UCLA
COMPUTATIONAL AND APPLIED MATHEMATICS

Capacitance Matrix Methods for the Solution of Elliptic Equations Defined by Level Set Functions

Christopher R. Anderson

May 2000

CAM Report 00-15

Department of Mathematics
University of California, Los Angeles
Los Angeles, CA. 90095-1555

http://www.math.ucla.edu/applied/cam/index.html

UCLA
COMPUTATIONAL AND APPLIED MATHEMATICS

Capacitance Matrix Methods for the Solution of Elliptic Equations Defined by Level Set Functions

Christopher R. Anderson

May 2000

CAM Report 00-15

Department of Mathematics
University of California, Los Angeles
Los Angeles, CA. 90095-1555

Capacitance Matrix Methods for the Solution of Elliptic Equations Defined by Level Set Functions

C.R. Anderson *

May 10, 2000

Abstract

We describe methods, based on the capacitance matrix technique, for accelerating the iterative solution of elliptic equations on irregular regions whose boundary is delineated by the contour of a level set function. The application of the technique and computational results are presented for Poisson's equation and the modified Helmholtz equation

1 Introduction

In recent years there have been tremendous advances in the capabilities to accurately compute interface motion in a wide variety of applications, from multi-phase fluid flow, to crystal growth, to the modeling of semiconductors. In many of these applications the computation of the interface motion requires the solution of elliptic equations on the domain whose boundary is delineated by the interface. The requirement that the elliptic equation be solved at each timestep in the interface evolution introduces great computational cost, and there is a demand for fast solvers to provide these solutions. A popular technique for computing the evolution of interfaces is the level set method; a method where the interface is defined implicitly as a contour of an evolving level set function (see [8] or [12] for general details on the level set method and references). Since the evolution of the level set function is often computed using a uniform rectangular grid, it is common to use discretizations of the elliptic problem that also use this underlaying uniform grid. For the Poisson or the modified Helmholtz equation, standard five point, second order difference approximations are used at points away from an interface and special difference approximations are used near an interface to account for the presence of the boundary. There are a variety of ways to create such discretizations and a good introduction to them is given in [15]. More recent work on this type of discretization is presented in [6][7][8]. However, once a choice of discretization has been made, the resulting system of linear equations must still be solved. This paper is concerned with this latter task, specifically, we discuss computationally efficient iterative methods for the solution of these equations.

The methods described here are capacitance matrix methods (see [14] for a general description and overview of early work) that consist of either preconditioned conjugate gradients (PCG) [4] or Generalized Minimum Residual (GMRES) [11] used in association with the equations arising from an application of the Sherman-Morrison-Woodbury formula [5]. The success of the approach

*Research supported in part by National Science Foundation/ARPA Grant NSF-DMS-961584.

depends upon the availability of a proconditioner that gives an approximate solution that satisfies a large number of the original equations exactly. In the case of discretizations of Poisson's and Helmholtz's equation, fast Poisson solvers based upon discrete Fourier transforms or block cyclic reduction [4][13] can be used. Given such a preconditioner, the use of the equations arising from the method of modification allows one to implement a PCG or GMRES iteration that is only applied to the unknowns associated with equations where the preconditioner and the original operator differ. This approach immediately leads to an increase in computational efficiency over iterative methods for the original system because of the reduced cost of implementing vector inner products and reduced storage requirements. (The storage savings is very significant if one is using GMRES). Further computational efficiencies are obtained by adding additional pre-conditioning to the system so that the iteration on the reduced set of unknowns converges more rapidly.

Implementation of a capacitance matrix method involves two basic steps

(i) Reduction of the problem to that of solving a system of equations for a set of unknowns near the interface

(ii) Creation of efficient methods for solving the reduced system of equations.

Existing capacitance matrix methods differ in their implementation of these steps. For example, completely algebraic approaches [2] and approaches motivated by the analytical structure of the elliptic differential equation [14] have been used to formulate the reduced set of equations. (The analytic based approach [14] was developed to avoid the ill-posed linear systems that can arise when using a completely discrete approach.) Similarly, there are a variety of techniques proposed for efficiently solving the reduced system of equations; most recently [3][9][10] these have taken the form of finding good pre-conditioners to be used in conjunction with iterative methods for solving these systems.

The nature of the elliptic problems arising in the context of level set methods dictate that the implementation of a capacitance method should

- avoid the explicit parameterization of the interface;
- accommodate general discretizations of the equations near the interface, in particular those based on interpolation [15] and those that lead to non-symmetric systems of equations;
- posses efficient preconditioners for the reduced system that does not depend strongly upon the differential equation being solved.

With these considerations in mind, the capacitance matrix methods that we describe here are formulated from a completely algebraic approach, e.g. we do not use any explicit knowledge of the underlying interface or the differential equation that is being solved. (The ill-posedness of the linear systems associated with Dirichlet problems is avoided by expanding the sets of unknowns used in the formulation of the reduced equations.) The implementation of the preconditioner for the reduced system is also completely algebraic.

In the first section we give a general derivation of the equations used in our implementation of the capacitance matrix method. We then describe the construction of PCG and GMRES iterative procedures that are used in conjunction with it. While the principle application in this paper is to equations arising from discretizations of Poisson's and Helmholtz's equations, the ideas are

applicable to any set of linear equations and so we provide a rather general description from a linear algebra perspective. In the second section we apply the method to the equations arising from discretizations of Poisson's equation and the modified Helmholtz equation. Computational results will be presented that demonstrate the efficiency of the computational methods. Additionally, for the modified Helmholtz equation, we compare the results of the solution procedures considered here with an iterative method that uses incomplete LU factorization (another popular method for solving the discrete equations). Lastly, in an appendix, we give the details concerning the construction preconditioners for modified Helmholtz problems with periodic boundary conditions.

2 The Capacitance Matrix Equations

The equations we are concerned with can be described as the $n \times n$ system of equations

$$\mathrm{A}x = b \tag{1}$$

where a preconditioner $\tilde{\mathrm{A}}$ is available that differs from A in k columns. The reduction of the problem (1) to one involving a system of size k can be obtained directly from the Sherman-Morrison-Woodbury formula. However, the direct application of this formula is not particularly motivating, so we digress briefly to simultaneously derive and apply the formulas; hopefully providing some insight.

Let C be the set of indices of the columns of A that differ from $\tilde{\mathrm{A}}$, let S_C denote the subspace associated with the variables whose indices are in C, and let P^{t} be the $k \times n$ projection matrix into the subspace S_C and $\mathrm{P} = (\mathrm{P}^{\mathrm{t}})^{\mathrm{t}}$ the $n \times k$ matrix that extends vectors in S_C to all of $\mathcal{R}^n$. Since $(\mathrm{A} - \tilde{\mathrm{A}})x$ vanishes for any component outside of S_C, one can derive a system equivalent to (1);

$$\begin{aligned} \mathrm{A}\,x &= b \\ \equiv \left[\tilde{\mathrm{A}} + (\mathrm{A} - \tilde{\mathrm{A}})\right] x &= b \\ \equiv \left[\tilde{\mathrm{A}} + (\mathrm{A} - \tilde{\mathrm{A}})\mathrm{P}\mathrm{P}^{\mathrm{t}}\right] x &= b \end{aligned}$$

If both sides are multiplied by $\tilde{\mathrm{A}}^{-1}$, then this yields

$$\left[\mathrm{I} + \tilde{\mathrm{A}}^{-1}(\mathrm{A} - \tilde{\mathrm{A}})\mathrm{P}\mathrm{P}^{\mathrm{t}}\right] x = \tilde{\mathrm{A}}^{-1} b$$

or, rearranging

$$x = \tilde{\mathrm{A}}^{-1} b - \tilde{\mathrm{A}}^{-1}(\mathrm{A} - \tilde{\mathrm{A}})\mathrm{P}\mathrm{P}^{\mathrm{t}} x \tag{2}$$

Thus, in the case when $\tilde{\mathrm{A}}$ and A differ in a few columns, one finds that the solution is completely determined by $\mathrm{P}^{\mathrm{t}}x$, the unknowns associated with the subspace S_C.

Equation (2) suggests a solution strategy of computing $\mathrm{P}^{\mathrm{t}}x$ first and then using (2) to obtain the remainder of the solution. As will be discussed below, if $\tilde{\mathrm{A}}$ and A are both symmetric positive definite, then one can use a PCG method to obtain $\mathrm{P}^{\mathrm{t}}x$ directly. To accommodate other types of linear systems, one needs to devise equations for $\mathrm{P}^{\mathrm{t}}x$.

To this end, let $y = \mathrm{P}^{\mathrm{t}}x$, and consider the sequence of equivalent systems

$$\begin{aligned} \mathrm{A}\,x &= b \\ -\mathrm{P}^{\mathrm{t}}x + y &= 0 \end{aligned}$$

$$\begin{aligned} \equiv \left[\tilde{\mathrm{A}} + (\mathrm{A} - \tilde{\mathrm{A}})\mathrm{P}\mathrm{P}^{\mathrm{t}}\right] x &= b \\ -\mathrm{P}^{\mathrm{t}}x + y &= 0 \end{aligned}$$

$$\begin{aligned} \equiv \quad \tilde{\mathrm{A}}x + (\mathrm{A} - \tilde{\mathrm{A}})\mathrm{P}y &= b \\ -\mathrm{P}^{\mathrm{t}}x + y &= 0 \end{aligned}$$

In block matrix form, an equivalent set of equations is thus

$$\begin{pmatrix} \tilde{A} & (A-\tilde{A})P \\ -P^t & I \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix}$$

One can derive equations for $y = (P^t x)$ by using block Gaussian elimination (e.g. forming the Shur compliment) on this last set of equations;

$$\left[I + P^t \tilde{A}^{-1} (A - \tilde{A}) P \right] y = P^t \tilde{A}^{-1} b \tag{3}$$

A solution of (1) can therefore be obtained by first solving

$$\left[I + P^t \tilde{A}^{-1} (A - \tilde{A}) P \right] (P^t x) = P^t \tilde{A}^{-1} b \tag{4}$$

for $P^t x$ and then using

$$x = \tilde{A}^{-1} b - \tilde{A}^{-1} (A - \tilde{A}) P (P^t x) \tag{5}$$

to obtain the remainder of the solution.

Alternately, if B is an $n \times n$ non-singular matrix and U and V are $n \times k$ matrices then the Sherman-Morrison-Woodbury formula [5] for $(B - U V^t)^{-1} b$ is given by

$$(B - U V^t)^{-1} b = B^{-1} b - B^{-1} U y \tag{6}$$

where

$$(V^t B^{-1} U - I) y = V^t B^{-1} b$$

If one sets $B = \tilde{A}$, $U = (A - \tilde{A})P$ and $V^t = P^t$ then $A = (B + U V^t)$ and equations (4) and (5) are obtained. The key computational task (however you derive the formulas) is to compute $P^t x$.

It should be noted that when this derivation of reduced equations is applied to discretizations of Poisson problems on irregular regions with Dirichlet boundary conditions, the equations one obtains are similar to those obtained with a dipole layer ansatz as described in [14], rather than those obtained with a single layer ansatz. The reason for this stems from the fact that the set of equations where A and $\tilde{A}$ differ includes all boundary points *and* all neighbors of boundary points. (The finite difference operators may agree at the neighbor points, but when one constructs the equations, the contribution to the finite difference operator due to the boundary value is moved to the right hand side, thus yielding an equation that differs from that of the $\tilde{A}$).

One additional point concerning the use of (5) is that solutions obtained with approximate values in $P^t x$ may have large residuals. Specifically, if the values of $P^t x$ are computed with some error δ, then the residual corresponding to a solution obtained with (5) is

$$r = - A \tilde{A}^{-1} (A - \tilde{A}) P \delta \tag{7}$$

If one makes the reasonable assumption that $A\tilde{A}^{-1} \approx I$, then one is led to the conclusion that the residual is proportional to $(A - \tilde{A})P\delta$. In the case that A and $\tilde{A}$ represent discretizations of differential operators, then $(A - \tilde{A})P\delta$ can be large, especially for small mesh sizes. The large size of the residuals doesn't necessarily indicate an inaccurate solution, it is just an artifact of using (5) to obtain the complete solution. Fortunately, the residual is non-zero only at the points where

the preconditioner and the original operator differ, and we have found that the residuals can be reduced to acceptable levels by performing a few sweeps of relaxation on the approximate solution. Additionally, if one wants very small residuals we have found that iterative improvement [4] works very well.

2.1 A Preconditioned Conjugate Gradient Procedure for Computing $\mathrm{P}^{\mathrm{t}}x$

If $\tilde{\mathrm{A}}$ and A are both symmetric positive definite then the following PCG method can be used to compute $\mathrm{P}^{\mathrm{t}}x$ directly. While the resulting method is not, strictly speaking, a capacitance matrix method, it is of interest because it is a more efficient implementation of the method that consists of the standard PCG iteration on the original system.

This procedure is constructed by simply observing that if the initial residual is confined to the subspace S_C, then the standard PCG iteration applied to A with preconditioner $\tilde{\mathrm{A}}$ can be implemented using vectors associated with the subspace S_C, and those in a slightly larger subspace. Specifically, let C' be the collection of indices of unknowns with non-zero coefficients occurring in the rows where A and $\tilde{\mathrm{A}}$ differ. Let $S_{C'}$ be the subspace spanned by these unknowns and $\hat{\mathrm{P}}$ be the projection onto this subspace. If the standard PCG iteration converges to the solution x of (1), then the following iteration converges to $y = \mathrm{P}^{\mathrm{t}}x$;

$\tilde{b} = \mathrm{P}\ (b - \mathrm{A}\ \tilde{\mathrm{A}}^{-1}b)$
$k=0;\ y_0 = 0; r_0 = \tilde{b}$
while $(r_k \neq 0)$
  $z_0 = \hat{\mathrm{P}}^{\mathrm{t}}\mathrm{A}^{-1}\mathrm{P}\, r_k$
  $k = k + 1$
  if$(k = 1)$
    $p_1 = z_0$
  else
    $\beta_k = r_{k-1}^{\mathrm{t}} r_{k-1} \,/\, r_{k-2}^{\mathrm{t}} r_{k-2}$
    $p_k = \hat{\mathrm{P}}^{\mathrm{t}}\mathrm{P}\, r_{k-1} + \beta_k p_{k-1}$
  end
  $\alpha_k = r_{k-1}^{\mathrm{t}} r_{k-1} \,/\, (\mathrm{P}^{\mathrm{t}}\hat{\mathrm{P}} p_k)^{\mathrm{t}}\mathrm{P}^{\mathrm{t}}\mathrm{A}\hat{\mathrm{P}}\, p_k$
  $y_k = y_{k-1} + \alpha_k \mathrm{P}^{\mathrm{t}}\hat{\mathrm{P}}\ p_k$
  $r_k = r_{k-1} - \alpha_k \mathrm{P}^{\mathrm{t}}\mathrm{A}\hat{\mathrm{P}}\, p_k$
end

Here $r_k, y_k, \tilde{b} \in S_C$ and $p_k, z_k \in S_{C'}$. Since this iteration is just the standard PCG iteration with the accumulation of $y_k = \mathrm{P}^{\mathrm{t}}x_k$, the number of iterations to convergence will be the same as that of the standard iteration. The computational efficiency results through the use of smaller vectors to implement the iteration. The only additional cost of this alternate formulation is the need to create the complete solution using (5) after y_k has converged.

2.2 A GMRES Based Procedure for Computing $\mathrm{P}^{\mathrm{t}}x$

For non-symmetric systems, GMRES can be used to iteratively solve (4). Unlike the result for a PCG iteration, the GMRES iteration applied to (4) is not equivalent to a GMRES iteration applied to the complete system. The computational results will show that the procedure is effective and provides results that are comparable to PCG.

Given the use of iterative methods to solve (4) leads one to consider additional preconditioning that might be employed to reduce the number of iterations required. One approach to formulating additional preconditioning is to consider the task of constructing a matrix R so that when $\tilde{\mathrm{A}}$ is used as a preconditioner for

$$\mathrm{R\ A}\ x{=}\mathrm{R}\ b$$

the GMRES iteration on the reduced system for these equations

$$\left[\mathrm{I}+\ \mathrm{P}^{\mathrm{t}}\tilde{\mathrm{A}}^{-1}(\mathrm{RA}-\tilde{\mathrm{A}})\mathrm{P}\ \right]y=\mathrm{P}^{\mathrm{t}}\tilde{\mathrm{A}}^{-1}b \tag{8}$$

converges rapidly. One constraint in the selection of R is that its use should not greatly increase the size of the subspace used in the formulation of (8). In many applications this can be satisfied by restricting R to differ from the identity only in the rows where A and $\tilde{\mathrm{A}}$ differ. Even with this restriction, there are a great many choices that can be made.

One choice that we have found particularly effective is to pick R so that $||(\mathrm{RA}{-}\tilde{\mathrm{A}})\mathrm{P}||$ is small; then (8) is a small perturbation of the identity and GMRES converges rapidly. Intuitively, the idea is to add additional preconditioning so that the equations represented by RA and $\tilde{\mathrm{A}}$ are in closer agreement. To construct such an R, let $\hat{\mathrm{P}}$ be the projection onto the subspace spanned by the unknowns associated with the rows where A and $\tilde{\mathrm{A}}$ differ. We pick $\hat{\mathrm{R}}$ so that $\hat{\mathrm{R}}(\hat{\mathrm{P}}^{\mathrm{t}}\mathrm{A}){-}(\hat{\mathrm{P}}^{\mathrm{t}}\tilde{\mathrm{A}})$ is small, specifically $\hat{\mathrm{R}}$ is chosen so that the norm of the transpose of this expression,

$$\left\|(\hat{\mathrm{P}}^{\mathrm{t}}\mathrm{A})^{\mathrm{t}}\hat{\mathrm{R}}^{\mathrm{t}}{-}\ (\hat{\mathrm{P}}^{\mathrm{t}}\tilde{\mathrm{A}})^{\mathrm{t}}\right\|_F$$

is minimized. (Here $\|\cdot\|_F$ is the Frobenius norm). With this choice of norm, the minimizing $\hat{\mathrm{R}}$ is the pseudo-inverse of $(\hat{\mathrm{P}}^{\mathrm{t}}\mathrm{A})$ [4],

$$\hat{\mathrm{R}}{=}(\hat{\mathrm{P}}^{\mathrm{t}}\tilde{\mathrm{A}})(\hat{\mathrm{P}}\mathrm{A})^{\mathrm{t}}((\hat{\mathrm{P}}\mathrm{A})(\hat{\mathrm{P}}\mathrm{A})^{\mathrm{t}})^{-1} \tag{9}$$

R is then obtained by extended $\hat{\mathrm{R}}$ to the whole space, namely

$$\mathrm{R}\ =\begin{pmatrix}\hat{\mathrm{P}} & \\ & \mathrm{I}\text{-}\hat{\mathrm{P}}\end{pmatrix}\begin{pmatrix}\hat{\mathrm{R}} & \\ & \mathrm{I}\end{pmatrix}\begin{pmatrix}\hat{\mathrm{P}}^{\mathrm{t}} & \\ & \mathrm{I}\text{-}\hat{\mathrm{P}}^{\mathrm{t}}\end{pmatrix}$$

The application of this additional preconditioning requires applying the operator (9). We found it effective to apply this operator directly to vectors, and evaluate $((\hat{\mathrm{P}}\mathrm{A})(\hat{\mathrm{P}}\mathrm{A})^{\mathrm{t}})^{-1}$ by solving $(\hat{\mathrm{P}}\mathrm{A})(\hat{\mathrm{P}}\mathrm{A})^{\mathrm{t}}z\ =\ f$, using a preconditioned conjugate gradient iteration with diagonal preconditioning.

3 Computational Results

3.1 Numerical solution of Poisson's equation on an irregular region.

The first test problem consisted of finding approximate solutions to the problem

$$\Delta u = f \qquad u = 0 \text{ on } \partial\Omega$$

where the boundary of the irregular region Ω was defined as the zero contour of the level set function $\phi(x, y) = 1 - (x^2 + y^2)$. The computational domain was the rectangular region [-2, 2]×[-2, 2], and a uniform rectangular grid of size N panels by N panels was used for creating finite difference approximations. At grid points at least one mesh width away from the boundary, the Laplace operator was approximated with a standard second order five point difference approximation. At grid points less than or equal to one mesh width away from the boundary, the discretization described in [8] was used. This latter discretization consists of linearly extrapolating the values of u across the boundary and then using the extrapolated values in a standard second order difference approximation. This discretization was chosen for simplicity and the fact that it leads to a symmetric discretization of the Laplace operator on the irregular domain. The symmetric nature of the discretization allows one to use iterative methods based on PCG, however, symmetry of the discretization is not a requirement for iterative methods based on GMRES.

In each of the iterative methods tested, the preconditioner used was the standard discretization of the Laplacian on the rectangular domain with homogeneous Dirichlet boundary conditions. The solutions of the preconditioning system were obtained with the subroutine HWSCRT [1]. Four different iterative methods were considered; a preconditioned conjugate gradient procedure on the complete system (PCG1), a preconditioned conjugate gradient procedure on the reduced set of unknowns as described in section 2.1 (PCGR), a GMRES iterative method applied to (4) (GMRES1) and a GMRES iterative method applied to the preconditioned system (8) (GMRES2). For the GMRES based methods, we used a Krylov subspace dimension of 20. In the implementation of these iterative methods, matrix representations were not formed, instead, the application of the operators in (4) and (8) were applied to vectors directly. As mentioned previously, the application of the preconditioner for the GMRES2 method required applying the additional preconditioning operator (9)

$$(\hat{\mathrm{P}}^{\mathrm{t}}\tilde{\mathrm{A}})(\hat{\mathrm{P}}\mathrm{A})^{\mathrm{t}}((\hat{\mathrm{P}}\mathrm{A})(\hat{\mathrm{P}}\mathrm{A})^{\mathrm{t}})^{-1}$$

The application of $((\hat{\mathrm{P}}\mathrm{A})(\hat{\mathrm{P}}\mathrm{A})^{\mathrm{t}})^{-1}$ was implemented by solving $(\hat{\mathrm{P}}\mathrm{A})(\hat{\mathrm{P}}\mathrm{A})^{\mathrm{t}}z = f$, using a preconditioned conjugate gradient iteration with diagonal preconditioning. This iteration converged very rapidly, typically in eight to ten iterations.

We investigated the behavior of the different methods with two different stopping criterion. In the first set of experiments the iterative methods were terminated when the estimated relative norm of the residual was less than $\alpha * dx^2$, where dx is the mesh width of the computational domain and the constant α was taken to be 10^{-3}. This stopping criterion was selected so that the approximate solutions generated would have an error that was on the order of the errors associated with the discrete approximations to the Laplace operator. In the second set of experiments, we chose the tolerance independent of the mesh size and stopped the iteration when the estimated relative norm of the residual was less than the tolerance. This latter criterion is commonly used in investigations of the effectiveness of iterative methods.

In Tables 1-3 we present the results obtained with the different iterative methods for grids of size $N = 100$, 200 and 400 panels in each direction. The function f was taken to be

$$f(x,y) = \begin{cases} -16\,(x^2+y^2) & x^2+y^2 \le 1 \\ 0 & x^2+y^2 > 1 \end{cases}$$

In the first column of the tables, we present the errors of the approximate solution. This error is discrete L^2 norm of the difference between the computed solution and the exact solution of the differential equation given by

$$u(x,y) = \begin{cases} 1-(x^2+y^2)^4 & x^2+y^2 \le 1 \\ 0 & x^2+y^2 > 1 \end{cases}$$

By comparing the results for different values of N, we observe that the computed solution is converging with second order accuracy. This demonstrates the quality of the discrete approximation as well as our choice of stopping criterion.

In the second, third and forth columns we give the difference between the computed solution and the solution of the discrete equations, the residual estimate obtained from the iterative method, and the residual estimate obtained by direct calculation. An examination of these results shows that the relationship between the size of the residual and the size of the error depends upon the iterative method used and the size of the problem. This suggests that some caution is required when switching iterative methods or changing the mesh size; the constant of proportionality between the residual and the solution error is likely to change. (The disparity of the results is not indicative of any failure of the iterative methods; for it is still true that when a given method is applied to a problem of fixed size, a reduction of the residual tolerance leads to a corresponding reduction in the error in the approximate solution.)

In the last two columns we present the number of iterations and the total time that the computation required. In the case of the iterative methods on the reduced set of equations, this computation time included that required to set up the discrete operators. For N=100, the difference in running times between the methods is not significant. As the number of unknowns increases, the methods that are applied to the reduced system exhibit their superiority. In particular, GMRES2 shows about a factor of five improvement over the standard PCG scheme PCG1.

Method	$\|u - u_{exact}\|$	$\|u - \hat{u}\|$	$\|r_{iteration}\|$	$\|r_{computed}\|$	Iterations	CPU sec.
PCG1	6.589×10^{-4}	8.201×10^{-6}	1.266×10^{-6}	8.632×10^{-4}	14	.281
PCGR	6.595×10^{-4}	1.081×10^{-5}	1.266×10^{-6}	1.090×10^{-4}	14	.203
GMRES1	6.576×10^{-4}	4.014×10^{-5}	6.937×10^{-8}	3.791×10^{-4}	11	.203
GMRES2	6.578×10^{-4}	4.787×10^{-6}	2.759×10^{-7}	1.683×10^{-4}	5	.156

Table 1 Iterative Method Results for N = 100 (100 X 100 grid).

Method	$\|u - u_{exact}\|$	$\|u - \hat{u}\|$	$\|r_{iteration}\|$	$\|r_{computed}\|$	Iterations	CPU sec.
PCG1	1.603×10^{-4}	2.851×10^{-6}	3.937×10^{-7}	1.790×10^{-3}	23	2.125
PCGR	2.034×10^{-4}	1.646×10^{-4}	3.984×10^{-7}	3.940×10^{-3}	23	1.422
GMRES1	1.592×10^{-4}	2.916×10^{-5}	3.929×10^{-8}	1.530×10^{-3}	16	1.234
GMRES2	1.601×10^{-4}	4.792×10^{-6}	1.146×10^{-7}	3.875×10^{-4}	7	.781

Table 2 Iterative Method Results for N = 200 (200 X 200 grid).

Method	$\|u - u_{exact}\|$	$\|u - \hat{u}\|$	$\|r_{iteration}\|$	$\|r_{computed}\|$	Iterations	CPU sec.
PCG1	4.007×10^{-5}	8.009×10^{-7}	5.743×10^{-8}	8.157×10^{-3}	47	21.000
PCGR	4.073×10^{-5}	9.258×10^{-6}	5.748×10^{-8}	1.126×10^{-3}	47	14.391
GMRES1	4.555×10^{-5}	3.136×10^{-5}	9.476×10^{-9}	3.914×10^{-3}	36	12.265
GMRES2	4.039×10^{-5}	1.085×10^{-5}	7.372×10^{-8}	2.135×10^{-3}	9	4.172

Table 3 Iterative Method Results for N = 400 (400 X 400 grid).

Results obtained with the more commonly used fixed tolerance stopping criterion are presented in Tables 4, 5, and 6. One draws the same conclusions regarding relative computational efficiency from these results as were drawn from the data in Tables 1-3. More interestingly, one finds in these results the penalty of using a fixed tolerance; for several of the methods the error with respect to the exact solution *increases* as the the mesh width tends to zero.

Method	$\|u - u_{exact}\|$	$\|u - \hat{u}\|$	$\|r_{iteration}\|$	$\|r_{computed}\|$	Iterations	CPU sec.
PCG1	6.572×10^{-4}	1.915×10^{-5}	3.443×10^{-6}	2.704×10^{-3}	12	.250
PCGR	6.610×10^{-4}	5.482×10^{-5}	3.441×10^{-6}	5.285×10^{-4}	12	.172
GMRES1	6.817×10^{-4}	3.763×10^{-4}	6.270×10^{-7}	3.520×10^{-3}	8	.172
GMRES2	6.531×10^{-4}	2.961×10^{-5}	1.955×10^{-6}	6.433×10^{-4}	4	.156

Table 4 Iterative Method Results for 100 X 100 grid with a fixed stopping tolerance of 10^{-5}.

Method	$\|u - u_{exact}\|$	$\|u - \hat{u}\|$	$\|r_{iteration}\|$	$\|r_{computed}\|$	Iterations	CPU sec.
PCG1	1.767×10^{-4}	1.205×10^{-4}	9.920×10^{-6}	4.524×10^{-2}	14	1.281
PCGR	1.714×10^{-4}	7.838×10^{-5}	9.920×10^{-6}	7.830×10^{-3}	14	.875
GMRES1	6.706×10^{-4}	8.709×10^{-4}	6.652×10^{-7}	2.907×10^{-2}	10	.844
GMRES2	6.264×10^{-4}	7.973×10^{-4}	8.638×10^{-6}	5.348×10^{-2}	3	.484

Table 5 Iterative Method Results for a 200 X 200 grid with a fixed stopping tolerance of 10^{-5}.

Method	$\|u - u_{exact}\|$	$\|u - \hat{u}\|$	$\|r_{iteration}\|$	$\|r_{computed}\|$	Iterations	CPU sec.
PCG1	5.978×10^{-5}	6.199×10^{-5}	4.443×10^{-6}	8.592×10^{-2}	24	10.609
PCGR	8.588×10^{-3}	1.112×10^{-2}	4.646×10^{-6}	6.676×10^{-1}	24	7.297
GMRES1	1.397×10^{-3}	1.810×10^{-3}	9.476×10^{-7}	3.581×10^{-1}	10	4.125
GMRES2	1.166×10^{-3}	1.509×10^{-3}	6.833×10^{-6}	3.516×10^{-1}	3	2.188

Table 6 Iterative Method Results for a 400 X 400 grid with a fixed stopping tolerance of 10^{-5}.

3.2 Numerical solution of the modified Helmholtz equation on an irregular region.

In our third set of experiments we compared the results of the procedure GMRES2 with results obtained with a preconditioned conjugate gradients method with incomplete LU factorization (ILU) [4] as a preconditioner.

The test problem concerned the solution of

$$\Delta u + \epsilon u = f \qquad u = 0 \text{ on } \partial\Omega \tag{10}$$

with periodic boundary conditions imposed on u at the edge of a rectangular computational domain. Here $\epsilon \leq 0$ and we were especially interested in the case when ϵ was small, on the order of 10^{-3}. As with the first set of test problems, the boundary of the irregular region Ω was defined as the zero contour of the level set function $\phi(x,y) = 1 - (x^2 + y^2)$. The function f was taken to be the constant value 1.

The success of the procedures described in this paper depend upon the availablity of preconditioners that are identical to the original operator at the majority of the unknowns. The formulation of such a preconditioner for this test problem is a challenge. Difficulties arise because with periodic boundary conditions the modified Helmholtz operator $\Delta u + \epsilon u$ is a singular operator when $\epsilon = 0$. Moreover, for small values of ϵ, the solution contains a large constant component (of size $\frac{1}{\epsilon}$) that significantly reduces the effectiveness of this operator as a preconditioner. One preconditioner without these problems consists of a discretization of $\Delta u + \epsilon u$ in a rectangular domain with the function value fixed at one point in the domain (e.g. the Laplace operator is replaced by the identity at one point). This preconditioner has the property that it is non-singular when $\epsilon = 0$ and is stable for non-zero values of epsilon in the limit as $\epsilon \to 0$. In the appendix we give the details of implementing this preconditioner using a fast solvers.

Method	Grid size	$\|u - \hat{u}\|$	Iterations	CPU sec.
PCG-ILU	100×100	9.497×10^{-4}	35	.401
PCG-ILU	200×200	3.512×10^{-4}	72	4.623
PCG-ILU	400×400	2.562×10^{-4}	144	40.148
GMRES2	100×100	6.165×10^{-4}	5	.297
GMRES2	200×200	4.850×10^{-4}	6	1.166
GMRES2	400×400	1.620×10^{-4}	8	6.391

Table 7 Comparison with ILU $\epsilon = 0$

Method	Grid size	$\|u - \hat{u}\|$	Iterations	CPU sec.
PCG-ILU	100×100	9.489×10^{-4}	35	.395
PCG-ILU	200×200	3.509×10^{-4}	72	4.520
PCG-ILU	400×400	2.560×10^{-4}	144	40.367
GMRES2	100×100	6.165×10^{-4}	5	.438
GMRES2	200×200	4.851×10^{-4}	6	1.982
GMRES2	400×400	1.620×10^{-4}	8	11.014.

Table 8 Comparison with ILU $\epsilon = -.001$

Method	Grid size	$\|u - \hat{u}\|$	Iterations	CPU sec.
PCG-ILU	100×100	1.046×10^{-4}	38	.437
PCG-ILU	200×200	6.743×10^{-4}	75	4.816
PCG-ILU	400×400	6.574×10^{-4}	149	41.747
GMRES2	100×100	1.171×10^{-4}	6	.319
GMRES2	200×200	4.262×10^{-4}	8	1.331
GMRES2	400×400	2.991×10^{-4}	10	6.776

Table 8 Comparison with ILU $\epsilon = -1.0$

In Tables 6, 7 and 8 we give results obtained with PCG and ILU preconditioning and those obtained GMRES2 for values of $\epsilon = 0, -.001, -1.0$. The stopping criterion chosen was the more restrictive one; the residual tolerance was taken to be $\alpha * dx^2$ with $\alpha = 10^{-3}$. As with our first two numerical experiments, for values of $N = 100$, the execution time of the capacitance matrix based method was not significantly less than that obtained with the ILU method. In one case ($\epsilon = -.001$) the implementation of the modified Holmholtz preconditioner requires two calls to the fast solver (see the appendix), thus, for $N = 100$, the method is actually slower then the ILU method. However, for larger N, the convergence rates of the ILU based method deteriorates, and the capacitance matrix methods are significantly faster. Again, for the $\epsilon = -.001$ case, the nature of the preconditioner requires two fast solves per application, so the improvement in efficiency isn't as great as the improvement in the other two cases $\epsilon = 0$ and $\epsilon = -1.0$

3.3 Concluding Remarks

This paper is concerned with the solution of the equations arising from discretizations of elliptic differential equations on irregular domains. Of particular interest are those discretizations that use a uniform rectangular mesh with the irregular nature of the boundary being incorporated by modifying the difference equations near the boundary. If one has a preconditioner for such equations that agrees with the original operator at a majority of the grid points, then efficient methods that can be created for solving the discrete equations. Following the general capacitance matrix method techniques, one reduces the problem to a problem involving only the unknowns on and near the boundary. The formulas used to construct the reduced problems can be derived directly or viewed as an application of the Sherman-Morrison-Woodbury formula. The equations for the reduced problem are solved iteratively, either using preconditioned conjugate gradients or the generalized minimum residual algorithm. Due to the decrease in number of unknowns, some computational efficiency is gained through this problem reduction. As the computational results illustrate, more dramatic computational efficiencies can be obtained by applying additional preconditioning that accelerates the convergence of the iterative methods for the reduced problem equations.

The approach described here, and the additional preconditioning that is proposed, is completely "discrete" and does not depend upon any particular technique for constructing the equations near the boundary. The use of GMRES based iterative methods accommodates discretizations on the irregular region that are non-symmetric. The application of this procedure to other elliptic problems depends upon the availability of suitable preconditioners in the computational domain. For Laplace's equation and the modified Helmholtz equation in which the computational domain is a rectangular, these preconditioners are readily available; for other types of equations this requirement leads to a new set of problems that are worthy of investigation.

4 Acknowledgments.

The author would like to thank Dr. M. Kang at UCLA for the use of his implementation of preconditioned conjugate gradients with an incomplete LU factorization preconditioner.

References

[1] Adams J., Swarztrauber P., and Sweet R., *Fishpak, a package of fortran subprograms for the solution of separable elliptic partial differential equations*, Ver. 3.1, National Center for

Atmospheric Research, Boulder Colorado, 1980.

[2] Buzbee, B.L., Dorr, F.W., George, J.A. and Golub, G.H., *The direct solution of the discrete Poisson equation on irregular regions*, SIAM J. Numer. Anal. vol. 8, 1971, pp. 722-736.

[3] Ellman H.C., and O'Leary D.P., *Efficient Iterative Solution of the Three-Dimensional Helmholtz Equation*, J. of Comp. Physics, 142: 163-18, 1998.

[4] G.H Golub, C.F. VanLoan, *Matrix Computations* (second edition), Johns Hopkins University Press, Baltimore Maryland, 1989.

[5] A.S. Householder, *The Theory of Matrices in Numerical Analysis,* Dover Publications, New York, 1964.

[6] R.J. LeVeque and Z. Li. *The immersed interface method for elliptic equations with discontinuous coefficients and singular sources*, SIAM J. Num. Anal., 31:1019-1044, 1994.

[7] Li, Z.,*A fast iterative algorithm for elliptic interface problems.* SIAM J. Numer. Anal. 35:230–254, 1998

[8] S.J. Osher, R.P. Fedkiw, *Level set methods*, UCLA Dept. of Mathematics CAM report 00-08, 2000.

[9] Proskurowski, W. and Vassilevski, S., *Preconditiong capacitance matrix problems in domain imbedding*, SIAM. J. Sci. Stat. Comput., 15:77-88, 1994.

[10] Proskurowski, W. and Vassilevski, S., *Preconditioning capacitance non-symmetric and indefinite capacitance matrix problems in domain imbedding.* , SIAM. J. Sci. Stat. Comput., 16:414-430, 1995.

[11] Y. Saad. *GMRES: A generalized minimal residual algorithm for solving non-symmetric linear systems.* SIAM J. Sci. Stat. Comput., 7:856-869, 1986.

[12] J.A. Sethian, *Level set methods: evolving interfaces in geometry,fluid mechanics, computer vision and materials sciences*, Cambridge University Press, 1996.

[13] Swartztrauber, P.N. *The methods of cyclic reduction, Fourier analysis and the FACR algorithm for the discete solution of Poisson's equation on a rectangle.* SIAM Rev. 19:490-501, 1977.

[14] O.B. Widlund,and P. Proskurowski, *On the numerical solution of Helmholtz's equation by the capacitance matrix method.* Math. Comp. 30, 433-468, 1976

[15] D.M. Young, R.T. Gregory, *A Survey of Numerical Mathematics*, Vol II, Dover Publications, New York, 1973.

Appendix : Preconditioners for the modified Helmholtz equation with periodic boundary conditions.

The application of the proposed techniques to the solution of

$$\Delta u + \epsilon u = f \qquad u = 0 \text{ on } \partial\Omega \tag{11}$$

when Ω is embedded in a periodic domain requires the construction of a preconditioner that agrees with the operator in (11) at most of the grid points. The natural choice of a preconditioner consists of solutions to

$$\Delta u + \epsilon u = f$$

in a rectangular domain with periodic boundary conditions. When $|\epsilon|$ is of moderate size (e.g. $|\epsilon| > 10^{-2}$) this preconditioner is satisfactory, and can be evaluated with a single call to a fast solver e.g. HWSCRT[1]. For other values of ϵ this operator is not satisfactory. When $\epsilon = 0$ the differential operator is singular and for small values of ϵ, the solution contains a large constant component (of size $\frac{1}{\epsilon}$) that significantly reduces the effectiveness of the operator as a preconditioner. A preconditioner without these problems consists of a discretization of $\Delta u + \epsilon u$ in a rectangular domain with the function value fixed at one point in the domain (e.g. the Laplace operator is replaced by the identity at one point). The difficulty in using this preconditioner is the construction of the solutions to these equations. In this appendix we give the details of a procedure for constructing such solutions.

Let p be a point in the computational domain with indices (i_p, j_p). We are concerned with the solution of

$$\begin{array}{ll} \Delta^h u + \epsilon u = f & (i,j) \neq (i_p, j_p) \\ u_{(i_p,j_p)} = 0 & (i,j) = (i_p, j_p) \end{array} \tag{12}$$

where Δ^h is the standard five point discretization of the Laplacian. (Non-zero values of $u_{(i_p,j_p)}$ can be accommodated by modifying f). We consider two cases, when $\epsilon = 0$ and when $\epsilon \neq 0$.

When $\epsilon = 0$, let $\bar{f}$, be the average over the domain, and define $\hat{f}$ by

$$\hat{f}_{(i,j)} = \begin{cases} f_{(i,j)} & (i,j) \neq (i_p, j_p) \\ f_{(i,j)} - \bar{f} & (i,j) = (i_p, j_p) \end{cases}$$

The average of $\hat{f}$ then equals 0, and so the solution u^* defined by

$$u^* = (\Delta^h)^{-1}\ \hat{f}$$

exists. We force u^* to be unique by requiring the average of u^* vanish. Here $(\Delta^h)^{-1}$is the inverse of the five point Laplace equation in a rectangular domain with periodic boundary conditions. (u^* can be obtained with one call to a fast solver.) The solution, $u_{(i,j)}$, to (12) is obtained by subtracting off a constant so that u^* vanishes at (i_p, j_p), specifically

$$u_{(i,j)} = u^*_{(i,j)} - u^*_{(i_p,j_p)}$$

The case when $\epsilon \neq 0$ requires a bit more work. As before, define $\bar{f}$ as the average of f over the computational domain. Define $\hat{f}_1$ and $\hat{f}_2$ by

$$\hat{f}_{1\,(i,j)} = \begin{cases} f_{(i,j)} & (i,j) \neq (i_p, j_p) \\ f_{(i,j)} - \bar{f} & (i,j) = (i_p, j_p) \end{cases}$$

$$\hat{f}_{2\,(i,j)} = \begin{cases} f_{(i,j)} & (i,j) \neq (i_p, j_p) \\ f_{(i,j)} - \epsilon\,\bar{f} & (i,j) = (i_p, j_p) \end{cases}$$

One then constructs u_1^* and u_2^* using two calls to a fast solver to evaluate

$$u_1^* = (\Delta^h + \epsilon\, u)^{-1}\, \hat{f}_1$$

$$u_2^* = (\Delta^h + \epsilon\, u)^{-1}\, \hat{f}_2$$

The solution, $u_{(i,j)}$, to (12) is obtained by forming a linear combination of u_1^* and u_2^* so that the boundary condition at (i_p, j_p) is satisfied. Specifically, let s be defined by

$$s\, u^*_{1(i_p,j_p)} + (1-s)\, u^*_{2(i_p,j_p)} = 0$$

then

$$u = s\, u_1^* + (1-s)\, u_2^*$$

The use of u_1^* and u_2^* to obtain u is motivated by the need to construct a solution using the inverse of $\Delta^h + \epsilon\, u$ applied to right hand sides that have either vanishing, or very small average values.