A Computer Programme
for the Solution of Elliptic Partial Differential Equations
(Potential Problems)

by

J.S. Hornsby
Propriété littéraire et scientifique réservée pour tous les pays du monde. Ce document ne peut être reproduit ou traduit en tout ou en partie sans l'autorisation écrite du Directeur général du CERN, titulaire du droit d'auteur. Dans les cas appropriés, et s'il s'agit d'utiliser le document à des fins non commerciales, cette autorisation sera volontiers accordée.

Le CERN ne revendique pas la propriété des inventions brevetables et dessins ou modèles susceptibles de dépôt qui pourraient être décrits dans le présent document; ceux-ci peuvent être librement utilisés par les instituts de recherche, les industriels et autres intéressés. Cependant, le CERN se réserve le droit de s'opposer à toute revendication qu'un usager pourrait faire de la propriété scientifique ou industrielle de toute invention et tout dessin ou modèle décrits dans le présent document.

© Copyright CERN, Genève, 1969

Literary and scientific copyrights reserved in all countries of the world. This report, or any part of it, may not be reprinted or translated without written permission of the copyright holder, the Director-General of CERN. However, permission will be freely granted for appropriate non-commercial use. If any patentable invention or registrable design is described in the report, CERN makes no claim to property rights in it but offers it for the free use of research institutions, manufacturers and others. CERN, however, may oppose any attempt by a user to claim any proprietary or patent rights in such inventions or designs as may be described in the present document.
A Computer Programme
for the Solution of Elliptic Partial Differential Equations
(Potential Problems)

by

J.S. Hornaby

GENEVA
11. **The Method of Classifying Mesh Points**
   Possible cases to be considered; necessity for deformation of the boundary; definition of number of mesh rows and columns; table of boundary values; of irregular points; of regular points; points close to a boundary treated as on the boundary.

12. **Restrictions**
   Restrictions on mesh size and size of tables.

13. **Results**
   Format and interpretation of output.

14. **References**

**APPENDICES**

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I - Data Fields</td>
<td>23</td>
</tr>
<tr>
<td>II - Preparation of Subroutines</td>
<td>25</td>
</tr>
<tr>
<td>III - Variables in COMMON</td>
<td>29</td>
</tr>
<tr>
<td>IV - Technical Details</td>
<td>31</td>
</tr>
<tr>
<td>V - Alternative Version of the Programme to Treat Larger Cases</td>
<td>33</td>
</tr>
<tr>
<td>VI - Case where Two Neumann Boundaries Meet</td>
<td>35</td>
</tr>
<tr>
<td>VII - Example</td>
<td>37</td>
</tr>
</tbody>
</table>

**Figures 1 - 6**
A Computer Programme for the Solution of Second-Order
Quasi-Linear Partial Differential Equations of Elliptic Type
using the Method of Successive Over-relaxation

J. S. Hornsby

1. Introduction

The formulation of problems in a number of fields of applied
mathematics (elasticity, electrostatics, subsonic aerodynamics, heat
conduction) leads to partial differential equations of elliptic type;
particular mention may be made of Laplace's equation, Poisson's equation
and the biharmonic equation. Where the physical region under study has
cylindrical symmetry or is very long, a two-dimensional approach is
appropriate.

In cases where the boundary of the two-dimensional region is
particularly simple, analytical methods of solution are available. In
particular, the method of integral transforms\(^1\) may be applied. In
other cases, however, no analytical solution can be found, and this has
led to the development of finite-difference methods\(^2\), \(^3\) which, with the
increasing use of high-speed computers with large storage capacity, have
become practicable for routine work.

The essence of the method is to superimpose upon the region of
interest a net, and to replace the partial differential equation to be
solved by an approximate difference equation at each nodal point of the
net. The problem of finding an efficient method of solving the resulting
large number of equations has received intensive study, and a number of
methods have been proposed\(^4\). To set up the equations, prior to solving
them, when the boundary of the region is curved, is no small task,
particularly when a fine net is used; and so the present programme is
designed to relieve the user of this burden. This requires a rather
involved analysis of the geometry of the boundary in the neighbourhood of
each node of the net, and indeed for fourth-order equations (e.g. the biharmonic) the analysis becomes so involved that we have preferred to exclude them, rather than abandon the idea of setting up the difference equations automatically. For precisely the same reason we do not treat boundary conditions involving a derivative unless the corresponding boundary is straight and coincident with a nodal line of the net.

The present programme, then, will solve equations of the form:

\[ \frac{\partial^2 u}{\partial x^2} + b \frac{\partial^2 u}{\partial y^2} + c \frac{\partial u}{\partial x} + d \frac{\partial u}{\partial y} + eu + f = 0 \]  \hspace{1cm} (1.1)

where \( a, b, 0 \) and \( a, b, c, d, e, f \) may be functions of \( x, y \) and \( u \). The solution will be found within a region bounded by simple closed curves \( C_1, C_2, \ldots, C_k \), where \( C_2, C_2, \ldots, C_k \) lie within the domain bounded by \( C_1 \). Along any segment of each \( C_j \) we must have either

(a) \( u \) known, or

(b) if the segment is straight, and a net can be chosen so that the segment is coincident with a nodal line, a condition of the form

\[ n \frac{\partial u}{\partial x} + m \frac{\partial u}{\partial y} + ru + s = 0 \]  \hspace{1cm} (1.2)

We shall refer to a segment of a \( C_j \) along which a condition of type (b) is valid as a 'Neumann' boundary.

2. The Finite-Difference Net

Fig. 1 shows a net superimposed upon a region (between \( C_1 \) and \( C_2 \)) within which the solution of an equation typified by (1.1) might be required. The size of the net is governed by the elements \( \Delta x \) and \( \Delta y \). The horizontal lines, defined by

\[ y_j = (j-1) \Delta y, \quad (j = 1, 2, \ldots, NR) \]  \hspace{1cm} (2.1)

we refer to as 'mesh rows'; and the vertical lines, defined by

\[ x_i = (i-1) \Delta x, \quad (i = 1, 2, \ldots, NC) \]  \hspace{1cm} (2.2)
we refer to as 'mesh columns'. The values of NR (Number of Rows) and NC (Number of Columns) depend upon the largest y and x co-ordinates respectively of the curve $C_1$, in a manner which will be defined precisely below. (Section II).

The points of intersection of the mesh rows and columns we refer to as 'nodal' or 'mesh' points: they are numbered as indicated in fig. 1, that is to say, from left to right and from row to row, row number increasing.

Curve $C_1$ must lie wholly in the positive x-y quadrant, although segments of it may be coincident with the axes.

3. The Finite-Difference Equations

Let $N$ be a mesh point between $C_1$ and $C_2$, and let the value of $u$ at $N$ be $U_N$. $N$ has four neighbours, one to the right, one above, one to the left and one below; by neighbour we mean either another mesh point, or a point on the boundary if the boundary cuts the mesh row or column between the current mesh point and the next one. The situation is shown in fig. 2 in which we have now adopted a localised numbering system: point $N$ becomes point 0 and its four neighbours are numbered 1, 2, 3 and 4 respectively. Let $h_j$ be the distance from point 0 to point $j$ ($j = 1, 2, 3, 4$) as a proportion of $\Delta x$ or $\Delta y$ as the case may be (i.e. $0 < h_j < 1$), and let $U_j$ be the value of $U$ at $j$.

Using the Taylor series expansion of $U$ at the appropriate points, it is easy to show that

$$\begin{align*}
(U_{xx})_0 &= \frac{2}{\Delta x^2 h_1 h_3 (h_1 + h_3)} \left\{ h_3 U_1 + h_1 U_3 - (h_1 + h_3) U_0 \right\} + \epsilon_{xx} \quad (3.1) \\
(U_{yy})_0 &= \frac{2}{\Delta y^2 h_2 h_4 (h_2 + h_4)} \left\{ h_4 U_2 + h_2 U_4 - (h_2 + h_4) U_0 \right\} + \epsilon_{yy} \quad (3.2)
\end{align*}$$
\[
(U_x)_o = \frac{1}{h_1 h_2 (h_1 + h_2)} \Delta x \left\{ h_3^2 U_1 - h_1^2 U_3 - (h_3^2 - h_1^2) U_0 \right\} + \varepsilon_x \tag{3.3}
\]

\[
(U_y)_o = \frac{1}{h_2^2 h_4 (h_2 + h_4)} \Delta y \left\{ h_4^2 U_2 - h_2^2 U_4 - (h_4^2 - h_2^2) U_0 \right\} + \varepsilon_y \tag{3.4}
\]

where \( \varepsilon_{xx}, \varepsilon_x \to 0 \) as \( \Delta x \to 0 \) and \( \varepsilon_{yy}, \varepsilon_y \to 0 \) as \( \Delta y \to 0 \).

(We may digress here to remark that a corresponding expression for \( \frac{\partial^2 U}{\partial x \partial y} \) involves values of \( U \) at points lying on the diagonals through 0 (fig. 2), and for this reason terms of this type are excluded from our general form (1.1)).

Substituting (3.1) – (3.4) into equation (1.1), neglecting the truncation terms and rearranging, we obtain

\[
U_0 = C_1 U_1 + C_2 U_2 + C_3 U_3 + C_4 U_4 + C_5 \tag{3.5}
\]

where \( C_1 - C_5 \) are functions of \( h_1 - h_4, \Delta x, \Delta y \) and (through the coefficients a - f) of the co-ordinates of the mesh point \( M. \) For each mesh point in the region of interest (between \( C_1 \) and \( C_2, \) fig. 1) we have such an equation, and correspondingly, an unknown value \( U_0. \) There are thus just sufficient equations to determine the set of unknown \( U_0. \)

The equation (3.5) for points near a boundary, such as point \( O, \) fig. 2, involves values of \( U \) at points on the boundary (points 2 and 3) which are not mesh points. In order to be able to treat all mesh points in exactly the same way, we allocate to points on the boundary, such as points 2 and 3, the number of the nearest external mesh point. That is to say, the boundary value of \( U \) at point 2 will be stored in the computer in a location which is referenced by the number of mesh point \( A, \) and the boundary value of \( U \) at point 3 will similarly be referenced by the number of mesh point \( B. \) It should be remarked that this does not correspond to deforming the boundary to bring points 2, 3 into coincidence with points \( A, B \) respectively: it is simply a method of attaching reference numbers to
boundary points, for which no other numbering system exists.

Conflicts may arise, for example in fig. 2 the number of mesh point A should be allocated to both points 2 and 5. The computer resolves such conflicts by assuming that the value of U at all conflicting points is the same and equal to the value at the boundary point nearest to the mesh point (point A in fig. 2) for which the boundary points conflict. The error introduced by such an assumption \( \rightarrow 0 \) as \( \Delta x, \Delta y \rightarrow 0 \); moreover, in many practical problems, U is constant along parts of a boundary (e.g. equipotentials), and then the assumption is valid.

Values of U are unknown along boundaries where Neumann boundary conditions exist, and in such cases we need additional equations of type (3.5) to meet the increased number of unknowns. Referring to fig. 3, an equation of the form (3.5) can be set up for point 0, but it introduces \( U_2 \), the value of U at a point outside the region about which nothing is known. However, along the Neumann boundary equation (1.2) is valid, and gives us an opportunity of finding an expression for the missing \( U_2 \). Specifically by substituting (3.3) and (3.4) into (1.2), neglecting the truncation terms and rearranging, we have an explicit expression for \( U_2 \) in terms of \( U_0 \), \( U_1 \), \( U_2 \) and \( U_4 \). By substituting this into (3.5) and rearranging, we get

\[
U_0 = C_1 U_1 + C_2 U_2 + C_4 U_4 + C_5
\]

which is the missing equation corresponding to the unknown \( U_0 \) on the Neumann boundary of fig. 3.

4. Solution of the Equations

Solution is by the method of successive over-relaxation, also known as the extrapolated Liebmann method, which is described exhaustively in the literature \(^{2,3,4}\). For the purpose of the subsequent discussion it is necessary to describe briefly the method here.

Values of U are assigned to each of the mesh points of the domain
(these values consist of any fixed specified values on the boundaries, and a 'guess' for all other points.) This trial solution is a quite arbitrarily chosen approximation to the true solution. A correction process is then applied at each of the mesh points of the domain in turn, starting with the lowest numbered mesh point for which an equation exists and proceeding from left to right along successive mesh rows (fig. 1) to the highest numbered point for which an equation exists. When the correction process has been applied once to every mesh point for which an equation exists, there are new values of \( U \) at all these points, which are an improved approximation to the true solution. Denoting by \( U_j^{(k)} \) the value of \( U \) at the \( j \)th mesh point (local numbering, fig. 2) after the correction process has been applied \( k \) times, the correction process at point 0 is described by the following equation:

\[
U_0^{(k+1)} = U_0^{(k)} + \beta (c_1 U_1^{(k)} + c_2 U_2^{(k)} + c_3 U_3^{(k+1)} + c_4 U_4^{(k+1)} + c_5 - U_0^{(k)})
\]  

(4.1)

where \( \beta \) is a suitably chosen constant called the 'accelerating factor'. Notice that when we apply (4.1) to a point 0, all points below and to the left will already have been corrected, so that \( (k+1) \) corrections will have been applied to points 3 and 4, while all points above and to the right will not yet have been corrected, so that only \( k \) corrections will have been applied to points 1 and 2.

The correction process is applied repeatedly at all mesh points until the values of \( U \) are judged (using a criterion we shall quote presently) to be sufficiently close to the true solution.

Choice of a suitable accelerating factor (in the range \( 1 \leq \beta < 2 \)) is extremely important: it has been shown that a suitable choice can speed convergence of the process considerably.

If \( U \) is a vector containing the unknown values of \( U \), then
the complete set of equations of the form (3.5) can be written in matrix notation.

\[ A U = b \quad , \]  

where \( A \) is a constant matrix containing the coefficients \( C_j \) in a certain arrangement, and \( b \) is a constant vector containing the \( C_5 \) coefficients and those terms of (3.5) which refer to constant boundary values of \( U \).

Matrix \( A \) can be written in the form

\[ A = \bar{E} + \bar{D} + \bar{F} \quad , \]  

where \( \bar{E} \) is a lower triangular matrix with zeros on the diagonal, \( \bar{D} \) is a diagonal matrix, and \( \bar{F} \) is an upper triangular matrix with zeros on the diagonal. It can be shown that the correction process (4.1) can be represented by the matrix iteration

\[ U^{(k)} = H U^{(k-1)} + \bar{E} \quad , \]

where

\[ H = - (\beta^{-1} \bar{D} + \bar{F})^{-1} (\bar{F} + (1 - \beta^{-1}) \bar{D}) \]  

\[ \bar{E} = (\beta^{-1} \bar{D} + \bar{F})^{-1} \bar{b} \]

The iteration (4.4) converges if and only if all the latent roots of \( H \) have modulus less than unity. Thus the convergence is influenced by the accelerating factor \( \beta \) which appears in the expression (4.5) for \( H \).

5. Estimation of the Accelerating Factor

It has been shown \(^3\) that all the latent roots \( \lambda \) of \( H \) are indeed of modulus less than unity, for any \( \beta \) in the range \( 0 < \beta < 2 \), provided

(i) all the latent roots \( \mu \) of matrix \( (A - I) \) are real and have modulus less than unity;

(ii) \( A \) is of a particular form: matrices which have the required form are said to possess 'Property A';

(iii) the equations are solved in a certain order, said to be 'consistent'.

6084/p
If the above conditions hold, it can be shown further that the optimum accelerating factor (i.e. that which gives the fastest convergence) is

$$\beta_o = 2 \left\{ 1 + \sqrt{1 - (\lambda_{\text{max}}^{\text{max}} + \beta - 1)^2 / \lambda_{\text{max}}^{\text{max}} \beta^2} \right\}^{-1} \tag{5.1}$$

where $\lambda_{\text{max}}$ is the latent root of largest modulus of $H$.

Now the present programme sets up matrix $A$ which does indeed have 'property A', and solves the equations in a consistent order. However, there is no way of guaranteeing that condition (i) holds, i.e., that all the latent roots of $(A - I)$ are real and have modulus less than unity. Nevertheless, the method may well converge even if some of the roots $\mu$ are complex, and we have based the programme on this method because the operating experience of others $^5,^6$ suggests that it is valid over a wide field of application.

The present programme uses a method due to Carré $^5$ for the automatic determination of $\beta_o$. The method hinges on the estimation of $\lambda_{\text{max}}$ for a chosen $\beta$, whence an estimate of $\beta_o$ is given by (5.1). $\lambda_{\text{max}}$ can be estimated from the behaviour over several iterations of the norm of the displacement vector

$$\delta(k) = U(k) - U(k-1) \tag{5.2}$$

The norm $n(k)$ is the sum of the absolute values of the elements of $\delta(k)$.

Carré's method is as follows:

1. carry out the first iteration with $\beta = 1$, then set $\beta = 1.375$;
2. carry out 12 more iterations with the most recent value of $\beta$, and from the resulting values of $n(k)$ estimate $\lambda_{\text{max}}$ and hence $\beta_o$ from (5.1);
3. determine a new $\beta$ from the relation

$$\beta = \beta_o - (2 - \beta_o)/4 \quad \tag{5.3}$$
(4) repeat steps (2) and (3) until the difference \( \Delta \beta \) between two successive estimates of \( \beta \) satisfies
\[
\frac{\Delta \beta}{(2 - \beta)} < 0.05
\]
(5.4)
(5) carry out all subsequent iterations with this final \( \beta \).

Thus, during the early iterations, we are using a value of \( \beta \) which should give a reasonably high convergence rate, and at the same time we are continually improving the estimate of \( \beta \).

In order to assess the accuracy of the solution at any time we use \( \delta_{\max}^{(k)} \), the arithmetically largest element of \( \delta^{(k)} \). We assume that the largest error \( e_{\max}^{(k)} \) is bounded by
\[
e_{\max}^{(k)} \leq \frac{\lambda e | \delta_{\max}^{(k)} |}{(1 - \lambda e)}
\]
(5.5)
where if a final estimate of \( \beta \) has not yet been made then \( \lambda e \) is the most recent assessment of \( \lambda_{\max} \), and if a final estimate of \( \beta \) is being used \( \lambda e = \beta - 1 \).

For further details and reasons, see Carré 5).

Carré's method of estimating \( \lambda_{\max} \) fails if the process passes through a phase in which the norm \( n^{(k)} \) is increasing with \( k \) for some iterations, as it may well do when there is a large number of equations. The present programme senses when this is the case; it then uses for all subsequent iterations the most recent estimate of \( \beta \), obtained before \( n^{(k)} \) started to increase.

6. Quasi-Linear Equations

If none of the coefficients \( a, b, \ldots, f \) in equation (1,1) are functions of \( u \), the equation is linear. In our development of the theory so far we have tacitly assumed that such is the case. When one or more of \( a, b, \ldots, f \) is a function of \( u \), then one or more of the coefficients
$C_1 \ldots \ldots C_5$ in the corresponding difference equation (3.5) is also a function of $U_0$, i.e. we have to solve a non-linear set of simultaneous equations. Our method is as follows:

1. calculate $C_1 - C_5$ from an initial guess at the set of values $U$.
2. solve the resulting set of linear equations by the Extrapolated Liebmann Method as described, obtaining a new estimate of the set of values $U$.
3. re-calculate $C_1 - C_5$ using the values of $U$ obtained in (2).
4. solve the set of equations again, using the set of values of $U$ obtained in (2) as the first guess for the Extrapolated Liebmann Method.
5. repeat (3) and (4) as often as necessary to obtain the required accuracy.

Whether or not such a process will converge depends entirely on the equation being solved and the boundary values involved.

7. **First Approximation to a Solution**

   Normally the computer itself provides a first guess of zero for the values of $U$ at all mesh points for which fixed boundary values of $U$ are not available. However there is a facility for reading in from magnetic tape a set of values of $U$ to be taken as the first guess; this is useful when a problem is to be solved which is very similar to a problem for which the solution has already been obtained.

8. **Accuracy Requirement**

   The computer must be given some criterion from which to judge when the iterative method of solving the difference equations has converged to adequate accuracy. In the present programme, the process ceases as soon as

   $$c_{\text{max}} < \frac{c}{100} \cdot |U_i|$$

   6084/p
where \( e_{\text{max}} \) is a bound on the maximum error defined by the right-hand side of (5.5), \( U_i \) is the value of \( U \) at the mesh point at which the largest error occurred, and \( e \) is a percentage accuracy criterion specified by the user.

9. Specifying the Problem

The equation to be solved, (1.1), must be manipulated into the form (3.5) using the substitutions (3.1) - (3.4), giving explicit expressions for the coefficients \( C_1 - C_5 \). To distinguish this set of expressions from other sets arising from other equations or from Neumann boundaries, we allocate to it a 'basic code number', \( K_0 \), which is an integer in the range \( 1 \leq K_0 \leq 255 \).

For each Neumann boundary we obtain an expression of the form (3.6) with one of \( U_1 \), \( U_2 \), \( U_3 \) or \( U_4 \) missing, using the procedure indicated in the later paragraphs of section 3, so obtaining further sets of expressions for \( C_1 - C_5 \). To identify the set of expressions for the \( j \)th Neumann boundary we allocate a code-number \( k_j \) in the same range as \( K_0 \).

A FORTRAN subroutine having the name GETCO must be prepared for the calculation of \( C_1 - C_5 \). The main programme supplies the subroutine with values of \( h_1 \), \( h_2 \), \( h_3 \) and \( h_4 \) and the code number \( K \), together with other relevant information; the subroutine, by inspection of \( K \), decides which set of expressions for \( C_1 - C_5 \) to use, and carries out the relevant computation. Details of how to write the subroutine are given in Appendix II.

Two further FORTRAN subroutines USER 1 and USER 2 must be prepared. The first may be used for example to read in further data, e.g. constants which might appear in the terms \( a \), \( b \), \( c \)..... of (1.1); and the second might be used for additional output, or for additional calculations in the outer cycle of a quasi-linear problem. Details are given in Appendix II.
The boundary curves are defined by specifying the co-ordinates of a set of points on each curve. The boundary value of $U$ at each such point must be known. Between any two neighbouring specified points the boundary is assumed to be straight, and $U$ is assumed to vary linearly, except that if the two points are labelled with the same code number $K_j$, the line joining them is the $j^{th}$ Neumann boundary, along which $U$ is unknown.

To complete the necessary information, $\Delta x$, $\Delta y$ and $\varepsilon$ must be specified.

10. Data Preparation

Table I shows how the data should be presented to the computer. It should be written on standard 709 data forms, which are ruled vertically to represent the 80 columns of a card. Fields 1, 2 and 3 are decimal number fields, fields 4 and 5 are integer fields (see Appendix I). In explanation of the table, the following remarks are pertinent:

(i) $N_c$ is the number of simple closed curves making up the boundary. For the region of fig. 1, $N_c = 2$

(ii) Run no. is any identification number chosen by the user; it is reproduced at the head of the output results

(iii) $K_u$ is the basic code number, which in effect defines the differential equation being solved

(iv) $m$ is the number of mesh points at which Neumann boundaries intersect at right angles. For many problems $m = 0$, and when this is the case the following section, labelled 'information about nodes at intersection of two Neumann boundaries', is omitted. If $m \neq 0$, refer to Appendix VI.

(v) $N_1$ is the number of points on the first boundary curve, and is followed by the co-ordinates $(x_1, y_1)$, boundary value $U_1$ and code number $K_j$ (if any) of each point in turn, in their correct order round the boundary. The code numbers will be left blank, except where two entries in the table represent the end-points of a Neumann boundary; in such a
### Table I - Data Preparation

<table>
<thead>
<tr>
<th>1-15</th>
<th>16-30</th>
<th>31-45</th>
<th>46-50</th>
<th>51-55</th>
<th>FIELD</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Delta x$</td>
<td>$\Delta y$</td>
<td>$N_c$</td>
<td>Run no.</td>
<td>$K_o$</td>
<td>m</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Information about nodes at intersection of two Neumann Boundaries</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$(\epsilon_1)_{11}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$(\kappa_2)_{11}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$\cdots$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$\cdots$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$N_m$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$(\epsilon_1)_m$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$\kappa_{2m}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$(\epsilon_2)_m$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$y_1$</th>
<th>$u_1$</th>
<th>$N_1$</th>
<th>Code nos.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_2$</td>
<td>$y_2$</td>
<td>$u_2$</td>
<td>$k_j$</td>
<td>first boundary curve</td>
</tr>
<tr>
<td>$\cdots$</td>
<td>$\cdots$</td>
<td>$\cdots$</td>
<td>$\cdots$</td>
<td>necessary</td>
</tr>
<tr>
<td>$x_{N_1}$</td>
<td>$y_{N_1}$</td>
<td>$u_{N_1}$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$y_1$</th>
<th>$u_1$</th>
<th>$N_2$</th>
<th>Code nos.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_2$</td>
<td>$y_2$</td>
<td>$u_2$</td>
<td>$k_j$</td>
<td>second boundary curve (if any)</td>
</tr>
<tr>
<td>$\cdots$</td>
<td>$\cdots$</td>
<td>$\cdots$</td>
<td>$\cdots$</td>
<td>necessary</td>
</tr>
<tr>
<td>$x_{N_2}$</td>
<td>$y_{N_2}$</td>
<td>$u_{N_2}$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| ETC. | |

| $\beta$ | $e$ | Marker |

Further boundary curves (if any)
case the relevant $K_j$ is entered against the co-ordinates of the two
points concerned. Note particularly that the table is 'cyclic', i.e.
points numbers $1$ and $N_1$ are geometrically neighbours.

Values for the second and any subsequent boundary curve follow
in exactly the same way.

(vi) Since the programme itself will attempt to determine the
optimum acceleration factor $\beta$, it is unnecessary to specify any value
in the data. However, if an accurate estimate of $\beta$ is known either from
the results of previous runs, or from theoretical or empirical calculations,
then this value of $\beta$ may be entered in the data and will be used for all
iterations except the first.

(vii) $\epsilon$ is the percentage accuracy criterion whose use is defined
in section 8.

(viii) Marker is used to indicate whether or not an array of values
of potential at the mesh points representing an approximation to the
wanted solution should be read in from a magnetic tape. Such an approxi-
mation will normally exist as the result of a previous run of this programme
on a similar problem. The old and the new problems must have the same
finite difference net in the sense that the number of rows, NR, and the
number of columns, NC, must be the same for both, as well as the mesh
intervals $\Delta x$ and $\Delta y$.

Marker  = 0: no approximation to be read in
Marker  = 1: an approximation is to be read in

In the latter case, the programme will check that the approximation
is suitable by verifying that $\Delta x$ and $\Delta y$ are the same for both. If not,
it will not use the approximation. If it is suitable, then in addition,
the value of $\beta$ used for the old problem is read in, and replaces any value
that may have been written in the data.

Table I and the above remarks describe data which are required
for all problems. If further data must be input for a particular case,
such data will follow the data described in Table I; they will be input under the control of subroutine USER 1 which must be written specially for the particular case; the order and arrangement of such further data will depend upon the way subroutine USER 1 is written.

Some examples of the way in which data is prepared will be found in Appendix VII.

11. The Method of Classifying Mesh Points

Prior to setting up the difference equations, the computer takes each mesh point in turn, and considers the following possibilities:

(i) is the point on a boundary at which the value of $U$ is known, or is it, although outside the region, a point at which a boundary value of $U$ should be stored (such as points A and B in fig. 2), or is it, although inside the region, so close to the boundary that it ought to be considered as a boundary point?

(ii) is the point on a Neumann boundary and if so what is the relevant code number $K_j$?

(iii) is the point inside the region and near to a boundary, and if so, what are the relevant neighbour distances $h_j$ ($j = 1, 2, 3, 4$)? (e.g. point 0 in fig. 2)

(iv) is the point inside the region and well away from any boundary, (a so-called regular interior point) so that all the neighbour distances $h_j$ are equal to unity?

(v) is the point outside the region and so far away from a boundary that it can safely be ignored?

It has been found that the computer is helped in its deliberations if none of the points tabulated in Table I (defining the boundaries) lies exactly on a mesh row or mesh column. The programme therefore in effect deforms the boundary slightly by altering the co-ordinates of some of the data boundary points according to the following rule:- if a data point lies within .00005 $\Delta y$ of a mesh row (on either
side) then .0001 \( \Delta y \) is added to its \( y \) co-ordinate; similarly, if it lies within .00005 \( \Delta x \) of a mesh column, then .0001 \( \Delta x \) is added to its \( x \) co-ordinate.

To ensure that the mesh completely covers the region, the programme takes

\[
\text{number of mesh columns} = NC = \left\lceil \frac{\text{max.}(x_i) + .0001 \Delta x}{\Delta x} \right\rceil + 2
\]

\[
\text{number of mesh rows} = NR = \left\lceil \frac{\text{max.}(y_i) + .0001 \Delta y}{\Delta y} \right\rceil + 2
\]

(11.1)

where \( \lceil X \rceil \) means 'the integral part of \( X \).

Mesh points which come in category (i) are recorded in a table of boundary values; this is a two-column table in which values in the left-hand column are mesh point numbers, and values in the right-hand column are the associated boundary values of \( U \). Mesh points which come in categories (ii) and (iii) are recorded in a table of irregular points; this is a six-column table in which column 1 gives the mesh-point number, columns 2 - 5 give the neighbour distances \( h_1 \) - \( h_4 \) respectively, and column 6 gives the code number \( K_j \) or \( K_o \). Mesh points which come in category (iv) are recorded in a table of regular points; this is a two-column table in which, in any row, the first and second columns contain mesh-point numbers, say \( n_1 \) and \( n_2 \); signifying that all mesh points with numbers from \( n_1 \) to \( n_2 \) inclusive are in category (iv).

Before classifying a point, its neighbour distances are computed; where a neighbour distance is less than .0002, it is replaced by zero, and where a neighbour distance is more than .9998 it is replaced by unity. Thus points which are close to a boundary are treated as being on the boundary. This has the effect of nullifying rounding errors occurring at input time, and some of the errors due to the deformation of the boundary; some of the errors due to the latter cause persist however, and may be visualised as an alteration of the boundary near a

6084/p
few boundary points (usually at sharp corners) by an amount less than the mesh interval.

12. Restrictions

The configuration of the boundary and the choice of mesh intervals must be such that the following conditions are satisfied:

(a) \( NR \leq 200, \ NC \leq 200, NC, NR \leq 3250 \)

(b) Total number of boundary points in table I,
\[ N_1 + N_2 + \ldots \ldots \leq 499 \]

(c) Number of intersections of any mesh row or column with a boundary \( \leq 20 \)

(d) Total number of mesh points at which a difference equation will be set up, i.e., number of interior mesh points plus number of mesh points on a Neumann boundary, \( \leq 2900 \).

(e) Number of rows in the table of boundary values \( \leq 780 \)

(f) Number of rows in the table of irregular points \( \leq 780 \)

(g) Number of rows in the table of regular points \( \leq 390 \).

Conditions (a), (b) and (c) are easy to check from the data. Conditions (d) to (g) are more difficult to check, but will be satisfied for most 'reasonable' boundaries provided (a) to (c) are satisfied. However, if (a) is only just satisfied or if (c) is close to the limit of 20, it would be wise to check conditions (d) to (g) from a rough sketch of the region with the net superimposed.

The largest square region which can be treated is limited by (d) to \( NC = NR = 55 \), although (a) suggests that \( NC = NR = 57 \) could be treated.

13. Results

The results are output on two tapes; there are therefore two output listings.

The normal FORTRAN output tape contains information about the
progress of the iterative process for solving the set of difference equations. It consists essentially of an eight-column table, in which:

column 1 = iteration number

column 2 = value of \( \beta \) used in the iteration; this is always unity for the first iteration and 1.375 for the next 12 iterations. Subsequent values are given by equation (5.3). [However, if the user specifies a value of \( \beta \) on the data sheet as described in paragraph 10, remark (vi), this is used in all iterations after the first].

column 3 = the norm \( n^{(k)} \) of the displacement vector \( \delta^{(k)} \)
(the displacement vector is the set of changes in \( U^{(k)} \) after one iteration, equation (5.2)).

column 4 = the mesh-point number \( m \) of the point at which the displacement was largest.

column 5 = the largest element of the displacement vector, \( \delta^{(k)}_{\text{max}} \).

column 6 = the corresponding value of \( U, U_{\text{m}}^{(k)} \).

column 7 = \( \lambda \), an estimate of the value of the latent root of largest modulus of the matrix of coefficients of the set of equations being solved. This is arbitrarily set equal to 0.95 during the first 13 iterations, and is subsequently estimated as prescribed by Carré 5).

column 8 = an estimate of the actual error in \( U_{\text{m}}^{(k)} \)
expressed as a proportion of \( U_{\text{m}}^{(k)} \);
\[ e_{\text{max}} = \frac{\lambda \delta_{\text{max}}}{(1-\lambda)U_{\text{m}}^{(k)}} \] (see (5.5))

After the 13th and 25th iterations there will be, and after each subsequent 12 iterations there may be, an additional line of output of 5 quantities. The first three of these are the ratio of the norm for an iteration to the norm for the previous iteration, evaluated for the three most recent iterations. The fourth is a new estimate of \( \lambda \), and is equal to the most recent norm ratio except after the 13th iteration,
where an Aitken extrapolation may have been carried out. The fifth quantity is a new estimate of \( \beta_0 \) obtained by equation (5.1).

As the process proceeds, values in columns 3, 5 and 8 must tend to zero. The solution is regarded as satisfactory when \( \epsilon_{\text{max}} \) becomes less than \( \epsilon /100 \), \( \epsilon \) being the percentage accuracy criterion specified in the data.

The second output tape contains the solution to the problem. The listing consists of:

(a) Two rows of 5 numbers:

<table>
<thead>
<tr>
<th>Run No</th>
<th>( \Delta x )</th>
<th>( \Delta y )</th>
<th>NR</th>
<th>NC</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \beta_p )</td>
<td>( \lambda )</td>
<td>( \beta )</td>
<td>( \beta_e )</td>
<td>( N_i )</td>
</tr>
</tbody>
</table>

where

- \( \beta_p \) is either the optimum accelerating factor as found by the programme, or is zero if the programme has not yet found the optimum accelerating factor, or is the value of \( \beta \) (if any) specified by the user in the input data;
- \( \lambda, \beta \) are the latest values of the corresponding quantities mentioned above;
- \( \beta_e \) is either the latest estimate of the optimum accelerating factor (may be equal to \( \beta_p \)), or is zero if the value of \( \beta \) was specified by the user in the input data;
- \( N_i \) is the total number of iterations performed.

(b) The values of \( U \): these are listed 7 per row in order of increasing mesh-point number from \( M = 1 \) to \( M = NR,NC \). Points outside the region have \( U = 0 \), except that mesh points outside but near the boundary 'carry' the neighbouring boundary values of \( U \). (See remarks following equation (3.5) concerning points such as 2 and 3 in fig. 2).

(c) Three integers, being the number of rows, respectively, in the three tables which follow.
(d) Table of boundary values
(e) Table of irregular points \{ \}
(f) Table of regular points

\text{see Section II.}
14. References


2. Modern Computing Methods, Chapter 4. London H.M.S.O. 1961 (for a brief introduction)


Appendix I - Data Fields

Decimal Number Field

A decimal number field must contain either
(i) a decimal number as defined below, or
(ii) a decimal number followed by an exponent part.

A decimal number here has the conventional meaning, i.e. a number including a decimal point. The decimal point is essential, unless the number is zero, and there may be digits before and/or after it. There must be no blank spaces between the digits, or between the decimal point and the preceding or following digit. There may be a sign preceding the first digit: in the absence of a sign, the number is taken as positive.

The exponent part consists of a sign (this is essential) followed by one or two digits, with no blank spaces between the digits.

A decimal number when appearing in a decimal number field may be preceded and/or followed by blank spaces. If there is an exponent part it must be at the extreme right-hand side of the decimal number field, i.e. the last digit of the exponent part must appear in the right-most column of the field.

Examples

<table>
<thead>
<tr>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>1027.3</td>
</tr>
<tr>
<td>1027.3 + 2</td>
</tr>
<tr>
<td>10273 - 1</td>
</tr>
<tr>
<td>0010273 + 6</td>
</tr>
</tbody>
</table>
Integer Field

An integer field must contain a decimal integer with no
blank spaces between the digits, and the units digit in the right-most
column of the field. There may be a sign preceding the first digit:
in the absence of a sign, the number is taken as positive.
Appendix II - Preparation of Subroutines

The programme is arranged as a CHAIN job with two links. The first link reads all the data listed in Table I except the last card, prepares in COMMON the tables of boundary values, irregular and regular points, and calls the second link.

The second link carries out the iterative processes in accordance with the following very simplified flow diagram:

No

Is this the first entry to this link?

Yes

Set NLITS = 0, read remaining data, set OPM ≠ 0.

USER 1

For each equation in turn, use GETCO to calculate $C_1, C_2, \ldots, C_5$.

Extrapolated Liebhmann Iteration

Is error < EWANT? No

Adjust $\beta$ using Carré's method.

Yes

Is OPM = 0?

Yes

No

Output results

USER 2

6084/p
We first discuss subroutines USER 1 and USER 2. These have no argument lists: they use and alter information in COMMON, listed in Appendix III. In linear problems, USER 2 is not reached until the problem is solved, so that it must consist only of the statements PAUSE and CALL EXIT. USER 1 may be used to read additional data to be placed in COMMON (in the spare locations labelled DUMMY if sufficient) for subsequent use by subroutine GETCO. Often, however, there is nothing to be done, and then USER 1 may consist only of the statement RETURN.

In quasi-linear problems, it would be advisable to set $OPM = 0$ in USER 1, since one would not want to output results until some outer iterative process had converged. The main body of this outer iteration would be in USER 2, which would test for convergence and finally set $OPM \neq 0$ for a last run through. If this part of the calculation is non-trivial, it is advisable to embody it in a third link, to be called from USER 2. The main programme is then re-entered by recalling link 2, instead of using the usual RETURN statement. If extensive data prepared by USER 2 are required by GETCO for the recalculation of coefficients, they should be written onto magnetic tape.

If a basic code $K_0 > 200$ is chosen, then just before exit from link 1 to link 2, some information is written on to logical tape 3 in order to make it available to USER 1 and USER 2. The information may be collected by including in the subroutine the statements

```
READ TAPE 3, NPTS
READ TAPE 3, (XDATA(I), YDATA(I), I = 1, NPTS)
```

NPTS is the total number of boundary points in Table 1,

$$NPTS = N_1 + N_2 + N_3 + \ldots \ldots \ldots$$

while XDATA, YDATA are respectively the set of all $x$ co-ordinates and $y$ co-ordinates in Table 1.

Subroutine GETCO has the following argument list:

```
SUBROUTINE GETCO (X, Y, H, M, KODE, COFFT, JEQU)
```

with DIMENSION H(4), COFFT(5)
The following information is supplied by the main programme to the subroutine:

- \( K, Y \): co-ordinates of the mesh point being considered;
- \( M \): number
- \( h_j \) (\( j = 1, 2, 3, 4 \)): neighbour distances defined in section 3;
- \( KOD J \): code number \( K_c \) or \( K_j \) classifying the point, and defined in Section 9;
- \( JEQ U \): number of the difference equation corresponding to the mesh point being considered.

From this information, together with other information available in COMMON, the subroutine must compute the values of \( C_1, C_2, \ldots, C_5 \) and place them in \( COFFT(I) \), \( I = 1, 2, \ldots, 5 \).
Appendix III - Variables in COMMON

The following is a list of names of variables in COMMON. Numbers in brackets indicate names of arrays, and give their dimensions.

C5, C95, NT3(780), UT3(780), NT4(780), HT4(780, 4),
KT4(780), NT5(390, 2), NE3, NE4, NE5, NC, NR, KODBAS,
DX, DY, JOBNUM, UMAT(3250), NEQU, NPIA, EWANT,
EFINAL, NLITS, DUMMY(10), OPM, BETA, BSQPR,
EIGEN, NIT, BUNK, NUMPT(2900), DIFCO(2900, 5)

These variables have the following significance:

C5 = 0.0002, C95 = 0.9998. These are the critical values used to decide if a mesh point ought to be considered as exactly on a boundary (see last paragraph of Section 11).

They are set in S1 and can be changed. However, if C5 is changed, then an obvious change must be correspondingly made to one of the first statements of S13.

NT3, UT3 are respectively the first and second columns of the table of boundary values. (Section 11.)

NT4, HT4, KT4 are respectively the first, next four and last columns of the table of irregular points. (Section 11)

NT5 is the table of regular points. (Section 11)

NE3, NE4, NE5 are respectively the numbers of rows used in the above three tables.

NC, NR are the numbers of columns and rows respectively in the finite-difference net. (Equation 11.1)

KODBAS = basic code number K (Section 9)

DX, DY = Δx, Δy respectively

JOBNUM is the user's run number (Section 10, remark (ii))

UMAT is the matrix of values of U, i.e. UMAT(I) is the value of U at mesh point I.

NEQU is the number of finite-difference equations to be solved.
NPIA = NC.NR.
EWANT = /100 (Section 10, remark (vii)).
BFINAL = P (Section 13, (a)).
NLITS = number of iterations performed in a quasi-linear problem.
DUMMY is a group of spare locations.
CPM is an output marker.
BETA = (Section 13, (a)).
BUNK is an unused location.
BEEP = $\beta_E$
EIGEN = $\lambda$
NIT = $N_i$
NUMPT is a list of the numbers of all mesh points at which there is a difference equation to be solved, in increasing order.

DIFCO contains five quantities in the Ith row. These are the coefficients $C_1, C_2, \ldots, C_5$ in the equation for the mesh point whose number is recorded in NUMPT(I).
Appendix IV - Technical Details

Since this is a chain job, common tapes are required on units A4 and B1. The results are output on unit B2. If a first guess at the solution exists (Section 10, item (viii)) it must be on unit B3. If a job overruns, it can be safely interrupted by lowering sense switch 1 and saving the tape from B2. Subsequently the job may be continued by loading this tape into B3 and rerunning the programme with the same data except that Marker = 1.

The instructions given to the operator at run time are as follows:

00015C2
THIS IS A CHAIN JOB. COMMON TAPES WILL BE REQUIRED ON UNITS A4 AND B1 DURING BOTH COMPILATION AND EXECUTION;

AT PAUSE 0, LOAD THE SPECIFIED TAPE INTO UNIT B2 NOT FILE-PROTECTED IF A TAPE IS SPECIFIED FOR UNIT B3, LOAD THIS ALSO (FILE-PROTECTED).
PRESS START.

IF PROGRAMME HAS NOT TERMINATED WITHIN SPECIFIED TIME, LOWER SENSE SWITCH 1

PROGRAMME ENDS AT PAUSE 525. REMOVE TAPES (THAT FROM UNIT B2 IS TO BE LISTED). RAISE SENSE SWITCH 1, IF NECESSARY.

AT PAUSE 777 (IF ANY), PRESS START.

DO NOT OMIT TO LIST THE MONITOR OUTPUT TAPE, ON UNIT A3.

The input deck must consist of the following cards:

00015C2
CHAIN (1,A4)
00015B50
CHAIN (2,A4)
USER 1
USER 2
GETCO
00015B51
Data

B50 consists of subprogrammes B1 to B9 inclusive, and B51 consists
of subprogrammes B11, B12 and B13. Included with both B50 and B51 is an IOU table which gives the following logical/physical tape unit identifications:

<table>
<thead>
<tr>
<th>Tape Unit</th>
<th>Logical Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>A2</td>
<td>1</td>
</tr>
<tr>
<td>A3</td>
<td>2</td>
</tr>
<tr>
<td>B1</td>
<td>3</td>
</tr>
<tr>
<td>B2</td>
<td>4</td>
</tr>
<tr>
<td>B3</td>
<td>5</td>
</tr>
<tr>
<td>B5</td>
<td>6</td>
</tr>
<tr>
<td>A4</td>
<td>7</td>
</tr>
<tr>
<td>A1</td>
<td>8</td>
</tr>
</tbody>
</table>
Appendix V

Alternative Version of the Programme to Treat Larger Cases.

An alternative version of the programme exists in which the coefficients $C_1, \ldots, C_5$ are not stored separately for each equation. Instead, $C_j$ is stored only if it differs from the $C_j$ for the previous equation.

An array WUMPT of 1334 words is set up which is to be considered as a sequence of 8004 6 bit sets which are in one-to-one correspondence with mesh points numbers 1 to 8004. The first of each set of 6 bits is a 1 if, and only if, there exists a difference equation at the corresponding mesh point. The remaining 5 bits are 0 or 1 respectively according as $C_1, \ldots, C_5$ is or is not the same for the current difference equation as for that preceding. Corresponding to each bit 1 a new value of some $C_j$ is to be found in the next available location of an array DIFCO of 10000 words.

This version is slightly slower than that described in the text owing to the time spent unpacking the bit codes.

Most of what has gone before applies to this version also, but the following differences should be noted.

Section 12 - Restrictions:

Restrictions (b), (c), (e), (f) and (g) still apply.

Also

(a) $NR \leq 200$, $NC \leq 200$, $NC,NR \leq 6000$

(d) Storage for coefficients as described above is restricted to 10000 numbers.

Appendix III - Variables in COMMON:

UMAT (3250) becomes UMAT (6000), and the last few variables are now

.........EIGEN, NIT, WUMPT(1334), DIFCO(10000)
Appendix IV - Technical Details:

The input deck should be

CO015C2
CHAIN (1,A4)
CO015B50
CHAIN (2,A4)
USER 1
USER 2
GETCO
CO0015B52
Data

B52 consists of subprogrammes B24, B26, B111, B112 and B113 and an IOU table.
Appendix VI

Case where Two Neumann Boundaries Meet

Since Neumann boundaries are restricted to be coincident with either a mesh row or a mesh column, two Neumann boundaries which meet must be either co-linear or at right angles.

In the former case, the co-ordinates of the meeting point are entered twice in Table I, once with the code number of the first Neumann boundary, and once with that of the second.

When two Neumann boundaries meet at right angles, we have to supply additional information about the meeting point (which is a mesh point) and we have to make a small deformation of the boundaries. Referring to figs. 4 or 5, let \( P_{2,2} \) and \( P_{2,3} \) be Neumann boundaries whose code numbers are \( K_1 \) and \( K_3 \) respectively. At \( P_2 \) there will be a difference equation of a special form identified by a code number \( K_2 \).

Instead of entering the true co-ordinates of \( P_2 \) in Table I, they must be altered by small amounts \( \eta_x \) or \( \eta_y \) in such a way that \( P_2 \) is displaced away from the mesh point (number \( M \), say) with which it is theoretically coincident, and into the interior of the region.

\[ 5 \times 10^{-5} < |\eta_x| < 2 \times 10^{-4}, \quad 5 \times 10^{-5} < |\eta_y| < 2 \times 10^{-4} \]

but are otherwise arbitrary.

The arrows in figs. 4 and 5 indicate the directions of these displacements. The co-ordinates of the displaced \( P_2 \) are entered twice in Table I, once with code number \( K_1 \) and again with code number \( K_3 \).

Now define two numbers \( \ell_1 \) and \( \ell_2 \) giving the directions of the arms of the right angle formed by the boundary at \( P_2 \) as follows:
\[ \varepsilon = 1 \]
\[ \varepsilon = 2 \]
\[ \varepsilon = 3 \]
\[ \varepsilon = 4 \]

the boundary runs in the direction of \( \begin{cases} \text{positive } x \\ \text{positive } y \end{cases} \) from M.

In fig. 4, \( \varepsilon_1 = 1, \varepsilon_2 = 2 \) or vice versa. In fig. 5, \( \varepsilon_1 = 1, \varepsilon_2 = 4 \) or vice versa.

Now for the whole boundary let there be \( m \) points at which Neumann boundaries meet at right angles, and let \( M_i, (K_i)_1, (\varepsilon_1)_i \), and \( (\varepsilon_2)_i \) be the mesh point number, code number and directions for the \( i \)th point \((i = 1, 2, \ldots, m)\). These numbers appear in Table I in the manner indicated.
Appendix VII - Example

As an example we consider to determination of $U$ satisfying

$$\frac{\partial^2 U}{\partial r^2} + \frac{1}{r} \frac{\partial U}{\partial r} + \frac{\partial^2 U}{\partial z^2} = 0 \quad (A7.1)$$

in the region illustrated in fig. 6.

On the boundary between points 2 and 3 and points 6 and 7 respectively, $U$ varies linearly, and between points 1 and 8 the condition $\frac{\partial U}{\partial r} = 0$ is to be satisfied.

**Analysis**

Identifying $\frac{\partial^2 U}{\partial r^2}$ with $U_{yy}$ and $\frac{\partial^2 U}{\partial z^2}$ with $U_{xx}$,

we substitute (3.1) - (3.4) into (A7.1), and after rearranging

we get (3.5) with

$$C_1 = 2\lambda h_3 / K, \quad C_2 = \mu h_4 (2 + \frac{h_4}{r}) / K$$

$$C_3 = 2\lambda h_1 / K, \quad C_4 = \mu h_2 (2 - \frac{h_2}{r}) / K$$

$$C_5 = 0$$

$$\mu = \left[ \Delta r^2 h_2 h_4 (h_2 + h_4) \right]^{-1} \quad (A7.2)$$

$$\lambda = \left[ \Delta z^2 h_1 h_3 (h_1 + h_3) \right]^{-1}$$

and

$$K = \mu (h_2 + h_4) (2 + \frac{h_4}{r} \Delta r) + 2\lambda (h_1 + h_3)$$

$$\left( A7.3 \right)$$

We agree to identify these expressions by the basic code number $K_0 = 100$

The $z$-axis is a Neumann boundary for this problem.
With the condition \( \frac{\partial U}{\partial r} = 0 \), equation (A7.1) reduces to

\[
2 \frac{\partial^2 U}{\partial r^2} + \frac{\partial^2 U}{\partial z^2} = 0
\]  
(A7.4)

and again using (3.1) - (3.4) we get (3.5) with

\[
\begin{aligned}
C_1 &= (\Delta r h_2)^2 \lambda h_2 / K , \\
C_2 &= 2 / K ,
\end{aligned}
\]

\[
\begin{aligned}
C_3 &= (\Delta r h_2)^2 \lambda h_1 / K , \\
C_4 &= C_5 = 0
\end{aligned}
\]  
(A7.5)

where now

\[
K = 2 + (\Delta r h_2)^2 \lambda (h_1 + h_2) .
\]  
(A7.6)

We agree to identify these expressions for the Neumann boundary by the code number \( K_1 = 101 \).

**Programming**

Subroutine GETCO, which must be designed to calculate the difference coefficients, has the following form.

**CALCULATION OF DIFFERENCE EQUATION COEFFICIENTS FOR LAPLACE'S EQUATION IN CYLINDRICAL CO-ORDINATES WITH AXIAL SYMMETRY (X=Z, Y=R)**

**SUBROUTINE GETCO** (X,Y,H,L,M,KODE,COPFT,JEQU)

**COMMON** DUMMY,DX,DY

**DIMENSION** HL(4),COPFT(5),DUMMY(7028)

**INTERIOR POINTS HAVE CODE 100, POINTS ON AXIS, CODE 101.**

CLAM=1.0/(HL(1)*HL(3)*(HL(1)+HL(3))*DX*DY**2)

IF(KODE=100)6,5,6

5 CMU=1.0/(HL(2)*HL(4)*(HL(2)+HL(4))*DY**2)

CKAY=(2.0+(HL(4)-HL(2))*DY/Y)/(HL(2)*HL(4)*DY**2)

1+2.0/(HL(1)*HL(3)*DX**2)

COPFT(1)=2.0*CLAM*HL(3)/CKAY

COPFT(2)=CMU*HL(4)*(2.0+HL(4)*DY/Y)/CKAY

COPFT(3)=2.0*CLAM*HL(1)/CKAY

COPFT(4)=CMU*HL(2)*(2.0-HL(2)*DY/Y)/CKAY

8 COPFT(5)=0

RETURN

6 IF(KODE=101)10,7,10

7 CKAY=2.0+((DY*HL(2)/DX)**2)/(HL(1)*HL(3))
COFFT(1)=(DY*HL(2)**2*CLAH*HL(3)/CKAY
COFFT(2)=2.0/CKAY
COFFT(3)=(DY*HL(2)**2*CLAH*HL(1)/CKAY
COFFT(4)=0
GO TO 8
10 WRITE OUTPUT TAPE 2,100,M,KODE
CALL EXIT
100 FORMAT (7HO POINT, I5,26H HAS UNDEFINED CODE NUMBER,I5)
END

Since there are no additional calculations to be carried out, subroutine USER 1 consists only of a RETURN statement, and subroutine USER 2 contains only PAUSE and CALL EXIT statements.

**Data Preparation**

Taking $\Delta r = \Delta z = 2.0$, we have a mesh with 29 rows and 52 columns. Referring to section 12, we see that this is well within the capacity of the faster version of the programme (see appendix V for a version which treats larger problems).

We ask for 0.5% accuracy in the solution of the difference equations so that, aside from discretisation errors, results should be correct to one unit of $U$.

Table II is the data sheet for this problem.

(Compare Table I)
<table>
<thead>
<tr>
<th>1</th>
<th>6</th>
<th>21</th>
<th>16</th>
<th>21</th>
<th>26</th>
<th>31</th>
<th>36</th>
<th>41</th>
<th>46</th>
<th>51</th>
<th>56</th>
<th>61</th>
<th>66</th>
<th>71</th>
<th>76</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25.0</td>
<td>55.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30.0</td>
<td>55.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40.0</td>
<td>30.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>80.0</td>
<td>30.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>95.0</td>
<td>55.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100.0</td>
<td>55.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>0.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

CERN 709 DATA FORM

TABLE II
Fig. 1

Fig. 2

Fig. 3