Algorithm 732: Solvers for Self-Adjoint Elliptic Problems in Irregular Two-Dimensional Domains

PATRICK F. CUMMINS
Institute of Ocean Sciences
and
GEOFFREY K. VALLIS
University of California, Santa Cruz

Software is provided for the rapid solution of certain types of elliptic equations in rectangular and irregular domains. Specifically, solutions are found in two dimensions for the nonseparable self-adjoint elliptic problem $\nabla \cdot (g\nabla\psi) = f$, where g and f are given functions of x and y, in two-dimensional polygonal domains with Dirichlet boundary conditions. Helmholtz and Poisson problems in polygonal domains and the general variable coefficient problem (i.e., $g \neq 1$) in a rectangular domain may be treated as special cases. The method of solution combines the use of the capacitance matrix method, to treat the irregular boundary, with an efficient iterative method (using the Laplacian as preconditioner) to deal with nonseparability. Each iterative step thus involves solving the Poisson equation in a rectangular domain. The package includes separate, easy-to-use routines for the Helmholtz problem and the general problem in rectangular and general polygonal domains, and example driver routines for each. Both single- and double-precision routines are provided.

Second-order-accurate finite differencing is employed. Storage requirements increase approximately as $p^2 + n^2$, where p is the number of irregular boundary points and where n is the linear domain dimension. The preprocessing time (the capacitance matrix calculation) varies as $pn^2 \log n$, and the solution time varies as $n^2 \log n$. If the equations are to be solved repeatedly in the same geometry, but with different source or diffusion functions, the capacitance matrix need only be calculated once, and hence the algorithm is particularly efficient for such cases.

Categories and Subject Descriptors: G.1.0 [**Numerical Analysis**]: General—*numerical algorithms*; G.1.8 [**Numerical Analysis**]: Partial Differential Equations—*elliptic equations*

General Terms: Algorithms

Additional Key Words and Phrases: Capacitance iteration, capacitance matrix, elliptic equations, fast Poisson solvers, Green's function

This work was partially funded by ONR and the NSF.
Authors' addresses: P. F. Cummins, Institute of Ocean Sciences, Sidney, British Columbia, Canada V8L-4B2; and G. K. Vallis, Department of Physics, University of California, Santa Cruz, CA 95064.

1. INTRODUCTION

The rapid numerical solution of elliptic equations is an important problem in numerical analysis with many practical ramifications in the applied sciences. In this paper we present algorithms for the solution of certain elliptic problems in two dimensions. The methods are "fast" in the sense that they are built on a rapid direct solution to the Poisson equation in a regular domain. In several disciplines where numerical models are important, elliptic equations must be solved repeatedly, perhaps thousands of times, with different forcing functions but in the same geometry. The software package has been designed to be especially efficient for such problems.

This section contains a brief discussion of the algorithms used. In Section 2 the difference forms of the equations are explicitly stated, and in Section 3 the software package itself is described. This includes a brief description of each subroutine; a longer description of the calling parameters and arguments for one routine are given in the Appendix. Example driver routines for each algorithm are included in the package. Section 4 provides a brief description of the performance of certain aspects of the routines.

1.1 Analytic Foundation

The building block for all of the algorithms presented in this paper is a method for the rapid solution of Poisson's equation in a regular domain, to wit:

$$\nabla^2\psi = f, \tag{1.1}$$

where f is a given source and ψ is the required solution. Various rapid direct methods for solving this equation exist, notably, the Fourier transform method, the method of cyclic reduction, and a combination of both—the FACR class of methods (see, e.g., Swarztrauber [1977]). The generalization to the Helmholtz equation,

$$(\nabla^2 - \lambda)\psi = f, \tag{1.2}$$

is straightforward. The FISHPACK software package [Swarztrauber and Sweet 1979], which uses cyclic reduction, provides efficient solvers for this type of problem.

A number of packages for solving elliptic problems using iterative or relaxation methods exist [Rice and Boisvert 1985], but are generally not as efficient as direct solvers based on FACR or similar methods. However, for self-adjoint problems of the form

$$\nabla \cdot (g(x, y)\nabla\psi) = f, \tag{1.3}$$

where the diffusion function g is spatially varying, the iterative procedure

$$\nabla^2\psi^{n+1} = \frac{f - \nabla g \cdot \nabla\psi^n}{g} \tag{1.4}$$

has proved to be an efficient and robust method [Concus and Golub 1973; Pares-Sierra and Vallis 1989]. Given a first guess for ψ, the right-hand side is

treated as a known source, and a fast solver is employed to determine the next iterate of ψ. Often, very few (1–10) iterations are required for an accurate result. The efficiency of the iteration is due to the use of the Laplacian as a preconditioner.

The fast solution of the Poisson and Helmholtz problems in irregular domains can be achieved with the capacitance matrix method [Hockney 1970; Proskurowski and Widlund 1976], implemented, for example, by Proskurowski [1983]. For a detailed description of the capacitance matrix method as implemented here; see Cummins and Mysak [1988]. It is most easily described for the Poisson problem (1.1), with a trivial generalization for the Helmholtz problem. The essence of the method lies in replacing boundary conditions by appropriately chosen source terms (a pseudosource) on the right-hand side. Problem (1.1) is first solved in an enclosing rectangular region $\mathscr{R}$ with the given source over the irregular domain $\mathscr{I}$, yielding values of ψ on $\partial\mathscr{I}$ of ψ_b, say. This is equivalent to a solution over the irregular domain with boundary conditions $\psi = \psi_b$. Therefore, the addition of this to a solution to the homogeneous problem

$$\nabla^2\psi_1 = 0, \tag{1.5}$$

with $\psi_1 = -\psi_b + \phi$ on $\partial\mathscr{I}$, will yield the desired solution to the Poisson problem with boundary values ϕ on $\partial\mathscr{I}$.

The solution to the homogeneous problem on $\mathscr{I}$ is found by first obtaining the appropriate Green's function or inverse capacitance matrix, $\mathbf{G}$. The columns of $\mathbf{G}$ are determined by solving (1.1) repeatedly, each time with a unit source at a different grid point along $\partial\mathscr{I}$. Then we may obtain the correct pseudosource f_s on $\partial\mathscr{I}$ by noting that

$$\phi - \psi_b = \mathbf{G}f_s. \tag{1.6}$$

Given f_s, the full solution to the problem on $\mathscr{I}$ is finally obtained as a solution to the equation

$$\nabla^2\psi = f + f_s, \tag{1.7}$$

where the equation is actually solved on $\mathscr{R}$. The values of f outside the domain of interest and the boundary conditions on $\mathscr{R}$ do not matter. In fact, the solution is obtained with zero values on the boundary on $\mathscr{R}$ outside the domain of interest.

Our implementation involves first determining the Green's function for the given irregular area. The subroutine may then return without calculating the full solution, in which case it must be called again with the Green's function as an input parameter in order to compute the solution. This allows the solution to a problem in an unchanging domain with a varying source or diffusion function to be found with only a single calculation of the Green's function. Alternatively, the Green's function and solution may be computed by a single call to the subroutine.

The capacitance-iteration (CI) method [Pares-Sierra and Vallis 1989] combines the capacitance matrix method with the iterative method (1.4), enabling solutions to be obtained for elliptic equations of the form of (1.3) in irregular domains. The algorithm implemented here is "Method 1" of Pares-Sierra and Vallis [1989]. The Green's function for the Laplacian operator is found, just as for the Poisson problem. Then, given an estimate for the solution ψ^n, we use

$$\nabla^2\psi^{n+1} = \frac{f - \nabla\psi^n \cdot \nabla g}{g}, \tag{1.8}$$

to obtain a better solution, on $\mathscr{R}$. Using the Green's function, we then obtain a boundary source

$$f_s = \mathbf{G}^{-1}(\phi - \psi_b) \tag{1.9}$$

and obtain an intermediate solution on $\mathscr{I}$ using

$$\nabla^2\psi^{n+1} = \frac{f - \nabla\psi^n \cdot \nabla g}{g} + f_s. \tag{1.10}$$

A single step in the iteration procedure consists of the successive application of steps (1.8)–(1.10). Note that in steps (1.8) and (1.10) the Poisson equation is obtained (directly) in $\mathscr{R}$ and that it is the use of the pseudosource f_s that yields the correct solution in $\mathscr{I}$. The process is repeated until satisfactory convergence is achieved.

The iterative procedure may be characterized as an outer iteration, with the Laplacian acting as a preconditioner. The convergence properties of the procedure are not fully understood, although they may be expected to be similar to those of the problem in a rectangular domain, discussed by Concus and Golub [1973], which does not involve the use of the capacitance matrix step. In particular, convergence is not guaranteed for all diffusion functions g (even if one-signed). However, our experience suggests that the procedure is actually very robust, with convergence achieved for highly discontinuous functions g; some empirically determined convergence properties are discussed in Section 4.

This package consists of routines that solve both the constant coefficient (i.e., Poisson and Helmholtz) and variable coefficient problems in irregular domains. Clearly, the constant coefficient problem is just a special case of the more general problem, and one could use the same routine with $g \equiv 1$. However, for most applications it is more convenient and faster to have different solvers. The package also includes a separate routine to solve the nonconstant coefficient problem in a rectangular domain, without use of the capacitance matrix methodology. We do not explicitly document a routine to solve the Poisson or Helmholtz equations in a regular domain, since there are many available [Swarztrauber and Sweet 1979]. The included subroutine POISS (or an easily modified version of RECCN) may, however, be used for this purpose. The routines are designed to be very straightforward to use;

generally, a call to a single subroutine is all that is needed. The limitations of the algorithms presented here are that the boundary of the region is a polygon described by lines parallel to, or diagonal to, the mesh points, and only Dirichlet conditions may be applied. (Note that a package for the Helmholtz equation in a more general irregular domain is available [Proskurowski 1983].)

The cost of preprocessing (the capacitance matrix calculation) varies approximately as $pn^2 \log n$, where p is the number of irregular boundary points and n is the number of grid points across the whole domain. (This is because the inverse capacitance matrix is obtained by solving the Poisson equation p times, with a δ-function source at each irregular boundary point.) The solution cost per iteration scales approximately as $n^2 \log n$. The number of iterations is largely independent of resolution and in typical cases is between 2 and 10, depending on the convergence parameter (ϵ) tolerance and the first guess. Storage requirements increase as $In^2 + p^2$, where I is a small integer, differing slightly in each code.

2. DIFFERENCE EQUATIONS AND NUMERICAL DETAILS

In this section the explicit difference forms of the equations are given. In all cases, boundary conditions on ψ are specified along $\partial\mathcal{S}$ (Dirichlet boundary conditions), and Cartesian coordinates are used.

Define the coordinates $x_i = i\Delta x$ and $y_j = j\Delta y$, where Δx and Δy are the discretization intervals in the x and y directions, and let $\phi_{i,j}$ denote $\phi(x_i, y_j)$. It is useful to define the following difference operators:

$$\delta_{xx}\phi_{i,j} = \frac{\phi(x_i + \Delta x, y_j) + \phi(x_i - \Delta x, y_j) - 2\phi(x_i, y_j)}{(\Delta x)^2},$$

$$\delta_{yy}\phi_{i,j} = \frac{\phi(x_i, y_j + \Delta y) + \phi(x_i, y_j - \Delta y) - 2\phi(x_i, y_j)}{(\Delta y)^2},$$

$$\overline{\phi_{i,j+\frac{1}{2}}}^{x} = \frac{1}{2}\left[\phi\left(x_i + \frac{\Delta x}{2}, y_j + \frac{\Delta y}{2}\right) + \phi\left(x_i - \frac{\Delta x}{2}, y_j + \frac{\Delta y}{2}\right)\right],$$

$$\overline{\phi_{i+\frac{1}{2},j}}^{y} = \frac{1}{2}\left[\phi\left(x_i + \frac{\Delta x}{2}, y_j + \frac{\Delta y}{2}\right) + \phi\left(x_i + \frac{\Delta x}{2}, y_j - \frac{\Delta y}{2}\right)\right].$$

2.1 Helmholtz Equation

The second-order-accurate finite-difference approximation to Helmholtz's equation is

$$\delta_{xx}\psi_{i,j} + \delta_{yy}\psi_{i,j} - \lambda\psi_{i,j} = f_{i,j}. \tag{2.1}$$

This equation must be satisfied at each grid point within $\mathcal{S}$.

2.2 Variable-Coefficient Elliptic Equation

The finite-difference form of (1.3) is

$$\frac{\overline{g_{i+\frac{1}{2},j}}^{y}(\psi_{i+1,j}-\psi_{i,j})-\overline{g_{i-\frac{1}{2},j}}^{y}(\psi_{i,j}-\psi_{i-1,j})}{(\Delta x)^2}$$
$$+\frac{\overline{g_{i,j+\frac{1}{2}}}^{x}(\psi_{i,j+1}-\psi_{i,j})-\overline{g_{i,j-\frac{1}{2}}}^{x}(\psi_{i,j}-\psi_{i,j-1})}{(\Delta y)^2}=f_{i,j}. \qquad (2.2)$$

Note that the diffusion function is staggered with respect to the source function and the solution.

2.3 Convergence Criteria

The capacitance iteration algorithm (1.4) is assumed to have converged when the inequalities

$$\frac{\|\psi^n-\psi^{n-1}\|}{\|\psi^n\|}<\epsilon \qquad (2.3a)$$

and

$$\frac{\|\mathscr{L}(\psi^n)-f\|}{\|f\|}<\epsilon \qquad (2.3b)$$

are both satisfied, where $\|\cdot\|$ denotes the L_2 norm over the rectangle and $\mathscr{L}$ denotes the elliptic operator. That is, both the relative residual and change in iterates must be small. If $\|f\|\equiv 0$, then the second criterion is replaced by $\|L(\phi^n)\|/\|\phi^n\|<\epsilon$. The convergence parameter, ϵ, must be specified by the user. With four-byte words ("single precision" on most, but not all, computers, or REAL*4, and about 7 digits of accuracy), values of ϵ from 10^{-3} to 10^{-5} are recommended; much smaller values do not improve the accuracy of the solution. With eight-byte words ("double precision," or REAL*8, and about 15 digits of accuracy), values of ϵ down to 10^{-12}–10^{-13} are possible. (With higher resolution, i.e., a larger number of grid points, the smallest usable value of ϵ tends to increase.) Use of double precision is recommended for the iterative algorithms if high accuracy is required.

If the iteration is failing to converge, then the subroutine will attempt to detect this and exit. In particular, if $\|\psi^n-\psi^{n-1}\|/\|\psi^{n-1}-\psi^{n-2}\|$ does not decrease almost monotonically, the iteration will exit (although this feature may be overridden and the iterations forced to continue).

3. PACKAGE DESCRIPTION

3.1 General Remarks

The package consists of three user-callable FORTRAN subroutines, plus a fast Poisson solver and some Linpack routines [Dongarra et al. 1979] used by the capacitance matrix procedures. (Double-precision versions of all the subroutines are included in the package. These routines generally have identical

names except that the prefix "D" is appended.) The Poisson solver (POISS) uses a fast direct method and is a modification of one written by C. Temperton (personal communication). There is normally little need to call it or the Linpack routines directly. Users may with little effort substitute their own packages if there is an efficient local implementation of these routines. This may be easiest for the Linpack routines, where some installations have efficient machine-coded versions, although the effect of these on this package will be minimal. The package here is complete and contains no machine-dependent code. POISS performs sine transforms in the x-direction and then solves the resulting tridiagonal system in the y-direction. The sine transforms are done with a mixed radix FFT allowing prime factors up to 5. This permits some flexibility in the choice of the number of grid points in the x-direction. The routines also allow for different grid spacings in the x- and y-directions. Some subroutines have been optimized for vectorization on Alliant and Cray vector computers, and compiler directives are inserted where appropriate. These have no effect on the operation on nonvectorizing computers.

The routines are designed with the intention of keeping the definition of the irregular domain geometry simple for the user. The boundary curve of the irregular domain is assumed to lie along the discrete nodes of a two-dimensional mesh. An embedding rectangle of mesh points is defined, and grid points along any portion of the boundary of the irregular domain that coincide with the edges of the rectangle are referred to as the "regular" boundary points. Boundary points that do not lie on the edges of the rectangle are referred to as the "irregular" boundary points. Two one-dimensional arrays are defined to supply the routines with the coordinates of the irregular boundary points. Simply or multiply connected domains are easily defined in this way. The Dirichlet boundary values at the regular and irregular boundary points are also supplied to the routines through one-dimensional arrays. The size of the capacitance matrix increases as the square of the number of irregular boundary points. To keep the size of this matrix to a minimum, the irregular domain should be positioned within the embedding rectangle in such a way as to maximize the number of regular boundary points and to minimize the number of irregular boundary points. Illustrative examples are given in the comments of the code itself.

For the nonseparable solvers in irregular domains a "MASK" array must also be supplied by the user. The values of this array must be 1 inside the domain of interest and 0 elsewhere, including the boundary points. (A check is made that this condition is consistent with the definition of the irregular boundary.) The main purpose of this array is to define the domain of interest and to ensure that convergence checks are made only within that domain. It does not preclude multiply connected domains or multiply unconnected domains.

To apply the solvers, just a single call to a subroutine is required. If the subroutine is being called for the first time, the inverse capacitance matrix is generated and factored. This matrix, in general, depends only on the geome-

try of the domain and, in the case of the Helmholtz equation, also on the constant λ. The routine then obtains the solution for the given right-hand side. Provided that the geometry and λ do not change, the inverse capacitance matrix need not be evaluated on subsequent calls to the subroutine, even with different source functions and different diffusion functions. This enables subsequent solutions to be obtained much more rapidly than the first time.

3.2 Subroutine and Variable Definitions

Three subroutines are available, plus double-precision analogues:

—(D)CAPC solves the Helmholtz equation in irregular, polygonal domains.
—(D)CAPCN solves the variable coefficient self-adjoint elliptic equation (1.3) in irregular, polygonal domains.
—(D)RECCN solves the variable coefficient self-adjoint elliptic equation (1.3) in a rectangular domain.

In addition, the package contains

—POISS: a fast Poisson solver, utilizing the FACR(0) algorithm; and
—LINPACK: a selection of Linpack routines.

The user normally calls (D)CAPC, (D)CAPCN, or (D)RECCN. The LINPACK and POISS routines are included in the code and are called by the main routines. These both contain appropriate single- and double-precision code. The LINPACK (or even the POISS) routines may be separated from the other routines by the user, if the substitution of faster machine-dependent versions is desired. The naming of the LINPACK routines is standard.

The calling parameters for the variable-coefficient elliptic solver (D)CAPCN are outlined in the Appendix. The calling parameters for the two other routines are not included here, but are generally similar to the one presented below. A detailed description of these is found in the software. Three test (driver) programs, one for each of the routines, are included in the package.

4. RESULTS AND PERFORMANCE

In this section we briefly describe the performance of the package on various test problems. All of the results here use an IBM RS/6000 with the XLF FORTRAN compiler, although all of the code is written in FORTRAN 77 and results vary little from machine to machine. First, we describe the output for a simple test problem.

The routine listed in the Appendix is a driver for CAPCN and solves (1.3) in the domain $\Omega \in |x - 0.5| + |y - 0.5| \leq 0.375$, embedded in the unit rectangle. The discretization interval is $1/16$, although this may be changed easily. The source function is given by $f(x, y) = \cos \pi x \times \sin 2\pi y$; the diffusion function is given by $g(x, y) = e^{x+y}$; and the value of CCON, the tolerance, is 10^{-4}.

Table I. Number of Iterations Required in CAPCN as a Function of Value C in the Diffusion Function $g(x, y) = e^{x+y} - C$, for Two Values of Convergence Parameter CCON.

C	−1000	−100	−10	0	0.5	1.0	1.4	1.6	1.8	1.867
Number of iterations with CCON = $1\,10^{-4}$	3	3	4	5	5	6	7	8	9	12
Number of iterations with CCON = $1\,10^{-10}$	4	5	7	11	11	13	15	17	22	27

The source function is $f(x, y) = \cos(\pi x) \times \sin(2\pi y)$, and the resolution is 16×16. Results vary little with resolution.

The output of the code is as follows:

```
ITERATION 1 ANORM = 1.0000 RESIDUAL = 6.03776E-02
ITERATION 2 ANORM = .10641 RESIDUAL = 6.53740E-03
ITERATION 3 ANORM = 1.24130E-02 RESIDUAL = 8.2605E-04
ITERATION 4 ANORM = 1.84225E-03 RESIDUAL = 1.08370E-04
ITERATION 5 ANORM = 2.24527E-04 RESIDUAL = 1.93784E-05
ITERATION 6 ANORM = 3.28904E-05 RESIDUAL = 1.50112E-05
NUMBER OF ITERATIONS REQUIRED = 6
NUMBER OF CORRECT DIGITS IN RESIDUAL = 4.82
NORMALIZED L2 NORM OF RESIDUAL = 1.50112E-05
LARGEST ABSOLUTE RESIDUAL = 1.25202E-05 AT (I, J) = 11 12
```

The value of ANORM is the normalized L_2 norm of the difference between successive iterations, see Eq. (2.3a). The RESIDUAL is the normalized L_2 difference between the source function and the computed left-hand side of Eq. (2.3b). The actual values are slightly machine dependent, but decrease with each iteration. When ANORM and the RESIDUAL are both less than CCON, the iterations cease. Printing these diagnostics is optional (determined by the value of IWRITE, disabled for IWRITE = 0).

The driver code then uses the computed solution and the diffusion function to reconstruct the source function. The normalized L_2 residual between this and the original, and the largest absolute residual, are reported. These values are somewhat machine dependent, being the difference between two approximately equal values. The number of correct digits is the logarithm to base 10 of the L_2 norm. The L_2 norm is normalized by the source function, whereas the largest absolute residual is not and, therefore, may appear large if the source term is large. The value of the L_2 norm of the residual can be no greater than the value of the specified tolerance.

Table I gives the number of iterations required for two values of the tolerance parameter CCON, with a diffusion function given by $\exp(x + y) - C$, where C is allowed to vary, for a problem at a resolution of 16×16. The smaller value of the convergence parameter CCON requires that double precision (REAL*8) be used. The number of iterations increases both as the convergence parameter CCON is made smaller and as the problem approaches singularity. (The diffusion function must be positive where MASK = 1, which restricts C to the interval $-\infty < C < \exp(0.625) \approx 1.868$.)

To examine the properties of the iterative convergence and the expense of the capacitance matrix solver, it is most illustrative to use RECCN and CAPC

separately. Table II illustrates how the rate of convergence varies with resolution using RECCN in double precision. In general, there are three factors that determine the accuracy of a solution, namely, the truncation error of the discretization (here controlled by the grid resolution), the convergence parameter, and machine precision. For this problem, at a given convergence parameter there is little dependence on the resolution of either the residual (the difference between the original source and that computed from the numerical solution) or the rapidity of convergence. The error in the solution (the difference between the computed and the analytic solutions) decreases with the square of the grid interval, characteristic of second-order-accuracy numerical methods. The error cannot always be reduced by decreasing the convergence parameter if the resolution is too coarse, although the residual may be made as small as machine precision allows. There is, of course, no gain in making the convergence parameter smaller than machine precision, and for this reason double precision is recommended if accuracy is required. (All single-precision routines use generic calls to elementary functions, like sin or log, and on many machines the single-precision code may be automatically compiled to run in double precision or to run on 64-bit machines automatically at that precision, if required.)

Table III illustrates how, for a continuous diffusion function, the number of iterations required for convergence does not depend on the resolution, but on the properties of the diffusion function itself: The number of iterations is almost constant with resolution, but increases as the gradient of the diffusion function increases with the parameter D. Table IV illustrates the performance with a discontinuous diffusion function. In this case the grid is covered with a checkerboard pattern: the transition between checkerboard squares is one grid point in each case. For each column of the table, the problem is the same (i.e., the boundaries of the subregions coincide), and only the resolution varies. Here, the number of iterations varies weakly with resolution.

In the algorithms implemented here, we do not employ any additional acceleration or preconditioning methods [Concus and Golub 1973]. Extensive experimentation revealed that, although in some cases they may improve convergence, no general choice proved robust and satisfactory. However, note that, for the test problem in Table III, scaling the problem following Concus and Golub [1973] may enable convergence at higher values of D, and for particular cases users may wish to modify the code to employ such methods. There are clearly some diffusion functions for which the method as implemented will not converge.

The preprocessing time, that is, the time taken to set up the capacitance matrix, is illustrated using CAPC. Table V illustrates the time taken at various resolutions to obtain the matrix and then to solve Poisson's equation in the domain $\Omega \in |x - 0.5| + |y - 0.5| \leq 0.375$. For a single solution of the problem, the preprocessing time is a substantial fraction of the total time, especially at high resolution, but will become small for multiple solutions. Theoretically, we expect the preprocessing time to scale approximately as NP $N^2 \log N$, and the solving time to scale approximately as $N^2 \log N$ where N

Table II. Performance of RECCN at Various Resolutions in Double Precision with $g(x, y) = 2 + \cos \pi x \times \cos \pi y$ and $f(x, y) = -2\pi^2 \sin \pi x \times \sin \pi y(2 + 2 \cos \pi x \times \cos \pi y)$

	$\epsilon = 10^{-4}$			$\epsilon = 10^{-8}$		
Resolution	Number of iterations	Residual	Error	Number of iterations	Residual	Error
16×16	5	$1.60\ 10^{-5}$	$3.12\ 10^{-3}$	10	$2.25\ 10^{-10}$	$3.12\ 10^{-3}$
32×32	5	$1.73\ 10^{-5}$	$7.81\ 10^{-4}$	10	$2.65\ 10^{-10}$	$7.80\ 10^{-4}$
64×64	5	$1.74\ 10^{-5}$	$1.94\ 10^{-4}$	10	$2.76\ 10^{-10}$	$1.95\ 10^{-4}$
128×128	5	$1.76\ 10^{-5}$	$4.70\ 10^{-5}$	10	$2.78\ 10^{-10}$	$4.88\ 10^{-5}$
256×256	5	$1.77\ 10^{-5}$	$1.17\ 10^{-5}$	10	$2.80\ 10^{-10}$	$1.22\ 10^{-5}$
512×512	5	$1.77\ 10^{-5}$	$5.50\ 10^{-6}$	10	$3.36\ 10^{-10}$	$3.05\ 10^{-6}$
1024×1024	5	$1.77\ 10^{-5}$	$5.47\ 10^{-6}$	10	$1.66\ 10^{-9}$	$7.62\ 10^{-7}$

The analytical solution is $\psi(x, y) = \sin \pi x \times \sin \pi y$. *Residual* is the normalized L_2 norm of the difference between the original source, f, and that computed from the numerical solution (eq. (2.3b)), while *Error* is the normalized L_2 norm of the difference between the numerical and analytical solutions. The table illustrates the variation of accuracy with resolution and with the convergence parameter ϵ.

Table III. Number of Iterations Required for Convergence in RECCN at Various Resolutions in Double Precision

	Value of D					
Resolution	1	2	3	4	5	6
16×16	11	17	26	44	86	*
64×64	11	17	26	45	94	*
256×256	11	17	26	46	94	*
1024×1024	11	17	26	46	95	*

The diffusion function is $g(x, y) = e^{D(x+y)}$, and the source term is $f(x, y) = \cos \pi x \times \sin \pi 2y$. CCON = 10^{-8}. Asterisks indicates nonconvergence.

Table IV. Number of Iterations Required for Convergence in RECCN at Various Resolutions

	Number of checkerboard squares per side			
Resolution	4	8	16	32
8×8	25	—	—	—
16×16	29	27	—	—
32×32	53	42	28	—
64×64	65	56	43	29
128×128	76	69	58	45
256×256	86	80	71	60
512×512	97	92	85	75

The diffusion function is a checkerboard pattern, with alternating squares with values in each of 1 and 10. The source function is that for Table III. CCON = $1.2\ 10^{-10}$. The entries in each column of the table solve the same problem at different resolutions.

Table V. Efficiency of CAPC at Various Resolutions

NX, NY (resolution)	NP (number of irregular points)	Preprocessing time	Solving time	Preprocess time / Solve time	$NP \times$ Solve time / Preprocess time
16, 16	24	0.064	0.0055	11.6	2.06
32, 32	48	0.43	0.019	22.6	2.12
64, 64	96	3.6	0.077	47.0	2.05
128, 128	192	32.0	0.33	97.0	1.98

The preprocessing time (in CPU seconds) and the time to process, or solve, are listed at four different resolutions. The ratio of the preprocessing time to solving time is also presented. The value of the last column should be asymptotically constant.

is the linear dimension of the total domain and NP is the number of irregular boundary points. These are approximately satisfied. The method is well suited to situations such as numerical models, where the solution must be obtained many times in a given geometry, for then the preprocessing time can become negligible. For such models the iterative algorithms (CAPCN and RECCN) are also well suited, because a good first guess is often available. In such cases only a few iterations may be required to achieve acceptable accuracy.

APPENDIX. CALLING SEQUENCE FOR CAPCN

Here we give the calling parameters for CAPCN. Additional information is given in the code itself, where the details for RECCN and CAPC may also be found. The sequences for the double-precision versions are identical.

```
CALL
(D)CAPCN( F,G,U,P,WORK,MASK,TRIGS,NX,NY,DX,DY,C,CHARGE,IPS,
          VP,IP,JP,NP,ICFLAG,BA,BB,BC,BD,ITMAX,CCON,NITT,
          IWRITE,IERROR)
```

Input Variables

F — Real array dimensioned NX by NY. The forcing function. Values of F outside the domain of interest (MASK = 0) may be set to any value. This array is unchanged on exit.

G — Real array dimensioned NX-1 by NY-1, containing the values of the diffusion function. G should be positive within the irregular domain (defined by MASK = 1). Over those regions of the rectangle that are outside the irregular domain (MASK = 0), G may be set to any value. This array is unchanged on exit.

P — Real array dimensioned NX by NY. The initial estimate to the solution; if the user does not have an initial estimate, this array may be set to zero. The array is overwritten on exit.

WORK — Real array dimensioned NX by NY. A work-space array.

MASK Integer array dimensioned NX by NY. This array is used to identify the mesh points within the rectangle that lie inside the polygonal domain(s) in which solutions are required. The user must set MASK(I,J) = 1 for all values inside the irregular domain (i.e., where a solution is required) and set MASK(I,J) = 0 for other points, including boundary points. The mesh of grid points so defined coincides with the source function and the solution. This array is unchanged on exit.

NX Integer constant. The number of grid points in the x-direction. The value of NX is restricted such that (NX-1) is of the form $(2^i * 3^j * 5^k)$, where i is an integer greater than or equal to 1 and where j and k are integers greater or equal to 0.

NY Integer constant. The number of points in the y-direction.

NP Integer constant. The number of irregular boundary points.

DX Real constant. The discretization interval in the x-direction.

DY Real constant. The discretization interval in the y-direction.

CHARGE Real array dimensioned NP. A work-space array.

BA Real array dimensioned NY. The boundary values on the regular boundary at (I = 1, J = 2,NY-1).

BB Real array dimensioned NY. The boundary values on the regular boundary at (I = NX, J = 2,NY-1).

BC Real array dimensioned NX. The boundary values on the regular boundary at (I = 1,NX, J = 1).

BD Real array dimensioned NX. The boundary values on the regular boundary at (I = 1,NX, J = NY).

VP Real array dimensioned NP. The boundary values on the irregular boundary.

IP Integer array dimensioned NP. The coordinates in the x-direction of the irregular boundary points. The condition 1 < IP(N) < NX must hold for all N. The coordinates of the irregular boundary must not coincide with the boundaries of the rectangular domain.

JP Integer array dimensioned NP. The coordinates in the y-direction of the irregular boundary points. The condition 1 < JP(N) < NY must hold for all N. Note: the order in which the irregular grid points are specified is immaterial. However, all grid points must be distinct. Failure to ensure this will result in error condition IERROR = 7.

ICFLAG Integer constant. A flag: if ICFLAG = 0, the routine will compute the inverse capacitance matrix, C, and the pivot indices, IPS. If ICFLAG = 1, the routine will compute the inverse capacitance matrix, C, and the pivot indices, IPS, and solve the elliptic problem. If ICFLAG = 2, the routine will only solve the elliptic problem, assuming that preprocessing has already been done. ICFLAG is set to 2 on return.

CCON Real constant. The convergence criterion for the iteration. Iterations will proceed until solutions are achieved to any accuracy determined by CCON, until a lack of convergence is detected, or until the maximum number of iterations is reached.

ITMAX Integer constant. The absolute value, |ITMAX|, determines the maximum number of iterations permitted. If negative values of ITMAX are supplied, the routine suppresses the iterative check on convergence.

IWRITE Integer constant. Logical unit number to which to write out the iteration number and convergence conditions. If IWRITE = 0, this output is suppressed.

Output Variables

U — Real array dimensioned NX by NY. The solution in the irregular domain with U(IP(N),JP(N)) = VP(N) for N = 1, NP on the irregular portion of the boundary, and also with U(1,J) = BA(J) for J = 2, NY; U(NX,J) = BB(J) for J = 2, NY-1; U(I,1) = BC(I) for I = 1, NX; and U(I,NY) = BD(I) for I = 1, NX.

NITT — Integer constant. The number of iterations required to get the solution within the prescribed margin of error given by CCON.

C — Real array dimensioned NP by NP. The (factored) inverse capacitance matrix. This array is set by calling the routine with ICFLAG = 0 or 1. The array should not be altered on successive calls to the routine with different right-hand sides.

IPS — Integer array dimensioned NP. An integer vector of pivot indices. This array also should not be altered on successive calls with different right-hand sides.

TRIGS — Real array dimensioned 2*(NX-1). A real vector of coefficients required by the FFT routine employed by the Poisson solver. This array is set on the first call to the routine or when the flag ICFLAG = 0 or 1 is set. It must not be altered between successive calls to the routine when ICFLAG = 2.

IERROR — Integer constant. An error flag on input variables. On return values of IERROR may be interpreted as follows:

IERROR = 0—Normal status. No error detected.

IERROR = 1—Value of NX-1 or NY-1 is less than 8.

IERROR = 2—Illegal value for NX. See description of NX for allowed values.

IERROR = 3—Illegal value for IP or JP. See descriptions for allowed values.

IERROR = 4—Boundary points in IP or JP are inconsistent with the MASK array.

IERROR = 5—Illegal value for ICFLAG.

IERROR = 6—An illegal value (zero or negative) of G detected inside the domain.

IERROR = 7—An error occurred while trying to factor the inverse capacitance matrix. The matrix is probably singular. This will arise if there are duplicate boundary points.

IERROR = 8—Maximum number of iterations reached.

IERROR = 9—Nonconvergence of the iteration detected.

IERROR = 10—CCON is not positive.

IERROR = 11—Illegal value for DX or DY.

Note: Error checking is normally done only on the first call to the routine or when preprocessing is done. It may be forced by setting IERROR = −1.

ACKNOWLEDGMENTS

We thank W. Proskurowski and R. Boisvert for their meticulous reviews, which led to several improvements in the paper and in the algorithms. Clive Temperton kindly provided the fast Helmholtz solver used in this package.

REFERENCES

CONCUS, P., AND GOLUB, G. H. 1973. Use of fast direct methods for the efficient numerical solution of nonseparable elliptic equations. *SIAM J. Numer. Anal.* 10, 1103–1120.

CUMMINS, P. F., AND MYSAK, L. A. 1988. A quasi-geostrophic circulation model of the Northeast Pacific. Part I: A preliminary numerical experiment. *J. Phys. Ocean. 18*, 1261–1286.

DONGARRA, J. J., MOLER, C. B., BUNCH, J. R., AND STEWART, G. W. 1979. *LINPACK User's Guide*. SIAM Publications, Philadelphia, Pa.

HOCKNEY, R. W. 1970. The potential calculation and some applications. In *Methods of Computational Physics*, vol. 9, B. Adler, S. Fernbach, and M. Rotenberg, Eds. Academic Press, New York, 135–211.

PARES-SIERRA, A., AND VALLIS, G. K. 1989. A fast semi-direct method for the numerical solution of non-separable elliptic equations in irregular domains. *J. Comput. Phys. 82*, 398–412.

PROSKUROWSKI, W. 1983. A package for the Helmholtz equation in nonrectangular planar regions. *ACM Trans. Math. Softw. 9*, 1, 117–124.

PROSKUROWSKI, W., AND WIDLUND, O. 1976. On the numerical solution of Helmholtz's equation by the capacitance matrix method. *Math. Comput. 30*, 433–468.

RICE, J. R., AND BOISVERT, R. F. 1985. *Solving Elliptic Problems using ELLPACK*. Springer-Verlag, New York, 497pp.

SWARZTRAUBER, P. N. 1977. The methods of cyclic reduction, Fourier analysis and the FACR algorithm for the discrete solution of the Poisson equation on a rectangle. *SIAM Rev. 19*, 490–501.

SWARZTRAUBER, P. N., AND SWEET, R. A. 1979. Efficient Fortran subprograms for the solution of separable elliptic partial differential equations. *ACM Trans. Math. Softw. 5*, 352–364.

Received March 1992; revised February 1993 and April 1993; accepted April 1993