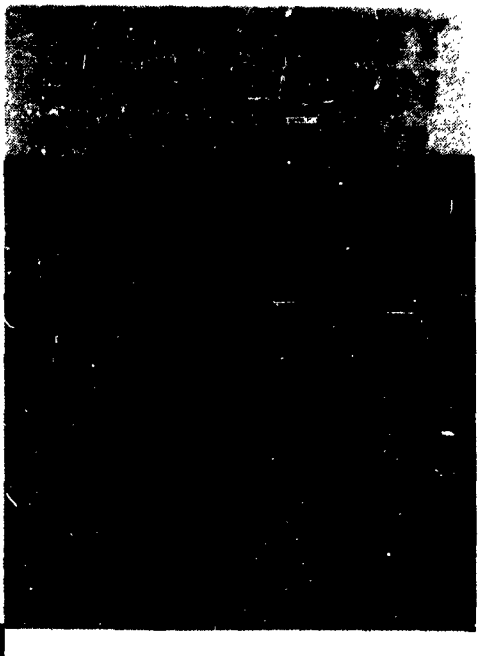
63-3-2



MATHEMATICS RESEARCH CENTER

MATHEMATICS RESEARCH CENTER, UNITED STATES ARMY
THE UNIVERSITY OF WISCONSIN

Contract No.: DA-11-022-ORD-2059

ON THE APPROXIMATE SOLUTION OF $\Delta u = F(u)$

D. Greenspan and M. Yohe

MRC Technical Summary Report #384
March 1963

Madison, Wisconsin

ABSTRACT

Three dimensional Dirichlet problems for $\Delta u = F(u)$, $F_u \geq 0$, are treated numerically by an exceptionally fast, exceptionally accurate numerical method. Programming details, numerous examples, and mathematical theory are supplied. Extension of the method in a natural way to n-dimensional problems is indicated by means of a 4-dimensional example.

ON THE APPROXIMATE SOLUTION OF $\Delta u = F(u)$

D. Greenspan and M. Yohe

1. Introduction. Because of its importance in such fields as potential theory, automorphic functions, and electron radiation [1]-[5], this paper will be concerned primarily with Dirichlet problems in three dimensions. It will be assumed that the problems need not be reducible to ones in two dimensions so that special assumptions, like axial symmetry, will be precluded.

Precisely, we will consider:

Problem D. Let G be a three dimensional closed, bounded, simply connected (contractible) domain in E^3 whose interior is R and whose boundary is S. For all real u, let $F(u)$ be defined and differentiable and let

$$F_u \geq 0 \quad . \tag{1.1}$$

Then if $\phi(x, y, z) \in C(S)$, find a function $u(x, y, z)$ which is a solution on R of

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} = F(u) \tag{1.2}$$

and which satisfies both

$$u \equiv \phi, \quad (x, y, z) \in S \tag{1.3}$$

$$u \in C(G) \quad . \tag{1.4}$$

Sponsored by the Mathematics Research Center, United States Army, Madison, Wisconsin under Contract No.: DA-11-022-ORD-2059.

Under quite general assumptions on S, such as possession of the cone property [1, p. 233], it is known that Problem D has a unique solution [1, p. 372], and it is only with such problems that we shall be concerned. But since it is not known, in general, how to give this solution in closed form, attention will be directed toward approximating it. A finite difference, digital computer technique which has proved exceptionally fast and exceptionally accurate will be described and both mathematical and experimental support for the method will be provided.

Extension of the method in a natural way to any number of dimensions will be indicated by means of a four-dimensional example (a two dimensional example already having been given elsewhere [6]).

2. The Numerical Method. Let $(\bar{x}, \bar{y}, \bar{z})$ be an arbitrary point of G and let h be a positive constant. The set of points $(\bar{x} + ph, \bar{y} + qh, \bar{z} + rh)$; $p = 0, \pm1, \pm2, \ldots$; $q = 0, \pm1, \pm2, \ldots$; $r = 0, \pm1, \pm2, \ldots$, is called a set of grid points. Two grid points are said to be adjacent if their distance apart is h . The set of all lines, each one of which contains at least one pair of adjacent grid points, is called a lattice. Denote by G_h those points which are either grid points in G or are points of intersection of S and the lattice. If $(x, y, z) \in [S \cap G_h]$, then (x, y, z) is called a boundary lattice point and the set of all boundary lattice points is denoted by S_h . The set of all points of G_h which are not elements of S_h is called the set of interior lattice points and is denoted by R_h .

If G_h consists of n points, we shall number these in a one-to-one fashion with the positive integers 1, 2, ..., n and if $(x, y, z) \in G_h$ and has been numbered t , then $u(x, y, z)$ will be denoted by u_t .

Method D. At each point $(x, y, z) \in S_h$, $u(x, y, z) = \phi(x, y, z)$ so that the exact solution is known. At each point $(x, y, z) \in R_h$, let $(x+h_1, y, z)$, $(x-h_2, y, z)$, $(x, y+h_3, z)$, $(x, y-h_4, z)$, $(x, y, z+h_5)$, $(x, y, z-h_6)$ be those points of G_h which are nearest to (x, y, z) in the positive and negative x, y and z directions, respectively (consult Diagram 2.1). Of course $0 < h_i \leq h$, $i = 1, 2, \ldots, 6$. If then (x, y, z), $(x + h_1, y, z)$, $(x - h_2, y, z)$, $(x, y+h_3, z)$, $(x, y - h_4, z)$, $(x, y, z+h_5)$, $(x, y, z-h_6)$ are numbered 0-6, respectively, then write down the difference equation

$$(2.1)\quad \left(-\frac{1}{h_1h_2}-\frac{1}{h_3h_4}-\frac{1}{h_5h_6}\right)u_0+\frac{1}{h_1(h_1+h_2)}u_1+\frac{1}{h_2(h_1+h_2)}u_2+\frac{1}{h_3(h_3+h_4)}u_3+\frac{1}{h_4(h_3+h_4)}u_4$$

$$+\frac{1}{h_5(h_5+h_6)}u_5+\frac{1}{h_6(h_5+h_6)}u_6=\frac{F(u_0)}{2}$$

If R_h consists of m points, application of (2.1) exactly once at each point of R_h yields a system of m algebraic equations in, say, $u_1, u_2, \ldots, u_m$, the solution of which, say, $U_1, U_2, \ldots, U_m$, constitutes the numerical solution. The final step then is to solve this algebraic system.

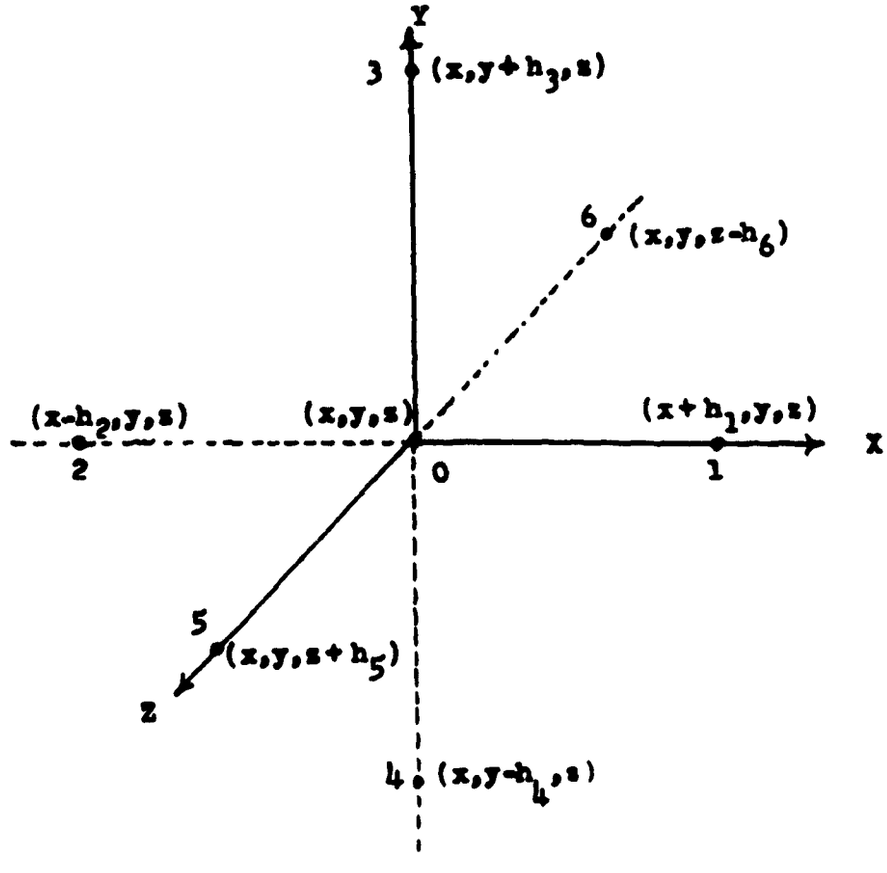


Diagram 2.1

3. Mathematical Basis of Method D.

The derivation of (2.1) is a straightforward generalization of that given for the two dimensional analogue [6].

In the important case when $F \equiv 0$, the algebraic system generated in Method D is linear and, by means of the elementary techniques of [6]-[9], it is easily shown that the algebraic solution vector exists and is unique, that linear over-relaxation converges, and that the numerical solution converges to the analytical solution in a suitable class of functions. The general case, when F need not be identically zero and the algebraic system need not be linear, can be supported in a fashion completely analogous to that of Bers [10]. But though existence and uniqueness of the solution of the algebraic system do follow and convergence of the numerical to the analytical solution for solutions of class $C^2(G)$ can be established, one can rely at all times only on the extended Liebmann method for solving the nonlinear algebraic system. As yet, no general theoretical basis has been developed to support nonlinear overrelaxation.

4. Programming Method D. In this section we give programming details for running Method D with $F \equiv 0$. Only minor, natural modifications are necessary to extend the program to the case $F \neq 0$ and to n-dimensional problems.

The program will require the following information:

(a) the mesh size h. (h may be different for the x, y, and z directions if desired; we will assume, however, that the same h is to be used for all directions).

(b) the base point, $(\bar{x}, \bar{y}, \bar{z})$, chosen so that, for all points $(x, y, z) \in G$, $\bar{x} \leq x$, $\bar{y} \leq y$, and $\bar{z} \leq z$.

(c) integers Lx, Ly, and Lz such that the point $(\bar{\bar{x}}, \bar{\bar{y}}, \bar{\bar{z}}) = (\bar{x} + (Lx-1)h, \bar{y} + (Ly-1)h, \bar{z} + (Lz-1)h)$ satisfies the condition $x \leq \bar{\bar{x}}$, $y \leq \bar{\bar{y}}$, $z \leq \bar{\bar{z}}$ for all points $(x,y,z) \in G$.

(d) the 6 functions B_i, defined for all points of the form $(\bar{x} + n_x h, \bar{y} + n_y h, \bar{z} + n_z h)$, $0 \leq n_x < L_x$, $0 \leq n_y < L_y$, $0 \leq n_z < L_z$. If a point (x,y,z) is in G, then $B_i(x,y,z)$ gives the distance from (x, y, z) to S along the ray passing through point i (see Diagram 2.1). If $(x, y, z) \notin G$, $B_i(x, y, z)$ must be negative for at least one i.

(e) the boundary value function $\phi(x, y, z)$

(f) the 7 formulas for the coefficients of $u_0, u_1, u_2, u_3, u_4, u_5, u_6$ in the difference equation (2.1).

(g) the over-relaxation factor, ω.

(h) the convergence criterion, ϵ.

The program will analyze this information, calculate the coefficients of the difference equations and set up an efficient procedure for the iterative solution of the system of difference equations. The program will then perform the iteration, and, finally, print the results.

Before discussing the particulars of the program, an example is in order. For the solution of Example 1, section 5, the following information was supplied:

(a) $h = 0.1$

(b) $(\bar{x}, \bar{y}, \bar{z}) = (0, 0, 0)$

(c) $L_x = L_y = L_z = 11$

(d) $B_1 = \text{sgn}(\theta_x)\sqrt{|\theta_x|} - x$, where $\theta_x = 1-(y^2+z^2)$; $\text{sgn}(\theta_x) = \begin{cases} +1 \text{ if } \theta_x \geq 0 \\ -1 \text{ if } \theta_x < 0 \end{cases}$

$B_2 = x$

$B_3 = \text{sgn}(\theta_y)\sqrt{|\theta_y|} - y$, where $\theta_y = 1-(x^2+z^2)$

$B_4 = y$

$B_5 = \text{sgn}(\theta_z)\sqrt{|\theta_z|} - z$, where $\theta_z = 1-(x^2+y^2)$

$B_6 = z$

(e) $\varphi(x, y, z) = x^2 + 2y - z^2$

(f) the 7 formulas as given in equation (2.1)

(g) $\omega = 1.8$

(h) $\epsilon = 10^{-9}$

Although the method in no way depends on the computer being used to solve the problem (as long as enough high speed storage is available) or on the programming language, we will assume for the sake of clarity that the programming will be done in FORTRAN II for the Control Data Corporation model 1604 computer.

The item of major concern to us is the manner in which arrays are stored in the computer's memory. If we are given a 3-dimensional array A of dimension $m \times n \times p$, we will give the FORTRAN program a DIMENSION statement as follows:

```
DIMENSION A(M, N, P)
```

where, of course, the actual integers involved must be specified. If we want to reference a specific element of A, say a_{ijk}, we use the following formula:

$$q = i + m \cdot (j - 1 + n \cdot (k-1)) \ ;$$

we reference a_{ijk} by referring to the storage location $A + q$. The number q, calculated as above, will be called the INDEX of the point a_{ijk} in the array A. (we note that q is uniquely determined by the subscripts i, j, and k).

Memory must be allocated for storage of the following data:

(a) two arrays of dimension $L_x \times L_y \times L_z$, one of which will be used to store the solution vector, and the other of which will be used to control the iteration procedure.

(b) the coordinate vectors V_x, V_y, and V_z

(c) seven arrays, one for each coefficient of the difference equation (2.1).

Coefficients will be stored in sets; that is, the set consisting of the i^{th} element from each of the seven coefficient vectors will comprise the coefficients of one difference equation. Duplicate sets will be stored only once, and several distinct equations may well use the same coefficient set. The number of sets depends on the problem; it is suggested that the program be given an upper limit and that a check be made to see that this limit is not exceeded. It is advisable to point out the number of sets actually computed by the program; this gives a basis for estimation for future problems as well as providing a check on the problem being solved.

The program starts with the values $\bar{x}$, $\bar{y}$, and $\bar{z}$, and generates the coordinate vectors. It then examines each point of the form $(\bar{x}+n_x h, \bar{y}+n_y h, \bar{z}+n_z h)$ to determine whether it is in the region R. (The functions B_i are used to make this determination). If the point is in R, the coefficients of the corresponding difference equation are computed, the solution vector entry is initialized, and the appropriate control information is computed and stored. If the point is not in the region R, the solution vector entry is set to 1.0, and the control entry is made negative.

After this analysis is complete, we have in storage the following information:

(a) The "solution vector" array U. The entries u_{ijk} are initialized to zero if a difference equation is to be solved for the corresponding point; otherwise u_{ijk} must contain 1.0.

(b) The control array KON . This array is the heart of the computational procedure. KON_{ijk} is negative if no difference equation is to be solved for the corresponding point. If a difference equation is to be solved for the corresponding point, then KON_{ijk} contains two items of information: the lower half of the memory cell contains an integer which specifies the index I of the NEXT point at which a difference equation is to be solved; the upper half of the memory cell contains the integer J which specifies the coefficient set to be used in solving the NEXT difference equation. If the point in question is the last point, the entry is zero. The information for the first point is stored in KON(0)

(c) The coefficient vectors; these vectors contain the coefficient sets to be used in computing the solution at the various points. This technique of keeping only the unique coefficient sets, suggested by D. Van Egeren, effects a considerable saving in high speed storage requirements, and can be applied whenever two integers can be stored in each element of the control array.

The iteration is now carried out as follows: KON(0) supplies the index I of the first part to be considered and the number J of the coefficient set to be used. We then compute U(I) as follows:

$$U(I) = U(I) + \omega\Big\{-U(I) + \Big[C1(J) * U(I+k_1) + C2(J) * U(I-k_1) + C3(J) * U(I+k_2) + C4(J) * U(I-k_2) + C5(J) * U(I+k_3) + C6(J) * U(I-k_3)\Big]/C0(J)\Big\} .$$

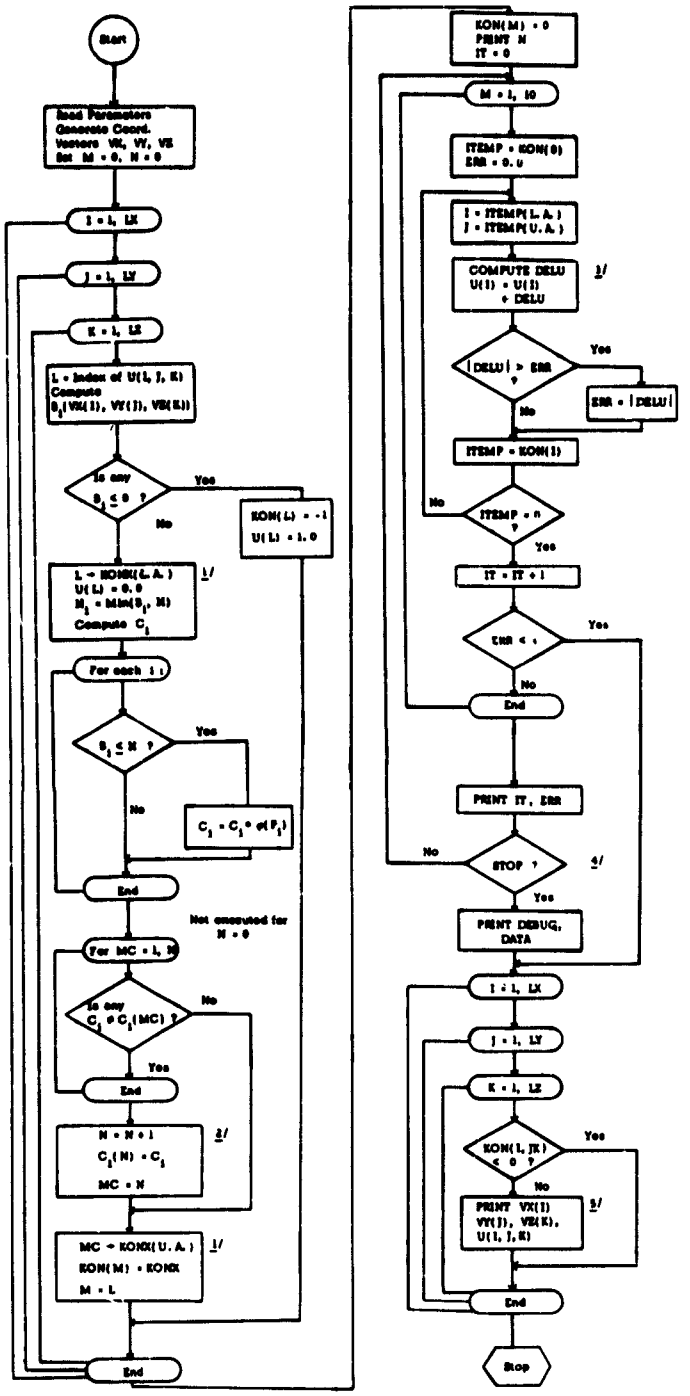
We record the "error" term, which is all of the right hand side of the equation except the U(I) term.

When this calculation is completed, we pick up KON(I) which gives us the next values of I and J . When KON(I) is zero, this iteration is complete, and we check for convergence and repeat the entire iteration procedure if necessary.

The entire programming procedure is shown in detail in the attached block diagram (Diagram 4.1). The diagram should be self-explanatory, but notes are appended in certain places to aid in tracing the program steps.

In order to abbreviate the diagram, we will use the following conventions:

Symbol "i" will indicate that the procedure is to be carried out in each of the six directions. That is, the statement "Compute B_i" means compute B_1, B_2, B_3, B_4, B_5, and B_6 for the given values of x, y, and z .



1/ (U.A.) = Upper Address
(L.A.) = Lower Address

2/ If a test is made for too many coefficient sets, it should be made here.

3/ The formula for DELU is:

DELU = OMEGAP(C1(J)*U(I+1) + C2(J)*U(I-1) + C3(J)*U(I+LX) + C4(J)*U(I-LX) + C5(J)*U(I+LX*LY) + C6(J)*U(I-LX*LY) - U(I)).

This will undoubtedly need to be re-written to fit the Fortran system.

4/ This is the place to test for divergence or for operator interrupt by means of a console switch.

5/ Print in a suitable format. It may be desired to print the number of points computed, too.

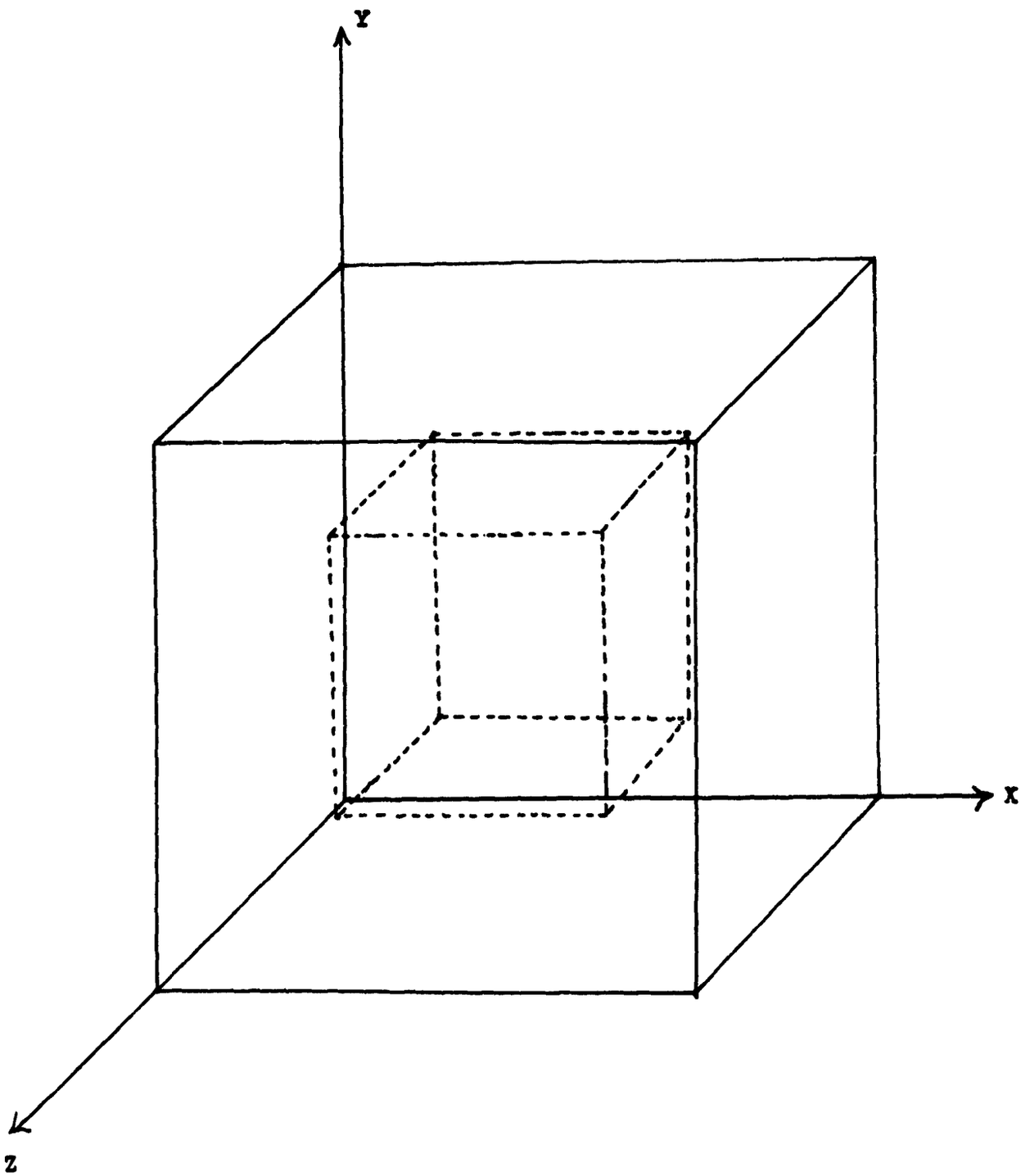
5. Examples. To support our contention that Method D is superior, from the point of view of both speed and accuracy, to any other method, whether discrete or continuous, typical numerical evidence will now be presented by means of several examples. Each example was run on the CDC 1604 at the University of Wisconsin.

Example 1. Let G be the spherical sector in the first octant bounded by the surfaces whose equations are $x = 0$, $y = 0$, $z = 0$, $x^2 + y^2 + z^2 = 1$. Let $F \equiv 0$, $(\bar{x}, \bar{y}, \bar{z}) = (0, 0, 0)$, $h = .1$ and $\phi = x^2 + 2y - z^2$. R_h contains 410 points. Using overrelaxation with a zero initial vector, Method D yielded results correct to at least nine decimal places to the exact solution $u = x^2 + 2y - z^2$. The running time was 1 minute 50 seconds.

Example 2. Example 1 was modified by setting $\phi = x^3y - xy^3 - 5z$. Selected, but typical, results are recorded under $u^{(1)}$ in Table 1. The running time was 1 minute 53 seconds and the exact solution was $u = x^3y - xy^3 - 5z$.

Example 3. Example 2 was modified by refining the grid to $h = .05$. R_h contained 3721 points. Selected, but typical, results are recorded under $u^{(2)}$ in Table 1. The running time was 8 minutes 11 seconds.

Example 4. Let G be the spherical sector in Example 1. Let $F \equiv e^u$ (see [5]), $(\bar{x}, \bar{y}, \bar{z}) = (0, 0, 0)$, $h = .1$ and $\phi = x + 2y + z^2$. R_h contains 410 points. Method D was applied with nonlinear overrelaxation [14] and with zero initial vector. The running time was 2 minutes 33 seconds. Selected, but typical, results are recorded under $u^{(1)}$ of Table 2.


Y
X
Z

Example 5. An extended Liebman method [10] was applied in Example 4 in place of nonlinear overrelaxation. Newton's method was applied to solve each equation. The running time was 4 minutes 20 seconds. The results agreed with those of Example 4 to at least eight decimal places. (With regard to this example, one should also consult [15].)

Example 6. Let G be the four dimensional spherical sector defined by $G = \{(x,y,z,\omega);\ x \geq 0,\ y \geq 0,\ z \geq 0,\ \omega \geq 0,\ x^2+y^2+z^2+\omega^2 \leq 1\}$. Let $F \equiv 0$, $(\bar{x}, \bar{y}, \bar{z}, \bar{\omega}) = (0, 0, 0, 0)$, $h = .1$ and $\phi = x^2 - 2y^2 + 3z^2 - 2\omega^2$. R_h contained 803 points. The difference analogue of the four dimensional Laplace equation, that is, the extension of (2.1) to four dimensions can be written for this problem in the form

$$(5.1) \qquad \sum_{i=1}^{4} \left[\frac{-u_0}{h_{2i-1}h_{2i}} + \frac{u_{2i-1}}{h_{2i-1}(h_{2i-1}+h_{2i})} + \frac{u_{2i}}{h_{2i}(h_{2i-1}+h_{2i})} \right] = 0 \ .$$

Using overrelaxation with initial vector the zero vector, Method D approximated the exact solution $u = x^2 - 2y^2 + 3z^2 - 2\omega^2$ at each point to at least nine decimal places. The running time was 3 minutes 58 seconds.

Example 7. In this example we will show how Method D can be applied even when G is not simply connected. Let S_1 be the "outer" cubic surface with vertices (0, 0, 0), (2, 0, 0), (0, 2, 0), (0, 0, 2), (2, 2, 0), (2, 0, 2), (0, 2, 2), (2, 2, 2) and let S_2 be the "inner" cubic surface with vertices $(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$, $(\frac{3}{2}, \frac{1}{2}, \frac{1}{2})$, $(\frac{1}{2}, \frac{3}{2}, \frac{1}{2})$, $(\frac{1}{2}, \frac{1}{2}, \frac{3}{2})$, $(\frac{3}{2}, \frac{3}{2}, \frac{1}{2})$, $(\frac{3}{2}, \frac{1}{2}, \frac{3}{2})$, $(\frac{3}{2}, \frac{3}{2}, \frac{1}{2})$, $(\frac{3}{2}, \frac{3}{2}, \frac{3}{2})$, as shown in Diagram 5.1.

Let R be the region between S_1 and S_2. Let $S = S_1 \cup S_2$ and $G = R \cup S$. Also, it is given that $\phi \equiv 1$ on S_1, $\phi \equiv 0$ on S_2, and $F \equiv 0$. Then, setting $(\bar{x}, \bar{y}, \bar{z}) = (0, 0, 0)$ and $h = .1$, Method D was applied to the resulting Dirichlet problem. R_h contained 784 points. The symmetry of the solution was incorporated into the program so that only one eighth of G had to be considered. Over-relaxation was applied with initial vector the zero vector and selected, but typical, results are recorded under $u^{(1)}$ of Table 3. The running time was 2 minutes 26 seconds.

Example 8. Example 7 was modified by refining the grid to $h = .05$. R_h contained 6669 points. Selected, but typical, results are recorded under $u^{(2)}$ in Table 3. The running time was 12 minutes 49 seconds.

TABLE 1

x	y	z	Approx. Sol.: $u^{(1)}$	Approx. Sol.: $u^{(2)}$	Exact Sol.: $u=x^3y-xy^3+5z$
.1	.1	.1	-0.50000000	-0.50000000	-.50000000
.1	.1	.7	-3.50000000	-3.50000000	-3.50000000
.1	.2	.2	-1.00059993	-1.00060000	-1.00060000
.1	.3	.1	-0.50239984	-0.50239998	-0.50240000
.1	.3	.6	-3.00239930	-3.00239992	-3.00240000
.1	.4	.2	-1.00599922	-1.00599990	-1.00600000
.1	.4	.8	-4.00599842	-4.00599974	-4.00600000
.1	.6	.2	-1.02099693	-1.02099962	-1.02100000
.1	.7	.1	-0.53359660	-0.53359961	-0.53360000
.1	.7	.7	-3.53361012	-3.53360023	-3.53360000
.1	.8	.4	-2.05039221	-2.05039855	-2.05040000
.2	.1	.1	-0.49940004	-0.49940000	-0.49940000
.2	.1	.6	-2.99940018	-2.99940002	-2.99940000
.2	.2	.1	-0.50000000	-0.50000000	-0.50000000
.2	.2	.6	-3.00000000	-3.00000000	-3.00000000
.2	.3	.1	-0.50299980	-0.50299997	-0.50300000
.2	.4	.1	-0.50959933	-0.50959992	-0.50960000
.2	.4	.6	-3.00959709	-3.00959968	-3.00960000
.2	.5	.5	-2.52099448	-2.52099937	-2.52100000
.2	.6	.2	-1.03839447	-1.03839930	-1.03840000
.2	.6	.6	-3.03838782	-3.03839868	-3.03840000
.2	.7	.6	-3.06297700	-3.06299759	-3.06300000
.3	.1	.1	-0.49760016	-0.49760002	-0.49760000
.3	.1	.6	-2.99760070	-2.99760008	-2.99760000
.3	.2	.1	-0.49700020	-0.49700002	-0.49700000
.3	.2	.6	-2.99700092	-2.99700010	-2.99700000
.3	.3	.1	-0.50000000	-0.50000000	-0.50000000

TABLE 1 (Continued)

x	y	z	Approx. Sol.: $u^{(1)}$	Approx. Sol.: $u^{(2)}$	Exact Sol.: $u=x^3y-xy^3+5z$
.3	.3	.9	-4.50000000	-4.50000000	-4.50000000
.3	.4	.8	-4.00839981	-4.00839951	-4.00840000
.3	.5	.6	-3.02399285	-3.02399923	-3.02400000
.3	.6	.4	-2.04858917	-2.04859867	-2.04860000
.3	.7	.3	-1.58398473	-1.58399789	-1.58400000
.3	.8	.2	-1.13198012	-1.13199693	-1.13200000
.3	.9	.2	-1.19437299	-1.19439290	-1.19440000
.4	.1	.4	-1.99400130	-1.99400016	-1.99400000
.4	.1	.8	-3.99400158	-3.99400026	-3.99400000
.4	.2	.8	-3.99040373	-3.99040039	-3.99040000
.4	.3	.5	-2.49160217	-2.49160024	-2.49160000
.4	.4	.1	-0.50000000	-0.50000000	-0.50000000
.4	.5	.2	-1.01799753	-1.01799967	-1.01800000
.4	.6	.4	-2.04798864	-2.04799869	-2.04800000
.4	.7	.3	-1.59238087	-1.59239767	-1.59240000
.4	.8	.4	-2.15360595	-2.15359356	-2.15360000
.5	.1	.7	-3.48800346	-3.48800042	-3.48800000
.5	.2	.5	-2.47900552	-2.47900062	-2.47900000
.5	.3	.3	-1.47600437	-1.47600055	-1.47600000
.5	.4	.3	-1.48200338	-1.48200042	-1.48200000
.5	.5	.1	-0.50000000	-0.50000000	-0.50000000
.5	.5	.7	-3.50000000	-3.50000000	-3.50000000
.5	.7	.2	-1.08398798	-1.08399811	-1.08400000
.6	.1	.2	-0.97900307	-0.97900037	-0.97900000
.6	.1	.6	-2.97900485	-2.97900064	-2.97900000
.6	.2	.4	-1.96160854	-1.96160104	-1.96160000
.6	.3	.2	-0.95140680	-0.95140090	-0.95140000

TABLE 1 (Continued)

x	y	z	Approx. Sol.: $u^{(1)}$	Approx. Sol.: $u^{(2)}$	Exact Sol.: $u=x^3y-xy^3+5z$
.6	.3	.6	-2.95141421	-2.95140156	-2.95140000
.6	.4	.5	-2.45201490	-2.45200137	-2.45200000
.6	.5	.5	-2.46700303	-2.46700050	-2.46700000
.6	.6	.4	-2.00000000	-2.00000000	-2.00000000
.7	.1	.4	-1.96640694	-1.96640094	-1.96640000
.7	.2	.3	-1.43701210	-1.43700156	-1.43700000
.7	.3	.2	-0.91601231	-0.91600171	-0.91600000
.7	.4	.1	-0.40760854	-0.40760124	-0.40760000
.7	.4	.5	-2.40764318	-2.40760312	-2.40760000
.7	.5	.5	-2.41599728	-2.41599952	-2.41600000
.7	.7	.1	-0.50000000	-0.50000000	-0.50000000
.8	.2	.1	-0.40401347	-0.40400140	-0.40400000
.8	.3	.4	-1.86803535	-1.86800377	-1.86800000
.8	.5	.1	-0.34404041	-0.34400209	-0.34400000
.9	.1	.4	-1.92799112	-1.92800167	-1.92800000
.9	.4	.1	-0.26598085	-0.26599834	-0.26600000

TABLE 2

x	y	z	$u^{(1)}$	x	y	z	$u^{(1)}$
.1	.1	.1	.33379803	.4	.1	.3	.77843093
.1	.2	.4	.73417575	.4	.2	.1	.88700006
.1	.3	.5	1.04443022	.4	.3	.3	1.28020294
.1	.4	.3	1.08918215	.4	.3	.7	1.61368477
.1	.5	.2	1.23012008	.4	.4	.4	1.58080909
.1	.6	.4	1.57000816	.4	.5	.7	1.95821289
.1	.7	.4	1.75731038	.4	.6	.3	1.89143826
.1	.8	.2	1.81786813	.4	.7	.5	2.12651462
.1	.9	.4	2.06963333	.4	.8	.2	2.13369786
.2	.1	.5	.72353422	.5	.1	.4	.95402334
.2	.2	.3	.80203491	.5	.2	.3	1.14019526
.2	.2	.9	1.44682719	.5	.3	.2	1.29857485
.2	.3	.9	1.63582348	.5	.4	.4	1.66722297
.2	.4	.8	1.71152454	.5	.5	.7	1.99755020
.2	.5	.8	1.87751902	.5	.7	.2	2.06195601
.2	.7	.1	1.70257965	.6	.1	.3	.97486870
.2	.8	.1	1.88993511	.6	.2	.6	1.45675948
.3	.1	.1	.55309066	.6	.4	.3	1.66927338
.3	.1	.6	.94374260	.6	.5	.6	1.98270586
.3	.2	.7	1.30608156	.6	.7	.3	2.13505733
.3	.3	.5	1.33513052	.7	.1	.7	1.39334550
.3	.4	.4	1.47053424	.7	.4	.2	1.59571747
.3	.5	.2	1.51616497	.7	.5	.3	1.89236582
.3	.6	.3	1.79105773	.8	.1	.2	1.09268284
.3	.6	.4	1.85851258	.8	.2	.5	1.48122715
.3	.7	.4	2.01526321	.8	.4	.3	1.75110486
.3	.9	.1	2.14891247	.9	.1	.3	1.21296598

TABLE 3

x	y	z	$u^{(1)}$	$u^{(2)}$
1.0	1.0	1.6	.21318413	.21340617
1.0	1.4	1.6	.25474801	.26000151
1.0	1.6	1.3	.23039266	.23203625
1.0	1.8	1.9	.93922824	.94029832
1.0	1.9	1.6	.87962760	.88193903
1.1	1.5	1.7	.52860578	.53955766
1.1	1.6	1.6	.46730212	.47796716
1.1	1.8	1.5	.70210223	.70815556
1.1	1.9	1.0	.81346190	.81395052
1.2	1.1	1.9	.81812623	.81902279
1.2	1.6	1.1	.22076057	.22147398
1.2	1.7	1.1	.43191770	.43329169
1.2	1.8	1.1	.63036172	.63180745
1.2	1.9	1.7	.91165183	.91351895
1.3	1.5	1.7	.54872314	.56194398
1.3	1.7	1.5	.54872314	.56194398
1.3	1.8	1.2	.64900339	.65215013
1.4	1.0	1.7	.47342317	.47933409
1.4	1.2	1.8	.67122668	.67620583
1.4	1.6	1.4	.29590727	.30704619
1.4	1.8	1.9	.94863874	.95008233
1.5	1.2	1.6	.31897168	.33873950
1.5	1.6	1.5	.41057277	.45137840
1.5	1.8	1.4	.73764103	.74655664
1.5	1.9	1.6	.90921585	.91255196
1.6	1.1	1.8	.75512719	.76003480
1.6	1.2	1.9	.88318714	.88577147

TABLE 3 (Continued)

x	y	z	$u^{(1)}$	$u^{(2)}$
1.6	1.5	1.6	.56670759	.58864922
1.6	1.7	1.6	.75991236	.76967969
1.6	1.9	1.3	.88834726	.89126535
1.7	1.0	1.0	.42086795	.42134851
1.7	1.2	1.0	.42989416	.43111151
1.7	1.4	1.3	.49616024	.50444129
1.7	1.6	1.5	.70081037	.71379555
1.7	1.8	1.5	.86041861	.86533931
1.8	1.1	1.3	.64295832	.64555233
1.8	1.2	1.7	.82021476	.82397741
1.8	1.4	1.4	.70371313	.71143949
1.8	1.6	1.7	.88560303	.88971671
1.8	1.9	1.3	.94416441	.94549284
1.9	1.0	1.4	.83606017	.83822357
1.9	1.3	1.1	.82513613	.82661512
1.9	1.3	1.7	.91578506	.91787661
1.9	1.5	1.7	.93179483	.93410454
1.9	1.7	1.0	.90873890	.91041024
1.9	1.9	1.0	.96974268	.97026699

REFERENCES

1. R. Courant and D. Hilbert, Methods of Mathematical Physics, vol. II, Interscience, New York, 1962, pp. 275-290, 305-306.

2. I. N. Sneddon, Elements of Partial Differential Equations, McGraw-Hill, New York, 1957, pp. 141-180.

3. O. D. Kellog, Foundationsof Potential Theory, Springer, Berlin, 1929.

4. W. J. Sternberg and T. I. Smith, The Theory of Potential and Spherical Harmonics, Univ. Toronto Press, Toronto, 1946.

5. P. Frank and R. V. Mises, Die Differential - und Integralgleichungen der Mechanik und Physik, vol. I, Second Edition, Rosenberg, New York, 1943, pp. 838-845.

6. D. Greenspan, "On the numerical solution of problems allowing mixed boundary conditions", Technical Summary Report 372, Mathematics Research Center, U.S. Army, Univ. Wisconsin, Madison, Wisconsin, 1962, pp. 28, 30-31.

7. D. Greenspan, "On the approximate solution of a class of strongly elliptic linear differential equations", Jour. Franklin Institute, 271, 1961, pp. 471-487.

8. D. M. Young, "Iterative methods for solving partial difference equations of elliptic type", Trans. Amer. Math. Soc., 76, 1954, pp. 92-111.

9. E. Batschelet, "Uber die numerische Auflosung von Randwertproblemen bei elliptishen partiellen Differentialgleichungen", ZAMP, 3, 1952, pp. 165-193.

10. L. Bers, "On mildly nonlinear partial difference equations of elliptic type", J. Res. Nat. Bur. Standards, 51, 1953, pp. 229-236.

11. D. Greenspan and R. M. Warten, "On the approximate solution of Dirichlet-type problems with singularities in the boundary", Jour. Franklin Inst., 273, 1962, pp. 187-200.

12. R. M. Warten, On the Approximate Solution of Axially Symmetric Problems by Means of Finite Differences, Ph. D. thesis, Purdue University, August, 1961.

13. D. Greenspan, "The approximate solution of axially symmetric problems", Tech. Rpt. 377, Math. Res. Ctr., U.S. Army, Univ. Wisconsin, 1963.

14. H. M. Lieberstein, "A numerical test case for the nonlinear overrelaxation algorithm", Tech. Report 122, Math Res. Center, U.S. Army, Univ. Wisconsin, 1960.

15. S. Schechter, "Iteration methods for nonlinear problems", Trans. A.M.S., 1962, pp. 179-189.