THE USE OF DIRECT METHODS FOR THE SOLUTION OF THE DISCRETE POISSON EQUATION ON NON-RECTANGULAR REGIONS

BY

J. ALAN GEORGE

STAN-CS-70-159
JUNE 1970

COMPUTER SCIENCE DEPARTMENT
School of Humanities and Sciences
STANFORD UNIVERSITY

THE USE OF DIRECT METHODS FOR THE SOLUTION OF THE DISCRETE POISSON EQUATION ON NON-RECTANGULAR REGIONS*

by

J. Alan George

* The title of this report was originally announced as:
"An Imbedding Approach to the Solution of Poisson's Equation on an Arbitrary Bounded Region".

The preparation of this report was sponsored by the Office of Naval Research under grant number N0013-67-A-0112-0029, the National Science Foundation under grant number NSF GJ 408 and the Atomic Energy Commission under grant number AT (04-3) 326,PA 30.

THE USE OF DIRECT METHODS FOR THE SOLUTION OF THE DISCRETE POISSON EQUATION ON NON-RFXTANGULAR REGIONS*

J. Alan George

1. Introduction

In recent years several special direct methods have been developed for solving the discrete Poisson equation on rectangular domains. These methods take advantage of the regular block structure of the coefficient matrix, and some of them require an amount of computation which is close to being directly proportional to the number of grid points (equations) in the discretized problem. Dorr [4] presents an excellent survey of these methods. A considerable number of these algorithms suffer from numerical instability and are not suitable for large problems. An analysis of stability of several methods appears in [10].

In this paper we describe ways in which these direct methods can be used to solve non-rectangular Poisson problems. We will not concern ourselves with which of the direct methods is to be utilized; we merely observe that a number of satisfactory ones are available. Notable among them are Buneman's version of the method of odd/even reduction [1,2], and methods based on Fourier analysis [7,8].

* This work was supported in part by the Office of Naval Research under grant N0013-67-A-00112-0029, the Atomic Energy Commission under grant AT(04-3)326, PA30, and an IBM Graduate Fellowship.

The basic procedure is as follows. The domain R of the given problem is enclosed in a rectangle over which a uniform mesh is placed. The usual five-point Poisson difference operator is applied over the entire rectangle, yielding a block tri-diagonal system of equations. The given problem, however, determines only those elements of the right-hand side which lie in R; the remaining elements can be treated as parameters. Furthermore, the "solution" of the enclosing rectangular "problem" which we have generated will have certain constraints imposed upon it by the presence within the rectangle of the boundary S of the given (or imbedded) problem. Dirichlet boundary conditions will require the solution on the rectangle to have specified values at grid points which lie on S; other types of boundary conditions will require specific relations to hold between values at grid points lying on and/or adjacent to S.

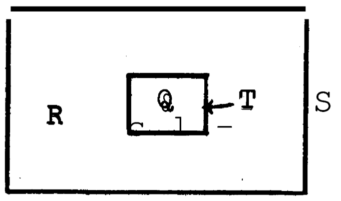
We now summarize our situation. We have a fast, efficient method for solving a specific system of equations, and we cannot delete or modify equations of the system because the method depends upon the structure of the coefficient matrix. We generate a system of equations which has this appropriate form, but for which some of the right-hand sides are unspecified, and where the solution must satisfy certain constraints. This paper describes methods for solving this problem.

2. Notation and a Representative Problem

For definiteness, we consider the following problem:

(2.1) $Au = f$ in R

$u = g$ on $S \cup T$



We superimpose a uniform grid on the rectangle S, and for simplicity we assume that T lies on grid points and on lines adjoining adjacent grid points. Approximating the differential operator with the usual five-point difference operator, and writing out an equation for <u>every</u> grid point in S, we obtain an $N \times N$ system of equations

(2.2) $$Av = h,$$

where the vectors v and h are defined on the grid, and N is the total number of grid points in the rectangle. For expository purposes only, we write (2.2) in the following partitioned form:

(2.3) $$\begin{pmatrix} A_{11} & A_{12} & \\ A_{21} & A_{22} & A_{23} \\ & A_{32} & A_{33} \end{pmatrix} \begin{pmatrix} v_R \\ v_T \\ v_Q \end{pmatrix} = \begin{pmatrix} h_R \\ h_T \\ h_Q \end{pmatrix} ,$$

where the vector partitions with subscripts Q, R, and T contain elements corresponding to grid points lying in Q, R, and on T, respectively. We will denote the number of elements in these partitions by N_Q, N_R, and N_T. We emphasize that this reordering cannot be done in practice because the special direct methods depend upon A having the regular block structure which occurs only if the grid points are numbered row by row or column by column.

It should be clear that if h_T and h_Q are assigned values so that the solution to (2.3) satisfies the boundary conditions (i.e., if v_T has the correct values), then v_R will be the correct discrete solution to our given problem. In Section 3 a method is presented for finding the values to be assigned to h_T and h_Q so that the above is achieved.

3. Direct Solution: Method 1

This method has been described by Hockney [7], and is closely connected to the discrete Green's function [6]. Formally, we can invert the partitioned matrix in (2.3) to obtain

(3.1)
$$\begin{pmatrix} v_R \\ v_T \\ v_Q \end{pmatrix} = \begin{pmatrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \\ B_{31} & B_{32} & B_{33} \end{pmatrix} \begin{pmatrix} h_R \\ h_T \\ h_Q \end{pmatrix} ,$$

and solving for $B_{22}\, h_T$, we have

(3.2)
$$B_{22}\, h_T = v_T - B_{21}\, h_R - B_{23}\, h_Q .$$

Since B_{22} is non-singular (it is positive definite), we have set h_Q to zero. We have an efficient method for solving (3.1) (in a reordered form) so we can easily obtain $B_{21}\, h_R$ as z_T from the solution of the system

(3.3)
$$\begin{pmatrix} A_{11} & A_{12} & \\ A_{21} & A_{22} & A_{23} \\ & A_{32} & A_{33} \end{pmatrix} \begin{pmatrix} z_R \\ z_T \\ z_Q \end{pmatrix} = \begin{pmatrix} h_R \\ 0 \\ 0 \end{pmatrix} .$$

The vector h_T is then obtained by solving

(3.4)
$$B_{22}\, h_T = v_T -- z_T .$$

Thus, we need B_{22}, which means that we need the N_T corresponding columns of the inverse of the coefficient matrix A. This method, therefore, requires solving $N_T + 2$ systems of the form (2.2), and the solution of the N_T linear equations (3.4). If we assume that the number of arithmetic operations required to solve (2.2) is kN,- then[1/] the total number of operations is about $kN_T(N + \frac{1}{3} N_T^2)$. Since N_T will typically be $O(\sqrt{N})$, the amount of work will be roughly proportional to N_T. If we suppose S is a square with $n^2 = N = 10^4$ grid points in it, and that N_T is 2n (it could easily be this large for typical regions), then the number of arithmetic operations required is $O(N^3)$. This is not likely to compare favourably with solving for v_R using SOR, especially when we consider how little programming overhead there is for the SOR process. Note, however, that the matrix B_{22} depends only on the-geometry of the problem. Thus, if we wish to solve a time-dependent problem, one with a non-linear right-hand side, or many problems with the same geometry, then this procedure may very well be the best one to use. It will almost certainly be the best if N_T and N_Q are small relative to N_R.

[1/] The factor k is actually a very slowly increasing function of N, of the form $\ell \log_2 \sqrt{N}$, ℓ a constant.

4. Direct Solution: Method II

An alternate approach which is more general than that of the method of section 3 is the following: We replace equations $A_{21}v_R + A_{22}v_T + A_{23}v_Q = h_T$ in (2.3) with the equations $0v_R + Iv_T + 0v_Q = v_T$ by adding a suitable correction to A. Obviously, theresulting solution v will have the correct v_R, regardless of the value of h_Q. Defining F and G by

$$F = \begin{pmatrix} 0 \\ I \\ 0 \end{pmatrix}, \quad G = \begin{pmatrix} -A_{21}^T \\ -A_{22}^T + I \\ -A_{23}^T \end{pmatrix}, \tag{4.1}$$

and denoting the coefficient matrix of (2.3) by A, we can write the equation

$$\begin{pmatrix} A_{11} & A_{12} & 0 \\ 0 & I & 0 \\ 0 & A_{32} & A_{33} \end{pmatrix} \begin{pmatrix} v_R \\ v_T \\ v_Q \end{pmatrix} = \begin{pmatrix} h_R \\ v_T \\ 0 \end{pmatrix} \tag{4.2}$$

as

$$(A + FG^T)v = h. \tag{4.3}$$

It can be shown [9] that

$$(A + FG^T)^{-1} = A^{-1} - A^{-1}F(I + G^TA^{-1}F)^{-1}G^TA^{-1}. \tag{4.4}$$

Thus, the procedure is

a) solve $AW = G$;

b) solve $Ay_1 = h$;

c) compute $y_2 = G^T y_1$ and $Y = I + W^T F$;

d) solve $Yy_3 = y_2$;

e) solve $Ay_4 = Fy_3$,

f) compute $v = y_1 - y_4$.

Note that this method is very flexible. It allows us to replace any equation by another at the expense of one solution of (2.2).

The amount of computation and storage required is virtually the same as for method I. However, since method II is somewhat more complicated, method I seems preferable unless the increased generality provided by method II is necessary.

5. Iterative Solution Based on Method I

We now turn to potentially more efficient ways to utilize direct methods to solve non-rectangular Poisson problems. Our basic problem is to find a solution to (3.4), and the major expense in the algorithm results from the generation of B_{22}. Hence, we would like to arrive at an iterative scheme which generates an (approximate) solution $h_T^{(k)}$ without actually requiring B_{22}. First note that for an arbitrary vector $h_T^{(k)}$, (3.1) implies

$$v_T^{(k)} = z_T + B_{22}\, h_T^{(k)} \tag{5.1}$$

or

$$B_{22}\, h_T^{(k)} = v_T^{(k)} - z_T = w + r^{(k)}, \tag{5.2}$$

where

$$w = v_T - z_T \tag{5.3}$$

and

$$r^{(k)} = v_T^{(k)} - v_T. \tag{5.4}$$

Here $r^{(k)}$ is the residual of the linear system (3.4) and is the difference between the solution on T generated by $h_T^{(k)}$ and the required values v_T. Our problem is equivalent to minimizing the quadratic function

$$\frac{1}{2}\, h_T^T B_{22}\, h_T - h_T^T w, \tag{5.5}$$

where we do not know B_{22} but can compute the gradient of φ. We are obviously free to use any of the many iterative methods for solving a system of linear equations or minimizing a quadratic function that is bounded from below.

However, because the residual (gradient) calculation is expensive it is natural to use a relatively powerful function minimizer or linear equation solver. For example, we could use the conjugate gradient method, or one of the several variable metric algorithms which have been developed [3,5]. In section 8 we compare SOR to two of these iterative forms of method I, making use of the conjugate gradient method in one and the Davidon-Fletcher-Powell algorithm [5] in the other. We shall refer to this class of methods as _iterative imbedding_ algorithms.

6. Iterative Solution Based on Method II

Using equation (4.3) as a basis, we consider the following iterative scheme[1/]:

$$(6.1)\quad \begin{pmatrix} A_{11}+\alpha I & A_{12} & A_{13} \\ A_{21} & A_{22}+\alpha I & A_{23} \\ A_{31} & A_{32} & A_{33}+\alpha I \end{pmatrix} \begin{pmatrix} v_R^{(k+1)} \\ v_T^{(k+1)} \\ v_Q^{(k+1)} \end{pmatrix} = \begin{pmatrix} h_R \\ \beta v_T \\ 0 \end{pmatrix}$$

$$+ \begin{pmatrix} \alpha I & 0 & 0 \\ A_{21} & A_{22}+(\alpha-\beta)I & A_{23} \\ 0 & 0 & \alpha I \end{pmatrix} \begin{pmatrix} v_R^{(k)} \\ v_T^{(k)} \\ v_Q^{(k)} \end{pmatrix}$$

, α and β real positive constants.

Denoting the coefficient matrix by A_α, and expressing the matrix on the right-hand side of (6.1) by $(FG^T + \alpha I)$, where

$$(6.2)\qquad F = \begin{pmatrix} 0 \\ I \\ 0 \end{pmatrix} \quad \text{and} \quad G = \begin{pmatrix} A_{21}^T \\ A_{22}^T - \beta I \\ A_{23}^T \end{pmatrix}$$

we obtain the error equation immediately as

$$\begin{pmatrix} e_R^{(k+1)} \\ e_T^{(k+1)} \\ e_Q^{(k+1)} \end{pmatrix} = A_\alpha^{-1}(FG^T + \alpha I) \begin{pmatrix} e_R^{(k)} \\ e_T^{(k)} \\ e_Q^{(k)} \end{pmatrix}$$

$$= A_\alpha^{-1}\left\{(FG^T + \alpha I)A_\alpha^{-1}\right\}^k (FG^T + \alpha I)e^{(0)}.$$

[1/] We assume that the fast direct methods applicable to solving $Av = h$ can also be used to solve $(A + \alpha I)v = h$, $\alpha > 0$.

Now the matrix in the braces is

$$\begin{pmatrix} \alpha I & 0 & 0 \\ A_{21} & A_{22}+(\alpha-\beta)I & A_{23} \\ 0 & 0 & \alpha I \end{pmatrix} \underbrace{\begin{pmatrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \\ B_{31} & B_{32} & B_{33} \end{pmatrix}}_{A_\alpha^{-1}}$$

$$= \begin{pmatrix} \alpha B_{11} & \alpha B_{12} & \alpha B_{13} \\ -\beta B_{21} & (I-\beta B_{22}) & -\beta B_{23} \\ \alpha B_{31} & \alpha B_{32} & \alpha B_{33} \end{pmatrix} = B_{\alpha,\beta} .$$

Hence, we have

$$e^{(k+1)} = A_\alpha^{-1} B_{\alpha,\beta}^k (FG^T + \alpha I)e^{(0)}.$$

The rate of convergence obviously depends critically on $\|B_{\alpha,\beta}\|$, and there appears to be no easy way to determine optimal α and β. For $\alpha = 0$, it is easy to show that β should be set to $2/(\lambda_{max}+\lambda_{min})$ where λ_{max} and λ_{min} are the largest and smallest eigenvalues of B_{22}. $B_{0,\beta}$ then has as its largest (in magnitude) eigenvalue $\mu = (\lambda_{max}-\lambda_{min})/(\lambda_{max}+\lambda_{min})$, and the iteration then converges. The problem is, of course, the difficulty in determining estimates of λ_{max} and λ_{min}. Numerical experiments and further analysis of this method are currently being pursued.

7. Error Bounds and Convergence Criteria

One of the most difficult problems in the application of an iterative process is the determination of a safe and meaningful convergence criterion. For a short and very good account of this problem with SOR see [8]. Briefly, the problem is as follows: Since we do not know the true (discrete) solution, the error at each stage of the iteration must be estimated on the basis of such measurable quantities as the size of the residuals or the size of the last correction vector. Unfortunately, small residuals or small changes in successive iterates do not guarantee correspondingly small errors in the computed solution. For rather ordinary problems the error can be several orders of magnitude larger.

The iterative imbedding algorithms seem particularly attractive with regard to the above problem, as the following theorem demonstrates.

Theorem 1. Let v be the true discrete solution on the enclosing rectangle, and let v^* be the computed solution, where v_T^* satisfies the (Dirichlet) boundary condition to within some value ϵ, i.e.,

$$\|v_T - v_T^*\|_\infty \le \epsilon. \tag{7.1}$$

Then

$$\|v_R - v_R^*\| \le \epsilon. \tag{7.2}$$

Proof: Let L be the discrete Laplacian operator. Then the following equations are satisfied:

$$Lv_R = h_R, \tag{7.3}$$

$$Lv_R^* = h_R. \tag{7.4}$$

Denoting the error in the computed solution by e, we have from (7.1), (7.3), and (7.4) that

(7-5) $\quad Le_R = 0$

and

(7.6) $\quad \|e_T\|_\infty \leq \epsilon.$

Since -L is an operator of "positive type," we can apply the well-known maximum principle to conclude from (7.5) and (7.6) that $\|e_R\|_\infty = \|v_R - v_R^*\| \leq \epsilon$.

Thus, we can determine when to stop the iteration simply by examining the largest element of $|e_T|$. Since it is difficult to imagine an iterative scheme which would not make use of e_T (it is the residual of (3.4)), the cost of determining $\|e_T\|_\infty$ should be negligible.

8. Numerical Experiments

We now present some numerical experiments for a problem of the type (2.1), where S is covered with a square mesh having< m rows, n columns, and mesh width h, and where Q is a $kh \times \ell h$ rectangle. The "southwest" corners of S and Q are at grid positions (0,0) and (j_1,j_2) respectively.

The implementation of the SOR algorithm provides for an initial approximate solution on a coarse grid (with mesh width 2h) which is then used to furnish a starting solution on the fine net by using linear interpolation. Thirty iterations were carried out on the coarse mesh to obtain the initial solution, and these iterations and the time required for them are not included in the tables below. An acceleration parameter ω of 1.8 was used on the coarse mesh for the first 25 iterations, followed by 5 iterations with $\omega = 1$ to estimate the optimal $\omega = \omega^*$ for the coarse mesh. The value $\omega^* + .55\ (2-M)$ was found to be near optimal for the fine mesh.

The number of iterations reported for the iterative imbedding methods requires some discussion. Obviously each iteration requires substantially more work than an SOR iteration. The ratio will depend on the size of the mesh since the computation required for the direct methods is not quite directly proportional to mn. Also, the relative sizes of N_R and $N_R + N_T + N_Q$ will affect the ratio because the SOR iterations will (at least ideally) only involve grid points in R. A factor of about 10 seems reasonable for typical problems having fewer than 20,000 points.

The time required to compute the right-hand sides of the equations is not included in the tables. All times are for execution of ALGOL W programs on an IBM 360/67.

Case I: $f = (2-100\, y^2) \cos (10x)$

$u = g = y^2 \cos (10x)$

$h = 0.0125,\ m = 49,\ n = 127$

$(j_1,j_2) = (64,32),\ k = \ell = 10$

Case II: Same as Case I except $(j_1,j_2) = (20,40)$ and $k = \ell = 20$.

	Method	Iterations	Maximum Error	Time (Seconds)
Case I	SOR	70	4.7×10^{-4}	42
	Imbedding I	5	2×10^{-4}	24.0
	Imbedding II	5	2×10^{-4}	23.7
	Direct	N.A.'	2×10^{-4}	9.6*
Case II	SOR	70	4.2×10^{-4}	41
	Imbedding I	6	2×10^{-4}	28.6
	Imbedding II	6	2×10^{-4}	28.5
	Direct	N.A.	2×10^{-4}	9.6*

Imbedding I - method of Section 5 using the Davidon-Fletcher-Powell algorithm [5].

Imbedding II - method of Section 5 using the conjugate gradient algorithm.

Direct - method of Section 3.

The maximum errors for the direct method and the imbedding methods are all the same because the error is due entirely to the truncation errar of the difference operator. The error in the discrete solution for these methods is below that level.

* Does not include the time required (approximately 3 minutes and 6 minutes, respectively, for Cases I and II) to generate and decompose B_{22} (see Section 3).

9. Remarks and Conclusions

The reported times at first do not appear particularly impressive, although the times required for the imbedding methods are substantially less than for the SOR process. It is important to keep in mind, however, that during the calculation using the methods of Section 5, we have precise information concerning how close our computed solution is to the true discrete solution. This is obviously highly important in a practical situation where the solution to our problem is not known. As we mentioned in Section 7, it is extremely difficult when applying SOR to ascertain how close the computed solution is to the true discrete solution. (For example, the maximum change for the last step of SOR in Case I above was 8.1×10^{-5}.)

As one might expect, the rate of convergence of the iterative imbedding algorithms depends on N_T. However, quite extensive experiments seem to suggest that the number of iterations does not increase very rapidly with N_T, and $\sqrt{N_T}$ iterations are usually sufficient.* Problems with singularities also do not appear to greatly affect the rate of convergence.

When N_Q and N_T are relatively large, and R can be subdivided into a number of rectangular blocks (R might be H- or L-shaped, for example), a direct method described in [2] may be more efficient than the method described in Section 3. It is not obvious if or when its iterative analog converges and, even if it does, no a posteriori bounds are available because the "parameters" are grid values lying in R rather than on a boundary.

We have not discussed the direct method used to solve our rectangular problems. As we mentioned earlier, many of the methods discussed by Dorr [4] suffer from numerical instability and are not suitable for large problems. We have used a method due to Buneman [1] which appears to be stable even for very large problems. For a qualitative discussion explaining this stability, see [2]. Hockney's algorithm POT I [7] could in theory reduce the times for the imbedding algorithms and the direct method by a factor of two, although in practice program overhead would reduce some of the advantage of the lower operation count.

Note that no use has been made of the particular geometry of the problem we have discussed other than it is enclosed by a rectangle. The methods we have described are applicable to arbitrary domains, and their efficiency will depend upon the subjective factors discussed at the end of Section 3.

10. Acknowledgments

I am pleased to acknowledge the generous assistance of Professor George E. Forsythe in the preparation of this report. Appreciation is also extended to Dr. Fred W. Dorr and Professor Gene H. Golub for their helpful suggestions and encouragement, and to Mr. M. A. Saunders for providing and modifying his SOR program to run the numerical experiments.

REFERENCES

[1] Buneman, O., "A Compact Non-Iterative Poisson Solver," SUIPR Report No. 294, Institute of Plasma Research, Stanford University, April 1969.

[2] Buzbee, B. L., G. H. Golub, and C. W. Nielson, "On Direct Methods for Solving Poisson's Equations," to appear in SIAM J. on Numer. Anal.

[3] Davidon, W. C., "Variance Algorithm for Minimization," Comput. J., vol. 10, 1968, pp. 406-10.

[4] Dorr, F. W., "The Direct Solution of the Discrete Poisson Equation on a Rectangle," SIAM Rev., Vol. 12, No.*-2, 1970, pp. 248-263.

[5] Fletcher, R., and M. J. D. Powell, "A Rapidly Convergent Descent Method for Minimization, Comput. J., Vol. 6, 1963, pp. 163-68.

[6] Forsythe, G. E., and W. R. Wasow, Finite Difference Methods for Partial Differential Equations< John Wiley and Sons, Inc., 1960, pp. 314-18.

[7] Hockney, R. W., "The Potential Calculation," Langley Working, Paper No. 673, Langley Station, Hampton, Va., September 1968.

[8] Hockney, R. W. in Methods of Computational Physics (B. Adler, S. Fernbach, and M. Rotenberg, eds.), Vol. 9, Academic Press, New York and London 1969.

[9] Householder, A. S., Principles of Numerical Analysis, McGraw-Hill, 1953, pp. 79-83.

[10] Rosser, J. B., "The Direct Solution of Difference Analogs of Poisson's Equation," MRC Tech. Report No. 797, Mathematics Research Center, University of Wisconsin, October 1967.