JOURNAL OF COMPUTATIONAL PHYSICS 82, 398–412 (1989)

A Fast Semi-direct Method for the Numerical Solution of Non-separable Elliptic Equations in Irregular Domains

ALEJANDRO PARES-SIERRA AND GEOFFREY K. VALLIS

Scripps Institution of Oceanography, University of California San Diego, La Jolla, California 92093

Received April 6, 1988; revised September 9, 1988

We investigate new algorithms for the solution of nonseparable elliptic equations in irregular domains. Such equations arise frequently in fluid dynamics and other branches of continuum mechanics. We show that the combined use of a fast iteration (involving the use of fast Poisson solvers (FPS) in rectangular domains) and the capacitance matrix method can lead to algorithms which are several times faster than traditional methods of successive over relaxation (SOR), even when the latter are vectorized. We also show that use of a certain acceleration procedure enables problems in irregular domains to be solved with only slightly more computational effort than in regular domains. © 1989 Academic Press, Inc.

1. INTRODUCTION

In many fields of physics, notably those in which numerical models play a significant role (e.g., plasma physics, fluid dynamics, magnetohydrodynamics) elliptic equations occur which must be solved numerically. In this paper we describe a method for the fast solution of non-separable elliptic equations in irregular domains and some illustrative computational experiments. The examples we have in mind relate to ocean modelling and are mainly 2-dimensional, although the methods used are more general.

The simplest nontrivial case of an elliptic equation is perhaps Laplace's or Poisson's equation in a rectangular domain, namely

$$\nabla^2\psi = q. \tag{1.1}$$

In quasi-geostrophic dynamics this equation occurs as a diagnostic equation for the stream-function ψ in terms of the forcing function, the quasi-geostrophic potential vorticity q. This equation would have to be solved, in two dimensions with given boundary conditions, at each timestep of a numerical model. An idealised model would perhaps have a rectangular domain, whereas if the model were attempting to cover a realistic ocean the domain would obviously be irregular.

In "primitive equation" numerical models (i.e., typically in oceanography those

0021-9991/89 $3.00

models derived from the Navier–Stokes equations only with the Boussinesq approximation and the assumption of hydrostatic balance) a more complicated, generally non-separable, equation ensues. Non-separable equations do not allow their solution to be written in the form $p(x)r(y)$, where p and r are arbitrary functions of the co-ordinates x and y, respectively, a fact which precludes the direct use of fast solvers. The equation arises specifically from the imposition of a (artificial) rigid lid at the top of the fluid. This prevents the propagation of the "external barotropic mode," namely, a wave of the shallow water equations which propagates at an approximate speed c given by $c=\sqrt{gh}$. For an ocean 5 km deep, c is approximately 0.2 km/s. With a grid of say 25 km this would necessitate a CFL limited timestep of 2.5 min, clearly unrealistic for integrations of many years. Fortunately, this mode is believed unimportant for large scale ocean dynamics, and it may be eliminated by a rigid lid approximation without corrupting the dynamics. (Other methods of treating it are also possible, typically involving the time-stepping procedure, but we are not concerned with that here.)

Invoking the rigid lid approximation leads to a 2-dimensional diagnostic equation for the pressure field at the physical top of the ocean. We refer the reader to Holland and Lin [1] or Bryan [2] for derivations. The equation is of the form:

$$\nabla\cdot(g(x,y)\nabla\psi)=\rho(x,y). \tag{1.2}$$

If the domain is regular (e.g., rectangular) we shall call the domain $\mathscr{R}$ with boundary $\partial\mathscr{R}$. An irregular domain we shall refer to as $\mathscr{S}$ with boundary $\partial\mathscr{S}$. Equation (1.2) also occurs frequently in plasma dynamics. In (1.2) ψ is the unknown field (with given boundary conditions, say $\psi=\psi_b$ on $\partial\mathscr{R}$ or $\partial\mathscr{S}$), $g(x, y)$ is a given function physically related to the underlying topography (and usually constant in time), and $\rho(x, y)$ is a time varying forcing function. The problem, then, is to solve (1.2) in a irregular domain, $\mathscr{S}$, many times with different forcings but with the same geometry. In many modelling applications in fluid dynamics (1.2) has to be solved at each timestep, and there may easily be several thousand timesteps. Thus, frequently, very extensive preliminary computational work which may be necessary to solve (1.2) is quite acceptable, since its overall contribution to the computational load then becomes insignificant. This is obviously not the case if the geometry changes each timestep.

The important differences between (1.2) and the simplest case (1.1), then, are: (i) Eq. (1.2) is in general non-separable and (ii) we wish to solve (1.2) in an irregular domain. For these two reasons slow iterative methods (e.g., successive over relaxation) have in the past often been used. Solving (1.2) this way is often the largest expense in fluid codes. Our purpose is to show that, given a fast solver in a rectangular region, much faster methods can be employed. In the next section we shall describe these methods. In Section 3 we shall describe some numerical results.

2. METHODOLOGY

The method is a combination of the capacitance matrix method (Buzbee *et al.* [3]; Proskurowski and Widlund [4]) and a fast iteration (Concus and Golub [5]), which itself uses a fast Poisson solver (FPS) such as multiple Fourier transforms or the FACR algorithm (Swartztrauber [6]). We shall give a brief review of each method and then show how they may be combined. Our discussion will be aimed more toward the physicist than the numerical analyst and no proofs of consistency or convergence are offered.

2.1. *Heuristic Description of the Capacitance Matrix Method*

The physical origin of the method lies in the replacement of boundary conditions by point charges (e.g., Hockney [7]). The problem is to solve an elliptic equation in an irregular domain; for simplicity we shall consider for now only Poisson's equation with Dirichlet boundary conditions:

$$\nabla^2\psi = \rho(x, y). \tag{2.1}$$

The method is trivially extended to Helmholtz's equation $\nabla^2\psi - \kappa^2\psi = \rho(x, y)$. The general philosophy is as follows. We first solve (2.1) in a regular (say rectangular) region $\mathscr{R}$, with any smooth (typically zero) boundary conditions for ψ on $\partial\mathscr{R}$, by any available method. (A typical fast algorithm might involve double Fourier transforms, or be the FACR method.) Embedded within this is the region of interest, say $\mathscr{S}$ with boundary $\partial\mathscr{S}$ (Fig. 1). The source term ρ is given everywhere in $\mathscr{S}$, is zero on $\partial\mathscr{S}$ and its value is irrelevant elsewhere. Let the solution of (2.1) on $\mathscr{R}$ be ψ_1. Suppose its value on $\partial\mathscr{S}$ is ψ_b. Then we have found the (unique) solution to (2.1) on $\mathscr{S}$ with boundary condition ψ_b. Suppose now we find the solution to Laplace's equation

$$\nabla^2\psi_2 = 0 \tag{2.2}$$

with $\psi_2 = -\psi_b$ on $\partial\mathscr{S}$. Since the equations are linear, we may add the solutions and find

$$\nabla^2\psi = \rho, \tag{2.3}$$

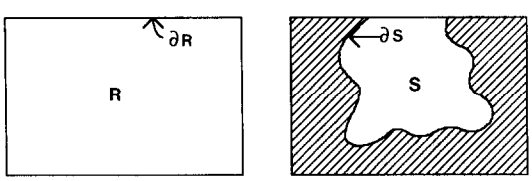

FIG. 1. Schema of regular domain $\mathscr{R}$ with boundary $\partial\mathscr{R}$, and irregular domain $\mathscr{S}$ with boundary $\partial\mathscr{S}$. $\mathscr{S}$ may be wholly embedded within $\mathscr{R}$, or may share a common boundary.

where $\psi = \psi_1 + \psi_2$ (where $\nabla^2\psi_1 = \rho(x, y)$ and $\nabla^2\psi_2 = 0$) and ψ will be the solution of (2.1) on $\mathscr{S}$ with $\psi = 0$ on $\partial\mathscr{S}$. This is the required, and unique, solution to our problem.

The only difficulty, then, is to solve (2.2) with the required boundary condition on $\partial\mathscr{S}$. Evidently there are at least two methods to do this. The first ("A") is direct but requires a lot of storage (of the order $M \times N$, where M is the number of grid points in the domain $\mathscr{R}$, and N is the number of irregular grid points, i.e., the number of grid points on $\partial\mathscr{S}$). The second ("B") requires another call to the fast solver, but requires only $N \times N$ words of storage.

(a) *Method A.* The solution to $\nabla^2\psi_2 = 0$ with $\psi_2 = -\psi_b$ on $\mathscr{S}$ is given by

$$\psi_2(x, y) = -\int \psi_b(s')\mathscr{G}_b(x, y \mid s')\, ds', \tag{2.4}$$

where $\mathscr{G}_b$ is a boundary Green's function and the integration ds is along $\partial\mathscr{S}$ (Morse and Feshback [8]). Numerically, we would implement this by first evaluating the appropriate discrete form of $\mathscr{G}_b$ (an $M \times N$ matrix), multiply it by ψ_b and evaluate the integral as a sum along $\partial\mathscr{S}$. The sum $\psi_1 + \psi_2$ is the required solution. Since $\mathscr{G}_b$ is a very large matrix, the use of this method is impractical for interesting problems with current computers. Thus we shall leave this method aside.

(b) *Method B.* This method relies on one lemma. It is that the solution of (2.2) may be obtained by replacing the inhomogeneous boundary conditions with an appropriate distribution of "source" (i.e., terms which would appear on the right-hand side) along the boundary $\partial\mathscr{S}$. This is a well-known result (e.g., Morse and Feshback [8]), although perhaps not immediately obvious.

This means the following. Let the solution of the homogeneous problem (with inhomogeneous boundary conditions on $\partial\mathscr{S}$) be ψ_2. Then we can write

$$\psi_2(x, y) = -\int \hat{\rho}(x', y')\mathscr{G}(x, y \mid x', y')\, dx'\, dy',$$

where $\hat{\rho}(x, y)$ (the extra source) *has nonzero values only along the boundary* $\partial\mathscr{S}$. Thus we can immediately write

$$\psi_2(x, y) = -\int \hat{\rho}(s')\mathscr{G}(x, y \mid s')\, ds'. \tag{2.5}$$

This is very similar to (2.4), except that the Green's function is different, and the source term $\hat{\rho}$ is not the actual boundary value of streamfunction. It is in fact unknown at this stage. It is impractical to keep $\mathscr{G}(x, y \mid s')$. However, if we can evaluate $\hat{\rho}$ we can obtain ψ_2 directly by a call to our fast solver. Now, consider (2.5) on $\mathscr{S}$. Then

$$\psi_2(s) = \psi_b = -\int \hat{\rho}(s')\mathscr{G}(s \mid s')\, ds'. \tag{2.6}$$

We shall call $\mathcal{G}(s \mid s')$ a partial Green's function. It is just the projection of the Green's function on $\partial\mathcal{S}$, in the sense of Proskurowski and Widlund [4]. We can write (2.6) in a matrix form $\boldsymbol{\psi}_b = \hat{\boldsymbol{\rho}}_s \mathbf{G}$ where $\boldsymbol{\psi}_b$ is the vector of values of ψ_b along S, $\hat{\boldsymbol{\rho}}_s$ is to be determined and $\mathbf{G}$ is the appropriate discrete partial Green's function. Then simply

$$\hat{\boldsymbol{\rho}}_s = \boldsymbol{\psi}_b \mathbf{G}^{-1}. \tag{2.7}$$

Given $\hat{\rho}_s$, (2.3) is equivalently written $\nabla^2 \psi_2 = \hat{\rho}_s$ (where the discrete form of all operations will henceforth be implicit). The solution of this will be equivalent to the solution of (2.3). Given $\hat{\rho}_s$, the solution to our problem is, in fact, most easily obtained as the direct solution to the equation

$$\nabla^2 \psi = \rho + \hat{\rho}_s. \tag{2.8}$$

In summary, a recipe for using the capacitance matrix method is to first determine the partial Green's funtion for the particular geometry by solving (2.1) many times, each time with a unit source on each irregular gridpoint. The discrete partial Green's function is simply the matrix of solutions at each boundary point for each source. (It is clear from (2.7) that it is an "impulse-response" matrix, giving the response ψ from a source ρ.) Then, each timestep one first solves (2.1) in a regular domain $\mathcal{R}$ with the known source. Using (2.7) one obtains the modified source, and the solution with the required boundary conditions is obtained by another call to the fast solver to obtain the solution of (2.8). We note that the explicit capacitance matrix method described above necessitates determining the Green's function, or capacitance matrix. Although this is rather time consuming, as stated in the Introduction it may then be used many thousands of times and its relative burden then becomes negligible. For further numerical discussion, the reader is referred to Prokurowski and Widlund [4].

2.2. *Fast Iterative Methods for Non-separable Elliptic Equations*

The solution of (1.2) in a rectangular domain is obtained by the iterative procedure

$$\nabla^2 \psi^{n+1} = (\rho - \nabla \psi^n \cdot \nabla g)/g, \tag{2.9}$$

where superscript n denotes the iterate. At each iteration the right-hand side is assumed to be a known forcing, and a fast solver can be employed for what is simply Poisson's equation in a regular domain. There are many ways to speed convergence; here we mention only two, drawn from Concus and Golub [5]. The first is a form of preconditioning, in which the variable ψ is replaced by $\phi(x, y) = [g(x, y)]^{1/2} \psi(x, y)$. Equation (1.2) becomes

$$\nabla^2 \phi - g' \phi = f', \tag{2.10}$$

where $g'(x, y) = g^{-1/2}\nabla^2 g^{1/2}$ and $f'(x, y) = g^{-1/2}\rho(x, y)$. Second, Chebychev acceleration may be employed. This involes

$$\psi^{n+1} = \omega^{n+1}(\tilde{\psi}^{n+1} - \psi^{n-1}) + \psi^{n-1},$$

where $\omega^1 = 1$, $\omega^2 = 2/(2-p^2)$, $\omega^{n+1} = (1 - p^2\omega^n/4)^{-1}$, and p is the spectral ratio of the iteration matrix and $\tilde{\psi}$ refers to a preliminary value of ψ. However, neither of these add-ons are, in principle, essential although in practice they may be very useful.

2.3. *Non-separability and Irregularity*

To solve (1.2) in an irregular domain we combine the methods of Sections 2.1 and 2.2. There are two somewhat distinct ways of doing this, which we shall describe separately. The second perhaps appears at first sight more natural, but the first is faster.

(a) *Method* 1. In this method we obtain the partial Green's function for the Laplacian operator in $\mathscr{S}$, just as in Section 2.1b. Then each iteration step simply involves first solving (2.9) on $\mathscr{R}$, then applying the capacitance matrix multiplication (2.7) to obtain the modified source, and then solving (2.9) again but with a modified source, as in (2.8). Explicitly this algorithm goes as follows.

Solve

$$\nabla^2\psi^{n+1} = (\rho - \nabla\psi^n \cdot \nabla g)/g. \tag{2.11a}$$

Now obtain a boundary source:

$$\hat{\rho}_s = \psi_b \mathscr{G}^{-1}. \tag{2.11b}$$

Now solve again:

$$\nabla^2\psi^{n+1} = (\rho + \hat{\rho}_s - \nabla\psi^n \cdot \nabla g)/g. \tag{2.11c}$$

Repeat until convergence.

Note that at each iteration step the boundary conditions will be exactly satisfied. The procedure may be significantly accelerated by replacing (2.11c) by

$$\nabla^2\psi^{n+2} = (\rho + \hat{\rho}_s - \nabla\psi^{n+1} \cdot \nabla g)/g \tag{2.11d}$$

which has the flavour of the Gauss–Seidel method. At each iteration the boundary conditions are no longer exactly satisfied but will be satisfied when convergence is achieved, to the desired accuracy. If it is felt necessary to exactly satisfy the boundary conditions, then the last iteration could use (2.11c) instead of (2.11d). Using (2.11d), we are in a loose sense getting two iterations for the price of one, and the procedure may now be expected to converge almost as rapidly as if there

were no need to transform from $\mathscr{R}$ to $\mathscr{S}$. If so, then since (2.11b) is very fast, the entire method will be only a little slower than solving (1.2) in a rectangular region.

(b) *Method* 2. This involves first evaluating the partial Green's function, not for the Laplacian operator, *but for the operator* $\nabla(g \cdot \nabla *)$ on $\mathscr{S}$. To do this one must solve

$$\nabla \cdot (g(x, y)\nabla\psi) = \delta_{\mathbf{x}, \mathbf{x}'} \tag{2.12}$$

several times, each time with x' being a different gridpoint along $\partial\mathscr{S}$. This can be done with a solver in a rectangle, but we note that the equation is non-separable, so no very fast solver is available. Thus, the iteration procedure of (2.9) must be employed. The procedure is fairly slow just because we must solve (2.12) as many times as there are irregular gridpoints, and each time an iteration procedure must be used. Nevertheless, one has a good first guess each time (namely the solution from the adjacent delta function source, which can also be shifted a grid point or two for an even better first guess). Further, for applications where the geometry and diffusion function are unchanging this need only be calculated once and for all, and the associated computer time is then negligible. Given the Green's function, the algorithm goes as follows. We calculate the solution to (1.2) in $\mathscr{R}$ using the iteration (2.9). We then calculate the modified source by the capacitance matrix multiplication (2.7). Then we calculate the solution to (1.2) in $\mathscr{S}$; that is, solve (1.2) in $\mathscr{R}$ with a modified source:

$$\nabla \cdot (g(x, y)\nabla\psi) = \rho(x, y) + \hat{\rho}(s). \tag{2.13}$$

This is done using the iteration (2.9). On convergence we have the required solution. Explicitly, we solve (1.2) in $\mathscr{R}$ by iterating as many times as needed:

$$\nabla^2\psi^{n+1} = (\rho - \nabla\psi^n \cdot \nabla g)/g. \tag{2.14}$$

Then we obtain a boundary source

$$\hat{\rho}_s = \psi_b \mathscr{G}^{-1}. \tag{2.15}$$

Then we solve (2.12) by the iteration:

$$\nabla^2\psi^{n+1} = (\rho + \hat{\rho}_s - \nabla\psi^n \cdot \nabla g)/g. \tag{2.16}$$

The capacitance matrix multiplication (2.15) is performed only once. The boundary conditions on $\partial\mathscr{S}$ will not be satisfied exactly by (2.16) until convergence has been achieved.

3. Numerical Results

In this section we present some results of the numerical experiments used to illustrate the ideas of the previous sections. The algorithms for Methods 1 and 2 described in Section 2 explicitly proceed in the following step:

Method 1. 1. Preprocessing. Obtain capacitance matrix $\mathscr{G}$ with the elements $\mathscr{G}_{ij}$ being the solution values at the ith irregular point resulting from a unit source at the jth irregular point. It is the matrix formed of the vectors P_i^j, where

$$\nabla^2 P_i^j = \delta_i^j, \qquad \delta_i^j = \begin{cases} 1, & \text{at irregular point } j; \\ 0, & \text{elsewhere.} \end{cases}$$

Whereas the solution is naturally otained over the whole domain, the Green's function is only saved along the irregular gridpoints (denoted here by subscript i) giving us the partial Green's function.

2. Solve $\nabla^2 \psi^{n+1} = [\rho - \nabla\psi^n \cdot \nabla g]/g$ using a FPS. ρ is the original source, zero on $\partial\mathscr{S}$.

3. Obtain boundary source vector: $\hat{\rho}_s = -\mathscr{G}^{-1}\psi_B$, where $\psi_B = \psi^{n+1}(x \in s)$.

4. Solve $\nabla^2 \psi^{n+2} = [\rho + \hat{\rho}_s - \nabla\psi^{n+1} \cdot \nabla g]/g$.

5. Check convergence, $\psi^{n+2} \to \psi^n$, and repeat from step 2 until convergence.

Method 2. 1. Preprocessing. Get capacitance matrix $\mathscr{G}$, where the element $\mathscr{G}_{ij}$ is the value at the ith irregular point of the solution of:

$$\nabla \cdot g\nabla P_i^j = \delta_i^j, \qquad \delta_i^j = \begin{cases} 1, & \text{at irregular point } j; \\ 0, & \text{elsewhere.} \end{cases}$$

2. Solve iteratively using an FPS and (2.9) the equation: $\nabla \cdot g\nabla\psi_1 = \rho$.

2. Get boundary source vector: $\hat{\rho}_s = \mathscr{G}^{-1}\psi_B$, where $\psi_B = \psi_1$ $(x \in s)$.

3. Solve iteratively using FPS the equation: $\nabla \cdot g\nabla\psi = \hat{\rho}_s + \rho$.

The two forcing (ρ) and four diffusion functions (g) used in this study are shown in Table I. These functions were made up, but cover a broad range of possibilities which may arise in physical applications. In oceanography the diffusion function defined by (1.2) is related to the topography and hence it is essentially random. Results of our experiments are shown in Table II and III. All cases were done on

TABLE I

Forcing (ρ) and Diffusion (g) Functions Used in the Numerical Experiments

ρ	1	Const = 5
	2	$10\cos[10\pi(x+y)]$
g	1	$[0.5 + \cos(\pi(x+y)/6)]^2$
	2	$\exp[-(x+y)]$
	3	$1/x$
	4	$100x$

Note. The x and y values both run between 0 and 1 in $\mathscr{R}$.

TABLE II

Comparison of the time of convergence (in s) for the five solvers described in the text

ρ	g	Method 1	Method 2	Method 3	SOR2	SOR1
1	1	53.50	301.80	53.50		
		3.91	4.69	2.60	21.00	74.88
		19.15	15.97	28.80	3.56	1.00
2	1	53.50	301.80	53.50		
		3.98	4.07	1.87	19.77	72.10
		18.12	17.71	38.56	3.65	1.00
1	2	53.50	473.66	53.50		
		5.77	6.37	3.16	20.06	74.53
		12.92	11.70	23.58	3.72	1.00
2	2	53.50	473.66	53.5		
		4.85	5.14	1.96	19.86	73.64
		15.18	14.33	37.57	3.71	1.00
1	3	53.50	546.66			
		9.62	10.44	[a]	22.86	85.56
		8.89	8.19		3.74	1.00
2	3	53.50	546.66			
		4.80	5.80	[a]	20.24	75.11
		15.65	12.95		3.75	1.00
1	4	53.50	329.76			
		6.76	6.91	[b]	21.37	7871
		11.64	11.39		3.68	1.00
2	4	53.50	329.76			
		3.87	4.60	[b]	19.91	73.01
		18.87	15.87		3.67	1.00

[a] Very slowly convergent.

[b] Not convergent.

Note. The domain is that of Fig. 2 with a size of 128 * 128 and 173 irregular points. The forcing function (ρ) and diffusion function (g) are given in Table I. The first, second, and third values in each cell represent the time of preprocessing, net solution time, and speedup ratio relative to SOR1, respectively.

an Alliant FX8 computer. We do not expect relative timings to be significantly different on any other vectorizing machine. We do not use machine coded FFTs for the fast Poisson solver, simply for portability and consistency between methods. However, such FFTs are available for many machines (including the Alliant). If used, they would speed up the FPS by about a factor of two, further adding to the relative speedup.

Two SOR methods were implemented for timing references: SOR1 refers to the usual successive over-relaxation method while SOR2 is the black–white checker-

TABLE III

Same as Table II but for a Problem Size of 64 * 64 Mesh Points and 84 Irregular Points

ρ	g	Method 1	Method 2	Method 3	SOR2	SOR1
1	1	7.88	40.32	7.88		
		1.08	1.32	0.57	2.69	9.46
		8.75	7.17	16.60	3.52	1.00
2	1	7.88	40.32	7.88		
		0.84	1.02	0.58	2.38	8.34
		9.93	8.18	14.38	3.50	1.00
1	2	7.88	51.71	7.88		
		1.35	1.78	0.57	2.58	9.05
		6.70	5.08	1.88	3.51	1.00
2	2	7.88	51.71	7.88		
		1.08	1.32	0.59	2.38	8.47
		7.84	6.42	14.36	3.56	1.00
1	3	7.88	60.05	7.88		
		1.91	2.77	18.94[a]	2.79	9.88
		5.17	3.57	0.52	3.54	1.00
2	3	7.88	60.05	7.88		
		1.11	1.33	10.86[a]	2.51	8.83
		7.95	6.64	0.81	3.52	1.00
1	4	7.88	31.70			
		1.36	1.97	[b]	2.75	9.62
		7.07	4.88		3.50	1.00
2	4	7.88	31.70			
		0.64	1.00	[b]	2.40	8.39
		9.99	8.39		3.50	1.00

[a] Very slowly convergent.
[b] Not convergent.

board technique described by Buzbee *et al.* [9]. This latter, "coloured" SOR version allows for vectorization and concurrency, greatly improving its performance on vector machines. On a scalar machine no such speedup would be apparent. For both SOR methods the optimal overrelaxation parameter (determined empirically) was used. The fifth method considered (Method 3 in Tables II, III) is a scaled-shifted version of Method 1. The scaling and shifting were done following the ideas of Concus and Golub [5]. For this method, the scaled (2.10) is solved iteratively as in Method 1 after first shifting the differential operator by K, i.e., solve

$$[\nabla_h^2 - K]\psi^{n+1} = [g' - K]\psi^n + f',$$

where the shifting parameter K is given by

$$K = [\min(g') + \max(g')]/2.$$

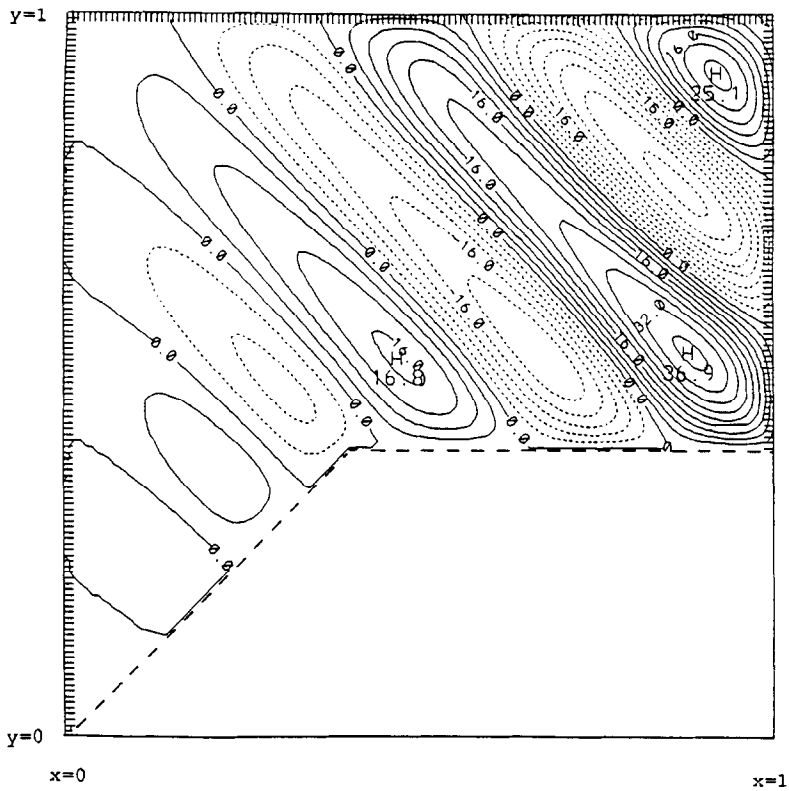

FIG. 2. Domain used in the numerical experiments listed in the tables (i.e., $\mathscr{S}$ and extended rectangular region $\mathscr{R}$ used in the capacitance method). Broken line marks the position of the irregular points. We used FACR for the solver on $\mathscr{R}$, with $\psi = 0$ on $\partial\mathscr{R}$. The contours are from the experiment with ρ_2 and g_3.

The shape of the domain was the same for all cases reported here (see Fig. 2). It is rather simple with relatively few irregular grid points, just for demonstration purposes. Two sizes are reported: A $128 * 128$ matrix with 173 irregular points (Table II) and a $64 * 64$ matrix with 84 irregular points (Table III). The power of 2 for the size of the domain is not crucial; it was chosen to optimize the efficiency of the particular FFT used in the fast Poisson–Helmholtz solvers (FPS) of Methods 1, 2, and 3. Mixed radix FFTs are of course available; using one would not significantly slow the results provided the numer of grid points has no prime factors higher than 5, not an especially restrictive requirement. In all cases the initial guess was zero and the iteration was stopped when the norm of the residual had been reduced by a factor of 10^{-3}. We have performed many other experiments with more irregular domains, with essentially similar results. The method is limited only by the size of the capacitance matrix, which should ideally fit in the fast machine core for speed and ease of coding on non-virtual machines. For a three

megaword machine (e.g., most Cray X-MPs one could easily use a domain of 500×500 with 750 irregular gridpoints. In an eight megaword machine (Alliant, Cray 2, etc.) once could easily fit a problem of 1000×1000 with 2000 irregular gridpoints in core.

Tables II and III show the time of preprocessing (i.e., computation of the capacitance matrix) as well as the time employed at getting the solution once the capacitance matrix is known. The third value in each cell gives the ratio of the time spent with respect to SOR1. For the type of application in mind (i.e., solving non-separable elliptic equations coming from time dependent problems) the diffusion function, as well as the position and number of irregular points do not change with time. This allows for the capacitance matrix to be calculated and factored out once and for all as a preprocessing of the problem. For a long enough time integration, (a typical oceanographic application could involve the integration of one year with a 30 min time interval, i.e., 17280 time steps!) the relative importance of the preprocessing stage becomes negligible and, as shown in Tables II and III, the superiority of the capacitance-iteration method over SOR is evident. Another extra benefit of the fact that neither the differential operator nor the geometry changes in a time stepping problem is the great improvement in speed that can be attained by using, as an initial guess at a given time step, the solution from the previous timestep instead of a zero field as was done for the present examples. In Table IV the iteration by iteration details are given for the example shown in Fig. 2 (experiment with ρ_2 and g_3 of Table I).

The g_1 and g_2 used (Table I) are such that the scaled function $g' = g^{-1/2}\nabla(g^{1/2})$ is nearly constant. As shown by Concus and Golub [5] this condition is the optimum for the scale-shifting procedure. This is clearly reflected in our results; Method 3 (the scaled-shifted version of Method 1) gives the best timing for g_1 and g_2, it is about twice as good as the unscaled version (Method 1) and more than 30

TABLE IV

Iteration by Iteration Details for the Example Given in Fig. 2 (f_2, g_3)

	Domain 64×64 84 irregular points		Domain 128×128 173 irregular points	
Iteration n	$\frac{\|\psi^n - \psi^{n-1}\|}{\|\psi^n\|}$	Max error	$\frac{\|\psi^n - \psi^{n-1}\|}{\|\psi^n\|}$	Max error
1	1.0	5.1(−1)	1.0	1.9
3	6.1(−3)	2.6(−2)	6.7(−3)	1.2(−1)
5	3.3(−4)	1.5(−3)	7.1(−4)	1.2(−2)
7	4.4(−5)	2.0(−4)	1.2(−4)	3.0(−3)
9	5.9(−6)	2.8(−5)	1.5(−5)	3.7(−4)
11	6.1(−7)	3.7(−6)	3.8(−6)	7.4(−5)

Note. The maximum error was computed with respect to an "exact" solution defined as that with $\|\psi^n - \psi^{n-1}\|/\|\psi^n\| < 1.0 \cdot 10^{-15}$

TABLE V

Comparison of the Time of Convergence (in s) for Method 1 Described in the Text for a Regular and Irregular Domain

	Rectangular	Irregular
$\rho = 10[y - 1/2]$ $g = \exp[-(x+y)]$	1.31	1.66
$\rho = 10\cos[5\pi(x+y)]$ $g = 1/x$	1.15	1.36

times faster than the regular SOR. For a realistic case, however, the g will not in general satisfy this condition and the scaling-shifting can make things worse (as in the case of g_3) or even make the procedure not convergent at all as when the very simple g_4 was used.

Method 1 gives consistently very good timings compared with all other methods. In particular, not considering preprocessing time, Table II shows it to be from 3–7 and 8–20 times faster than the coloured and regular SOR, respectively. Method 2 gives consistently better times than SOR, but in general it is slower than Method 1. It might be thought that Method 2 should be a little faster than Method 1 (again not considering preprocessing time), since it saves $N-1$ matrix multiplications needed to get the extra source, with N being the number of iterations needed. However, these multiplications are fast and the fact that Method 1 allows for the acceleration procedure implied in (2.11d) makes this method faster than Method 2. If acceleration is not used, the method is slowed by almost a factor of 2. Furthermore, the acceleration makes the solution on an irregular domain (using Method 1) almost as fast as the solution of the corresponding problem on a rectangular domain. Table V presents the results of an experiment to analyse this point. It shows that, for the same problem characteristics (i.e., same functions, level of convergence, size, etc.) the presence of the irregular boundary does not significantly slow down the solution.

4. Discussion

The results obtained above show that fast Poisson solvers in a rectangle can be used as building blocks to obtain efficient algorithms for more general non-separable equations in irregular domains. The results generally speak for themselves. To conclude, we shall simply outline various relevant points.

(i) A combination of the capacitance matrix method and a fast iteration produces a method which is several times faster than even a vectorized (coloured) SOR method.

(ii) The method is most efficient for problems with unchanging geometry, where the equation must be solved several times, and the initial proprocessing time becomes negligible. This is a typical situation in numerical models of fluid dynamics. Here, the capacitance matrix may be evaluated once and for all.

(iii) It is *not* necessary that the diffusion function be a constant (with time), since only the capacitance matrix of the Laplacian operator need be evaluated.

(iv) Because of a certain acceleration procedure, the computation of the solution in irregular domains is little more expensive than the computation in regular domains, not counting the time taken to compute the capacitance matrix itself.

(v) The very fast method of Proskurowski and Widlund [4] of computing the capacitance matrix, utilising the translation invariance of the problem, could be utilized for Method 1, but not for Method 2. This was not implemented here because preprocessing is not an important time factor in time-dependent modelling problems. Use of this method with those described above could lead to efficient algorithms even for single use implementations.

(vi) The principal limitation of the method lies in the need to keep a potentially large capacitance matrix. This may mean that for *very* large problems, or for problems where the boundary is very irregular, the coloured SOR method may be preferable. However, since the number of operations in the fast methods goes approximately as the number to do an FFT, namely $N \log N$, where N is the total number of gridpoints, (since the capacitance matrix multiplications are not time consuming) the capacitance-iteration method will still be relatively more efficient for large problems as indicated by a comparison of Tables II and III. The implicit capacitance matrix method (Proskurowski [10], Faber *et al.* [11]) overcomes the large storage requirements of the explicit methods described, since it does not require storage of the capacitance matrix, only that its product with a known vector can be calculated. There is perforce some additional cost in computational effort. Nevertheless, it could be used in conjunction with the above methods.

(vii) In time-stepping problems, at each timestep the first guess will be relatively good, being that from the previous timestep. Very few (perhaps 1–3) iterations are all that may be necessary.

(viii) We have not performed any numerical experiments with Neumann boundary conditions, for which the boundary condition $\psi = 0$ on $\partial\mathcal{S}$ is replaced by $\partial\psi/\partial n = 0$ on $\partial\mathcal{S}$. However, in principle this should cause no problem, since each iteration (2.11) is just the application of now standard capacitance matrix techniques, and Neumann boundary conditions on $\partial\mathcal{S}$ can be implemented at this stage (i.e., at each iteration). See, for example, Proskurowski and Widlund [4] or Buzbee *et al.* [3].

(ix) The addition of non-zero (Dirichlet) boundary conditions on $\partial\mathcal{S}$ causes no problems. The easiest way is to use the standard "trick" of using the required boundary value as an additional source one gridpoint from the boundary (if using

a five point Laplacian stencil). This simple method works only because the method ensures a solution with $\psi_b = 0$ on $\partial\mathscr{S}$; the boundary value is applied after the solution has been obtained with the extra source. This rather coarse approximation is acceptable when the boundary coincides with grid points.

(x) It would be interesting to compare the methods described herein with conjugate gradient and multi-grid methods. These may be attractive alternatives.

ACKNOWLEDGMENTS

We thank W. Holland of NCAR for introducing us to the problem and providing a fast Poisson solver. We are indebted to W. Proskurowski of USC for useful conversations and encouragement and for a very useful review. The work was funded by ONR under a University Research Initiative award (USN N00014-86-K-0752) to Scripps.

REFERENCES

1. W. R. HOLLAND AND L. B. LIN, *J. Phys. Oceanogr.* **5**, 642 (1975).
2. K. BRYAN, *J. Comput. Phys.* **1**, 347 (1969).
3. B. L. BUZBEE, F. W. DORR, J. A. GEORGE, AND G. H. GOLUB, *SIAM J. Numer. Anal.* **4**, 722 (1971).
4. W. PROKUROWSKI, *ACM Trans. Math. Software* **5**, 36 (1979).
5. P. CONCUS AND G. H. GOLUB, *SIAM J. Numer. Anal.* **10**, 1103 (1973).
6. P. N. SWARZTRAUBER, *SIAM Rev.* **19**, 490 (1977).
7. R. HOCKNEY, *Methods Comput. Phys.* **9**, 135 (1970).
8. P. M. MORSE AND H. FESHBACK, *Methods of Theoretical Physics* (McGraw–Hill, New York, 1953), p. 791.
9. B. L. BUZBEE, G. H. GOLUB AND J. A. HOWELL, in *Proceedings, Symposium on High Speed Computer and Algorithm Organization, Urbana, Illinois 1977*, edited by D. Kuck, D. Lawrie, and A. Sameh (Acadmic Press, New York/London, 1977), p. 255.
10. W. PROSKUROWSKI AND O. WIDLUND, *Math. Comput.* **30**, 433 (1976).
11. V. FABER, A. WHITE, AND R. SWEET, in *Advances in Computer Methods for Partial Differential Equations—IV, 1981*, edited by R. Vichnevetsky and R. S. Steplemen (IMACS, New Brunswick, NJ, 1981).