that will be discussed in this chapter.

A quite satisfactory alternative to a closed-form solution is an approximate solution of (1.1) in the form of a tabulation. Given

integers M and N we first define a grid of points (x_i, y_j) for

$$x_i = a + i\delta x \qquad i = 0, \ldots, M \tag{1.2}$$

$$y_j = c + j\delta y \qquad j = 0, \ldots, N, \tag{1.3}$$

where $\delta x = \dfrac{b-a}{M}$ and $\delta y = \dfrac{d-c}{N}$. The goal is to compute an approximate value $u_{i,j}$ of the solution $u(x_i, y_j)$ at each of the grid points. To this end we require the $u_{i,j}$ to satisfy a finite difference approximation of (1.1), which is developed in the next section. The fact that $u_{i,j}$ is only an approximate solution is of little concern since it is known (for example, see [19]) that we can make it as accurate as we like by increasing M and N in order to make δx and δy sufficiently small. Of course, this is limited by the available computing resource, but for most applications one can obtain a result that is accurate to three or four decimal digits with only a moderate amount of computing, particularly since the advent of the fast Poisson solvers.

Fast Poisson solvers came into being in 1965 when Hockney [15] used Fourier analysis to solve Poisson's equation. This approach was superior to any other at that time and it continues to be one of the most successful of the fast Poisson solvers. The method was developed further and in more detail in [16], which is also an important paper on the subject. Since then a great deal has been written about the method, and we will refer to some of this literature throughout the chapter.

In both [15] and [16], Hockney discusses a method, developed in collaboration with Golub, called cyclic reduction. In its original form, cyclic reduction was quite unstable. However this fact was initially not noticed since cyclic reduction was combined with the Fourier method in a manner that reduced both error growth and computing time. This combination is called the FACR(ℓ) algorithm. Nevertheless, cyclic reduction by itself applies to a larger class of problems than the Fourier method and hence Buneman [5] made a significant contribution when he stabilized the cyclic reduction method, making it a viable "fast Poisson solver" with all the attributes that had distinguished the Fourier method. The method of cyclic reduction, including Buneman's algorithm and variants, is

discussed in detail by Buzbee et al. [6]. This is one of the most important papers on cyclic reduction and is recommended for those who wish to continue reading about this subject. The Fourier and cyclic reduction methods are much superior to the iterative or relaxation methods that were previously used to solve Poisson's equation.

a) They require much less computing time; the speed-up was comparable to the speed-up achieved by using the fast Fourier transform over the slow Fourier transform. There is a vast amount of literature about the fast Fourier transform. For an account of recent work and a list of references see Swarztrauber [29].
b) They require half the storage of the previous methods. The computations can be done in place, i.e., the solution can be stored over the right hand side $f_{i,j} = f(x_i, y_j)$. This is an important advantage for these problems, which characteristically require a large amount of storage.
c) The Fourier and cyclic reduction methods are classified as direct methods. Using a direct method, the exact solution is obtained in a finite number of operations. This is in contrast with the iterative methods that theoretically require an infinite number of operations to obtain an exact solution. Of course, in practice, the iteration stops once the error has been reduced to an acceptable level called the error tolerance. For many problems the selection of the error tolerance is a nontrivial matter and a large amount of computing time can be wasted if it is improperly specified. The point of this is that an error tolerance is not needed when using a direct method. It is an important but little publicized advantage of the direct methods.
d) Because of the speed and reliability of the fast Poisson solvers, they are attractive candidates for implementation in software packages. Indeed, several packages exist, including that of Swarztrauber and Sweet [30] which is specifically designed for use by scientists. Machura and Sweet [21] have recently given a comprehensive description of software that is available for partial differential equations.

It is important to note that, although the fast Poisson solvers are clearly superior to the iterative methods for problems that both methods can solve, the iterative methods can be used on a much larger class of problems and still provide the most effective means of solving many elliptic partial differential equations. Brandt [4] has developed an efficient iterative method called the multigrid method that can be applied to a wide class of problems. The conjugate gradient method is also an effective method which, along with multigrid, is currently receiving much attention (see Concus, Golub and O'Leary [9]).

To a large extent the Fourier and cyclic reduction methods complement one another. The fast Fourier transform is very inefficient for prime M or N. On the other hand, as a result of the work of Sweet [31], cyclic reduction is quite efficient for any N. This point may not be too important since N or M can usually be selected as non prime, or at least highly composite, in which case the Fourier method has been found to be somewhat faster than cyclic reduction. The difference is not great and depends on a number of factors including the machine, compiler, and program [17], [28].

Although in practice the Fourier method is faster than cyclic reduction, it is important to note that cyclic reduction can be used to solve a larger class of problems. Schumann and Sweet [25] have developed variants of cyclic reduction for "staggered" grids and Swarztrauber [26] has developed a generalized cyclic reduction algorithm for solving an arbitrary separable elliptic equation. The degree to which the Fourier and cyclic reduction methods complement one another is exemplified by the fact that they can be combined into a third method, called the FACR method, which is faster than either the Fourier or cyclic reduction methods used separately. More on this later.

Prior to the seventies, the subject of fast Poisson solvers was limited to the Fourier and cyclic reduction methods. Dorr [12] reviews the status of these methods at that time as well as a number of other "not so fast" direct methods. In the past decade, a number of important fast Poisson solvers have been developed, and consequently, for those individuals who intend to pursue the subject, we will devote the remainder of this section to a brief review of some of the more recent results.

In order to quantify the relevance of these developments we need some measure of the computing resource that is required by a method. If we define an operation as either a multiplication, division, addition, or subtraction then it is common to use the number of operations, or operation count, as a measure of the resource that is required. For an N by N grid the total operation count consists of an expression which has several terms, each of which is a function of N. The *asymptotic* operation count is defined as the term in the operation count that dominates the expression as N gets very large. We will ignore any proportionality constants. For example, the asymptotic operation count of SOR is proportional to $N^3 \log N$, which is expressed by $O(N^3 \log N)$. The alternating-direction implicit (ADI) method [19], which is also an iterative method, has an asymptotic operation count $O(N^2 \log^2 N)$. The asymptotic operation count for both the Fourier and cyclic reduction methods is $O(N^2 \log N)$ but as mentioned above, the Fourier method is somewhat faster in practice because it has a smaller proportionality constant. These counts are compared in detail in [28] and [32].

The fastest known numerically stable method for solving Poisson's equation combines both the Fourier and cyclic reduction methods. Hockney [16] introduced the FACR(ℓ) algorithm in which ℓ steps of cyclic reduction were combined with the Fourier method. He also observed experimentally that an optimum value of ℓ existed. Swarztrauber [28] showed that ℓ proportional to $\log N$ is optimal and that with this value of ℓ the asymptotic operation count is $O(N^2 \log\log N)$. For a 64 by 64 grid the FACR method is only about 20% faster than the Fourier method since there is not that much difference between $\log N$ and $\log\log N$. It is interesting to note that, for all practical purposes, $\log\log N$ is constant. For an enormous grid, say, $N = 10^6$, $\log_2 \log_2 N \approx 4.3$; hence for grids that are within current or even anticipated capabilities of computers, $\log_2 \log_2 N$ is bounded by 5.

In spite of this observation, a question very naturally arises, namely, does a method exist with an asymptotic operation count that is proportional to N^2? The answer to this question is a qualified yes. Actually, there are several methods but they are not as accurate as the methods that we have already mentioned. Dorr [13] describes the shooting or marching algorithm which is $O(N^2)$

and he observes that it has been around for a number of years. The difficulty with this method is that it is extremely unstable, i.e., the error as a function of N grows rapidly and the method cannot be used in practice except for quite small grids where N is less than about 10, depending on the accuracy of the computer.

Roughly speaking, algorithms are classified as stable or unstable depending on whether the asymptotic error grows as a monomial or exponential function of N, respectively. On the other hand, the utility of a particular algorithm depends on the proportionality constant and the size of N. Nevertheless, this definition of stability seems quite satisfactory from a practical standpoint since the algorithms that are classified as unstable are generally not found to be useful—until they are stabilized like cyclic reduction!

Bank [1] and Bank and Rose [2], [3] develop a multiple marching method in which the marching and Fourier methods are combined into a new method that also has an asymptotic operation count that is proportional to N^2. Although it is not as accurate as the Fourier method, it is sufficiently accurate to make it useful in practice. Working independently, Lorenz [20] also developed a multiple marching method that he implemented in a FORTRAN program on the CDC 7600. He solved Poisson's equation on a 64 by 64 grid in .0195 seconds, which is equivalent to *two* SOR iterations. This is the fastest known solution of this problem (on the 7600). For the same problem the Fourier method took .062 seconds and cyclic reduction took .073 seconds. Each of these times are at least an order of magnitude less than the time required by SOR.

The multiple marching method gains its speed at some expense of accuracy. Lorenz's solution was accurate to 8 decimal digits compared to 12 digits for the Fourier method. On the other hand 8 digits is certainly sufficient for most applications and so it seems reasonable to use the multiple marching method and take advantage of the speed. But one should also note that if the multiple-shooting method were run on a less accurate computer, say, one with 8 digits compared with the 15 digits on the CDC 7600, then only one or possibly none of the digits would be accurate. As mentioned before, the best method depends on many factors. Hockney has examined the adaption of fast Poisson solvers to vector and parallel computers in [18].

There are other interesting $O(N^2)$ methods, including the total reduction method of Schroeder et al. [24] and the point-cyclic reduction method of Detyna [10]. Each of these methods has its own particular advantages but again none are as accurate as the Fourier method. Important work remains to be done in this area; in particular the question remains, does an $O(N^2)$ method exist that is as accurate as the Fourier method? If so, then it is asymptotically the best, since there are N^2 values of $u_{i,j}$.

The fast Poisson solvers, at least as they were originally developed, could be used only if the problem was defined on a rectangle. Much has changed since then and indeed now the solution can be obtained on irregular regions with very little sacrifice in performance. Buzbee et al. [7] use the capacitance matrix method with the fast Poisson solver to compute the solution on an irregular region. Proskurowski and Widlund [23] have developed the method further and Proskurowski [22] has implemented it in software. Fast Poisson solvers can also be used to solve more complex elliptic equations using a method called D'yakonov iteration, which is the approach taken by Concus and Golub [8].

Since this chapter is intended as an introduction to fast Poisson solvers, we will limit our discussion in succeeding sections to the Fourier and cyclic reduction methods. Because of this limitation we can describe these methods in some detail with the hope that the reader can, in fact, gain a working knowledge of the methods. The exposition proceeds in terms of examples in which four problems (A through D) are solved. In the next section, these problems are defined and Poisson's equation is solved in each, but subject to different boundary conditions. A large system of linear equations for the approximate solution of each problem is also derived in Section 2. These systems are solved using the Fourier method in Section 3 and the cyclic reduction method in Section 4. In Section 5 we deal with several computational problems that must be solved before the methods are useful in practice. We show that certain problems can only be solved in the least squares sense. We also provide an efficient method for computing the matrix polynomials that are associated with cyclic reduction. Finally, at the end of the section, we present Buneman's stable cyclic reduction algorithm.

2. FINITE DIFFERENCE APPROXIMATIONS

In this section we will define and discretize four sample problems by replacing the continuous differential problem with a large system of equations for the approximate solution. It would be ideal to obtain an exact closed-form solution of Poisson's equation; however, as noted in the introduction, this is not possible in most cases and hence we look for an approximate solution. The goal is to compute an approximate value $u_{i,j}$ of the solution $u(x_i, y_j)$ at each of the grid points defined in (1.1) and (1.2).

If the $u_{i,j}$ are solutions to an approximate Poisson equation then it seems reasonable that they will approximate the solution of the Poisson equation. This intuitive observation is, in fact, correct under most circumstances, as shown by formal mathematical development given in a number of places, including [19]. Therefore, we will require the $u_{i,j}$ to satisfy a finite difference approximation to the Poisson equation that is obtained from (1.1) by replacing the second partial derivatives by centered second-difference quotients

$$\frac{u_{i+1,j}-2u_{i,j}+u_{i-1,j}}{\delta x^2}+\frac{u_{i,j+1}-2u_{i,j}+u_{i,j-1}}{\delta y^2}=f_{i,j}. \tag{2.1}$$

The problem now becomes one of finding the solution of a very large linear system of equations with right-hand side $f_{i,j}=f(x_i, y_j)$ and unknowns $u_{i,j}$. The system (2.1) is not yet complete and must be augmented with linear equations obtained from the boundary conditions. We will consider three possible boundary conditions that occur frequently when solving the Poisson equation.

a) **The value of the solution is specified at the boundary.**

If the solution is specified at $x=a$, for example, then we are given a function, say, $b_a(y)$, such that $u(a, y)=b_a(y)$ for $c \leqslant y \leqslant d$. In this case $u_{0,j}=b_a(y_j)$ is known and the index i of the unknowns

begins with $i=1$. The equation for $i=1$ is obtained by substituting the boundary condition into (2.1)

$$\frac{u_{2,j}-2u_{1,j}}{\delta x^2}+\frac{u_{1,j+1}-2u_{1,j}+u_{1,j-1}}{\delta y^2}=f_{1,j}-\frac{b_a(y_j)}{\delta x^2}. \tag{2.2}$$

Hence the boundary conditions enter the linear system of equations through the right hand side.

b) **The derivative of the solution is specified at the boundary.**

If the derivative of the solution is specified at $x=a$ then a function, say, $b_a(y)$, is given such that $\frac{\partial u}{\partial x}(a,y)=b_a(y)$. To obtain the discrete form of this boundary condition, replace the partial derivative with a finite difference approximation centered at $x=a$ or $i=0$:

$$\frac{u_{1,j}-u_{-1,j}}{2\delta x}=b_a(y_j). \tag{2.3}$$

Note that this introduces a point outside the boundary at $x_{-1}=a-\delta x$ and a corresponding unknown $u_{-1,j}$. If we evaluate (2.1) at $x=a$ or $i=0$, then we obtain

$$\frac{u_{1,j}-2u_{0,j}+u_{-1,j}}{\delta x^2}+\frac{u_{0,j+1}-2u_{0,j}+u_{0,j-1}}{\delta y^2}=f_{0,j}. \tag{2.4}$$

The unknown $u_{-1,j}$ outside the boundary can now be eliminated between equations (2.3) and (2.4):

$$\frac{2u_{1,j}-2u_{0,j}}{\delta x^2}+\frac{u_{0,j+1}-2u_{0,j}-u_{0,j-1}}{\delta y^2}=f_{0,j}+\frac{2}{\delta x}b_a(y_j). \tag{2.5}$$

When a derivative boundary condition is specified, system (2.1) is augmented with (2.5), and the boundary condition enters the linear system through the right-hand side. Since the solution is unknown at $x=a$, the i index of the unknowns $u_{i,j}$ begins with $i=0$.

The temporary introduction of a point at $x=a-\delta x$ is a common computational technique called the virtual point method. By using this method, all finite differences are centered with the result that all the approximations have the same accuracy. If the solution is sufficiently differentiable at the boundary, then this technique can be justified by the fact that the solution can be continued across the boundary.

c) **The solution is periodic.**

If $u(a+x,y)=u(b+x,y)$ for all x, then the solution is said to be periodic in the x direction. In terms of the discrete variables, this condition takes the form $u_{-1,j}=u_{M-1,j}$ and $u_{M,j}=u_{0,j}$. The set $u_{i,j}$ for $i=0,\ldots,M-1$ is the smallest set of nonredundant unknowns. If they are determined then $u_{i,j}$ can be determined everywhere.

Since the solution is periodic, we know that the Poisson equation and (2.1) are valid at all x and in particular at $x=a$ and $x=b$. If we evaluate (2.1) at $i=0$ subject to the discrete periodic boundary condition, then we obtain

$$\frac{u_{1,j}-2u_{0,j}+u_{M-1,j}}{\delta x^2}+\frac{u_{0,j+1}-2u_{0,j}+u_{0,j-1}}{\delta y^2}=f_{0,j}. \tag{2.6}$$

Similarly, if we evaluate (2.1) at $i=M-1$ subject to the discrete periodic boundary conditions, then we obtain

$$\frac{u_{0,j}-2u_{M-1,j}+u_{M-2,j}}{\delta x^2}+\frac{u_{M-1,j+1}-2u_{M-1,j}+u_{M-1,j-1}}{\delta y^2}=f_{M-1,j}. \tag{2.7}$$

When a periodic boundary condition is specified, system (2.1) is augmented with both (2.6) and (2.7).

This development must be repeated for the remaining sides of the rectangle with their corresponding boundary conditions in order to obtain a complete set of equations for the unknowns $u_{i,j}$. Also, the equations at the corners contain contributions from both boundary conditions. There are a number of possible combinations of boundary conditions. Any of the conditions given above can occur in either the x or the y direction, for a total of 25 possible combinations. In the remainder of this section we will describe four of these combinations in some detail. A complete set of finite difference equations will be derived for each of these four problems, including the boundary and corner equations.

As we will see, the main difference between the problems occurs on the boundary. The problems were chosen to illustrate the solution of the Poisson equation subject to all possible combinations of the boundary conditions. Although only four problems are described, they can be used to develop the method for any combination of boundary conditions. For example, if the solution is specified at $x=a$ and $x=b$ and the derivative of the solution is specified at $y=c$ and $y=d$, then the discretization in the x direction would follow the development given in Problem A, which follows, and the discretization in the y direction would follow the development that is given in Problem B. Once the finite difference equations are derived, the problem reduces to solving a very large system of equations. It *is* therefore quite reasonable to ask why Gaussian elimination is *not* used to solve the equations. The answer is that Gaussian elimination is *less* efficient than the direct methods that we will discuss. The asymptotic operation count for Gaussian elimination is $O(N^4)$ compared with $O(N^2\log N)$ for the direct methods that are developed in the subsequent sections.

Problem A. Given functions $b_a(y)$, $b_b(y)$, $b_c(x)$ and, $b_d(x)$ defined on the boundary and $f(x, y)$ defined in the interior of the rectangle, we wish to determine a function $u(x, y)$ such that

$$u(a, y)=b_a(y), \qquad u(b, y)=b_b(y),$$
$$u(x, c)=b_c(x), \qquad u(x, d)=b_d(x)$$

and

$$u_{xx}(x, y)+u_{yy}(x, y)=f(x, y) \tag{2.8}$$

are on the interior of the rectangle. This problem is known as Dirichlet's problem.

As we discussed above, the goal is to find an approximate solution of this problem, and to that end we proceed to derive the finite difference approximations. There are a total of $(M-1)(N-1)$ unknowns $u_{i,j}$ on the interior of the region, and hence we must derive $(M-1)(N-1)$ equations for their solution. The equations on the interior of the region are not affected by the boundary condition, and, therefore, we can obtain the following $(M-3)(N-3)$ equations from (2.1) for $i=2,\ldots,M-2$ and $j=2,\ldots,N-2$:

$$\frac{u_{j+1,j}-2u_{i,j}+u_{i-1,j}}{\delta x^2}+\frac{u_{i,j+1}-2u_{i,j}+u_{i,j-1}}{\delta y^2}=f_{i,j}=g_{i,j}. \tag{2.9}$$

The grid function $g_{i,j}$ is introduced in order to facilitate the incorporation of the boundary conditions into the right-hand side of the equations. As we will show below, $g_{i,j}$ is equal to $f_{i,j}$ except on or near the boundary, where $g_{i,j}$ is equal to $f_{i,j}$ plus a term or terms that result from the boundary conditions.

In this example, we assume that the solution is specified on all sides of the rectangle. Thus we assume that we are given four functions, $b_a(y)$, $b_b(y)$, $b_c(x)$ and, $b_d(x)$ that specify the value of the solution on the boundaries $x=a$, $x=b$, $y=c$ and, $y=d$, respectively. If we repeat the discretization of boundary condition (a) at all four boundaries, then we obtain the following equations from (2.1), which is altered near the boundary. At $x=a+\delta x$ we obtain the following equations for $i=1$ and $j=2,\ldots,N-2$:

$$\frac{u_{2,j}-2u_{1,j}}{\delta x^2}+\frac{u_{1,j+1}-2u_{1,j}+u_{1,j-1}}{\delta y^2}=f_{1,j}-\frac{b_a(y_j)}{\delta x^2}=g_{1,j}. \tag{2.10}$$

At $x = b - \delta x$ we obtain the following equations for $i = M-1$ and $j = 2, \ldots, N-2$:

$$\frac{-2u_{M-1,j} + u_{M-2,j}}{\delta x^2} + \frac{u_{M-1,j+1} - 2u_{M-1,j} + u_{M-1,j-1}}{\delta y^2}$$

$$= f_{M-1,j} - \frac{b_b(y_j)}{\delta x^2} = g_{M-1,j}. \tag{2.11}$$

At $y = c + \delta y$ we obtain the following equations for $j = 1$ and $i = 2, \ldots, M-2$:

$$\frac{u_{i+1,1} - 2u_{i,1} + u_{i-1,1}}{\delta x^2} + \frac{u_{i,2} - 2u_{i,1}}{\delta y^2} = f_{i,1} - \frac{b_c(x_i)}{\delta y^2} = g_{i,1}. \tag{2.12}$$

at $y = d - \delta y$ we obtain the following equations for $j = N-1$ and $i = 2, \ldots, M-2$:

$$\frac{u_{i+1,N-1} - 2u_{i,N-1} + u_{i-1,N-1}}{\delta x^2} + \frac{-2u_{i,N-1} + u_{i,N-2}}{\delta y^2}$$

$$= f_{i,N-1} - \frac{b_d(x_i)}{\delta y^2} = g_{i,N-1}. \tag{2.13}$$

Near the four corners of the rectangle, the right-hand side of (2.9) is modified to include contributions from two boundary conditions. The discrete Poisson equation (2.9) centered at $i = j = 1$ takes the form

$$\frac{u_{2,1} - 2u_{1,1}}{\delta x^2} + \frac{u_{1,2} - 2u_{1,1}}{\delta y^2} = f_{1,1} - \frac{b_a(y_1)}{\delta x^2} - \frac{b_c(x_1)}{\delta y^2} = g_{1,1}. \tag{2.14}$$

At $i = M-1$ and $j = 1$,

$$\frac{-2u_{M-1,1} + u_{M-2,1}}{\delta x^2} + \frac{u_{M-1,2} - 2u_{M-1,1}}{\delta y^2}$$

$$= f_{M-1,1} - \frac{b_b(y_1)}{\delta x^2} - \frac{b_c(x_{M-1})}{\delta y^2} = g_{M-1,1}. \tag{2.15}$$

At $i = 1$ and $j = N-1$,

$$\frac{u_{2,N-1} - 2u_{1,N-1}}{\delta x^2} + \frac{-2u_{1,N-1} + u_{1,N-2}}{\delta y^2}$$

$$= f_{1,N-1} - \frac{b_a(y_{N-1})}{\delta x^2} - \frac{b_d(x_1)}{\delta y^2} = g_{1,N-1}. \tag{2.16}$$

At $i = M-1$ and $j = N-1$,

$$\frac{-2u_{M-1,N-1} + u_{M-2,N-1}}{\delta x^2} + \frac{-2u_{M-1,N-1} + u_{M-1,N-2}}{\delta y^2}$$

$$= f_{M-1,N-1} - \frac{b_b(y_{N-1})}{\delta x^2} - \frac{b_d(x_{M-1})}{\delta y^2} = g_{M-1,N-1}. \tag{2.17}$$

Equations (2.9) through (2.17) total $(M-1)(N-1)$ equations for the $(M-1)(N-1)$ unknowns $u_{i,j}$ in the interior of the rectangle.

Problem B. Given functions $b_a(y)$, $b_b(y)$, $b_c(x)$ and $b_d(x)$ defined on the boundary and $f(x, y)$ defined in the interior of the rectangle, we wish to determine a function $u(x, y)$ such that

$$\frac{\partial u}{\partial x}(a, y) = b_a(y) \tag{2.18}$$

$$\frac{\partial u}{\partial x}(b, y) = b_b(y) \tag{2.19}$$

$$\frac{\partial u}{\partial y}(x, c) = b_c(x) \tag{2.20}$$

$$\frac{\partial u}{\partial y}(x, d) = b_d(x) \tag{2.21}$$

and on the interior of the rectangle

$$u_{xx}(x,y)+u_{yy}(x,y)=f(x,y). \tag{2.22}$$

As before, we seek an approximate solution to this problem, which is called the Neumann problem. The finite difference equations can be obtained following the discretization of the derivative boundary condition (b) that was given at the beginning of this section. The solution is unknown on the boundaries and, hence, there are $(M+1)(N+1)$ unknowns, namely, $u_{i,j}$ for $i=0,\dots,M$ and $j=0,\dots,N$. Therefore we must determine a system of $(M+1)(N+1)$ equations in these unknowns. The equations on the interior of the region are not affected by the boundary condition and, hence, we obtain the following $(M-1)(N-1)$ equations from (2.1) for $i=1,\dots,M-1$ and $j=1,\dots,N-1$:

$$\frac{u_{i+1,j}-2u_{i,j}+u_{i-1,j}}{\delta x^2}+\frac{u_{i,j+1}-2u_{i,j}+u_{i,j-1}}{\delta y^2}=f_{i,j}=g_{i,j}. \tag{2.23}$$

Following the discretization of boundary condition (b) we obtain the following equations for $i=0$ and $j=1,\dots,N-1$:

$$\frac{2u_{1,j}-2u_{0,j}}{\delta x^2}+\frac{u_{0,j+1}-2u_{0,j}+u_{0,j-1}}{\delta y^2}=f_{0,j}+\frac{2}{\delta x}b_a(y_j)=g_{0,j}. \tag{2.24}$$

At $x=b$, we obtain the following equations for $i=M$ and $j=1,\dots,N-1$

$$\frac{-2u_{M,j}+2u_{M-1,j}}{\delta x^2}+\frac{u_{M,j+1}-2u_{M,j}+u_{M,j-1}}{\delta y^2}=f_{M,j}-\frac{2}{\delta x}b_b(y_j)=g_{M,j}. \tag{2.25}$$

At $y=c$ we obtain the following equations for $i=1,\dots,M-1$ and $j=0$:

$$\frac{u_{i+1,0}-2u_{i,0}+u_{i-1,0}}{\delta x^2}+\frac{2u_{i,1}-2u_{i,0}}{\delta y^2}=f_{i,0}+\frac{2}{\delta y}b_c(x_i)=g_{i,0}. \tag{2.26}$$

At $y=d$ we obtain the following equations for $i=1,\dots,M-1$ and $j=N$:

$$\frac{u_{i+1,N}-2u_{i,N}+u_{i-1,N}}{\delta x^2}+\frac{-2u_{i,N}+2u_{i,N-1}}{\delta y^2}=f_{i,N}-\frac{2}{\delta y}b_d(x_i)=g_{i,N}. \tag{2.27}$$

The remaining equations are obtained at the corners of the rectangle. At $i=j=0$ the right-hand side includes contributions from two boundary conditions,

$$\frac{2u_{1,0}-2u_{0,0}}{\delta x^2}+\frac{2u_{0,1}-2u_{0,0}}{\delta y^2}=f_{0,0}+\frac{2}{\delta x}b_a(c)+\frac{2}{\delta y}b_c(a)=g_{0,0}. \tag{2.28}$$

At $i=M$ and $j=0$, we obtain

$$\frac{2u_{M-1,0}-2u_{M,0}}{\delta x^2}+\frac{2u_{M,1}-2u_{M,0}}{\delta y^2}=f_{M,0}-\frac{2}{\delta x}b_b(c)+\frac{2}{\delta y}b_c(b)=g_{M,0}. \tag{2.29}$$

At $i=0$ and $j=N$, we obtain

$$\frac{2u_{1,N}-2u_{0,N}}{\delta x^2}+\frac{2u_{0,N-1}-2u_{0,N}}{\delta y^2}=f_{0,N}+\frac{2}{\delta x}b_a(d)-\frac{2}{\delta y}b_d(a)=g_{0,N}. \tag{2.30}$$

Finally, at $i = M$ and $j = N$ we obtain

$$\frac{2u_{M-1,N}-2u_{M,N}}{\delta x^2}+\frac{2u_{M,N-1}-2u_{M,N}}{\delta y^2}$$

$$=f_{M,N}-\frac{2}{\delta x}b_b(d)-\frac{2}{\delta y}b_d(b)=g_{M,N}. \tag{2.31}$$

Equations (2.23) through (2.31) total $(M+1)(N+1)$ equations for the $(M+1)(N+1)$ unknowns $u_{i,j}$ on the rectangle. Actually this system of equations is singular and its "solution" presents some special difficulties that are discussed in Section 5.

Problem C. Given functions $b_a(y)$, $b_b(y)$, $b_c(x)$ and, $b_d(x)$ defined on the boundary and $f(x, y)$ defined in the interior of the rectangle, we wish to determine a function $u(x, y)$ such that

$$u(a, y)=b_a(y) \tag{2.32}$$

$$\frac{\partial u}{\partial x}(b, y)=b_b(y) \tag{2.33}$$

$$u(x, c)=b_c(x) \tag{2.34}$$

$$\frac{\partial u}{\partial y}(x, d)=b_d(x) \tag{2.35}$$

and on the interior of the rectangle

$$u_{xx}(x, y)+u_{yy}(x, y)=f(x, y). \tag{2.36}$$

As before, we seek an approximate solution of the problem and to this end we will derive the finite difference equations using the discretizations that were given in examples (a) and (b) at the beginning of this section. The solution is given at $x = a$ and $y = c$; however, it is unknown at $x = b$ and $y = d$ and, hence, there are MN unknowns, namely, $u_{i,j}$ for $i=1,\ldots,M$ and $j=1,\ldots,N$. Therefore we must derive a system of MN equations in these

unknowns. The equations on the interior of the region are not affected by the boundary condition and, therefore, we can obtain the following $(M-2)(N-2)$ equations from (2.1) for $i=2,\ldots,M-1$ and $j=2,\ldots,N-1$:

$$\frac{u_{i+1,j}-2u_{i,j}+u_{i-1,j}}{\delta x^2}+\frac{u_{i,j+1}-2u_{i,j}+u_{i,j-1}}{\delta y^2}=f_{i,j}=g_{i,j}. \tag{2.37}$$

Following the discretization of boundary condition (a) in Section 2, we obtain the following equations for $i=1$ and $j=2,\ldots,N-1$:

$$\frac{u_{2,j}-2u_{1,j}}{\delta x^2}+\frac{u_{1,j+1}-2u_{1,j}+u_{1,j-1}}{\delta y^2}=f_{1,j}-\frac{1}{\delta x^2}b_a(y_j)=g_{1,j}. \tag{2.38}$$

At $x=b$ we obtain the following equations for $i=M$ and $j=2,\ldots,N-1$:

$$\frac{-2u_{M,j}+2u_{M-1,j}}{\delta x^2}+\frac{u_{M,j+1}-2u_{M,j}+u_{M,j-1}}{\delta y^2}$$

$$=f_{M,j}-\frac{2}{\delta x}b_b(y_j)=g_{M,j}. \tag{2.39}$$

At $y=c+\delta y$ we obtain the following equations for $i=2,\ldots,M-1$ and $j=1$:

$$\frac{u_{i+1,1}-2u_{i,1}+u_{i-1,1}}{\delta x^2}+\frac{u_{i,2}-2u_{i,1}}{\delta y^2}=f_{i,1}-\frac{1}{\delta y^2}b_c(x_i)=g_{i,1}. \tag{2.40}$$

At $y=d$ we obtain the following equations for $i=2,\ldots,M-1$ and $j=N$:

$$\frac{u_{i+1,N}-2u_{i,N}+u_{i-1,N}}{\delta x^2}+\frac{-2u_{i,N}+2u_{i,N-1}}{\delta y^2}=f_{i,N}-\frac{2}{\delta y}b_d(x_i)$$

$$=g_{i,N}. \tag{2.41}$$

The remaining equations are obtained at or near the corners of the rectangle. At $i=1$ and $j=1$ the right-hand side contains contributions from two boundary conditions

$$\frac{u_{2,1}-2u_{1,1}}{\delta x^2}+\frac{u_{1,2}-2u_{1,1}}{\delta y^2}$$

$$=f_{1,1}-\frac{1}{\delta x^2}b_a(\delta y)-\frac{1}{\delta y^2}b_c(\delta x)=g_{1,1}. \tag{2.42}$$

At $i=M$ and $j=1$ we obtain

$$\frac{2u_{M-1,1}-2u_{M,1}}{\delta x^2}+\frac{u_{M,2}-2u_{M,1}}{\delta y^2}$$

$$=f_{M,1}-\frac{2}{\delta x}b_b(\delta y)-\frac{1}{\delta y^2}b_c(b)=g_{M,1}. \tag{2.43}$$

At $i=1$ and $j=N$ we obtain

$$\frac{u_{2,N}-2u_{1,N}}{\delta x^2}+\frac{2u_{1,N-1}-2u_{1,N}}{\delta y^2}=f_{1,N}-\frac{1}{\delta x^2}b_a(d)-\frac{2}{\delta y}b_d(\delta x)$$

$$=g_{1,N}. \tag{2.44}$$

Finally at $i=M$ and $j=N$ we obtain

$$\frac{2u_{M-1,N}-2u_{M,N}}{\delta x^2}+\frac{2u_{M,N-1}-2u_{M,N}}{\delta y^2}$$

$$=f_{M,N}-\frac{2}{\delta x}b_b(d)-\frac{2}{\delta y}b_d(b)=g_{M,N}. \tag{2.45}$$

Equations (2.37) through (2.45) total MN equations for the MN unknowns $u_{i,j}$.

Problem D. Given a function $f(x,y)$, we seek a function $u(x,y)$ that is periodic in both x and y, i.e., for any x and y

$$u(a+x,y)=u(b+x,y) \quad \text{and} \quad u(x,c+y)=u(x,d+y) \tag{2.46}$$

and, on the interior of the rectangle,

$$u_{xx}(x,y)+u_{yy}(x,y)=f(x,y). \tag{2.47}$$

The development of the finite difference equations for the periodic boundary conditions follows the development given for condition (c) at the beginning of the section. The solution at $x=b$ and $y=d$ is identical to that at $x=a$ and $y=d$, respectively, and, therefore, it is not necessary to compute the solution on either $x=b$ or $y=d$. However, the solution is unknown at all other points on the rectangle and hence there are MN unknowns; namely, $u_{i,j}$ for $i=0,\ldots,M-1$ and $j=0,\ldots,N-1$. Therefore we wish to derive a system of MN finite difference equations in these unknowns. The equations on the interior of the region are not affected by the boundary condition and hence we can obtain the following $(M-2)(N-2)$ equations from (2.1) for $i=1,\ldots,M-2$ and $j=1,\ldots,N-2$:

$$\frac{u_{i+1,j}-2u_{i,j}+u_{i-1,j}}{\delta x^2}+\frac{u_{i,j+1}-2u_{i,j}+u_{i,j-1}}{\delta y^2}=f_{i,j}=g_{i,j}. \tag{2.48}$$

Following the discretization of boundary condition (c) we obtain the following equations for $i=0$ and $j=1,\ldots,N-2$:

$$\frac{u_{1,j}-2u_{0,j}+u_{M-1,j}}{\delta x^2}+\frac{u_{0,j+1}-2u_{0,j}+u_{0,j-1}}{\delta y^2}=f_{0,j}=g_{0,j}. \tag{2.49}$$

At $x = b - \delta x$ we obtain the following equations for $i = M-1$ and $j = 1, \ldots, N-2$:

$$\frac{u_{0,j} - 2u_{M-1,j} + u_{M-2,j}}{\delta x^2} + \frac{u_{M-1,j+1} - 2u_{M-1,j} + u_{M-1,j-1}}{\delta y^2} = f_{M-1,j} = g_{M-1,j}. \tag{2.50}$$

At $y = c$ we obtain the following equations for $i = 1, \ldots, M-2$ and $j = 0$:

$$\frac{u_{i+1,0} - 2u_{i,0} + u_{i-1,0}}{\delta x^2} + \frac{u_{i,1} - 2u_{i,0} + u_{i,N-1}}{\delta y^2} = f_{i,0} = g_{i,0}. \tag{2.51}$$

At $y = d - \delta y$ we obtain the following equations for $i = 1, \ldots, M-2$ and $j = N-1$:

$$\frac{u_{i+1,N-1} - 2u_{i,N-1} + u_{i-1,N-1}}{\delta x^2} + \frac{u_{i,0} - 2u_{i,N-1} + u_{i,N-2}}{\delta y^2} = f_{i,N-1} = g_{i,N-1}. \tag{2.52}$$

The remaining equations are obtained at or near the corners of the rectangle. At $i = 0$ and $j = 0$

$$\frac{u_{1,0} - 2u_{0,0} + u_{M-1,0}}{\delta x^2} + \frac{u_{0,1} - 2u_{0,0} + u_{0,N-1}}{\delta y^2} = f_{0,0} = g_{0,0}. \tag{2.53}$$

At $i = M-1$ and $j = 0$ we obtain

$$\frac{u_{0,0} - 2u_{M-1,0} + u_{M-2,0}}{\delta x^2} + \frac{u_{M-1,1} - 2u_{M-1,0} + u_{M-1,N-1}}{\delta y^2} = f_{M-1,0} = g_{M-1,0}. \tag{2.54}$$

at $i = 0$ and $j = N-1$ we obtain

$$\frac{u_{1,N-1} - 2u_{0,N-1} + u_{M-1,N-1}}{\delta x^2} + \frac{u_{0,0} + 2u_{0,N-1} - u_{0,N-2}}{\delta y^2} = f_{0,N-1} = g_{0,N-1}. \tag{2.55}$$

Finally at $i = M-1$ and $j = N-1$ we obtain

$$\frac{u_{0,N-1} - 2u_{M-1,N-1} + u_{M-2,N-1}}{\delta x^2} + \frac{u_{M-1,0} - 2u_{M-1,N-1} + u_{M-1,N-2}}{\delta y^2} = f_{M-1,N-1} = g_{M-1,N-1}. \tag{2.56}$$

Equations (2.48) through (2.56) total MN equations for the MN unknowns $u_{i,j}$. Actually this system of equations is singular and its "solution" presents some special difficulties that are discussed in Section 5. This completes the derivation of the finite difference equations for Problem D, which is the last sample problem that we will consider. In the remaining sections we will describe the Fourier and cyclic reduction methods for the rapid solution of these equations.

3. THE FOURIER METHOD

In this section we will use the Fourier method to solve the large systems of equations derived in the previous section. These equations will also be solved by cyclic reduction in the next section. Both methods are described as efficient direct methods in order to distinguish them from Gaussian elimination, which is also classified as a direct method.

The Fourier method can be divided into the following phases:

I. Transform the equations from physical into Fourier space.
II. Compute the Fourier coefficients of the solution.
III. Transform the solution from Fourier back into physical space.

The fast Fourier transform (FFT) is the key to the efficiency of this method. It also simplifies the method considerably since software is available for the FFT, with the result that phases I and III can be completed by simply calling subprograms that compute the transforms. However, each of the problems A through D requires a different Fourier transform. For example, Problem A requires a sine transform and Problem B requires a cosine transform. Like the full-complex Fourier transform, these transforms also have a "fast" counterpart. All of these fast transforms are available from the National Center for Atmospheric Research in a software package called FFTPACK. The development of the "fast" versions of the various transforms is described in [29].

Like the previous section, this section will be divided into four parts in which the Fourier method will be used to solve each of the four problems that were defined in the previous section

Problem A. In this part we will use the Fourier method to solve the Dirichlet problem, in which the solution is specified on the boundary. In the first phase of the Fourier method, the equations are transformed into Fourier space, where they decouple and can be easily solved. We will show that a solution can be obtained in the form

$$u_{i,j} = \sum_{k=1}^{M-1} \hat{u}_{k,j} \sin ki\frac{\pi}{M}. \tag{3.1}$$

The reason for seeking a solution in this form is that the Fourier coefficients $\hat{u}_{k,j}$ can be determined more easily than $u_{i,j}$. Once the $\hat{u}_{i,j}$ are determined, the solution $u_{i,j}$ can be determined quite efficiently from (3.1) using the fast Fourier transform. To this end we will transform equations (2.9) through (2.17) into Fourier space in order to obtain a system of equations for the $\hat{u}_{k,j}$.

Since the $g_{i,j}$ are known, we can first compute

$$\hat{g}_{k,j} = \frac{2}{M} \sum_{i=1}^{M-1} g_{i,j} \sin ik\frac{\pi}{M}. \tag{3.2}$$

Since the $\sin ik\frac{\pi}{M}$ are orthogonal we know that $g_{i,j}$ can also be expressed in terms of $\hat{g}_{k,j}$ by the inverse transform

$$g_{i,j} = \sum_{k=1}^{M-1} \hat{g}_{k,j} \sin ik\frac{\pi}{M}. \tag{3.3}$$

The transforms in (3.1) and (3.2) constitute phases III and I, respectively. Actually, (3.1) and (3.2) are not used directly since the FFT provides a much more efficient way to compute these transforms. After the tabulation of the trigonometric functions, the direct calculation of either (3.1) or (3.2) would require $O(N^2)$ operations, compared with $O(N \log N)$ operations for the FFT. We can ignore the operations that are required for the trigonometric functions since they can be computed and stored at the beginning and used repeatedly for all subsequent transforms. In this way they do not significantly contribute to the overall computing time. If we now substitute (3.1) and (3.3) into (2.9) we obtain

$$\begin{aligned} &\frac{1}{\delta x^2} \sum_{k=1}^{M-1} \hat{u}_{k,j} \left[\sin(i+1)k\frac{\pi}{M} - 2\sin ik\frac{\pi}{M} + \sin(i-1)k\frac{\pi}{M} \right] \\ &\quad + \sum_{k=1}^{M-1} \left(\frac{\hat{u}_{k,j+1} - 2\hat{u}_{k,j} + \hat{u}_{k,j-1}}{\delta y^2} \right) \sin ik\frac{\pi}{M} \\ &\quad = \sum_{k=1}^{M-1} \hat{g}_{k,j} \sin ik\frac{\pi}{M}. \end{aligned} \tag{3.4}$$

But the term in brackets has the form

$$\begin{aligned} &\sin(i+1)k\frac{\pi}{M} - 2\sin ik\frac{\pi}{M} + \sin(i-1)k\frac{\pi}{M} \\ &\qquad = \left(2\cos k\frac{\pi}{M} - 2\right)\sin ik\frac{\pi}{M} = -4\sin^2 k\frac{\pi}{2M} \sin ik\frac{\pi}{M}. \end{aligned} \tag{3.5}$$

Therefore, (3.4) can be written

$$\sum_{k=1}^{M-1}\left[\hat{u}_{k,j+1}-\left(2+4\rho^2\sin^2 k\frac{\pi}{2M}\right)\hat{u}_{k,j}+\hat{u}_{k,j-1}\right]\sin ik\frac{\pi}{M}$$

$$=\delta y^2\sum_{k=1}^{M-1}\hat{g}_{k,j}\sin ik\frac{\pi}{M}, \tag{3.6}$$

where $\rho=\dfrac{\delta y}{\delta x}$.

Equation (3.6) is valid for $i=2,\ldots,M-2$ and $j=2,\ldots,N-2$. If we repeat this derivation starting with equations (2.10) and (2.11), then we obtain two equations just like (3.6) that hold for $i=1$ and $i=M-1$. Therefore (3.6) holds for $i=1,\ldots,M-1$. Finally, since the set $\sin ik\dfrac{\pi}{M}$ is orthogonal, we can equate coefficients on both sides of (3.6) with the result that

$$\hat{u}_{k,j+1}-\left(2+4\rho^2\sin^2 k\frac{\pi}{2M}\right)\hat{u}_{k,j}+\hat{u}_{k,j-1}=\delta y^2\hat{g}_{k,j}. \tag{3.7}$$

This provides most of the equations for the coefficients $\hat{u}_{k,j}$. The remaining equations are obtained from (2.12) through (2.17). If we transform equations (2.12), (2.14), and (2.15) in a manner analogous to the transformations of (2.9), (2.10), and (2.11), then we obtain

$$\hat{u}_{k,2}-\left(2+4\rho^2\sin^2 k\frac{\pi}{2M}\right)\hat{u}_{k,1}=\delta y^2\hat{g}_{k,2}. \tag{3.8}$$

If we transform equations (2.13), (2.16), and (2.17), then we obtain

$$-\left(2+4\rho^2\sin^2 k\frac{\pi}{2M}\right)\hat{u}_{k,N-1}+\hat{u}_{k,N-2}=\delta y^2\hat{g}_{k,N-1}. \tag{3.9}$$

The coefficients $\hat{u}_{k,j}$ are determined in phase II. For any k we can determine $\hat{u}_{k,j}$ for $j=1,\ldots,N-1$ from the system of $N-1$ equations consisting of (3.8), (3.7) with $j=2,\ldots,N-2$, and (3.9). If we examine the coefficient matrix for these equations, it is

apparent that only nonzero elements occur on the diagonal as well as just above and below the diagonal. This system is called tridiagonal and it can be solved very efficiently using an algorithm that is described in a number of places, including [19]. Also, a number of programs are available for solving tridiagonal systems [11]. The coefficients $\hat{u}_{k,j}$ are determined by solving $M-1$ independent tridiagonal systems for $k=1,\ldots,M-1$, which completes the second phase of the Fourier method.

We end this part with a review of the Fourier method for solving Problem A, in which the solution is specified on the boundary.

I. Compute $\hat{g}_{k,j}$ from $g_{i,j}$ using the FFT to compute (3.2).
II. Compute $\hat{u}_{k,j}$ from $\hat{g}_{k,j}$ by solving the systems of tridiagonal equations (3.7), (3.8) and (3.9).
III. Compute $u_{i,j}$ from $\hat{u}_{k,j}$ using the FFT to compute (3.1).

Problem B. In this part we will use the Fourier method to solve the Neumann problem, in which the derivative of the solution is specified on the boundary. We present the solution in less detail since the development is similar to that given for Problem A. We begin with the first phase of the Fourier method, in which the equations are transformed into Fourier space, where they decouple into $M+1$ independent tridiagonal systems of linear equations that can be solved quite easily.

We will show that a solution can be obtained in the form

$$u_{i,j}=\sum_{k=0}^{M}{}''\,\hat{u}_{k,j}\cos ki\frac{\pi}{M}. \tag{3.10}$$

where the double prime notation on the sum indicates that the first and last terms are multiplied by one half. We note that (3.10) is different than the form (3.1) used for Problem A, where the solution was specified on the boundary. This difference is due to the difference between equations (2.10) and (2.24). Whereas the transformation (3.1) will decouple (2.10), the transformation (3.10) is required to decouple (2.24).

We proceed now to the first phase of the Fourier method, in which the equations (2.23) through (2.31) are transformed into

Fourier space in order to obtain a system of equations for $\hat{u}_{k,j}$. First we compute

$$\hat{g}_{k,j} = \frac{2}{M} \sum_{i=0}^{M}{}'' g_{i,j} \cos ik\frac{\pi}{M}, \tag{3.11}$$

which has the inverse

$$g_{i,j} = \sum_{k=0}^{M}{}'' \hat{g}_{k,j} \cos ik\frac{\pi}{M}. \tag{3.12}$$

As in Problem A, the FFT is used to compute the transforms (3.10) and (3.11), which constitute phases III and I, respectively. If we transform (2.23), (2.24), and (2.25) following the development after (3.3) above, then for $k = 0,\ldots, M$ and $j = 1,\ldots, N-1$ we obtain

$$\hat{u}_{k,j+1} - \left(2 + 4\rho^2 \sin^2 k\frac{\pi}{2M}\right)\hat{u}_{k,j} + \hat{u}_{k,j-1} = \delta y^2 \hat{g}_{k,j}. \tag{3.13}$$

If we transform (2.26), (2.28), and (2.29), then for $k = 0,\ldots, M$ we obtain

$$2\hat{u}_{k,1} - \left(2 + 4\rho^2 \sin^2 k\frac{\pi}{2M}\right)\hat{u}_{k,0} = \delta y^2 \hat{g}_{k,0}. \tag{3.14}$$

If we transform equations (2.27), (2.30), and (2.31), then for $k = 0,\ldots, M$ we obtain

$$-\left(2 + 4\rho^2 \sin^2 k\frac{\pi}{2M}\right)\hat{u}_{k,N} + 2\hat{u}_{k,N-1} = \delta y^2 \hat{g}_{k,N}. \tag{3.15}$$

In the second phase we can determine the coefficients $\hat{u}_{k,j}$ from $M+1$ tridiagonal systems each with $N+1$ equations consisting of (3.13), (3.14) with $j = 1,\ldots, N-1$, and (3.15). There is an important extra step that must be taken for this problem that was not necessary for Problem A, in which the solution, rather than the derivative of the solution, was specified. This step is discussed in the first subsection of Section 5.

The third and final phase of the Fourier method consists of computing the solution $u_{i,j}$ from $\hat{u}_{k,j}$, using (3.10) and a modified version of the FFT [29].

Problem C. In this part we will use the Fourier method to solve the problem in which the solution is specified on $x = a$ and $y = c$ and the derivative of the solution is specified on $x = b$ and $y = d$. We begin with the first phase of the Fourier method, in which the equations are transformed into Fourier space, where they decouple into M independent tridiagonal systems of linear equations that are easily solved.

We will show that a solution can be obtained in the form

$$u_{i,j} = \sum_{k=1}^{M} \hat{u}_{k,j} \sin(2k-1) i\frac{\pi}{2M}. \tag{3.16}$$

We note that this form is different than the previous forms. It will decouple equations (2.37) as well as the boundary equations (2.38) and (2.39) when they are transformed into Fourier space.

In the first phase of the Fourier method, equations (2.37) through (2.45) are transformed into Fourier space in order to obtain a system of equations for $\hat{u}_{k,j}$. First we compute

$$\hat{g}_{k,j} = \frac{2}{M} \sum_{i=1}^{M}{}' g_{i,j} \sin(2k-1) i\frac{\pi}{2M}, \tag{3.17}$$

where the prime notation on the sum indicates that the last term is multiplied by one half. Equation (3.17) has the inverse

$$g_{i,j} = \sum_{k=1}^{M} \hat{g}_{k,j} \sin(2k-1) i\frac{\pi}{2M}. \tag{3.18}$$

As in the previous problems, the FFT is used to compute (3.16) and (3.17), which constitute phases III and I, respectively. Follow-

ing the development after equation (3.3) above, we obtain

$$\hat{u}_{k,j+1}-\left(2+4\rho^2\sin^2(2k-1)\frac{\pi}{4M}\right)\hat{u}_{k,j}+\hat{u}_{k,j-1}=\delta y^2\hat{g}_{k,j} \tag{3.19}$$

for $k=1,\dots,M$ and $j=2,\dots,N-1$. If we transform (2.40), (2.42), and (2.43), then for $k=1,\dots,M$ we obtain

$$\hat{u}_{k,2}-\left(2+4\rho^2\sin^2(2k-1)\frac{\pi}{4M}\right)\hat{u}_{k,1}=\delta y^2\hat{g}_{k,1}. \tag{3.20}$$

If we transform equations (2.41), (2.44), and (2.45), then for $k=1,\dots,M$ we obtain

$$-\left(2+4\rho^2\sin^2(2k-1)\frac{\pi}{4M}\right)\hat{u}_{k,N}+2\hat{u}_{k,N-1}=\delta y^2\hat{g}_{k,N}. \tag{3.21}$$

In the second phase we can determine the coefficients $\hat{u}_{k,j}$ from M tridiagonal systems with N equations consisting of (3.19), (3.20) with $j=2,\dots,N-1$, and (3.21).

The third and final phase of the Fourier method consists of computing the solution $u_{i,j}$ from $\hat{u}_{k,j}$, using (3.16) and a modified version of the FFT [29].

Problem D. In this part we will use the Fourier method to solve the Poisson equation subject to periodic boundary conditions. For Problem D it is assumed that the solution is periodic in both x and y, i.e., for any x and y, the solution satisfies both $u(a+x,y)=u(b+x,y)$ and $u(x,c+y)=u(x,d+y)$. We begin with the first phase of the Fourier method, in which equations (2.48) through (2.56) are transformed into Fourier space, where they decouple into M independent tridiagonal systems of N linear equations in N unknowns that are easily solved.

If for the moment we assume that M is even, then a solution can be obtained in the form

$$u_{i,j}=\sum_{k=0}^{M/2}{}''\left[\hat{u}^{(0)}_{k,j}\cos ik\frac{2\pi}{M}+\hat{u}^{(1)}_{k,j}\sin ik\frac{2\pi}{M}\right]. \tag{3.22}$$

We note that this form is different than the previous forms. It will decouple equations (2.48) as well as the boundary equations (2.49) and (2.50) when they are transformed into Fourier space. We proceed now to transform (2.48) through (2.56) into Fourier space in order to obtain a system of equations for $\hat{u}^{(l)}_{k,j}$. First we compute

$$\hat{g}^{(0)}_{k,j}=\frac{2}{M}\sum_{i=0}^{M-1}g_{i,j}\cos ik\frac{2\pi}{M} \tag{3.23}$$

and

$$\hat{g}^{(1)}_{k,j}=\frac{2}{M}\sum_{i=1}^{M-1}g_{i,j}\sin\frac{2\pi}{M}, \tag{3.24}$$

which has the inverse

$$g_{i,j}=\sum_{k=0}^{M/2}{}''\left[\hat{g}^{(0)}_{k,j}\cos ik\frac{2\pi}{M}+\hat{g}^{(1)}_{k,j}\sin ik\frac{2\pi}{M}\right]. \tag{3.25}$$

As in the previous problems, the FFT is used to compute (3.24) and (3.25), which constitute phases III and I, respectively. Following the development after equation (3.3) we obtain

$$\hat{u}^{(l)}_{k,j+1}-\left(2+4\rho^2\sin^2k\frac{\pi}{M}\right)\hat{u}^{(l)}_{k,j}+\hat{u}^{(l)}_{k,j-1}=\delta y^2\hat{g}^{(l)}_{k,j} \tag{3.26}$$

for $k=0,\dots,M/2$; $j=1,\dots,N-2$ and $l=0,1$. If we transform (2.51), (2.53), and (2.54), then for $k=0,\dots,M/2$ and $l=0,1$ we obtain

$$\hat{u}^{(l)}_{k,1}-\left(2+4\rho^2\sin^2k\frac{\pi}{M}\right)\hat{u}^{(l)}_{k,0}+\hat{u}^{(l)}_{k,N-1}=\delta y^2\hat{g}^{(l)}_{k,0}. \tag{3.27}$$

If we transform equations (2.52), (2.55), and (2.56), then for $k=0,\dots,M/2$ and $l=0,1$ we obtain

$$\hat{u}^{(l)}_{k,0}-\left(2+4\rho^2\sin^2k\frac{\pi}{M}\right)\hat{u}^{(l)}_{k,N-1}+\hat{u}^{(l)}_{k,N-2}=\delta y^2\hat{g}^{(l)}_{k,N-1}. \tag{3.28}$$

In the second phase we determine the coefficients $\hat{u}^{(l)}_{k,j}$ from M periodic tridiagonal systems with N equations consisting of (3.26), (3.27) with $j=1,\ldots,N-2$, and (3.28). A method for solving periodic tridiagonal linear systems is presented at the end of Section 4. In Section 5 we discuss the "solution" of the singular tridiagonal system corresponding to $k=0$. The third and final phase of the Fourier method consists of computing the solution $u_{i,j}$ from $\hat{u}^{(l)}_{k,j}$ using (3.22) and the real periodic FFT. If M is odd then a straightforward modification of the transforms (3.22) and (3.25) can be used.

This completes the description of the Fourier method. The implementation of this method can be greatly simplified by using the transforms that are available in the software package FFTPACK which is described in [29].

4. CYCLIC REDUCTION

In this section we will use the cyclic reduction method to solve the large systems of equations that were derived in Section 2. Although the computational procedure is quite different than the Fourier method, both methods have the same asymptotic operation count and neither of them requires any additional storage. Each method has its own advantages and disadvantages. The method of cyclic reduction can be applied to a larger class of problems than the Fourier method, for example, see [25] and [26], but the Fourier method is usually faster when both methods are applicable.

Like the Fourier method, we will describe the method of cyclic reduction by using it to solve the problems described in the second section. We begin with the problem in which the solution is specified on all the boundaries.

Problem A. The exposition is greatly simplified if we write the large system of equations in matrix form. To this end we combine all of the unknowns on the jth grid line into the single vector $\mathbf{u}_j^T=(u_{1,j},u_{2,j},\ldots,u_{M-1,j})$ with $M-1$ components. This selection was arbitrary since we could also combine the components on the ith line, which would lead to an alternate description of cyclic

reduction with the same computational properties. In practice, these vectors should be made as large as possible. That is, the grid should be orientated so that $N \leqslant M$, since the operation count is proportional to $MN\log N$.

If we multiply the equations on the jth grid line by δy^2, then (2.9) can be written in matrix form

$$\mathbf{u}_{j+1}+\mathbf{A}\mathbf{u}_j+\mathbf{u}_{j-1}=\mathbf{g}_j, \tag{4.1}$$

where $\mathbf{g}_j^T=\delta y^2(g_{1,j},g_{2,j},\ldots,g_{M-1,j})$ and $\mathbf{A}$ is a matrix with order $M-1$ given by

$$\mathbf{A}=\rho^2\begin{bmatrix} -2\alpha & 1 & & & & \\ 1 & -2\alpha & 1 & & & \\ & 1 & \cdot & & & \\ & & & \cdot & & \\ & & & & \cdot & 1 \\ & & & & 1 & -2\alpha \end{bmatrix} \tag{4.2}$$

with $\alpha=1+\rho^{-2}$. For a sample case in which $N=8$, the complete system of equations (2.9) through (2.17) has the matrix form

$$\begin{aligned} \mathbf{A}\mathbf{u}_1+\mathbf{u}_2 &= \mathbf{g}_1 \\ \mathbf{u}_1+\mathbf{A}\mathbf{u}_2+\mathbf{u}_3 &= \mathbf{g}_2 \\ \mathbf{u}_2+\mathbf{A}\mathbf{u}_3+\mathbf{u}_4 &= \mathbf{g}_3 \\ \mathbf{u}_3+\mathbf{A}\mathbf{u}_4+\mathbf{u}_5 &= \mathbf{g}_4 \\ \mathbf{u}_4+\mathbf{A}\mathbf{u}_5+\mathbf{u}_6 &= \mathbf{g}_5 \\ \mathbf{u}_5+\mathbf{A}\mathbf{u}_6+\mathbf{u}_7 &= \mathbf{g}_6 \\ \mathbf{u}_6+\mathbf{A}\mathbf{u}_7 &= \mathbf{g}_7. \end{aligned} \tag{4.3}$$

The original cyclic reduction algorithm was highly *unstable*. Nevertheless, it will be presented here because it is quite intuitive and excellent for expository purposes. The method came into general use as a practical computational tool after a stable version was developed by Buneman [5], as discussed briefly in the third subsection of Section 5.

The method of cyclic reduction consists of three main phases: preprocessing, reduction and backsubstitution.