LA-4423

c.1

LOS ALAMOS SCIENTIFIC LABORATORY
of the
University of California
LOS ALAMOS • NEW MEXICO

Four Computer Programs Using Green's Third Formula to Numerically Solve Laplace's Equation in Inhomogeneous Media

UNITED STATES
ATOMIC ENERGY COMMISSION
CONTRACT W-7405-ENG. 36

Printed in the United States of America. Available from
Clearinghouse for Federal Scientific and Technical Information
National Bureau of Standards, U. S. Department of Commerce
Springfield, Virginia 22151
Price: Printed Copy $3.00; Microfiche $0.65

Written: April 1970
Distributed: June 1970

LA-4423
UC-32, MATHEMATICS
AND COMPUTERS
TID-4500

LOS ALAMOS SCIENTIFIC LABORATORY
of the
University of California
LOS ALAMOS • NEW MEXICO

Four Computer Programs Using Green's Third Formula to Numerically Solve Laplace's Equation in Inhomogeneous Media

by

John K. Hayes

CONTENTS

FOUR COMPUTER PROGRAMS USING GREEN'S THIRD FORMULA TO NUMERICALLY SOLVE LAPLACE'S EQUATION IN INHOMOGENEOUS MEDIA

by

John K. Hayes

ABSTRACT

This report serves as a user's manual and explains the theory behind four computer programs that can be used to numerically solve Laplace's equation. Laplace's equation in two dimensions and in three dimensions with axial symmetry is discussed. The numerical solution of both problems in inhomogeneous media is considered. A brief outline of applications to Poisson's equation is given.

I. INTRODUCTION

This report discusses four computer programs:

(i) LAPLACE solves the mixed boundary value problems for Laplace's equation in two dimensions.

(ii) LAPLARS solves the same problems in axially symmetric three-dimensional regions.

(iii) LAPLDDC solves the mixed boundary value problem for Laplace's equation in inhomogeneous media in two dimensions.

(iv) LAPLDRS solves the mixed boundary value problem for Laplace's equation in inhomogeneous media in axially symmetric three-dimensional regions.

The programs are called program (i), program (ii), and so forth. These programs are written in FORTRAN and are presently used on the CDC 6600. Programs (iii) and (iv) can be used to solve almost any problems that can be solved with programs (i) and (ii). However, programs (i) and (ii) are slightly faster, slightly easier to use, and much simpler as far as programming logic is concerned. All four programs use the same method to obtain a solution, have more or less the same input and output, and have the same structure. Because the programs are so similar, we emphasize the x-y plane with the understanding that the axially symmetric problems can be handled similarly.

The method used to obtain a numerical solution is different from that of most programs used to solve Laplace's equation, and makes the use of the program different. Suppose we want to find a numerical approximation to a solution of Laplace's equation in a two-dimensional region, G, with boundary S. For simplicity, assume that the region is a homogeneous medium. LAPLACE uses the fundamental formula

$$u(x,y) = \frac{1}{2\pi}\int_S \left[\ln\left(\frac{1}{r}\right)\frac{\partial u}{\partial \nu} - u\frac{\partial}{\partial \nu}\ln\left(\frac{1}{r}\right)\right]dS \qquad (x,y)\epsilon G \quad (1)$$

to approximate the desired solution, u, in G. See Sternberg and Smith,[1] p. 71. For this problem, either u or $\partial u/\partial \nu$ is known at every point of the boundary, S. The problem is to find u where $\partial u/\partial \nu$ is known, and $\partial u/\partial \nu$ where u is known. The values of u and $\partial u/\partial \nu$ that are sought are called the unknown boundary values. To approximate the unknown boundary values, we use a corresponding form of Eq. (1) for the boundary. Once the unknown boundary values have been computed, the solution or any of its derivatives can be approximated by using Eq. (1). Thus the use of LAPLACE is a two-step process:

1. Approximation of unknown boundary values, and

2. Approximation of the solution or its derivatives at desired points in G.

The program LAPLDDC also uses Eq. (1) to obtain a numerical solution. LAPLARS and LAPLDRS use the analog of Eq. (1) for the axially symmetric case.

II. GENERAL PROBLEM

In this section we detail the most general problem that can be handled by each of the four programs. More general problems can be solved, but they must be reducible to the form given here.

A. General Problem for LAPLACE

Consider a two-dimensional region, G, in the x-y plane with boundary S. Suppose we wish to solve for u(x,y), satisfying

$$\frac{\partial^2 u(x,y)}{\partial x^2} + \frac{\partial^2 u(x,y)}{\partial y^2} = 0 \qquad \text{in } G \quad ,$$

$$u = f \qquad \text{on } S_D \quad ,$$

and

$$\frac{\partial u}{\partial n} = g \qquad \text{on } S_N \quad .$$

Here $S_D \cup S_N = S$, which is the boundary of G, and $S_D \cap S_N$ is, at most, a finite number of points (for example, corners of S). S_N or S_D can be empty. We assume that S has a parametric representation $\{[x(t), y(t)] \mid t\epsilon(0,d)\}$ with respect to arc length. Here d is the length of S. Moreover, we assume that both x(t) and y(t) are piecewise smooth. The region G may be finite or infinite, but LAPLACE assumes that the boundary S is finite in length. The method can be extended to include infinitely long boundaries. Finally, Eq. (1) is not true, in general, for infinite regions G unless $u(x,y) \to 0$ as $(x,y) \to \infty$. For the other three programs (LAPLARS, LAPLDDC, and LAPLDRS), the assumptions concerning the parametric representation and infinite regions must also be true.

B. General Problem for LAPLARS

Essentially the same problems that can be solved with LAPLACE can be solved with LAPLARS. Using axial symmetry, we shall assume that the problem has been reduced from a problem in (z,r,ϕ) coordinates to a problem in (z,r) coordinates. Thus, only the z-r plane need be considered. Moreover, using the symmetry again, the problem will be restricted to the upper half-plane $\{(z,r) \mid r \geq 0\}$. Note that, in all diagrams that follow, the z-axis corresponds to what is normally the x-axis, and the r-axis corresponds to the y-axis. Now let G be a connected region in the upper half of the z-r plane, and let S be the boundary of G. Suppose we wish to solve for u(z,r) satisfying

$$\frac{\partial^2 u}{\partial r^2} + \frac{1}{r}\frac{\partial u}{\partial r} + \frac{\partial^2 u}{\partial z^2} = 0 \qquad \text{in } G \quad ,$$

$$u = f \qquad \text{on } S_D \quad ,$$

and

$$\frac{\partial u}{\partial n} = g \qquad \text{on } S_N \quad .$$

Here $S_D \cup S_N = S \cap \{(z,r) \mid r > 0\}$, and $S_D \cap S_N$ is, at most, a finite number of points. Notice that S_D and S_N do not contain parts of the z-axis. Because of the axial symmetry, the condition $\partial u/\partial n = -(\partial u/\partial r) = 0$ must always be satisfied on the z-axis.

C. General Problems for LAPLDDC

The physical problems that can be treated by LAPLDDC are given in Ref. 2, p. 391. For instance, one might have an electrostatic problem involving two or more different materials. The materials might be conductors, or might have different dielectric constants. Another possibility is a magnetic field problem with materials of different permeability. Now consider the general problem. In what follows, the choice of two σ's is for ease of explanation only. Let G be a connected region in the x-y plane. Suppose that G_1 and G_2 are connected subregions such that $G = G_1 \cup G_2$, and $G_1 \cap G_2$ is empty. Let S_1 be the boundary of G_1 and S_2 be the boundary of G_2. Define $C = S_1 \cap S_2$ to be the boundary common to G_1 and G_2. See Fig. 1. Let σ_1 be associated with G_1, and σ_2 with G_2. Define $\partial/\partial n_1$ to be the exterior normal derivative on $S_1 \cap C$, that is, exterior to G_1, and define $\partial/\partial n_2$ to be the exterior normal derivative on $S_2 \cap C$, that is, exterior to G_2. At every point of C, $\partial/\partial n_1 = -(\partial/\partial n_2)$. Finally, let S_D and S_N be such that $S-C = S_N \cup S_D$, and $S_D \cap S_N$ is, at most, a finite number of points. The problem is to find u(x,y), defined on G and $S = S_1 \cup S_2$, such that

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \qquad \text{in } G - C \quad ,$$

$$u = f \qquad \text{on } S_D \quad ,$$

$$\frac{\partial u}{\partial n} = g \qquad \text{on } S_N \quad ,$$

$$\lim_{\substack{(x,y)\to(x_o,y_o)\\(x,y)\in G_1}} u(x,y) = \lim_{\substack{(x,y)\to(x_o,y_o)\\(x,y)\in G_2}} u(x,y)$$

for each point $(x_o,y_o)\in C$, and

$$\sigma_1 \frac{\partial u(x_o,y_o)}{\partial n_1} = -\sigma_2 \frac{\partial u(x_o,y_o)}{\partial n_2} \quad \text{for each point } (x_o,y_o)\in C.$$

When, G_1 for instance, is empty, this problem reduces to the general problem stated for LAPLACE. If the stated problem were electrostatic, the σ's would correspond to the dielectric constants of the various materials. The boundary conditions on C are called the matching boundary conditions.

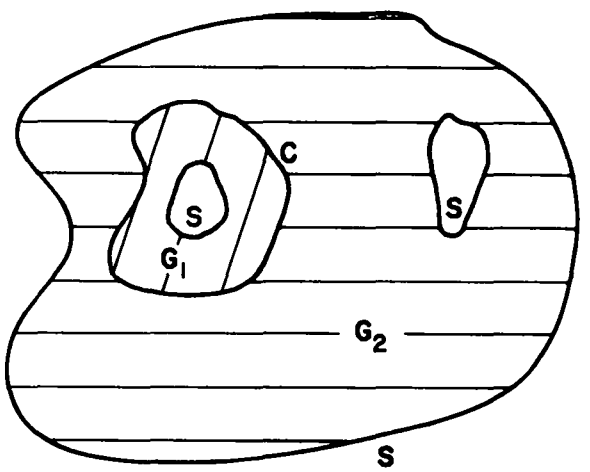

Fig. 1. Possible configuration for G.

D. General Problem for LAPLDRS

The general problem for LAPLDRS is the same as that for LAPLDDC, except that G must lie in the upper half of the z-r plane, and the partial differential equation corresponds to the axially symmetric case.

III. INPUT

In this section we discuss the input for the four programs. The input is slightly different from that given by Hayes,[3] but it is almost exactly the same for the four programs considered here. Assuming a compatible problem, the same input will work for each of the four programs. For programs (i) and (ii) there are two allowable boundary conditions: the Dirichlet boundary condition, i.e., u given, or the Neumann boundary condition, i.e., $\partial u/\partial n$ given. For problems in which only the Neumann condition is given, the solution to the general problem is unique only up to an additive constant. To make the solution to the interior Neumann problem unique, the approximate solution u_A is made to satisfy $\int_S u_A\, dS = 0$, for programs (i) and (iii), and

$$\int_S u_A r\, dS = 0 \quad ,$$

for the axially symmetric programs. For exterior problems, we assume that $u(v,w) \to 0$ as $(v,w) \to \infty$. This assumption makes the exterior Neumann problem uniquely solvable. However, using programs (i) and (iii), arbitrary Dirichlet boundary conditions cannot be imposed for the exterior problem because of this assumption.

To use the programs in nonhomogeneous media, we have another possible boundary condition, called a matching boundary condition. On the common boundary, the solution, u, must be continuous and the normal derivative must satisfy given jump conditions.

A. Boundary Description

Two different types of boundaries can be given as input. The first is called the regular boundary to distinguish it from the second. The regular boundary includes all of the common boundary and S_N. It also may contain all or part of S_D. The regular boundary must always be a finite number of closed curves. For programs (ii) and (iv), a closed curve can also be a curve beginning and ending on the z-axis. For instance, see Fig. 4. Moreover, boundary values given on the regular boundary always refer to one side of the boundary. For instance, the boundary value problem might be the interior of a circle or the exterior of a square.

For some other boundaries it is convenient and even desirable to assume that the region enclosed by a portion of the boundary is infinitely thin. For instance, in electrostatics, if one is setting the potential on a piece of foil, it is not unreasonable to assume that the foil is infinitely thin, if its actual thickness is very small compared to its other dimensions.

The second type of boundary encloses a region that is assumed to be infinitely thin, and is, therefore, called a thin-plate boundary. The values of the potential are assumed to be given on both sides of a thin-plate boundary. Moreover, the potential is

assumed to be the same on both sides. For this reason, thin plates must not form a closed curve. Thin plates can be used only with the Dirichlet boundary conditions.

1. Simplified Boundary Data. The boundary and boundary values can be given as input in two different ways. We will discuss the more general method later. For the simpler method, the boundary S is approximated by line segments, circular arcs, and complete circles. The boundary values are assumed to be constant on each section of the boundary, S. This type of input is satisfactory for most physical problems. A line segment is described by its two end points. The two end points and any interior point are used to describe a circular arc. For a complete circle, one must give the coordinates of the center of the circle, the radius of the circle, and the orientation. The orientation determines whether G is interior or exterior to the circle. A +1 denotes a positive orientation in which case G is interior to the circle, and -1 denotes a negative orientation, in which case G is exterior to the circle.

The orientation of the other boundary sections is implicit in the input. The region G must be to the left of the regular part of the boundary as one follows the curve S in the direction given in the input. The direction of the curve is determined by the order of the points used to describe line segments and circular sections and is from the first end point to the second end point. Thus, interior regions give the boundary S a positive (counterclockwise) orientation, and exterior regions give it a negative (clockwise) orientation.

All of the input used to describe the simple boundary conditions is given on data cards, each of which refers to a part of the boundary which is called a boundary section. Each card has ten fields. We will discuss only the first seven fields now. Each of the first seven fields is ten characters in length and is read with a 7E10.0 format. If the boundary section is not in the common boundary, the seventh field, i.e., columns 61 to 70, is used for the boundary value, be it Neumann or Dirichlet. If the boundary section is in the common boundary, then the seventh field is used to give the value of σ for the region on the left of the boundary section.

We discuss here only programs (i) and (iii); for programs (ii) and (iv), one simply replaces x-coordinates by z-coordinates and y-coordinates by r-coordinates. If a card is used to describe a line segment, the first four fields are used for the x- and y-coordinates of the first end point and for the x- and y-coordinates of the second end point, in that order. The fifth and sixth fields must be left blank. If a card is used to describe a circular arc, the first six fields are used to give the x- and y-coordinates of the first end point, of the interior point, and of the second end point, in that order. If a data card is used to describe a complete circle, the first two fields are used for the x- and y-coordinates of the center of the circle, the third and sixth fields must be left blank, and the fourth and fifth fields are used to give the radius and orientation, respectively. For data cards used to describe circular arcs and line segments, one can leave the first two fields blank if the first end point is the same as the second end point of the previous card. The data cards are printed as they are interpreted. A blank card must precede and follow the boundary data cards.

We will now consider four problems with simple boundary input. See Fig. 2 for the first example.

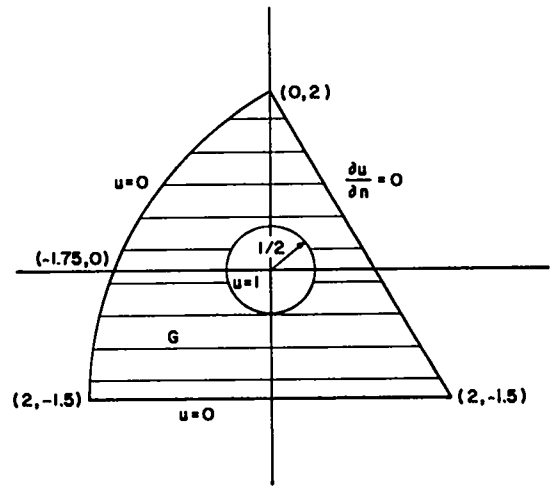

Fig. 2. Sample problem using the three types of boundary sections.

Here we want to find $u(x,y)$ satisfying $u_{xx} + u_{yy} = 0$ on G. G is the region between the circle and the three-sided section in Fig. 2. The boundary conditions are assumed to be $u = 1$ on the circle, $\partial u/\partial n = 0$ on the line segment $\overline{(2,-1.5)(0,2)}$, and $u = 0$ on the rest

of the boundary. Ignoring the information in columns 71 to 80, the first seven fields for the boundary input cards for this problem could be as follows.

80 COLUMN ENTRY

PROGRAMMER PROBLEM DATE PAGE OF

1–5	6–10	11–15	16–20	21–25	26–30	31–35	36–40	41–45	46–50	51–55	56–60	61–65	66–70	71–73	73–80
	2.		-1.5		0.		2.						0.	2	
	0.		2.		-1.75		0.		-2.		-1.5		0.	1	
	-2.		-1.5		2.		-1.5						0.	1	
	0.		0.				.5		-1.				1.	1	

The first two fields of the second and third cards could have been left blank.

In Fig. 2 the orientation of the circle is negative, and that of the three-sided section is positive. This is because G is interior to the three-sided section, but exterior to the circle. Care should be taken in deciding the correct orientation. Errors in orientation are easy to make and difficult to detect. Orientation is not used with thin plates because the region G is always exterior to a thin-plate boundary. In the example shown in Fig. 3, G is all of the interior of the circle except the line segment $\overline{(0,0)(10,0)}$. The boundary conditions are u = 1 on the circle and u = 0 on the line segment $\overline{(0,0)(10,0)}$. Ignoring the information in columns 71 to 80, the boundary input cards for this problem are:

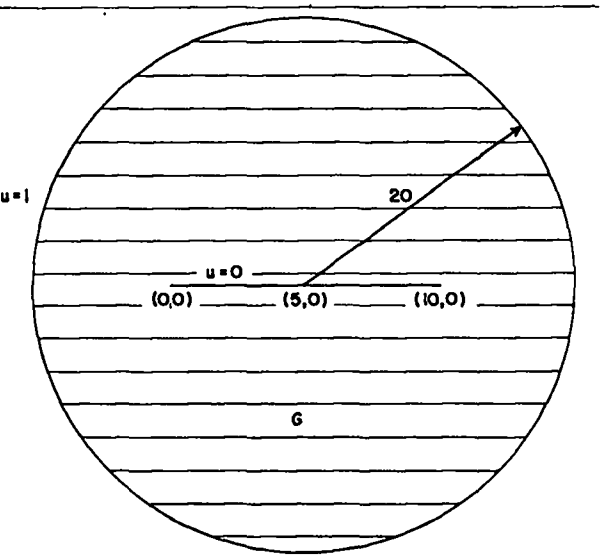

Fig. 3. Thin-plate boundary problem.

80 COLUMN ENTRY

PROGRAMMER PROBLEM DATE PAGE OF

1–5	6–10	11–15	16–20	21–25	26–30	31–35	36–40	41–45	46–50	51–55	56–60	61–65	66–70	71–73	73–80
	0.		0.		10.		0.						0.	3	21
	5.		0.				20.		1.				1.	1	31

At present, programs (iii) and (iv) are not capable of handling thin plates.

We will next consider an axially symmetric problem. Physically the problem corresponds to a region between a sphere and a right circular cylinder. The cylinder encloses the sphere, and the center of the sphere lies on the axis of the cylinder. On the cylinder the potential is 1, and on the sphere it is 0. See Fig. 4 for a diagram in the z-r plane. In this problem both curves are closed in our generalized sense. This is because the projection on the whole z-r plane would form closed curves. Ignoring columns 71 to 80, the boundary input cards would be

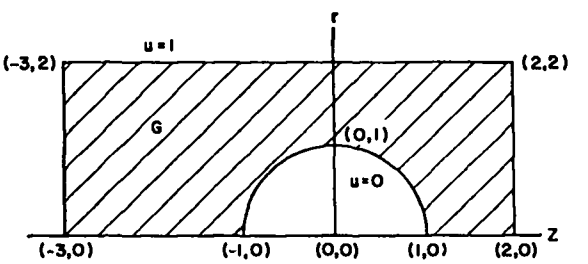

Fig. 4. Intersection of a sphere and a cylinder with the upper half-plane.

80 COLUMN ENTRY

PROGRAMMER PROBLEM DATE PAGE OF

1 5	6 10	11 15	16 20	21 25	26 30	31 35	36 40	41 45	46 50	51 55	56 60	61 65	66 70	71 73	80
	2.		0.		2.		2.						1.	1	
	2.		2.		-3.		2.						1.	1	
	-3.		2.		-3.		0.						1.	1	
	-1.		0.		0.		1.		1.		0.		0.	1	

In this type of problem it is easy to make an error in orientation. For the second and third cards, the first two fields could have been left blank.

In all of the above problems, the boundary input cards could have been put in any order. This is not true, in general, for problems in inhomogeneous media. For programs (iii) and (iv), the whole problem is considered as a sum of connected subproblems over subregions. Each of the subproblems is complete in the sense that if the potential on the common boundary were known, it could be determined independently on each subregion. This means that each section of the common boundary must be given as input twice, once for each of the two subproblems for which it is part of the boundary. To preserve the orientation of the boundary, the direction along the boundary sections is opposite for the two inputs.

The subproblems must be numbered, and the boundary data cards for each subproblem must be consecutive. The order of the cards within the subproblem is arbitrary. Columns 78 to 80 are used to give the number of the subproblem. The field is read with an I3 format. The numbering should be consecutive, starting with one. For clarification, consider the example shown in Fig. 5. This diagram is not drawn to scale, of course. It is an approximation to an example discussed in Sec. 4.03 of Ref. 4. The physical problem approximated is an infinite conducting cylinder surrounded by a layer of material of dielectric strength 5 in a uniform field of strength 1. To approximate the behavior in the conductor, we use a value of σ that is very large compared to the other values of σ. Ignoring columns 71 to 76, the boundary data cards are listed on the next page.

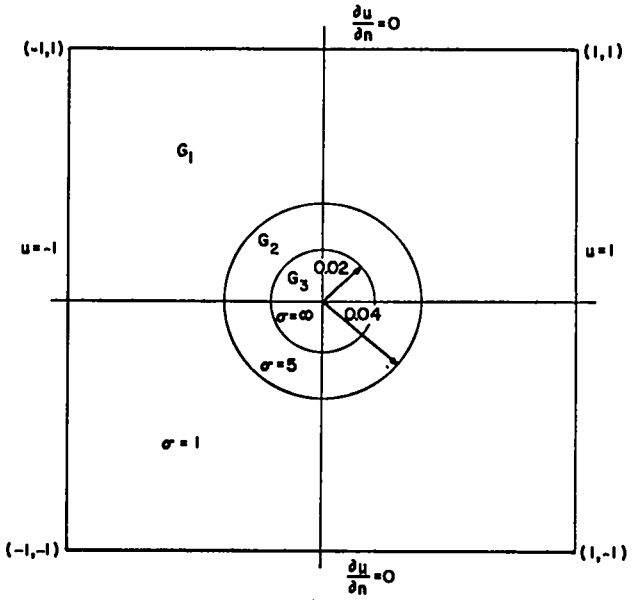

Fig. 5. Conducting and dielectric cylinders in a uniform field.

Some numerical results for this problem are given in Sec. V. If, instead of being a conductor, the inside cylinder were air, the data cards would be the same, except for the last card which would have a value of 1 instead of 1.E6 in the seventh field.

On each boundary data card, one must give two more input quantities. Column 72 is the eighth field, which is for the variable LT that determines the type of boundary condition on the boundary section referenced by the card. If LT = 1, the boundary is regular and u is given. If LT = 2, the boundary is regular and $\partial u/\partial n$ is given. If LT = 3, the boundary is thin-plate and u is given. If LT = 7, the boundary section is in the common boundary. These are the only four possibilities at present. Each of the examples has the variable LT given on the boundary data card.

80 COLUMN ENTRY

PROGRAMMER | PROBLEM | DATE | PAGE OF

1.	1.	1.	-1.	1.			0.	2	15 1
-1.	1.	1.	-1.	-1.			-1.	1	15 1
-1.	-1.	1.	1.	-1.			0.	2	15 1
1.	-1.	1.	1.	1.			1.	1	15 1
0.	0.	.04	-1.				1.	7	25 1
0.	0.	.04	1.				5.	7	25 2
0.	0.	.02	-1.				5.	7	25 2
0.	0.	.02	1.				1.E4	7	25 3

Columns 74 to 76 form the ninth field on the card. The ninth field is used to give the integer variable KV that tells how many points are to be used in approximating u and $\partial u/\partial \nu$ on a given boundary section. KV should be odd and greater than or equal to 3. If the value of KV given as input is less than 3, then KV is assumed to be 3. If the value of KV given as input is even, then KV is set equal to KV - 1. If a boundary section is in the common boundary, then the value of KV used for that section should be the same as the value of KV used for the other corresponding section in the common boundary. If the values are not the same, then the value of KV first given as input will be used for both of the sections. The variation in the boundary values, the shape of the boundary section, the desired accuracy, and the computer time available all determine KV; no set rule can be given. The user can try different values of KV on each boundary section; the change in the solution usually indicates the error. In the example in Fig. 3, we assumed that 31 approximation points on the circle and 21 on the line segment would give sufficient accuracy. The two boundary data cards have KV given.

Define NDCT to be the total number of boundary sections, and set

$$NT = \sum_{i=1}^{NDCT} KV_i \quad .$$

To compute the unknown boundary values, one must solve a matrix problem of the form $Az = b$, where b is a known vector of length NT and A is an NT x NT matrix. For $NT \leq 192$, the matrix can be stored in core on a 65,536-word machine. For very large problems, however, the matrix A must be stored on an external device. The extended core storage (ECS) on the CDC 6600 is convenient for this purpose. If the user wants to use the ECS to store the matrix A, he must request $(NT)^2 + 1500$ words of ECS. For the in-core version, the core itself simulates ECS. The reads and writes to the ECS are replaced by statements that move the data to another part of the core. The user must dimension a variable called A to be of at least $(NT)^2 + 1500$ length. The dimension statement must be placed at the start of the FORTRAN subroutine ECRD. ECRD also handles the ECS reads and writes when ECS is used. The size of the matrix A determines the largest allowable problem. At present it is assumed that $NT \leq 630$. By modifying the dimensions of the variables discussed in Sec. XI, one can raise or lower this figure.

2. Generalized Boundary Data. The subroutine BDRY is used for generalized boundary input. Whenever the simple boundary input is the only type used, BDRY shows up as an unsatisfied external. This is a normal procedure. The generalized boundary input can be used in conjunction with the simple boundary input, or it can be used as the only source of boundary data input. To use the generalized boundary input option, the user must write a subroutine that must be called BDRY. In what follows we will discuss programs (i) and (iii); the input for the other two programs is the same.

Assume that we have a boundary section, S_I, on which we wish to have variable boundary values. Let S_I have the parametric representation

$$\{[x_I(t), y_I(t)] | t\epsilon(0, LS_I)\}$$

with respect to src length. Here LS_I is the length

of the boundary section S_I. To keep the orientation correct for the regular boundary sections, the parametric representation must be such that as we traverse S_I with increasing t, the region remains on the left. An equivalent condition is that $[dy_I(t)/dt, -dx_I(t)/dt]$ forms the unit exterior normal vector. We must be able to compute $x_I(t)$, $y_I(t)$, $dx_I(t)/dt$, $dy_I(t)/dt$,

$$\frac{1}{2}\left[\frac{dx_I(t)}{dt}\frac{d^2y_I(t)}{dt^2} - \frac{dy_I(t)}{dt}\frac{d^2x_I(t)}{dt^2}\right] \quad ,$$

and the appropriate boundary value as a function of t for $t\epsilon(0, LS_I)$.

BDRY has the formal parameters (I, T, X, Y, XP, YP, F, V). I and T are input to BDRY. BDRY must give as output

$$X = X_I(T) \quad ,$$

$$Y = Y_I(T) \quad ,$$

$$XP = dX_I(T)/dT \quad ,$$

$$YP = dY_I(T)/dT \quad ,$$

and

$$V = \frac{1}{2}\left[\frac{dX_I(T)}{dT}\frac{d^2Y_I(T)}{dT^2} - \frac{dY_I(T)}{dT}\frac{d^2X_I(T)}{dT^2}\right] \quad ,$$

and the variable F must contain the appropriate boundary value. The boundary value can be

$$u[X_I(t), Y_I(t)] \text{ or } \frac{\partial}{\partial \nu} u[X_I(T), Y_I(T)] \quad .$$

Each boundary section with generalized boundary input must have a boundary data card. The first six fields must be left blank, and the last three fields must contain the same information as that for the simplified boundary input. The seventh field must contain the length of the boundary section. In this case, it will contain LS_I.

To clarify, consider the following example to be used as input for program (i). Let G be the square centered at the point (0,1) with sides of length 2. See Fig. 6. Let S_1 be the line segment from (1,0) to (1,2); S_2, that from (1,2) to (-1,2); S_3, that from (-1,2) to (-1,0); and S_4, that from (-1,0) to (1,0).

Assume that the boundary conditions are

$$\frac{\partial u}{\partial n} = 20(y-y^3) \quad \text{on } S_1 \quad ,$$

$$\frac{\partial u}{\partial n} = 5x^4 - 120x^2 + 80 \quad \text{on } S_2 \quad ,$$

$$u = 5y - 10y^3 + y^5 \quad \text{on } S_3 \quad ,$$

and

$$u = 0 \quad \text{on } S_4 \quad .$$

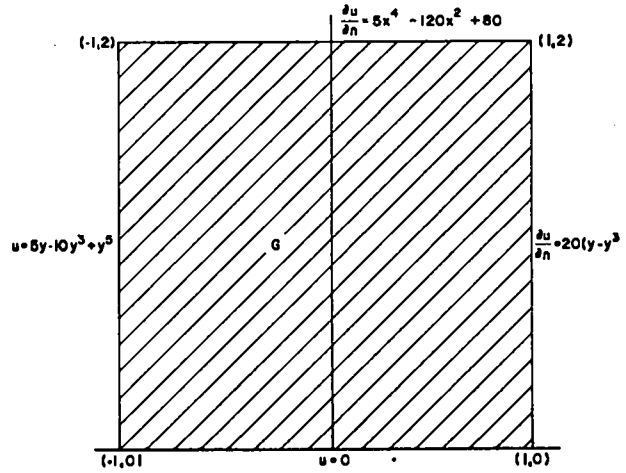

Fig. 6. Variable boundary data problem.

The actual solution for this problem is $u(x,y) = 5x^4y - 10x^2y^3 + y^5$. For the boundary section S_4, the simple type of boundary input can be used. To generate the input for the boundary sections S_1, S_2, and S_3, the subroutine BDRY must be used. One possible such routine is as follows.

PROBLEM DATE PAGE OF PROGRAMMER

C ← For comment / Statement number (1–5)	Continuation (6)	FORTRAN STATEMENT (7–72)	Identification (73–80)
		SUBROUTINE BDRY(I,T,X,Y,XP,YP,F,V)	
		V=0.	
		GOTO(1,2,3),I	
1		X=YP+1.	
		XP=0.	
		Y=T	
		F=20.*(Y-Y**3)	
		RETURN	
2		X=1.-T	
		XP=-1.	
		Y=2.	
		YP=0.	
		F=5.*X**4-120.*X**2+80.	
		RETURN	
3		X=YP=-1.	
		XP=0.	
		Y=2.-T	
		F=5.*Y-10.*Y**3+Y**5	
		RETURN	
		END	

If KV = 37 on each side of the square, then the boundary data cards for this problem are:

80 COLUMN ENTRY

PROGRAMMER PROBLEM DATE PAGE OF

1–5	6–10	11–15	16–20	21–25	26–30	31–35	36–40	41–45	46–50	51–55	56–60	61–65	66–70	71–72	73–80
													2.	2	37
													2.	2	37
													2.	1	37
	-1.		0.		1.		0.						0.	1	37

In some cases it is difficult, if not impossible, to compute x(t) and y(t) explicitly, as when the boundary curve is a section of a parabola or an ellipse. However, in these cases one can usually get an implicit expression and use it to compute x(t), y(t), and the desired derivatives. The speed of BDRY is not too important because the routine is called, at most, a few thousand times.

B. Superposition of Solutions and Magnetic Field Problems

Another option is available. Suppose we wish to find a function u(x,y) that does not satisfy all of the requirements for the general problem. For instance, the solution u(x,y) might have discontinuous boundary values or branch cuts. Suppose further, that we have a function v(x,y) such that u(x,y) - v(x,y) does satisfy the requirements for the general problem. Usually u(x,y) - v(x,y) has involved boundary conditions, and we prefer not to set up problems with such conditions. Using the following option, we can give the boundary conditions for u as input, if the function v and its first derivatives can be given as a function of x and y. The real function BC is used to generate branch cuts for magnetic fields that we will discuss later in this section. In this subroutine are two comment cards. After the first card, we must insert a statement,

$$BC = BC - V(X,Y) \quad ,$$

where V(X,Y) is the function, v, evaluated at X and Y. After the second comment card, we must insert a statement,

$$BC = BC - \frac{\partial V(X,Y)}{\partial X} * YP + \frac{\partial V(X,Y)}{\partial Y} * XP \quad .$$

Thus, v(x,y) will be subtracted from all input boundary data so that the programs can solve for u(x,y) - v(x,y), and then v(x,y) will be added to all output so that we get an approximate solution to u(x,y).

The following examples show how this option can be used. Consider again the example given in Fig. 5. The outside square was used to approximate the effect of a uniform electric field of strength 1. By using superposition with the function v(x,y) = x, we can put in the effect of the electric field directly and eliminate the need for the square boundary. The two statements needed for BC are

```
BC = BC - X
BC = BC - YP
```

The other input is the same except that the first four boundary data cards are not used. We also need to modify two statements in two subroutines. The modification is described later in this section.

Suppose we now want to solve for u(x,y) in the unit circle satisfying $\partial^2 u/\partial x^2 + \partial^2 u/\partial y^2 = 4$ with u = 1 on the boundary. In this example v is not unique. Arbitrarily choose $v = 2x^2$. The statements needed for BC are

```
BC = BC - 2.*X**2
BC = BC - 4.*X*YP
```

The other input for u is simply the boundary data card for the circle.

The application of these programs to Poisson's equation is difficult for the general problem. Suppose we wish to solve for u(x,y) satisfying

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x,y) \qquad \text{in } G$$

plus boundary conditions. This problem can be reduced to Laplace's equation by using the particular solution

$$v(x,y) = \frac{1}{2\pi} \int_G \int f(s,t) \ln\left[(x-s)^2+(y-t)^2\right]^{\frac{1}{2}} ds\, dt \quad .$$

The double integral can be difficult to deal with numerically. However, for many problems this technique is feasible and the problem can be solved. The author knows of one physical problem for which this is the only practical solution.

Next, we will discuss magnetic field calculations. Program (iii) can be easily modified to compute the scalar potential for a large class of problems. Program (iv) can also be modified to compute the scalar potential for axially symmetric problems, but the modification is more difficult because one must write an axially symmetric version of the program BC discussed in Sec. XI. The scalar potential is discussed in Sec. 7.28 of Ref. 4. The function u(x,y) here corresponds to the function Ω in Ref. 4.

Suppose we have a series of subregions G_1, G_2, . . . , G_{NRE} having constant permeabilities μ_1, μ_2, . . . , μ_{NRE}. Suppose we also have a number of point current sources of magnitude I_1, I_2, . . . , I_{NBC} at the points (X_1,Y_1), (X_2,Y_2), . . . , (X_{NBC},Y_{NBC}). The sign convention for the currents will be determined by Biot and Savart's law. See Sec. 7.14 of Ref. 4. For a single point source conductor, a positive I means a positive current of magnitude I into the paper, and this, in turn, gives B a clockwise direction. A negative I means a positive current out of the paper, which, in turn, gives B a counterclockwise direction. Let

$$G = \bigcup_{j=1}^{NRE} G_j$$

and let C be the common boundary in G. G will usually be all of the x-y plane. The problem is to find u(x,y) such that

$$\frac{\partial^2 u(x,y)}{\partial x^2} + \frac{\partial^2 u(x,y)}{\partial y^2} = 0 \qquad \text{in } G - C \quad ,$$

and for any closed curve, Γ,

$$\oint_\Gamma \nabla u \cdot d\Gamma = I_\Gamma \quad ,$$

where I_Γ is the algebraic sum of the currents inside Γ. Also, the function u(x,y) must satisfy the matching boundary conditions on C. Thus, if $(x_o,y_o) \epsilon C$, and G_i is on one side of C at (x_o,y_o) and G_j is on the other side of C at (x_o,y_o), then

$$\lim_{\substack{(x,y)\to(x_o,y_o)\\(x,y)\in G_i}} u(x,y) = \lim_{\substack{(x,y)\to(x_o,y_o)\\(x,y)\in G_j}} u(x,y)$$

and

$$\mu_i \frac{\partial u(x_o,y_o)}{\partial \nu_i} = -\mu_j \frac{\partial u(x_o,y_o)}{\partial \nu_j} \quad .$$

As the problem is stated, it does not satisfy the requirements for the general problem. Superposition must be used to satisfy the requirement with the integral. It is easy to construct a function v(x,y) such that

$$\oint_\Gamma \nabla v \cdot d\Gamma = I_\Gamma$$

for any closed curve, Γ, and $\Delta v = 0$. In fact,

$$v(x,y) = \frac{1}{2\pi} \sum_{k=1}^{NBC} I_k \tan^{-1}\left(\frac{y-y_k}{x-x_k}\right) \quad ,$$

with $-\pi < \tan^{-1}(\cdot) < \pi$. However, such a v does not, in general, satisfy the matching boundary conditions. Let $u_1 = u - v$. Obviously, then

$$\Delta u_1 = 0$$

and

$$\oint_\Gamma \nabla u_1 \cdot d\Gamma = 0$$

for any closed curve Γ. Moreover, it can be assumed that u - v is continuous across C even though v has branch cuts. However, for $(x_o,y_o)\in C$,

$$\mu_i \frac{\partial u_1}{\partial \nu_i} = \mu_i \left(\frac{\partial u}{\partial \nu_i} - \frac{\partial v}{\partial \nu_i}\right) = \mu_i \frac{\partial u}{\partial \nu_i} - \mu_i \frac{\partial v}{\partial \nu_i}$$

$$= -\mu_j \left(\frac{\partial u_1}{\partial \nu_j} + \frac{\partial v}{\partial \nu_j}\right) - \mu_i \frac{\partial v}{\partial \nu_i}$$

$$= -\mu_j \frac{\partial u_1}{\partial \nu_j} + \frac{\partial v}{\partial \nu_i} (\mu_j - \mu_i) \quad .$$

Program (iii) must be modified to handle this different boundary condition.

In Sec. XI we discuss the subroutines used by program (iii). The subroutines ROWSTOR and REGNSEL must be modified to handle this change in type of boundary condition. There are comment cards in both subroutines telling where the modifications go. The change involves the insertion of two statements. In REGNSEL, we insert the statement

```
IF(IC(J).NE.0)D(L) =
  D(L)+BCN(X(L),Y(L),YN(L),XN(L))*(1.+RA(J))  .
```

If ROWSTOR, we insert the statement

```
IF(IC(I).NE.0)FT(IG) =
  FT(IG)-BCN(X(M),Y(M),YN(M),XN(M))*(1.+RA(I))*CN(M).
```

This modification can be used for other problems where superposition is used and the matching boundary conditions are not satisfied. These two modifications are inefficient. For large numbers of current sources, one should use a table lookup for the modification to ROWSTOR.

Consider the input needed for point current sources. There is one data card for each current source. The format used to read the cards is FORMAT(3E10.0). The first field, that is columns 1 to 10, is used for the x-coordinate. The second field is used for the y-coordinate, and the last field is used for the current. The data cards for the currents must go in front of the blank card preceding the boundary data cards. The following example (Fig. 7). is a specific case of the example given in Sec. 7.26 of Ref. 4.

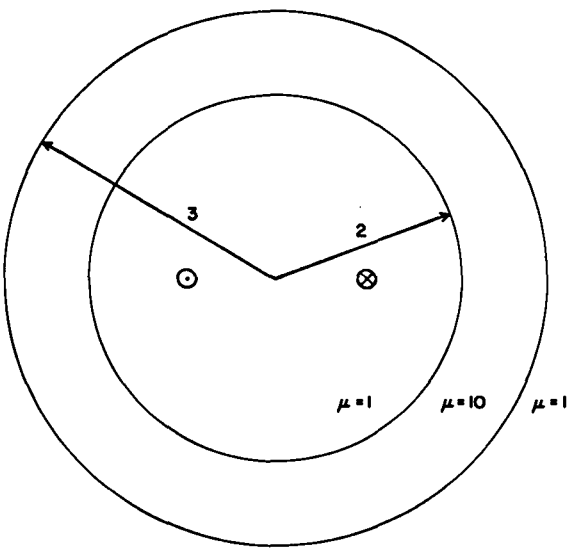

Fig. 7. Magnetic shielding of bifilar circuit.

This example corresponds to two infinite wires shielded by a cylinder of unit thickness and carrying a unit current in opposite directions. Here it is assumed that the cylinder has permeability 10. The following current data cards and boundary data cards

were used for this problem.

80 COLUMN ENTRY

PROGRAMMER PROBLEM DATE PAGE OF

1 5 6	10 11	15 16	20 21	25 26	30 31	35 36	40 41	45 46	50 51	55 56	60 61	65 66	70 71	73 80
Current data cards.														
	1.		0.		1.									
	-1.		0.		-1.									
Boundary data cards.														
	0.		0.				2.		1.				1.	7 25 1
	0.		0.				2.		-1.				10.	7 25 2
	0.		0.				3.		1.				10.	7 25 2
	0.		0.				3.		-1.				-1.	7 25 3

the boundary $y = 15$, $-\pi/2 \leq x < \pi/2$. The problem was run twice. The inputs are as follows.

The exact solution for subregion 3 of this problem is given by Smythe.[4] The computed and exact solutions are each given at a few randomly chosen comparison points in Table I.

Table I

RESULTS FOR THE PROBLEM OF FIGURE 7

x	y	Computed B_x	True B_x	Computed B_y	True B_y
4.	0.	$2. \times 10^{-8}$	0.	9.815×10^{-3}	9.826×10^{-3}
3.	4.	-5.646×10^{-3}	-5.652×10^{-3}	-1.824×10^{-3}	-1.823×10^{-3}
0.	6.	$-1. \times 10^{-8}$	0.	-4.072×10^{-3}	-4.077×10^{-3}
-8.	-7.	-1.315×10^{-3}	-1.316×10^{-3}	1.673×10^{-4}	1.675×10^{-4}

C. Nonsmooth Boundary Values

One of the basic assumptions of this method is that the boundary values have piecewise continuous third derivatives. It seems that most engineering problems of interest do not satisfy this assumption. Figure 3 shows such a problem. The singularity in this problem is similar to the one discussed by Whiteman,[5] and to the problem shown in Fig. 8. We will discuss two ways to minimize the figure loss due to nonsmooth boundary values.

The simplest way to minimize errors due to singularities is to distribute the points in the calculation so as to minimize the effect of the singularity. As an example, consider the first test case given by Schultz.[6] The example is in the x-y plane and is shown in Fig. 8. The change in boundary conditions at the origin is such that $\partial u/\partial n$ will not exist at that point. Because the boundary of the problem is infinite, the problem cannot be run exactly as it is. A reasonable approximation for this problem is to truncate the region at $y = 15$ and impose the boundary condition $u = 15. + \log(2.)$. along

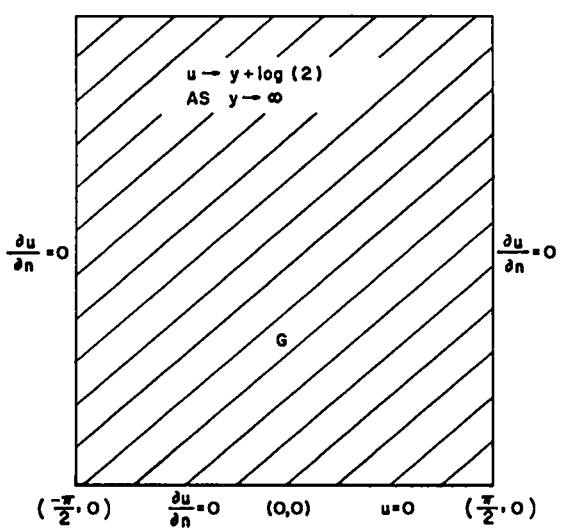

Fig. 8. Schultz's[6] first example.

80 COLUMN ENTRY

PROGRAMMER PROBLEM DATE PAGE OF

1–5	6–10	11–15	16–20	21–25	26–30	31–35	36–40	41–45	46–50	51–55	56–60	61–65	66–70	71	73–80
First approximation															
1.57	0.7963		0.	1.57	0.7963		15.						0.	2	59
				-1.57	0.7963		15.					15.6	93147	1	11
				-1.57	0.7963		0.						0.	2	59
					0.		0.						0.	2	23
				1.57	0.7963		0.						0.	1	23
Second approximation															
1.57	0.7963		0.	1.57	0.7963		15.						0.	2	59
				-1.57	0.7963		15.					15.6	93147	1	11
				-1.57	0.7963		0.						0.	2	59
					-0.7		0.						0.	2	11
					0.		0.						0.	2	11
					0.7		0.						0.	1	11
				1.57	0.7963		0.						0.	1	11

The only difference between the two inputs is the distribution of points along the x-axis. Some representative results are shown in Table II.

Table II

RESULTS FOR THE PROBLEM OF FIGURE 8

x	y	First approximation to u	Second approximation to u	True solution
$-\frac{\pi}{4}$	$\frac{\pi}{8}$	1.5938	1.6071	1.6090
$-\frac{3\pi}{8}$	$\frac{\pi}{8}$	1.7519	1.7645	1.7662
0	$\frac{\pi}{2}$	2.2436	2.2515	2.2530
$\frac{\pi}{4}$	π	3.7967	3.8030	3.8042

Redistributing the points has some disadvantages. In the first place, the optimal distribution of points is difficult, if not impossible, to determine. Even a good distribution of points is difficult to compute. In the second approximation to the above problem, the distribution of points is nowhere near optimal. Moreover, if the distribution of points is involved, the input is involved, and there is a greater chance for errors. Also, there is a tendency to put more points in the calculation, and this is time consuming. Finally, redistributing the points merely minimizes the error due to the singularity. The solution may be good in the region of interest, but the computed unknown boundary values near the singularity are sometimes nonsense.

A second way to minimize the figure loss due to a singularity is to subtract the singularity. This method, discussed by Fox,[7] can also be used for discontinuous boundary values given in the input. Essentially, the method reduces to finding the behavior of the solution at the singularity and then subtracting the singularity.

Consider the truncated version of the problem shown in Fig. 8. Using the formula in Ref. 7, p. 303, it follows that in polar coordinates the solution, u, has the form

$$u(r,\theta) = \sum_{n=1}^{\infty} b_n \, r^{n/2} \sin\left(\frac{n\theta}{2}\right) \quad . \tag{2}$$

Thus, $\partial u/\partial n$ is not well-defined at the origin if $b_1 \neq 0$. The term corresponding to $n = 3$ can also cause trouble, but we will not discuss it. To eliminate the singularity corresponding to the term $b_1 r^{\frac{1}{2}} \sin(\frac{\theta}{2})$, we need only use the option for superposition of solutions with

$$v(x,y) = b_1 \left[\frac{(x^2+y^2)^{\frac{1}{2}} - x}{2}\right]^{\frac{1}{2}} = b_1 \, r^{\frac{1}{2}} \sin(\theta/2) \quad .$$

Of course, b_1 must be found first, but that is easy because the program itself can be used to approximate the solution, u, and near the origin the term $b_1 \, r^{\frac{1}{2}} \sin(\theta/2)$ dominates all other terms in the series expansion.

This problem was run using the same input as that used in the first approximation of the same problem. Using the approximate solution, u_1, from this input and the relation

$$u_1(0.,0.06) \approx b_1(0.06)^{\frac{1}{2}} \sin(\frac{\pi}{4}) \quad ,$$

we computed an approximate value of b_1. Here the choice of the point(0.,.06) was arbitrary. Using the approximate value of b_1, we ran the problem again with the same input, but this time we subtracted the function $v(x,y) = b_1 r^{\frac{1}{2}} \sin(\theta/2)$ from the input data. We repeated this process until b_1 converged. For this problem, b_1 changed little after the first correction. Table III gives some representative results for the approximate solution using the first two guesses for b_1.

Table III

MORE RESULTS FOR THE PROBLEM IN FIGURE 8

x	y	Approximate u using 1st b_1 value	Approximate u using 2nd b_1 value	True solution
$-\frac{\pi}{4}$	$\frac{\pi}{8}$	1.6075	1.6090	1.6090
$-\frac{3\pi}{8}$	$\frac{\pi}{8}$	1.7647	1.7662	1.7662
0	$\frac{\pi}{2}$	2.2519	2.2529	2.2530
$\frac{\pi}{4}$	π	3.8033	3.8041	3.8042

Using the same number of points, the results here for the subtraction method are more accurate than the results for the method of redistributing the points. However, the subtraction method did require three times as much computation time.

The treatment of singularities on regions G having more than one σ value is more involved. In what follows, we will consider the problem shown in Fig. 9.

The method of redistributing the points works satisfactorily for this type of problem, but there are four boundary sections where points must be redistributed, instead of two as in the previous case. Thus, it is twice as expensive to add points to the calculation at the singularity.

The method of subtracting the singularities is also more difficult to apply in this case. We will assume that in some neighborhood of the singularity, the boundary is as shown in Fig. 9. Also, we will assume that in the subregion, G_1, the solution, u, is represented by

$$u(r,\theta) = \sum_{i=1}^{\infty} r^{\delta_i}\left[\alpha_i \cos(\delta_i\theta) + \beta_i \sin(\delta_i\theta)\right]$$

in polar coordinates, and that in the subregion, G_2, u is represented by

$$u(r,\theta) = \sum_{i=1}^{\infty} r^{\Delta_i}\left[a_i \cos(\Delta_i\theta) + b_i \sin(\Delta_i\theta)\right] \quad .$$

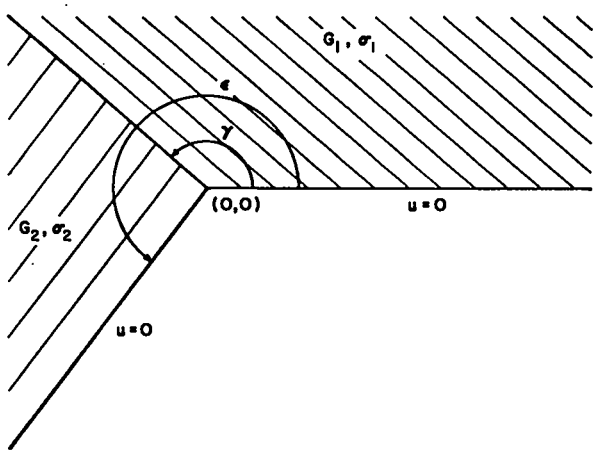

Fig. 9. Region G having two values of σ joining at a corner.

Here $0 \le \delta_1 < \delta_2 \ldots$ and $0 \le \Delta_1 < \Delta_2 \ldots$. Using the continuity across the common boundary, we immediately find that $\Delta_i = \delta_i$ for $i = 1, 2, \ldots, \infty$. Because $u = 0$ on both of the boundaries where the Dirichlet condition is given, we have $\delta_1 > 0$ and $0 = \alpha_1 = \alpha_2 = \ldots$.

Using the continuity of u across the common boundary again, we can equate coefficients of r^{δ_k} at $\theta = \gamma$ to get

$$\beta_k \sin(\delta_k\gamma) = a_k \cos(\delta_k\gamma) + b_k \sin(\delta_k\gamma) \quad .$$

Let $R = \sigma_2/\sigma_1$. Equating coefficients for the jump discontinuity in the normal derivative at $\theta = \gamma$, we get

$$\beta_k \cos(\delta_k\gamma) = R\left[-a_k \sin(\delta_k\gamma) + b_k \cos(\delta_k\gamma)\right] \quad .$$

Using $u = 0$ at $\theta = \epsilon$, we get the condition

$$a_k \cos(\delta_k\epsilon) + b_k \sin(\delta_k\epsilon) = 0 \quad .$$

The three equations written as a system are

$$a_k \cos(\delta_k\gamma) + b_k \sin(\delta_k\gamma) - \beta_k \sin(\delta_k\gamma) = 0 \quad ,$$

$$-a_k R \sin(\delta_k\gamma) + b_k R \cos(\delta_k\gamma) - \beta_k \cos(\delta_k\gamma) = 0 \quad ,$$

and

$$a_k \cos(\delta_k\epsilon) + b_k \sin(\delta_k\epsilon) = 0 \quad .$$

Setting the determinant of this system equal to zero, we have

$$\left[1+R\tan^2(\delta_k\gamma)\right]\tan(\delta_k\epsilon) - (1-R)\tan(\delta_k\gamma) = 0 \quad ,$$

or

$$\tan(\delta_k\epsilon) = \frac{(1-R)\tan(\delta_k\gamma)}{1 + R\tan^2(\delta_k\gamma)} \quad .$$

Using the formula for two $\tan^{-1}$ functions, we find that

$$\delta_k\epsilon = \delta_k\gamma - \tan^{-1}\left[R\tan(\delta_k\gamma)\right] \quad ,$$

and, hence,

$$\frac{\tan[\delta_k(\gamma-\epsilon)]}{\tan(\delta_k\gamma)} = R \quad .$$

The equation determines the sequence $0 < \delta_1 < \delta_2 \ldots$

From a numerical point of view, δ_1 is usually the only number of interest. Notice that if $\epsilon = 2\gamma$, then a solution does not exist for this equation unless $R = 1$. Notice also that once δ_1 and β_1 have been computed, one can compute a_1 and b_1 from the above system of equations. Moreover, a little analysis will show that

$(\epsilon-\gamma) > \gamma$ and $R > 1$ implies that $\frac{\pi}{\epsilon} > \delta_1 > \frac{\pi}{2(\epsilon-\gamma)}$,

$(\epsilon-\gamma) > \gamma$ and $R < 1$ implies that $\frac{\pi}{2\gamma} > \delta_1 > \frac{\pi}{\epsilon}$,

$(\epsilon-\gamma) < \gamma$ and $R > 1$ implies that $\frac{\pi}{2(\epsilon-\gamma)} > \delta_1 > \frac{\pi}{\epsilon}$,

and

$(\epsilon-\gamma) < \gamma$ and $R < 1$ implies that $\frac{\pi}{\epsilon} > \delta_1 > \frac{\pi}{2\gamma}$.

Thus, singularities can occur at points on the boundary where there is no reentrant corner, and singularities need not occur at reentrant corners.

Now consider how the subtraction method can be applied to a specific problem. As in the previous example, we must first use the routine LAPLDDC to compute the approximate behavior of the solution near the singularity. Using the approximate behavior of the solution in G_1 and the computed value of δ_1, we can approximate β_1 and, hence, a_1 and b_1. Then set

$$V = \begin{cases} \beta_1 r^{\delta_1}\sin(\delta_1\theta) & \text{in } G_1, \text{ and} \\ r^{\delta_1}\left[a_1\cos(\delta_1\theta)+b_1\sin(\delta_1\theta)\right] & \text{in } G_2 \quad , \end{cases}$$

and run the problem again to get the desired solution. If the common boundary is not a line segment, then the subroutines ROWSTOR and REGNSEL must be modified to take care of the fact that

$$\sigma_1\frac{\partial V}{\partial n_1} \neq -\sigma_2\frac{\partial V}{\partial n_2}$$

on the common boundary. This modification is given in Sec. III B. The modification is probably the easiest way to run the problem in any case.

Consider the example in Fig. 10. This problem

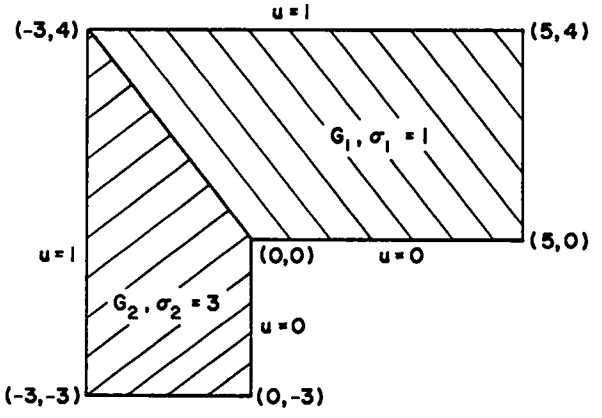

Fig. 10. Region with singularity on common boundary.

was run using the input shown on the next page. It was run a second time with the same input, except that KV was 21 for each boundary section, and a third time using KV = 41 for each boundary section. Finally, it was run a fourth time with KV = 11, and using the subtraction method described above. Some representative results are shown in Table IV.

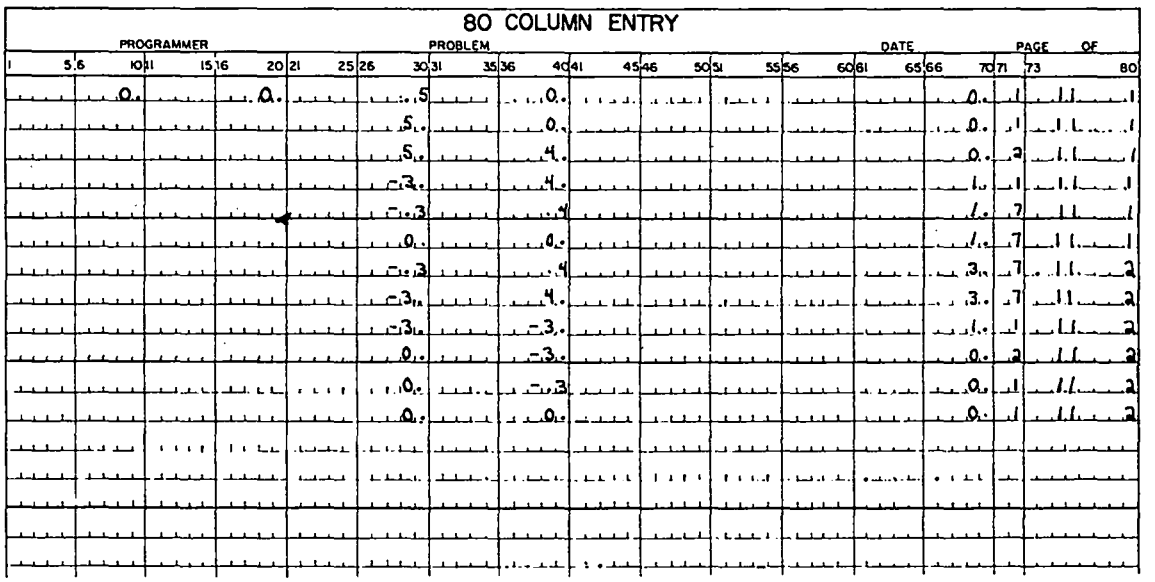

Table IV

RESULTS FOR THE PROBLEM OF FIGURE 10

x	y	Approximate u (KV = 11)	Approximate u (KV = 21)	Approximate u (KV = 41)	Approximate u (KV = 11 and subtraction)
-0.2	0.2	2.3337×10^{-1}	2.3354×10^{-1}	2.3361×10^{-1}	2.3365×10^{-1}
-0.1	0.1	1.4942×10^{-1}	1.4967×10^{-1}	1.4977×10^{-1}	1.4984×10^{-1}
-0.05	0.05	9.4565×10^{-2}	9.5578×10^{-2}	9.5735×10^{-2}	9.5847×10^{-2}

The author does not know of an analytic solution for this problem. However, using the subtraction method, the results shown are probably correct to four figures.

IV. OUTPUT

Using any of the four programs is a two-step process. The programs compute only the unknown boundary values. The call statement for each program is the same except for the program name. The call statement for program (i) is

```
CALL LAPLACE
```

No parameters are given in the call statement. Once the user has set up the input and called the subroutine to compute the unknown boundary values, he is faced with the question of output. For programs (i) and (iii) the output presently available is of three types: functional values at any given point in G, gradient values at any given point in G, and equipotential or gradient plots. In many cases the user would also like to have the computed unknown boundary values. The manner in which these boundary values are stored is discussed in Sec. XI. For programs (ii) and (iv) the output is the same, except that presently there is no explicit gradient routine. For all four programs, the call statements for the output are the same.

A. FN

After the unknown boundary values have been computed, it is simple to get the value of the approximate solution at any point. The real function FN(V,W) gives the value of the approximate solution at the point (V,W). For (V,W) not in G, FN gives the values -0.

B. GRADFN

Only programs (i) and (iii) presently have an explicit subroutine GRADFN. To compute an approximation to the gradient of the solution u(X,Y), one simply calls the subroutine GRADFN(X,Y,FX,FY). When the computer returns from GRADFN, the variables UX and UY contain $\partial u(X,Y)/\partial x$ and $\partial u(X,Y)/\partial y$, respectively. For program (iii) there is one more detail.

At any given time, one can use GRADFN only on the selected subregion. To select the Kth subregion, for instance, one merely uses the call statement

CALL REGNSEL(K) ,

or calls the routine FN(X,Y) with (X,Y) some point in the Kth subregion. Different subregions can be selected as often as desired. For (X,Y) not in G, or, in the case of program (iii), for (X,Y) not in the selected subregion, GRADFN is not well-defined.

C. Plot Routine

The plot routine is programmed to plot equipotential curves and gradient curves on film, using the Los Alamos Scientific Laboratory system. The routine can be easily modified to use other systems or devices for plotting. The two end points of a line segment and an integer, called K here, are given as input to the plot routine. The plot routine plots K equipotential curves or gradient curves depending on the call. The curves all cross the given line segment. If K > 1, then there are two curves at the ends of the segment and K-2 other curves crossing it at equal spacings. If K = 1, then one curve is plotted passing through the first end point of the line segment. The equipotential and gradient curves are a series of line segments of length HH. HH is given as input. We will say more later about the choice of HH. In what follows, we discuss the input variables for the plot routine. We discuss only the x-y plane because the plot routine in the z-r plane is exactly the same if one exchanges (z,r) for (x,y). A total list of input variables is as follows.

XMIN is the minimum X to be plotted.

YMIN is the minimum Y to be plotted.

XMAX is the maximum X to be plotted.

YMAX is the maximum Y to be plotted.

X1 is the first X coordinate of the line segment.

Y1 is the first Y coordinate of the line segment.

X2 is the second X coordinate of the line segment.

Y2 is the second Y coordinate of the line segment.

K is the number of equipotential or gradient curves to be plotted along the line segment.

HH is the step length to be used in constructing the curves.

There are four entry points into the plot routine: LAPLOT, LAPLOT1, LAPLOG, and LAPLOG1. LAPLOT and LAPLOT1 are both used to plot equipotential curves, and LAPLOG and LAPLOG1 are used to plot gradient curves. LAPLOT and LAPLOG advance the film; plot all of the boundary inside the rectangle formed by XMIN, YMIN, XMAX, and YMAX; and plot the desired curves inside the rectangle. LAPLOT1 and LAPLOG1 plot only the equipotential and gradient curves, and both assume that there has been a previous call to either LAPLOT or LAPLOG. LAPLOT1 and LAPLOG1 add more curves to previous plots. The input variables XMIN, YMIN, XMAX, and YMAX given in the call statements for LAPLOT1 and LAPLOG1 are used by these subroutines to determine the limits of any equipotential or gradient curves. The rectangle formed by these four points should be a subset of the rectangle given in the previous call to LAPLOT or LAPLOG. The call statement for LAPLOT is

CALL LAPLOT(XMIN,YMIN,XMAX,YMAX,X1,Y1,X2,Y2,K,HH) .

The call statement for the other three entry points is the same except for the name.

If K ≤ 0, only the boundary will be plotted. If there is an error in the input boundary data, and the unknown boundary values cannot be computed, then only the boundary will be plotted. The plots in this case are sometimes useful for finding errors. If XMIN ≥ XMAX or YMIN ≥ YMAX, then no equipotential or gradient curves will be plotted. If K > 50, only 50 curves will be plotted.

The plot routine will stop plotting an equipotential or gradient curve whenever any of the following occur: the curve runs outside the rectangle formed by XMIN, YMIN, XMAX, and YMAX; it runs into the part of the boundary S-C; it forms a closed curve; or it becomes too difficult to plot. The last condition occurs most often near corners in the boundary S, or when crossing a common boundary. Plotting equipotential or gradient curves is a time-consuming process. When computing each line segment of length HH used to approximate the curve, one must calculate at least one value of the potential and one gradient vector. Thus, the step HH should be as long as is practical. When one plots equipotential curves, the plot routine always requires that both ends of each line segment used to approximate the equipotential curve actually be on the curve. However, this is not possible with gradient curves. To follow the gradient

curves, the plot routine uses a Runge-Kutta method. The error on any given step is $O(HH^3)$. Locally the gradient curves are usually good, but the error over the length of the curve is cumulative.

V. MISCELLANEOUS PROGRAMMING INSTRUCTIONS

There are a few limitations on the programs that should be mentioned. When one has an exterior problem, the approximate solution tends to break down at large distances from the boundary (say 10^8 times the length of the boundary). However, the most important source of error seems to occur at the boundary. If H is the distance between approximation points on a given boundary section, then both FN and GRADFN start to break down at about 1.5H from the boundary section, except when the boundary section is linear. The error varies with the ratio of H to the minimum radius of curvature of the boundary section. It is easy to lose as much as half of the significant figures at the boundary. The same type of error causes trouble whenever two sections of the boundary are close relative to the distance between approximation points on either of them. As in the above case, this is not true when the boundary is linear.

The next two restrictions apply only to programs (ii) and (iv). Both are results of the approximations described in Sec. IX. If the restrictions are violated, there is a loss of accuracy proportional to the degree of violation. In the first place, any boundary section beginning or ending on the z-axis must intersect the axis at an angle greater than 12°. See Fig. 11.

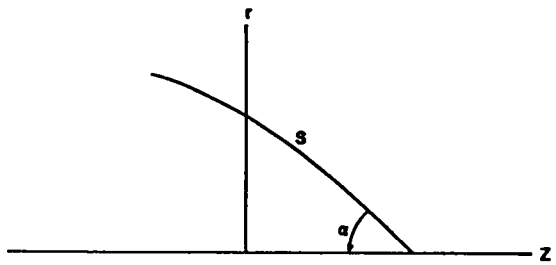

Fig. 11. Boundary section touching the z-axis.

Here the angle is α, and one should have $\alpha \geq 12^\circ$. The second restriction is difficult to explain exactly. We will discuss only linear boundary sections. Let S_j be a linear boundary section with end points (z_1,r_1) and (z_2,r_2). If S_j touches the z-axis, then it must satisfy the first restriction. If it does not touch the z-axis, then it must be true that

$$\frac{|z_2-z_1|}{KV_j^{-1}} \leq 10 \min(r_2,r_1) \quad .$$

See Sec. III for the definition of KV. For any given (z_1,r_1) and (z_2,r_2), there is a corresponding minimum value of KV that will satisfy the second restriction.

Probably we should say something about accuracy and timing. The discussion of accuracy will be limited to sample problems. Earlier we compared some numerical results to the explicit solution for a problem with singularities and a magnetic field problem with current sources. Now consider the problems shown in Figs. 5 and 6. These two problems were run with exactly the input shown. The problem in Fig. 6 was run with program (i), and that in Fig. 5 was run with program (iii). Some representative results for the problem in Fig. 5 are given in Table V.

Table V

RESULTS FOR THE PROBLEM OF FIGURE 5

x	y	Computed u	True u
-0.023	-0.023	-0.004090	-0.004087
0.03	0.	0.004766	0.004762
0.	0.036	2×10^{-9}	0.
0.9	0.9	0.8999	0.8993
-0.5	0.1	-0.4981	-0.4976
0.06	0.	0.03908	0.03905

For this problem there is more than one possible source of error. Using superposition to compute the approximate solution, as was discussed in Sec. III B, we get about the same solution in the dielectric material. However, the solution in air is much more accurate. For the problem given in Fig. 6 the error is entirely due to the discretization of the boundary values. Some results for this problem are given in Table VI.

Table VI

RESULTS FOR THE PROBLEM OF FIGURE 6

x	y	Computed u	True u	Computed $\partial u/\partial x$	True $\partial u/\partial x$
0.5	1.	-1.18762	-1.18750	-7.50007	-7.50000
0.25	0.25	-0.00394	-0.00391	-0.00002	0.0
0.05	0.4	0.00861	0.00865	-0.06303	-0.06300
-0.9	0.8	-1.19513	-1.19512	-2.44809	-2.44800

The times given here are for the CDC 6600, and they tend to be optimistic for small problems. The time required to compute the unknown boundary values is simple to calculate.

Let $N_i = \frac{1}{100} \sum_{\text{with } S_j \in G_i}^{} KV_j$, and let $NT = \sum_{i=1}^{NRE} N_i$. The quantity NRE is the total number of subregions. For programs (i) and (ii), NRE = 1, and NT = N_1. There are two main problems in computing the unknown boundary values. One is generating a matrix called A, and the other is the solution of the matrix problem Az = E for the vector z. The solution of the matrix problem requires about $1.1NT^3 + 0.3NT^2$ sec. For programs (i) and (iii), the time required to generate the matrix A is about $1.3\left(\sum_{i=1}^{NRE} N_i^2\right)$ sec. Thus, the total time required to compute the unknown boundary values for the problem in Fig. 6 was about 7 sec. To compute the value of the approximate potential in the ith subregion using the routine FN requires 0.013 N_i sec, and to compute the approximate gradient at a given point in the ith subregion using the routine GRADFN requires 0.015 N_i sec. For programs (ii) and (iv), it takes about a factor of six longer to generate the matrix A or to compute the approximate potential. The times given here assume two things. First, the only NT^3 operation for the matrix problem is written in machine language which cuts the time required for this operation in half. Second, the routine to evaluate the elliptic integrals is also written in machine language. The time savings for this operation is not known.

For a solution u(x,y) of Laplace's equation in the x-y plane, it must be true that

$$\int_S \sigma \frac{\partial u}{\partial \nu} \, dS = 0 \quad . \tag{3}$$

For an axially symmetric solution u(z,r) in the z-r plane, it must be true that

$$\int_S \sigma r \frac{\partial u}{\partial \nu} \, dS = 0 \quad . \tag{4}$$

For problems that have the Dirichlet condition given somewhere on S, it has been our experience that the ratios

$$\frac{\int_S \sigma \frac{\partial u}{\partial \nu} \, dS}{\int_S \sigma \left|\frac{\partial u}{\partial \nu}\right| dS} \quad \text{or} \quad \frac{\int_S \sigma r \frac{\partial u}{\partial \nu} \, dS}{\int_S \sigma \left|\frac{\partial u}{\partial \nu}\right| \, dS}$$

give a good indication of the correctness of the solution. Unless the ratio corresponding to the given problem is small, 10^{-4} or 10^{-5}, for instance, there is usually an error in the input, or a singularity on the boundary that has not been properly treated. The above ratios are easy to compute and are discussed in Sec. XI. When computing the ratios, one need not include the integrals over common boundaries, because they are actually zero. For programs (i) and (ii), σ is constant and can be eliminated from the computation.

VI. INTEGRAL EQUATIONS

We now reduce the problem of solving for the unknown boundary values for the general problem to solving three coupled integral equations. First we consider the x-y plane. The problem of solving for the unknown boundary values in a homogeneous medium is a subproblem of solving for the unknown boundary values in inhomogeneous media. Therefore, we consider only inhomogeneous media. Moreover, the problem for the homogeneous medium is discussed by Hayes.[3]

Assume that G consists of NRE subregions G_1, G_2, . . . , G_{NRE} with boundaries ∂G_1, ∂G_2, . . . , ∂G_{NRE}. Let C_1, C_2, . . . , C_{NRE} be the common boundaries for each of the subregions. For k = 1, 2, . . . , NRE, C_k must not be empty. Remember that σ_k is the value of σ for the region G_k. Let $\{[x(t),y(t)] | t \in (0,d)\}$ be a parametric representation of S with respect to arc length. Here d is the length of S. To make the notation easier, let $t \in \partial G_j$ mean that $[x(t),y(t)] \in \partial G_j$, and, similarly, $t \in C_j$ will mean that $[x(t),y(t)] \in C_j$. As a point of interest to be used later, note that for $t \in \partial G_j$, $[dy(t)/dt, -dx(t)/dt]$ forms the unit normal vector exterior to G_j at the point $[x(t),y(t)]$. For any point $t \in C = \bigcup_{k=1}^{NRE} C_k$, there is another point that will be called $\chi(t)$, such that $[x(t),y(t)] = [x(\chi(t)), y(\chi(t))]$. In a similar vein, define the function $\omega(t)$ mapping C onto the integers 1, 2, . . . , NRE by the relation $t \in C_{\omega(t)}$ for all $t \in C$. Also, define the function $\psi(t)$ mapping C onto the integers 1, 2, . . . , NRE by the relation $\psi(t) = \omega[\chi(t)]$. For k = 1, 2, . . . , NRE, define u_k to be the restriction of the solution u to the subregion G_k.

Because S is a finite number of piecewise smooth curves, there are, at most, a finite number of points $[x(\xi_i),y(\xi_i)]$ for i = 1, 2, . . . , NC where S has corners. Let α_{ξ_i} for i = 1, 2, . . . , NC be the angle inside G that the curve S makes with itself at $[x(\xi_i),y(\xi_i)]$. See Fig. 12.

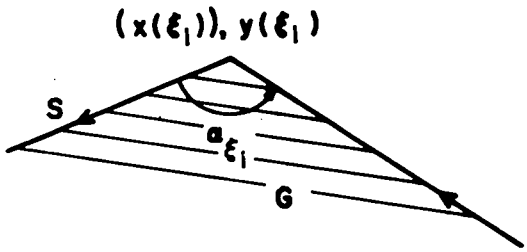

Fig. 12. Angle that S makes with itself at $[x(\xi_i), y(\xi_i)]$.

From our definition, we see that $0 \le \alpha_{\xi_i} \le 2\pi$. For $t\epsilon(0,d)$, define

$$\alpha(t) = \begin{cases} \pi & \text{if } t \neq \xi_i \text{ for } i = 1, 2, \ldots, NC \\ \alpha_{\xi_i} & \text{if } t = \xi_i \end{cases} ,$$

Set $r(s,t) = \left\{[x(s)-x(t)]^2 + [y(s)-y(t)]^2\right\}^{\frac{1}{2}}$, $u(s) = u[x(s),y(s)]$, and $\partial u(s)/\partial\nu = \partial u[x(s),y(s)]/\partial\nu$. Notice that Eq. (1) holds on any subregion, G_k. If we use the above definitions and assume that $u(s)$ and $\partial u(s)/\partial\nu$ are smooth, Eq. (1) becomes on the boundary

$$\alpha(t)u_k(t) = \int_{s\epsilon\partial G_k} \left\{\ln\left[\frac{1}{r(s,t)}\right] \frac{\partial u_k(s)}{\partial\nu} - u_k(s) \frac{\partial}{\partial\nu} \ln\left[\frac{1}{r(s,t)}\right]\right\} ds \quad . \tag{5}$$

For a proof of this result see Hayes,[3] p. 16. Actually Eq. (5) is a system of equations, because the result holds on ∂G_k for $k = 1, 2, \ldots, NRE$. The basic assumption of the scheme is that Eq. (5), along with the matching boundary conditions, can be inverted to solve for $\partial u/\partial\nu$ on S_D, u on S_N, and both $\partial u/\partial\nu$ and u on C. For $t\epsilon C \cap \partial G_k$ the matching boundary conditions are

$$\begin{aligned} u_k(t) &= u_{\psi(t)}[\chi(t)] \quad , \\ \text{and} \quad \sigma_k \frac{\partial u_k(t)}{\partial\nu} &= -\sigma_{\psi(t)} \frac{\partial u_{\psi(t)}[\chi(t)]}{\partial\nu} \quad . \end{aligned} \tag{6}$$

Again, Eq. (6) is really a system of equations, because the conditions must be true on each of the NRE common boundaries.

Define $E(t)$ for $t\epsilon(0,d)$, by the relation

$$E(t) = \begin{cases} \int_{S_D \cap \partial G_k} f(s) \frac{\partial}{\partial\nu} \ln[r(s,t)]ds - \alpha(t)f(t) \\ \quad - \int_{S_N \cap \partial G_k} g(s) \ln[r(s,t)]ds \quad \text{if } t\epsilon S_D \cap \partial G_k \\ \int_{S_D \cap \partial G_k} f(s) \frac{\partial}{\partial\nu} \ln[r(s,t)]ds - \int_{S_N \cap \partial G_k} g(s) \ln[r(s,t)]ds \\ \quad \text{if } t\epsilon(S_N \cup C) \cap \partial G_k \quad . \end{cases}$$

Here $f(s)$ and $g(s)$ are the known boundary values given in the statement of the general problem. Hence, $E(t)$ is a known function and can be computed. For ease of notation for $k = 1, 2, \ldots, NRE$, define $S_N^k = (S_N \cup C) \cap \partial G_k$ and $S_D^k = (S_D \cup C) \cap \partial G_k$. Assuming that Eqs. (5) and (6) can be inverted to solve for the unknown boundary values, one can formulate the problem as the following series of coupled equations.

$$\begin{aligned} &\alpha(t)u_k(t) - \int_{S_N^k} u_k(s) \frac{\partial}{\partial\nu} \ln[r(s,t)]ds \\ &\quad + \int_{S_D^k} \frac{\partial u_k(s)}{\partial\nu} \ln[r(s,t)]ds = E(t) \quad \text{if } t\epsilon S_N^k \quad , \\ &\quad - \int_{S_N^k} u_k(s) \frac{\partial}{\partial\nu} \ln[r(s,t)]ds \\ &\quad + \int_{S_D^k} \frac{\partial u_k(s)}{\partial\nu} \ln[r(s,t)]ds = E(t) \\ &\qquad \text{if } t\epsilon S_D \cap \partial G_k \\ &\text{for } k = 1, 2, \ldots, NRE, \end{aligned} \tag{7}$$

and

$$\begin{aligned} u_{\omega(t)}(t) &= u_{\psi(t)}[\chi(t)] \quad \text{if } t\epsilon C \quad , \\ \sigma_{\omega(t)} \frac{\partial u_{\omega(t)}(t)}{\partial\nu} &= -\sigma_{\omega(t)} \frac{\partial u_{\psi(t)}[\chi(t)]}{\partial\nu} \quad \text{if } t\epsilon C \quad . \end{aligned}$$

Equation (7) will be reduced to a system of three equations with one unknown function. First, more definitions are necessary. Define

$$C^1 = [s\epsilon C | \omega(s) < \psi(s)] \quad ,$$

$$C^2 = C - C^1 \quad ,$$

$$K_N(s,t) = \begin{cases} -\frac{\partial}{\partial \nu} \ln[r(s,t)] \text{ if } t\epsilon\partial G_k \text{ and } s\epsilon S_N^k \\ \qquad\qquad\qquad \text{for some } k \ , \\ 0 \quad \text{otherwise} \end{cases}$$

$$K_D(s,t) = \begin{cases} \ln[r(s,t] \text{ if } t\epsilon\partial G_k \text{ and } s\epsilon(S_D \cup C^1) \\ \qquad\qquad \cap\ \partial G_k \quad \text{for some } k \ , \\ -\left[\sigma_{\upsilon(s)}/\sigma_{\psi(s)}\right]\ln[r(s,t)] \text{ if } t\epsilon\partial G_k \\ \qquad \text{and } s\epsilon C^2 \cap \partial G_k \text{ for some } k \ , \\ 0 \quad \text{otherwise} \end{cases}$$

and

$$\tau(s) = \begin{cases} u_k(s) \quad \text{if } s\epsilon(S_N \cup C^2) \cap \partial G_k \\ \dfrac{\partial u_k(s)}{\partial \nu} \quad \text{if } s\epsilon(S_D \cup C^1) \cap \partial G_k \end{cases} \quad .$$

Thus, $\tau(s)$ is defined on $(0,d)$, and both K_N and K_D are defined on $(0,d) \times (0,d)$. Using the matching boundary conditions, if $s\epsilon C^1 \cap \partial G_k$, then $u_k(s) = \tau[\chi(s)]$, and if $t\epsilon\partial G_k$ and $s\epsilon C^2 \cap \partial G_k$, then

$$\frac{\partial u_k(s)}{\partial \nu} \ln[r(s,t)] = \tau[\chi(s)]K_D(s,t).$$

Using these definitions consider the first equation in Eqs. (7) with $k = 1$. For $t\epsilon(S_N \cup C^2) \cap \partial G$, this equation is equivalent to

$$\alpha(t)\tau(t) + \int_{S_N \cup C^2} \tau(s)\ K_N(s,t)ds + \int_{S_D \cup C^1} \tau(s)\ K_D(s,t)ds$$
$$+ \int_{C^1} \tau[\chi(s)]K_N(s,t)ds + \int_{C^2} \tau[\chi(s)]K_D(s,t)ds = E(t) \ . \tag{8}$$

For $t\epsilon C^1 \cap \partial G_1$, the first equation in Eqs. (7) with $k = 1$ is equivalent to

$$\alpha(t)\tau[\chi(t)] + \int_{S_N \cup C^2} \tau(s)\ K_N(s,t)ds + \int_{S_D \cup C^1} \tau(s)\ K_D(s,t)ds$$
$$+ \int_{C^1} \tau[\chi(s)]K_N(s,t)ds + \int_{C^2} \tau[\chi(s)]K_D(s,t)ds = E(t) \ . \tag{9}$$

For $t\epsilon S_D \cap \partial G_1$, the second equation in Eqs. (7) with $k = 1$ is equivalent to

$$\int_{S_N \cup C^2} \tau(s)\ K_N(s,t)ds + \int_{S_D \cup C^1} \tau(s)\ K_D(s,t)ds$$
$$+ \int_{C^1} \tau[\chi(s)]K_N(s,t)ds + \int_{C^2} \tau[\chi(s)]K_D(s,t)ds = E(t) \ . \tag{10}$$

However, Eqs. (8), (9), and (10) are true independent of k. Thus, the system (7), of 2(NRE + 1) equations is equivalent to the system:

$$\alpha(t)\tau(t) + \int_{S_N \cup C^2} \tau(s)K_N(s,t)ds + \int_{S_D \cup C^1} \tau(s)K_D(s,t)ds$$
$$+ \int_{C^1} \tau[\chi(s)]K_N(s,t)ds + \int_{C^2} \tau[\chi(s)]K_D(s,t)ds$$
$$= E(t) \text{ if } t\epsilon S_N \cup C^2 \ ,$$

$$\alpha(t)\tau[\chi(t)] + \int_{S_N \cup C^2} \tau(s)K_N(s,t)ds + \int_{S_D \cup C^1} \tau(s)K_D(s,t)ds$$
$$+ \int_{C^1} \tau[\chi(s)]K_N(s,t)ds + \int_{C^2} \tau[\chi(s)]K_D(s,t)ds$$
$$= E(t) \quad \text{if } t\epsilon C^1 \ ,$$

and

$$\int_{S_N \cup C^2} \tau(s)K_N(s,t)ds + \int_{S_D \cup C^1} \tau(s)K_D(s,t)ds + \int_{C^1} \tau[\chi(s)]K_N(s,t)ds$$
$$+ \int_{C^2} \tau[\chi(s)]K_D(s,t)ds = E(t) \text{ if } t\epsilon S_D \ .$$

Now set

$$K(s,t) = \begin{cases} K_N(s,t) \text{ for } s\epsilon S_N \cup C^2 \\ K_D(s,t) \text{ for } s\epsilon S_D \cup C^1 \end{cases} \ ,$$

and

$$\tilde{K}(s,t) = \begin{cases} K_D(s,t) \text{ for } s\epsilon C^1 \ , \\ K_N(s,t) \text{ for } s\epsilon C^2 \end{cases} \ .$$

The above system then becomes

$$\alpha(t)\tau(t) + \int_S \tau(s)K(s,t)ds + \int_C \tau[\chi(s)]\tilde{K}(s,t)ds = E(t) \text{ for } t\epsilon S_N \cup C^2 ,$$

$$\alpha(t)\tau[\chi(t)] + \int_S \tau(s)K(s,t)ds + \int_C \tau[\chi(s)]\tilde{K}(s,t)ds = E(t) \text{ for } t\epsilon C^1 , \quad (11)$$

and

$$\int_S \tau(s)K(s,t)ds + \int_C \tau[\chi(s)]\tilde{K}(s,t)ds = E(t) \text{ for } t\epsilon S_D .$$

Note that at every point where two boundary sections join, there is a possibility that τ, K, and $\tilde{K}$ are not well-defined. This will be covered later.

Now we will discuss the axially symmetric integral equation. Suppose we want to find an axially symmetric solution, $u(z,r,\varphi)$, to Laplace's equation on an axially symmetric region, G', in (z,r,φ) space. It is true that

$$u(z,r,\varphi) = \frac{1}{4\pi}\int\int_{S'}\left[\left(\frac{1}{d}\right)\frac{\partial u(\zeta,\rho,\theta)}{\partial \nu} - u(\zeta,\rho,\theta)\frac{\partial}{\partial \nu}\left(\frac{1}{d}\right)\right]dS' . \quad (12)$$

See Courant,[8] p. 257. Here S' is the boundary of G', and

$$d = \left\{(z-\zeta)^2+(r-\rho)^2+2r\rho[1-\cos(\theta-\varphi)]\right\}^{\frac{1}{2}}$$

is the distance from the point (z,r,φ) to the point (ζ,ρ,θ). Because $u(z,r,\varphi)$ is axially symmetric, we can drop the dependence on φ and assume that $\varphi = 0$. Also, using the half-angle formula for the sine function, we have

$$d = \left[(z-\zeta)^2+(r-\rho)^2+4r\rho\sin^2\left(\frac{\theta}{2}\right)\right]^{\frac{1}{2}} .$$

Let G be the restriction of G' to the upper half-plane and let S be the restriction of S' to the upper half-plane. Equation (12) can then be written in the form

$$u(z,r) = \frac{1}{4\pi}\int_S \frac{\partial u(\zeta,\rho)}{\partial \nu}\left(\int_0^{2\pi}\rho\,\frac{d\theta}{d}\right)dS - \frac{1}{4\pi}\int_S u(\zeta,\rho)\left[\int_0^{2\pi}\frac{\partial}{\partial \nu}\left(\frac{1}{d}\right)\rho d\theta\right]dS = \frac{1}{4\pi}\int_S \frac{\partial u(\zeta,\rho)}{\partial \nu}\left(\int_0^{2\pi}\frac{d\theta}{d}\right)\rho dS - \frac{1}{4\pi}\int_S u(\zeta,\rho)\frac{\partial}{\partial \nu}\left[\int_0^{2\pi}\left(\frac{1}{d}\right)d\theta\right]\rho dS . \quad (13)$$

Consider now only the integral $\int_0^{2\pi} d\theta/d$. Using the symmetry of the integrand, we have

$$\int_0^{2\pi}\frac{d\theta}{d} = \int_0^{2\pi}\frac{d\theta}{\left[(z-\zeta)^2+(r-\rho)^2+4r\rho\sin^2\left(\frac{\theta}{2}\right)\right]^{\frac{1}{2}}} = 2\int_0^{\pi}\frac{d\theta}{\left[(z-\zeta)^2+(r-\rho)^2+4r\rho\sin^2\left(\frac{\theta}{2}\right)\right]^{\frac{1}{2}}} = 4\int_0^{\pi/2}\frac{d\theta}{\left[(z-\zeta)^2+(r-\rho)^2+4r\rho\sin^2(\theta)\right]^{\frac{1}{2}}} .$$

Set

$$\beta = \frac{4r\rho}{(z-\zeta)^2+(r-\rho)^2}, \text{ and } d_o = [(z-\zeta)^2+(r-\rho)^2]^{\frac{1}{2}}.$$

Obviously $\beta \geq 0$. Assuming that $d_o \neq 0$,

$$\int_0^{\pi/2}\frac{d\theta}{\left[(z-\zeta)^2+(r-\rho)^2+4r\rho\sin^2(\theta)\right]^{\frac{1}{2}}} = \frac{1}{d_o}\int_0^{\pi/2}\frac{d\theta}{\left[1+\beta\sin^2(\theta)\right]^{\frac{1}{2}}} = \frac{1}{d_o}\int_0^{\pi/2}\frac{d\theta}{\left[1+\beta-\beta\cos^2(\theta)\right]^{\frac{1}{2}}} = \frac{1}{d_o(1+\beta)^{\frac{1}{2}}}\int_0^{\pi/2}\frac{d\theta}{\left[1-\frac{\beta}{1+\beta}\sin^2(\theta)\right]^{\frac{1}{2}}} = \frac{1}{d_o(1+\beta)^{\frac{1}{2}}}K\left(\frac{\beta}{1+\beta}\right) .$$

Here K is the complete elliptic integral of the first kind. See Ref. 9, p. 589-606. Set $m = \beta/1+\beta$. We then have $0 \leq m \leq 1$, and, thus, the following relation.

$$\int_0^{2\pi}\frac{d\theta}{d} = \frac{4}{d_o(1+\beta)^{\frac{1}{2}}}K(m) = \frac{4}{\left[(z-\zeta)^2+(\rho-r)^2+4r\rho\right]^{\frac{1}{2}}}K(m) .$$

Substituting this formula in Eq. (13), we get

$$u(z,r) = \frac{1}{2\pi}\int_S \left[\frac{\partial u(\zeta,\rho)}{\partial \nu}\frac{2K(m)\rho}{[(z-\zeta)^2+(\rho-r)^2+4r\rho]^{\frac{1}{2}}}\right.$$

$$\left. - u(\zeta,\rho)\frac{\partial}{\partial \nu}\frac{2K(m)\rho}{[(z-\zeta)^2+(\rho-r)^2+4r\rho]^{\frac{1}{2}}}\right] dS \quad . \quad (14)$$

Equation (14) is the axially symmetric version of Eq. (1). If we use the kernel $2K(m)\rho/[(z-\zeta)^2+(\rho-r)^2+4r\rho]^{\frac{1}{2}}$, instead of the kernel $\log(1/r)$, then the rest of the derivation for the integral equations is the same as for the x-y plane. The system of Eqs. (11) also holds for the z-r plane except that the kernel is different.

VII. DISCRETIZATION OF THE PROBLEM

In this section we derive a discrete approximation to the system of Eq. (11). The derivation holds for both the z-r and x-y planes. $\tau(s)$, $f(s)$, and $g(s)$ are approximated piecewise by polynomials of degree two. We assume that these functions have bounded third derivatives on S, except possibly at a finite number of points. We assume also that the boundary S is partitioned into sections S_j for $j = 1, 2, \ldots, NDCT$ such that: each section S_j is a smooth curve, i.e., S_j has no corners and is connected; each section S_j is contained entirely in either S_D, S_N, or C; and $f(s)$, $g(s)$, or $\tau(s)$, whichever is defined on S_j, has bounded third derivatives interior to S_j. Remember that NDCT is the total number of boundary sections. Usually S_j should be as large as possible while still satisfying the three given conditions.

At the end points of each section S_j for $j = 1, 2, \ldots, NDCT$, there is a possibility that $\tau(s)$ is not well-defined because there is a possible change in the type of boundary condition. Also, if there is a corner, the Neumann derivative need not be continuous around the corner. To correct the difficulty, we define an upper and lower limit at each $s\in(0,d)$ that is an end point of a boundary section. Let

$$\tau(s+) = \lim_{\substack{\delta\to 0 \\ \delta>0}} \tau(s+\delta) \quad ,$$

and

$$\tau(s-) = \lim_{\substack{\delta\to 0 \\ \delta<0}} \tau(s+\delta) \quad .$$

Similar definitions could be made for $K(s,t)$ and $\bar{K}(s,t)$, but, because these functions are always integrated, there is no problem with undefined points.

On each section S_j, $\tau(s)$ is approximated by using KV_j points for $j = 1, 2, \ldots, NDCT$. Because of the type of approximation used, each KV_j must be odd and greater than one. Remember that $NT = \sum_{j=1}^{NDCT} KV_j$, and LS_j is the length of S_j. Define $h_j = LS_j/(KV_j-1)$ for $j = 1, 2, \ldots, NDCT$. We next define arguments t_k for $k = 1, 2, \ldots, NT$ for the function $\tau(s)$. $\tau(s)$ is approximated at these points. First let $j(k)$ be the integer-valued function defined for $k = 1, 2, \ldots, NT$ such that

$$\sum_{i=1}^{j(k)-1} KV_i < k \le \sum_{i=1}^{j(k)} KV_i \quad .$$

Set

$$t_k = \begin{cases} \left[\sum_{i=1}^{j(k)} LS_i\right] - & \text{if } k = \sum_{i=1}^{j(k)} KV_i \quad , \\ \left[\sum_{i=1}^{j(k)-1} LS_i\right] + & \text{if } k = 1 + \sum_{i=1}^{j(k)-1} KV_i \quad , \\ \left[\sum_{i=1}^{j(k)-1} LS_i\right] + \left[k-1-\sum_{i=1}^{j(k)-1} KV_i\right] h_j(k) & \text{otherwise.} \end{cases}$$

Looking at the parametric representation for S, we see that for each section S_j, there are two values of t_k corresponding to its end points. The first point of S_j corresponds to an upper limit, and the last point corresponds to a lower limit. There are $KV_j - 2$ other points $[x(t_k),y(t_k)]$, equally spaced at a distance h_j along the boundary section. We assume that the ordering of the sections S_j corresponds to the ordering in the parametric representation.

Define $\tau_k = \tau(t_k)$ for $k = 1, 2, \ldots, NT$. The function $\tau(s)$ is approximated on (0,d) by a piecewise polynomial of degree two, called $\tau_A(s)$. On the interval (t_1,t_3), $\tau_A(s)$ is the polynomial of degree two having $\tau_A(t_1) = \tau_1$, $\tau_A(t_2) = \tau_2$, and $\tau_A(t_3) = \tau_3$. On (t_3,t_5), $\tau_A(s)$ is the polynomial of degree two having $\tau_A(t_3) = \tau_3$, $\tau_A(t_4) = \tau_4$, and $\tau_A(t_5) = \tau_5$. The approximation continues in this manner until the point t_{KV_1} is reached. t_{KV_1} corresponds to the end of the section S_1. Hence, S_1 is broken up into $(KV_1-1)/2$ subsections of equal length, and on each of the subsections $\tau_A(s)$ is a polynomial of degree two. Because t_{KV_1} corresponds to the end

of S_1, $\tau(s)$ is possibly discontinuous at this point. Therefore the upper limit τ_{KV_1+1} is used on the next interval. That is, for $s\epsilon(t_{KV_1+1}, t_{KV_1+3})$, $\tau_A(s)$ is the polynomial of degree two with $\tau_A(t_{KV_1+1}) = \tau_{KV_1+1}$, $\tau_A(t_{KV_1+2}) = \tau_{KV_1+2}$, and $\tau_A(t_{KV_1+3}) = \tau_{KV_1+3}$. This type of approximation is continued throughout (0,d). Using the function j(k) given above, we can define $\tau_A(s)$ by

$$\tau_A(s) = \begin{cases} \left[\tau_{k+2}(s-t_k)(s-t_{k+1}) - 2\tau_{k+1}(s-t_k)(s-t_{k+2}) \right. \\ \qquad \left. + \tau_{k+2}(s-t_{k+1})(s-t_{k+2})\right]/2h_{j(k)}^2 \\ \text{if } s\epsilon(t_k, t_{k+2}) \text{ and } \left[k - \sum_{i=1}^{j(k)-1} KV_i\right] \text{ is odd,} \\ \\ \tau_k \text{ if } s = t_k \text{ for some } k = 1, 2, \ldots, NT. \end{cases}$$

Notice that

$$\tau_A(s) = \sum_{k=1}^{NT} \theta_k(s)\tau_k \quad , \tag{15}$$

where each $\theta_k(s)$ is a piecewise polynomial of degree two, which is nonzero in, at most, an interval of length $4h_{j(k)}$ centered at t_k. An exact definition of θ_k is as follows.

$$\theta_k(s) = \frac{1}{2h_{j(k)}^2} \begin{cases} (s-t_{k+1})(s-t_{k+2}) \text{ if } \left[k - \sum_{i=1}^{j(k)-1} KV_i\right] \\ \qquad \text{is odd and less than } KV_{j(k)} \\ \qquad \text{and } s\epsilon(t_k, t_{k+2}), \\ (s-t_{k-1})(s-t_{k-2}) \text{ if } \left[k - \sum_{i=1}^{j(k)-1} KV_i\right] \\ \qquad \text{is odd and greater than one} \\ \qquad \text{and } s\epsilon(t_{k-2}, t_k), \\ -2(s-t_{k-1})(s-t_{k+1}) \text{ if} \left[k - \sum_{i=1}^{j(k)-1} KV_i\right] \\ \qquad \text{is even, and } s\epsilon(t_{k-1}, t_{k+1}), \\ 0 \quad \text{otherwise .} \end{cases}$$

To explain some of the above notation, consider the following example. Suppose S is the unit circle with center at the origin. Let $[x(t),y(t)] = [\cos(t), \sin(t)]$, $t\epsilon(0,2\pi)$ be the parametric representation for the boundary S. Assume that

$$S_1 = \{[x(t),y(t)] | t\epsilon(0,\pi/2)\}, \text{ and}$$

$$S_2 = \{[x(t),y(t)] | t\epsilon(\pi/2,2\pi)\} \quad .$$

Obviously NDCT = 2, d = 2π, $LS_1 = \pi/2$, and $LS_2 = 3\pi/2$. Assume that $KV_1 = 3$ and $KV_2 = 5$. It follows that NT = 8, $h_1 = \pi/4$, and $h_2 = 3\pi/8$. The locations of the arguments t_k, k = 1, 2, . . . , 8 are shown in Fig. 13.

If the Dirichlet (or Neumann) boundary condition were given on all of S and if f(s), (or g(s)), were continuous, then it would be true that $\tau_1 = \tau_8$ and $\tau_3 = \tau_4$. However, if the Dirichlet condition were given on one section and the Neumann condition on the other, then the pairs would not, in general, be equal.

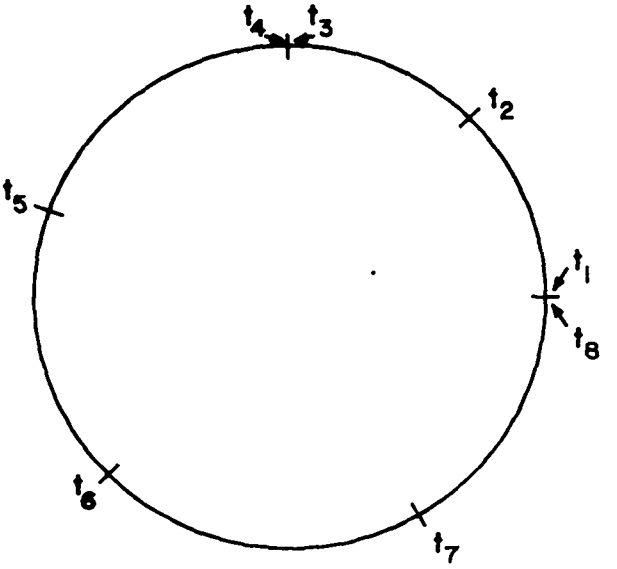

Fig. 13. Distribution of arguments t_k.

In what follows, the term $O(h^3)$ denotes both a scalar and a vector quantity. In the scalar notation, it has its usual meaning; in the vector notation, it denotes a vector with each component bounded by a constant times h^3. If $\tau(s)$ is assumed to have a bounded third derivative, then

$$\tau(s) = \tau_A(s) + O(h^3) \quad \text{for } s\epsilon(0,d) \quad , \tag{16}$$

where $h = \max(h_1, h_2, \ldots, h_{NDCT})$.

Next, we describe a method for using the coupled Eqs. (11) to approximate $\tau_A(s)$. Substituting the right-hand side of Eq. (16) in Eq. (11), we get

$$\alpha(t)\left[\tau_A(t)+O(h^3)\right] + \int_S \left[\tau_A(s)+O(h^3)\right] K(s,t)ds$$

$$+ \int_C \left\{\tau_A[\chi(s)]+O(h^3)\right\}\tilde{K}(s,t)ds = E(t) \text{ if } t\epsilon S_N \cup C^2 ,$$

$$\alpha(t)\left\{\tau_A[\chi(t)]+O(h^3)\right\} + \int_S \left[\tau_A(s)+O(h^3)\right]K(s,t)ds$$

$$+ \int_C \left\{\tau_A[\chi(s)]+O(h^3)\right\}\tilde{K}(s,t)ds = E(t) \quad \text{if } t\epsilon C^1 ,$$

and

$$\int_S \left[\tau_A(s)+O(h^3)\right]K(s,t)ds + \int_C \left\{\tau_A[\chi(s)]+O(h^3)\right\}\tilde{K}(s,t)ds$$

$$= E(t) \quad \text{if } t\epsilon S_D .$$

It is easy to show that both $\int_S |K(s,t)|ds$ and $\int_C |\tilde{K}(s,t)|ds$ are bounded functions of t with the bound dependent only on S. Hence, it follows that

$$\alpha(t)\tau_A(t) + \int_S \tau_A(t)K(s,t)ds + \int_C \tau_A[\chi(s)]\tilde{K}(s,t)ds$$

$$= E(t) + O(h^3) \quad \text{if } t\epsilon S_N \cup C^2 ,$$

$$\alpha(t)\tau_A[\chi(t)] + \int_S \tau_A(s)K(s,t)ds + \int_C \tau_A[\chi(s)]\tilde{K}(s,t)ds \tag{17}$$

$$= E(t) + O(h^3) \quad \text{if } t\epsilon C^1 ,$$

and

$$\int_S \tau_A(s)K(s,t)ds + \int_C \tau_A[\chi(s)]\tilde{K}(s,t)ds = E(t)+O(h^3)$$

$$\text{if } t\epsilon S_D .$$

Using the representation formula for $\tau_A(s)$ given in Eq. (15), we get

$$\alpha(t)\tau_A(t)+ \sum_{k=1}^{NT} \tau_k\left\{ \int_S \theta_k(s)K(s,t)ds+ \int_C \theta_k[\chi(s)]\tilde{K}(s,t)ds\right\}$$

$$= E(t) + O(h^3) \quad \text{if } t\epsilon S_N \cup C^2 ,$$

$$\alpha(t)\tau_A[\chi(t)]+ \sum_{k=1}^{NT} \tau_k\left\{ \int_S \theta_k(s)K(s,t)ds+ \int_C \theta_k[\chi(s)]\tilde{K}(s,t)ds\right\} \tag{18}$$

$$= E(t) + O(h^3) \quad \text{if } t\epsilon C^1 ,$$

and

$$\sum_{k=1}^{NT} \tau_k\left\{ \int_S \theta_k(s)K(s,t)ds + \int_C \theta_k[\chi(s)]\tilde{K}(s,t)ds\right\}$$

$$= E(t) + O(h^3) \quad \text{if } t\epsilon S_D .$$

Let I_1 be the set of indices k such that $t_k \epsilon S_N \cup C^2$; I_2, such that $t_k \epsilon C^1$; and I_3, such that $t_k \epsilon S_D$. Thus, $I_1 \cup I_2 \cup I_3$ is the set of indices k = 1, 2, . . . , NT. To simplify the problem of approximating $\tau_A(s)$, Eq. (18) will be required to hold only for NT values of the variable t. The choice of the NT points is arbitrary. Except for special cases to be discussed below, the points t_k for k = 1, 2, . . . , NT will be used. Define $E_i = E(t_i)$, $E = (E_1, E_2, \ldots, E_{NT})^T$, and $\alpha_i = \alpha(t_i)$. Given $t_i \epsilon C$, define i' to be that integer such that $t_{i'} = \chi(t_i)$. That such a correspondence does exist is guaranteed by our restrictions on the input. Using the above definitions, we get the following NT linear equations with constant coefficients.

$$\alpha_i\tau_i + \sum_{k=1}^{NT} \tau_k\left\{\int_S \theta_k(s)K(s,t_i)ds+ \int_C \theta_k[\chi(s)]\tilde{K}(s,t_i)ds\right\}$$

$$= E_i + O(h^3) \quad \text{if } i\epsilon I_1 ,$$

$$\alpha_i\tau_{i'} + \sum_{k=1}^{NT} \tau_k\left\{\int_S \theta_k(s)K(s,t_i)ds+ \int_C \theta_k[\chi(s)]\tilde{K}(s,t_i)ds\right\} \tag{19}$$

$$= E_i + O(h^3) \quad \text{if } i\epsilon I_2 ,$$

and

$$\sum_{k=1}^{NT} \tau_k\left\{\int_S \theta_k(s)K(s,t_i)ds + \int_C \theta_k[\chi(s)]\tilde{K}(s,t_i)ds\right\}$$

$$= E_i + O(h^3) \quad \text{if } i\epsilon I_3 .$$

For i, k = 1, 2, . . . , NT set

$$a_{ik} = \begin{cases} \alpha_i + \int_S \theta_k(s)K(s,t_i)ds + \int_C \theta_k[\chi(s)]\tilde{K}(s,t_i)ds \\ \quad \text{if } (i\epsilon I_1 \text{ and } i = k) \text{ or } (i\epsilon I_2 \text{ and } i' = k), \\ \int_S \theta_k(s)K(s,t_i)ds + \int_C \theta_k[\chi(s)]\tilde{K}(s,t_i)ds \\ \quad \text{otherwise.} \end{cases}$$

Now set $A = (a_{ik})$ and $\tau^o = (\tau_1, \tau_2, \ldots, \tau_{NT})^T$. Using these definitions, we can write the system of Eqs. (19) in the form

$$A\tau^o = E + O(h^3) . \tag{20}$$

All of the coefficients of A and E can be computed numerically to any desired accuracy from a knowledge of S, f(s), and g(s).

The matrix A does not, in general, have an inverse. We will now dicuss this problem and its remedies. The remedies described below, like the choice of approximation points in Eq. (19), are arbitrary. Considering each subregion separately, at the end points of each of the boundary sections S_j for j = 1, 2, . . . , NDCT, two ends of curves are joined. The curves may, or may not, join at a corner, and may, or may not, have the same boundary conditions. These conditions determine how the matrix A is to be modified so that it will have an inverse. For each joined pair of end points of boundary sections, we must determine which of the following possibilities is true.

1. One curve is in S_N, and the other is in S_D.
2. Both curves are in S_D, and there is a corner where the curves join.
3. Both curves are in S_D, and there is no corner where the curves join or both curves are in S_N.
4. One curve is in S_D, and the other is in C^2.
5. One curve is in S_D, and the other is in C^1.
6. One curve is in S_N, and the other is in C^1.
7. One curve is in S_N, and the other is in C^2.
8. Both curves are in C^2.
9. Both curves are in C^1, and there is a corner where the curves join.
10. Both curves are in C^1, and there is no corner where the curves join.
11. One curve is in C^1, and the other is in C^2.

For simplicity, assume that curve S_i joins S_k and that the corresponding values of t are t_{il} and t_{kl}. Moreover, assume that t_{il} is the start of S_i. In what follows we will assume in each subregion, that u(s) is continuous along S, and that $\partial u/\partial \nu$ is continuous along S except at corners.

For case 1, the matrix A need not be modified. However we can replace an approximate equation by an exact equation. To see this, assume that the Dirichlet boundary is given on S_i. Obviously, $\tau(t_{kl}) = f(t_{il})$, and, thus, $\tau_{kl} = f(t_{il})$. We can then replace the kl equation in Eqs. (19) by

$$\tau_{kl} = f(t_{il}) \quad .$$

In case 2 it is relatively easy to see that the kl equation in Eqs. (19) is the same as the il equation. Hence, A must certainly be singular. To correct this problem we need only replace the two equations by the third equation in Eqs. (18) evaluated at the two points $t = t_{il} + h_i/2$ and $t = t_{kl} - h_k/2$. The author has tried replacing only one of the two equations, but the results were not so satisfactory as when both equations were replaced.

For case 3, $\tau(t_{il}) = \tau(t_{kl})$ which implies that $\tau_{il} = \tau_{kl}$. If both curves are in S_D, then the il and kl equations in Eqs. (19) are the same. If both curves have the Neumann condition given, then the equations are the same except that the coefficients for τ_{il} and τ_{kl} are interchanged. In any case, one of the equations can be replaced by

$$\tau_{kl} - \tau_{il} = 0 \quad .$$

If both curves have the Neumann condition given, then the change does not have to be made.

Cases 4 and 5 are related. Assume that $S_i \epsilon S_D$ and $S_k \epsilon C$. For case 4, replace the il equation in Eqs. (19) by the third equation in Eqs. (18) evaluated at $t = t_{il} + h_i/2$, and replace the kl equation by the first equation in Eqs. (18) evaluated at $t = t_{kl} - h_k/2$. For case 5 we simply replace the kl equation by

$$\tau_{kl'} = f(t_{il}) \quad .$$

Cases 6 and 7 are treated almost the same. Assume that $S_i \epsilon S_N$ and $S_k \epsilon C$. For case 6, we simply replace the il equation in Eqs. (19) by

$$\tau_{il} - \tau_{kl} = 0 \quad .$$

For case 7, we simply replace the il equation in Eqs. (19) by

$$\tau_{il} - \tau_{kl'} = 0 \quad .$$

Cases 8, 9, and 10 are related. Case 10 could be handled by the same method as case 9, but the method of treatment in case 10 is more accurate. For cases 8 and 10, we replace either the il or kl equation in Eqs. (19) by

$$\tau_{kl} - \tau_{il} = 0 \quad .$$

For case 9, we simply replace the il and kl equations of Eqs. (19) by the second part of Eqs. (18) evaluated at $t = t_{il} + h_i/2$ and $t = t_{kl} - h_k/2$.

Case 11 can occur only when at least three subregions meet at one point. Assume that $S_i \epsilon C^1$ and $S_k \epsilon C^2$. Then change the kl equation to

$$\tau_{il'} - t_{kl} = 0 \quad .$$

If the stated problem happens to be interior and to have no Dirichlet boundary conditions, then the potential u is unique only up to an additive constant. To fix this, the restriction

$$\int_S \tau_A(s)ds = 0$$

is imposed for programs (i) and (iii), and the restriction

$$\int_S \tau_A(s)rds = 0$$

is imposed for programs (ii) and (iv). It has been found satisfactory to simply replace the equation corresponding to t_2 in Eqs. (19) by this restriction. For the Neumann problem on an infinite region G, no restrictions on $\tau_A(s)$ are necessary because of the assumption that $u(v,w) \to 0$ as $(v,w) \to \infty$.

VIII. APPROXIMATION OF A AND E IN THE x-y PLANE

The approximations to be discussed here can also be used to evaluate Eq. (1) once the unknown boundary values have been computed. For both the known and unknown boundary values, we approximate

$$u(s) \approx u(s)_A = \sum_{k=1}^{NT} u(t_k)\theta_k(s) \quad ,$$

and

$$\frac{\partial u(s)}{\partial \nu} \approx \left[\frac{\partial u(s)}{\partial \nu}\right]_A = \sum_{k=1}^{NT} \frac{\partial u(t_k)}{\partial \nu}\theta_k(s) \quad ,$$

where $\theta_k(s)$ is as defined above. Replacing the functions u and $\partial u/\partial \nu$ by their approximations in Eqs. (1) and (5) shows that one need only be able to approximate

$$\int_S \theta_k(s)\ln\left(\frac{1}{r}\right)ds \quad ,$$

and (21)

$$\int_S \theta_k(s)\frac{\partial}{\partial \nu}\ln\left(\frac{1}{r}\right)ds \quad .$$

Here $r = r(s,x,y) = \{[x(s)-x]^2 + [y(s)-y]^2\}^{\frac{1}{2}}$. To evaluate the approximate solution, the pair (x,y) must be allowed to range over all G and S. When we evaluate A and E, the pair (x,y) is restricted to the values $[x(t_k),y(t_k)]$ for $k = 1, 2, \ldots, NT$ and, possibly, the intermediate values at the end points of the boundary sections.

Three different methods are used to approximate the integrals in Eq. (21). To explain the three methods, we need more notation. Let $T_k = [s|\theta_k(s) \neq 0]$. Define the function h(s) for $s\epsilon(0,d)$ by

$$h(s) = h_k \text{ for } \sum_{i=1}^{k-1} LS_i < s \leq \sum_{i=1}^{k} LS_i \quad .$$

Given a fixed $\theta_k(s)$ and a point (x,y), the three methods correspond approximately to the three cases:

(a) For every $s\epsilon T_k$ $r(s,x,y) \geq 7h(s)$,

(b) For some $s\epsilon T_k$ $r(s,x,y) = 0$,

(c) The complement of (a) and (b).

For case (a), the functions $\partial/\partial\nu \ln[r(s,x,y)]$ and $\ln[r(s,x,y)]$ are smooth when thought of as functions of the variable s. The integrals in Eq. (21) can be approximated accurately by a Newton-Cotes formula. For example, assume that t_k is the start of the boundary section $S_j(k)$. Looking at the first integral in Eq. (21), we have

$$\int_S \theta_k(s)\ln\left[\frac{1}{r(s,x,y)}\right]ds$$

$$= \frac{1}{2h_{j(k)}^2}\int_{t_k}^{t_{k+2}} (s-t_{k+1})(s-t_{k+2})\ln\left[\frac{1}{r(s,x,y)}\right]ds$$

$$= \frac{1}{2h_{j(k)}^2}\int_0^{2h_{j(k)}} \left[s-h_{j(k)}\right]\left[s-2h_{j(k)}\right]\ln\left[\frac{1}{r(s+t_k,x,y)}\right]ds.$$

For i = 0, 1, 2, 3, 4, let $\xi_i = -\ln\left\{r\left[ih_{j(k)}/2+t_k,x,y\right]\right\}$. Using Eq. 3.5.13 of Ref. 6, we get

$$\frac{1}{2h_{j(k)}^2}\int_0^{2h_{j(k)}} \left[s-h_{j(k)}\right]\left[s-2h_{j(k)}\right]\ln\left[\frac{1}{r(s+t_k,x,y)}\right]ds$$

$$\approx \frac{h_{j(k)}}{45}\left[7(1)\xi_0+32\left(\tfrac{3}{8}\right)\xi_1+12(0)\xi_2+32\left(-\tfrac{1}{8}\right)\xi_3+7(0)\xi_4\right]$$

$$= \frac{h_{j(k)}}{45}\left(7\xi_0+12\xi_1-4\xi_3\right) \quad . \qquad (22)$$

By combining the terms, we need compute only one logarithm. The second integral in Eq. (21) can be

treated in exactly the same way. When $\Theta_k(s)$ is more involved, we have only to sum two integrals similar to the one given above. The method discussed for case (a) can be replaced by that used for case (c). However, the method used for case (a) is faster.

For cases (b) and (c), the approximation is more difficult. Looking at the definition of $\Theta_k(s)$, one can easily see that to compute the integrals in Eq. (21) it is sufficient to be able to compute

$$N(k,i) = \int_{t_k}^{t_{k+1}} (s-t_k)^i \ln\left[\frac{1}{r(s,x,y)}\right]ds \quad ,$$

and (23)

$$D(k,i) = \int_{t_k}^{t_{k+1}} (s-t_k)^i \frac{\partial}{\partial \nu} \ln\left[\frac{1}{r(s,x,y)}\right]ds$$

for $k = 1, 2, \ldots, NT-1$ and $i = 0, 1, 2$. Thus, we will discuss only the approximation of the integrals in Eq. (23). As a point of interest, note that $D(k,0)$ can always be computed explicitly. However, this fact will not be used.

Using the definition of $\partial/\partial\nu$ and $r(s,x,y)$, it follows that

$$D(k,i) = \int_{t_k}^{t_{k+1}} (s-t_k)^i \frac{\frac{\partial}{\partial \nu} r(s,x,y)}{r(s,x,y)}\, ds$$

$$= \int_{t_k}^{t_{k+1}} (s-t_k)^i \frac{\left\{[x(s)-x]\frac{dy(s)}{ds} - [y(s)-y]\frac{dx(s)}{ds}\right\}}{r^2(s,x,y)} ds.$$

In what immediately follows, we restrict the discussion to case (c). For (x,y) fixed and $s\epsilon(t_k,t_{k+1})$, the functions $\{[x(s)-x]y'(s) - [y(s)-y]x'(s)\}$ and $r^2(s,x,y)$ will be approximated by polynomials of degree two. Set

$$p(s-t_k) = as^2 + bs + c \approx r^2(s,x,y) \quad ,$$

and

$$q(s-t_k) = \alpha s^2 + \beta s + \gamma \approx \{[x(s)-x]y'(s) - [y(s)-y]x'(s)\} \quad , \qquad (24)$$

and choose a, b, c, α, β, and γ such that the above approximations are true at $s = t_k$, t_{k+1} and $(t_{k+1}+t_k)/2$. Define $D_A(k,i)$ and $N_A(k,i)$ as the approximations to the integrals $D(k,i)$ and $N(k,i)$ resulting from the use of the polynomials $p(s)$ and $q(s)$. Then we have

$$N(k,i) \approx N_A(k,i) = -\int_{t_k}^{t_{k+1}} (s-t_k)^i \ln[p(s-t_k)]ds$$

$$= -\int_0^{h_{j(k)}} s^i \ln[p(s)]ds \quad ,$$

and (25)

$$D(k,i) \approx D_A(k,i) = \int_{t_k}^{t_{k+1}} (s-t_k)^i \frac{q(s-t_k)}{p(s-t_k)}\, ds$$

$$= \int_0^{h_{j(k)}} \frac{s^i q(s)}{p(s)}\, ds \quad .$$

Both $N_A(k,i)$ and $D_A(k,i)$ can be computed explicitly for any given polynomials $p(s)$ and $q(s)$ of degree two. However, round-off can be a problem whenever $|ah^2_{j(k)}/c|$ gets small.

For case (b), $r(s,x,y) = 0$ for some $s\epsilon(t_k,t_{k+1})$. If we are generating the matrix A and the vector E, and $r(s,x,y) = 0$ for some $s\epsilon(t_k,t_{k+1})$, then either $r(t_k,x,y) = 0$, $r(t_{k+1},x,y) = 0$, or $r[(t_{k+1}+t_k)/2,x,y] = 0$. If we compute values of the solution, and $r(s,x,y)$ grows small or is zero for some $s\epsilon(t_k,t_{k+1})$, then we can use the values of u and $\partial u/\partial n$ computed at the closest point on the boundary to give an approximation to the solution. Hence, we can assume that $r(s,x,y)$ is only zero when generating A and E. Here we will assume that $r(t_k,x,y) = 0$. The other cases are treated similarly.

The integral $N(k,i)$ is approximated in somewhat the same way as in case (c). The only difference is that $p(s)$ is chosen differently. Here $p(s) = s^2(as+b)$, with a and b chosen so that $p(s-t_k) = r^2(s,x,y)$ at $s = t_{k+1}$ and $(t_k+t_{k+1})/2$. Hence, we also get $p(s-t_k) = r^2(s,x,y)$ and

$$\frac{d}{ds}p(s-t_k) = \frac{dr^2(s,x,y)}{ds} \quad \text{at } s = t_k.$$

Using the approximation

$$N(k,i) \approx N_A(k,i) = -\tfrac{1}{2}\int_{t_k}^{t_{k+1}} (s-t_k)^i \ln[p(s-t_k)]ds$$

$$= -\tfrac{1}{2}\int_0^{h_{j(k)}} s^i \ln[p(s)]ds \quad ,$$

it follows that $N_A(k,i)$ can be integrated explicitly.

However, there can be problems due to round-off if $|a_{h_{j(k)}}/b|$ is small.

For case (b) the integral D(k,i) is treated as in case (a). However, the integrand

$$\frac{\partial}{\partial \nu} \ln\left[\frac{1}{r(s,x,y)}\right] = -\frac{\{[x(s)-x]y'(s)-[y(s)-y]x'(s)\}}{r^2(s,x,y)}$$

is not well-defined at $s = t_k$ because both the numerator and denominator are zero. Using L'Hospital's rule, it is easy to show that

$$\lim_{s\to t_k} \frac{\partial}{\partial \nu} \ln\left[\frac{1}{r(s,x,y)}\right] = \tfrac{1}{2}[y'(t_k)x''(t_k)-y''(t_k)x'(t_k)] .$$

Now define a polynomial $q(s) = \alpha s^2 + \beta s + \gamma$ such that

$$q(0) = \tfrac{1}{2}[y'(t_k)x''(t_k)-x'(t_k)y''(t_k)] ,$$

and

$$q(s-t_k) = \frac{\partial}{\partial \nu} \ln\left[\frac{1}{r(s,x,y)}\right] \text{ for } s = t_{k+1}, \frac{(t_k+t_{k+1})}{2} .$$

Then, approximate

$$D(k,i) \approx D_A(k,i) = \int_{t_k}^{t_{k+1}} (s-t_k)^i q(s-t_k)ds = \int_0^{h_{j(k)}} s^i q(s)ds .$$

If the boundary is a line segment or a circular arc for $s\epsilon(t_k,t_{k+1})$, and $r(s,x,y) = 0$ for some $s\epsilon(t_k,t_{k+1})$, then $\partial/\partial\nu \ln[1/r(s,x,y)]$ is constant and the approximation is exact.

IX. APPROXIMATION OF A AND E IN THE z-r PLANE

The approximations discussed here are more involved than those in the previous section. The goal is to be able to approximate the integrals of Eq. (26), given below, to within a relative error of about 10^{-8}. This is absolutely the best error estimate that one can expect for any problem using programs (ii) and (iv). The method of approximation discussed here is applicable to at least one other partial differential equation and probably more.

The approximations here, just as in the x-y plane, can also be used to evaluate Eq. (14) as well as to compute A and E. Define $u_A(s)$ and $\partial u_A(s)/\partial\nu$ just as in Sec. VIII. Again, it is obvious that one need only be able to evaluate

$$\int_S \rho\theta_k \frac{2K(m)dS}{\left[(z-\zeta)^2+(\rho-r)^2+4r\rho\right]^{\frac{1}{2}}}$$

and (26)

$$\int_S \rho\theta_k \frac{\partial}{\partial\nu} \frac{2K(m)dS}{\left[(z-\zeta)^2+(\rho-r)^2+4r\rho\right]^{\frac{1}{2}}} .$$

Here $m = \dfrac{4r\rho}{(z-\zeta)^2+(r-\rho)^2+4r\rho}$.

The two integrals in Eq. (26) are evaluated by using Gaussian quadrature schemes. However, the number of points in the scheme and the type of Gaussian quadrature scheme vary. The number of points used in the Gaussian quadrature scheme for any given case was derived from a numerical study. As in the x-y plane, the integrals of Eq. (26) can be reduced to integrals of the form

$$\tilde{N}(k,i) = \int_{t_k}^{t_{k+1}} (s-t_k)^i \rho(s) \frac{2K[m(s)]ds}{\left\{[z-\zeta(s)]^2+[r-\rho(s)]^2+4r\rho(s)\right\}^{\frac{1}{2}}} ,$$

$$\tilde{D}(k,i) = \int_{t_k}^{t_{k+1}} (s-t_k)^i \rho(s) \frac{\partial}{\partial\nu} \frac{2K[m(s)]ds}{\left\{[z-\zeta(s)]^2+[r-\rho(s)]^2+4r\rho(s)\right\}^{\frac{1}{2}}} , \quad (27)$$

for i = 0, 1, 2, and k = 1, 2, . . . , NT - 1. There are four different types of Gaussian quadrature schemes used to approximate the integrals of Eqs. (27). They correspond to the four cases:

(a) $[z-\zeta(s)]^2 + [r-\rho(s)]^2 \neq 0$ for all $s\epsilon(t_k,t_{k+1})$ and $r \neq 0$,

(b) $[z-\zeta(s)]^2 + [r-\rho(s)]^2 = 0$ for some $s\epsilon(t_k,t_{k+1})$ and $r \neq 0$,

(c) $[z-\zeta(s)]^2 + [r-\rho(s)]^2 \neq 0$ for all $s\epsilon(t_k,t_{k+1})$ and $r = 0$,

(d) $[z-\zeta(s)]^2 + [r-\rho(s)]^2 = 0$ for some $s\epsilon(t_k,t_{k+1})$ and $r = 0$.

Before describing the schemes, we will examine the integrand for $\tilde{D}$ more closely. Let $d_o^2=(z-\zeta)^2+(\rho-r)^2$. Then

$$\frac{2K(m)}{\left[(z-\zeta)^2+(r-\rho)^2+4r\rho\right]^{\frac{1}{2}}} = \frac{2K(m)}{\left[d_o^2+4r\rho\right]^{\frac{1}{2}}} = \frac{K(m)m^{\frac{1}{2}}\rho^{-\frac{1}{2}}}{r^{\frac{1}{2}}} .$$

Using the derivative formula given in Ref. 10, p. 521, we get

$$\frac{1}{r^{\frac{1}{2}}}\frac{\partial}{\partial\nu}\left[K(m)m^{\frac{1}{2}}\rho^{-\frac{1}{2}}\right] = \frac{1}{r^{\frac{1}{2}}}\left\{\tfrac{1}{2}m^{-\frac{1}{2}}\rho^{-\frac{1}{2}}K(m)\frac{\partial m}{\partial\nu} - \tfrac{1}{2}m^{\frac{1}{2}}\rho^{-3/2}K(m)\frac{\partial\rho}{\partial\nu}\right.$$

$$\left. + \tfrac{1}{2}\left[\frac{E(m)-K(m)m'}{mm'}\right]m^{\frac{1}{2}}\rho^{-\frac{1}{2}}\frac{\partial m}{\partial\nu}\right\}$$

$$= \frac{1}{(4r\rho m)^{\frac{1}{2}}}\left\{K(m)\frac{\partial m}{\partial\nu} - m\frac{K(m)}{\rho}\frac{\partial\rho}{\partial\nu} + \left[\frac{E(m)}{m'} - K(m)\right]\frac{\partial m}{\partial\nu}\right\}$$

$$= \frac{1}{(4r\rho m)^{\frac{1}{2}}}\left[\frac{E(m)}{m'}\frac{\partial m}{\partial\nu} - m\frac{K(m)}{\rho}\frac{\partial\rho}{\partial\nu}\right] .$$

Here E(m) is the elliptic integral of the second kind, and m' = 1 - m. Now if t is the integration variable used on the curve S, then

$$\frac{\partial\rho(t)}{\partial\nu} = -\frac{d\zeta(t)}{dt} = -\dot{\zeta}(t) ,$$

and

$$\frac{\partial m(t)}{\partial\nu} = \frac{\left\{\left[d_o^2+4r\rho(t)\right]4\frac{\partial\rho(t)}{\partial\nu}r - 4\rho(t)r\left[\frac{\partial}{\partial\nu}\,d_o^2+4\frac{\partial\rho(t)}{\partial\nu}\,r\right]\right\}}{\left[d_o^2+4\rho(t)r\right]^2}$$

$$= \frac{\left[d_o^2 4r\,\frac{\partial\rho(t)}{\partial\nu} - 4\rho(t)r\,\frac{\partial d_o^2}{\partial\nu}\right]}{\left[d_o^2+4\rho(t)r\right]^2}$$

$$= \frac{\left(-d_o^2 4r\dot{\zeta}(t) - 8\rho(t)r\left\{\left[\zeta(t)-z\right]\dot{\rho}(t) - \left[\rho(t)-r\right]\dot{\zeta}(t)\right\}\right)}{\left[d_o^2+4\rho(t)r\right]^2} .$$

Hence,

$$\frac{\partial}{\partial\nu}\,\frac{2K(m)}{\left[d_o^2+4r\rho\right]^{\frac{1}{2}}} = \tfrac{1}{2}\left(\frac{m}{r\rho}\right)^{\frac{1}{2}}\left[\frac{E(m)}{mm'}\frac{\partial m}{\partial\nu} - \frac{K(m)}{\rho}\frac{\partial\rho}{\partial\nu}\right]$$

$$= \tfrac{1}{2}\left(\frac{m}{r\rho}\right)^{\frac{1}{2}}\left[\frac{E(m)\left(d_o^2+4\rho r\right)^2}{\left[4\rho r d_o^2\right]^{\frac{1}{2}}}\frac{\partial m}{\partial\nu} - \frac{K(m)}{\rho}\frac{\partial\rho}{\partial\nu}\right]$$

$$= \frac{K(m)\dot{\zeta}}{\rho\left(d_o^2+4r\rho\right)^{\frac{1}{2}}} + \frac{\left(d_o^2+4r\rho\right)^{3/2}E(m)}{4r\rho d_o^2}$$

$$\left\{\frac{-d_o^2 4r\dot{\zeta} - 8\rho r\left[(\zeta-z)\dot{\rho} - (\rho-r)\dot{\zeta}\right]}{\left(d_o^2+4r\rho\right)^2}\right\}$$

$$= \frac{K(m)\dot{\zeta}}{\rho\left(d_o^2+4r\rho\right)^{\frac{1}{2}}} + \frac{E(m)}{\rho\left(d_o^2+4r\rho\right)^{\frac{1}{2}}}\left\{-\dot{\zeta} - 2\rho\left[\frac{(\zeta-z)\dot{\rho} - (\rho-r)\dot{\zeta}}{d_o^2}\right]\right\}$$

$$= \frac{1}{\rho\left(d_o^2+4r\rho\right)^{\frac{1}{2}}}\left(K(m)\dot{\zeta} - E(m)\left\{\dot{\zeta} + 2\rho\left[\frac{(\zeta-z)\dot{\rho} - (\rho-r)\dot{\zeta}}{d_o^2}\right]\right\}\right) . \quad (28)$$

Thus, the integrand for $\tilde{D}(k,i)$ has a logarithmic singularity and a singularity similar to D(k,i) in the x-y plane. The integrand for $\tilde{N}$ has only a logarithmic singularity. Now we will discuss the Gaussian quadrature scheme corresponding to case (a). Let $P_1(s)$, $P_2(s)$, $P_3(s)$, and $P_4(s)$ be second-degree polynomials, and let $P_5(s)$ be a third-degree polynomial such that

1. $P_1(s-t_k) = \rho(s)$ for $s = t_k, \frac{t_{k+1}+t_k}{2}, t_{k+1}$,

2. $P_2(s-t_k) = d_o^2(s)$ for $s = t_k, \frac{t_{k+1}+t_k}{2}, t_{k+1}$,

3. $P_3(s-t_k) = [\zeta(s)-z]\dot{\rho}(s) - [\rho(s)-r]\dot{\zeta}(s)$ for $s = t_k, \frac{t_{k+1}+t_k}{2}, t_{k+1}$,

4. $P_4(s-t_k) = \dot{\zeta}(s)$ for $s = t_k, \frac{t_{k+1}+t_k}{2}, t_{k+1}$,

and

5. $P_5(s-t_k) = d_o^2(s)$ for $s = t_k, \frac{t_{k+1}+t_k}{2}, t_{k+1}$ and $\frac{d}{ds}P_5(s-t_k) = \frac{d}{ds}\,d_o^2(s)$ for $s = \frac{t_{k+1}+t_k}{2}$.

Define $EM(s-t_k) = 4rP_1(s-t_k)/[4rP_1(s-t_k)+P_5(s-t_k)]$. Let $w_1, w_2, \ldots, w_{LM}$ be the weights, and $\tilde{t}_1, \tilde{t}_2, \ldots, \tilde{t}_{LM}$ be the abscissas, for an LM point Gaussian quadrature scheme with weight function 1. That is,

$$\int_0^1 f(t)dt \approx \sum_{\ell=1}^{LM} w_\ell f(\tilde{t}_\ell) .$$

See Stroud and Secrest.[11] Set $\tilde{s}_\ell = \tilde{t}_\ell h_{j(k)}$ for $\ell = 1, 2, \ldots, LM$. Now, using the change of variables, $t = (s-t_k)/h_{j(k)}$, we appropriate the integral

$$\tilde{N}(k,i) = \int_0^1 \left[th_{j(k)}\right]^i \rho\left[th_{j(k)}+t_k\right] \frac{2K\left\{m\left[th_{j(k)}+t_k\right]\right\}h_{j(k)}dt}{\left\{d_o^2\left[th_{j(k)}+t_k\right]+4r\rho\left[th_{j(k)}+t_k\right]\right\}^{\frac{1}{2}}}$$

by

$$\tilde{N}(k,i) \approx h_{j(k)}\sum_{\ell=1}^{LM} w_\ell\left\{\left[(s_\ell)^i P_1(s_\ell)\right]\frac{2K[EM(s_\ell)]}{\left[P_5(s_\ell)+4rP_1(s_\ell)\right]^{\frac{1}{2}}}\right\} ,$$

and the integral

$$\tilde{D}(k,i) = \int_0^1 \left[th_{j(k)}\right]^i \rho\left[th_{j(k)}+t_k\right] \frac{\partial}{\partial \nu} \left(\frac{2K\{m[th_{j(k)}+t_k]\}h_{j(k)}}{\left\{d_o^2\left[th_{j(k)}+t_k\right]+4r\rho\left[th_{j(k)}+t_k\right]\right\}^{\frac{1}{2}}}\right) dt$$

by

$$\tilde{D}(k,i) = h_{j(k)} \sum_{\ell=1}^{IM} w_\ell(s_\ell)^i \left(\{K[EM(s_\ell)]-E[EM(s_\ell)]\}P_4(s_\ell)-E[EM(s_\ell)]P_1(s_\ell)P_3(s_\ell)/P_2(s_\ell)\right)/\left[P_5(s_\ell)+4rP_1(s_\ell)\right]^{\frac{1}{2}}.$$

The choice of IM here depends on the minimum values of the ratios $d_o(s)/h_{j(k)}$ and $\rho(s)/h_{j(k)}$ for $s\epsilon(t_k,t_{k+1})$. IM is chosen so that the integrals have a relative error of about 10^{-8}. One might reasonably ask why two different polynomials are used to approximate $d_o^2(s)$. The approximation to $d_o^2(s)$ is most critical. Hence, the third-degree polynomial $P_5(s)$ is used. The ratio $\{[\zeta(s)-z]\dot{\rho}(s)-[\rho(s)-r]\dot{\zeta}(s)\}/d_o^2(s)$ is also important. Moreover, as was mentioned for the x-y plane, this ratio is constant for many important cases. Without modifying the input, it is not practical to approximate the numerator by a similar third-degree polynomial, and if one does not, the ratio of the polynomial approximations is not constant for the important cases. Hence, we use polynomials that will make the ratio constant for these cases.

We discuss next the Gaussian quadrature scheme corresponding to case (b). For simplicity we assume that $d_o^2(t_k) = 0$. Because of our assumptions concerning the parametric representation of the boundary, there exists a smooth function $\tilde{d}_o^2(s)$, such that $d_o^2(s) = (s-t_k)^2\,\tilde{d}_o^2(s)$ and $\tilde{d}_o^2(t_k) = 1$. There also exist four polynomials Q_1, Q_2, Q_3, and Q_4 each of degree four such that

$$K(m) \approx Q_1(m') + Q_2(m')\log(m') \quad ,$$

$$E(m) \approx Q_3(m') + Q_4(m')\log(m') \quad ,$$

and the maximum relative error in these two approximations is 2×10^{-8}. See Ref. 9, pp. 591-592. Because

$$m' = 1 - m = d_o^2(s)/(d_o^2+4r\rho) = (s-t_k)^2\,\tilde{d}_o^2(s)/(d_o^2+4r\rho) \quad ,$$

we have

$$K(m) \approx Q_1(m') + Q_2(m')\log\left(\frac{\tilde{d}_o^2}{d_o^2+4r\rho}\right) + Q_2(m')\log[(s-t_k)^2] \quad ,$$

and

$$E(m) \approx Q_3(m') + Q_4(m')\log\left(\frac{\tilde{d}_o^2}{d_o^2+4r\rho}\right) + Q_4(m')\log[(s-t_k)^2] \quad .$$

For ease of notation below, set

$$\tilde{m}' = \tilde{d}_o^2/(d_o^2+4r\rho) \quad ,$$

$$K_1(m,\tilde{m}') = Q_1(m') + Q_2(m')\log(\tilde{m}') \quad ,$$

$$K_2(m) = Q_2(m')\log[(s-t_k)^2] \quad , \qquad (29)$$

$$E_1(m,\tilde{m}') = Q_3(m') + Q_4(m')\log(\tilde{m}')$$

and

$$E_2(m) = Q_4(m')\log[(s-t_k)^2] \quad .$$

Let P_1, P_5, and P_4 be the same polynomials defined above. Choose P_2 and P_3 such that

$$P_2(s-t_k) = \tilde{d}_o^2(s) \quad \text{at } s = t_k,(t_{k+1}+t_k)/2,\ t_{k+1} \quad ,$$

and

$$P_3(s-t_k) = \lim_{t\downarrow s} \{[\zeta(t)-z]\dot{\rho}(t)-[\rho(t)-r]\dot{\zeta}(t)\}/d_o^2(s) \quad \text{at } s = t_k,\ (t_{k+1}+t_k)/2,\ t_{k+1} \quad .$$

Set $EM(s) = 4rP_1(s)/[4rP_1(s)+P_5(s)]$ and $\widetilde{EM}(s) = P_2(s)/[4rP_1(s)+P_5(s)]$. Let $\omega_1, \omega_2, \ldots, \omega_{IM}$ be the weights and let $\xi_1, \xi_2, \ldots, \xi_{IM}$ be the abscissas for an IM point Gaussian quadrature scheme with weight function $\log(t)$. That is,

$$\int_0^1 \log(t)f(t)dt \approx \sum_{\ell=1}^{LM} w_\ell f(\xi_\ell) \quad .$$

See Stroud and Secrest.[11] Set $\eta_\ell = \xi_\ell h_j(k)$ for $\ell = 1, 2, \ldots, LM$. Using the change of variables $t = (s-t_k)/h_j(k)$, we approximate the integral $\mathfrak{N}(k,i)$ by

$$N(k,i) \approx h_j(k) \sum_{\ell=1}^{LM} w_\ell (s_\ell)^i P_1(s_\ell) \left\{ \frac{2K_1[EM(s_\ell),\widetilde{EM}(s_\ell)]}{[P_5(s_\ell)+4rP_1(s_\ell)]^{\frac{1}{2}}} \right\}$$

$$+ 2h_j(k) \sum_{\ell=1}^{LM} w_\ell (\eta_\ell)^i P_1(\eta_\ell) \left\{ \frac{2K_2[EM(\eta_\ell)]}{[P_5(\eta_\ell)+4rP_1(\eta_\ell)]^{\frac{1}{2}}} \right\},$$

and the integral $\mathfrak{D}(k,i)$ by

$$\mathfrak{D}(k,i) \approx h_j(k) \sum_{\ell=1}^{LM} w_\ell s_\ell^i \Big([K_1[EM(s_\ell),\widetilde{EM}(s_\ell)]$$

$$- E_1[EM(s_\ell),\widetilde{EM}(s_\ell)]]P_4(s_\ell) - E_1[EM(s_\ell),$$

$$\widetilde{EM}(s_\ell)]P_1(s_\ell)P_3(s_\ell)\Big)/[P_5(s_\ell)+4rP_1(s_\ell)]^{\frac{1}{2}}$$

$$+ 2h_j(k) \sum_{\ell=1}^{LM} w_\ell(\eta_\ell)^i \Big(\{K_2[EM(\eta_\ell)]-E_2[EM(\eta_\ell)]\}$$

$$P_4(\eta_\ell) - E_2[EM(\eta_\ell)]P_1(\eta_\ell)P_3(\eta_\ell)\Big)/[P_5(\eta_\ell)$$

$$+ 4rP_1(\eta_\ell)]^{\frac{1}{2}} \quad .$$

The s_ℓ's and w_ℓ's used here are the same as those used in the previous approximations for $\mathfrak{N}$ and $\mathfrak{D}$.

Finally let us consider the two cases when $r = 0$. The two integrals in Eq. (27) then reduce to

$$\mathfrak{N}(k,i) = \int_{t_k}^{t_{k+1}} \frac{(s-t_k)^i \rho(s)ds}{d_o(s)} \quad ,$$

and

$$\mathfrak{D}(k,i) = \int_{t_k}^{t_{k+1}} \frac{(s-t_k)^i \rho(s)}{d_o^3} \left\{ \frac{[\zeta(s)-z]\dot{\rho}(s)-\rho\dot{\zeta}}{} \right\} ds \quad . \quad (30)$$

For case (c), P_1, P_2, and P_3 are the same as in case (a). $\mathfrak{N}$ and $\mathfrak{D}$ are approximated by

$$\mathfrak{N}(k,i) \approx h_j(k) \sum_{\ell=1}^{LM} w_\ell (s_\ell)^i P_1(s_\ell)/[P_2(s_\ell)]^{\frac{1}{2}} \quad ,$$

and

$$\mathfrak{D}(k,i) \approx h_j(k) \sum_{\ell=1}^{LM} w_\ell (s_\ell)^i P_1(s_\ell)P_3(s_\ell)/[P_2(s_\ell)]^{3/2} \quad .$$

For case (d), the approximation is similar, except that the polynomials are defined differently. Assuming that $d_o^2(t_k) = 0$, then P_1, P_2, and P_3 are defined by the relations

$$P_1(s-t_k) = \rho(s)/(s-t_k) \text{ for } s = (t_{k+1}+t_k)/2, \; t_{k+1}$$

$$P_1(s-t_k) = d\rho(s)/ds \text{ for } s = t_k \quad ,$$

$$P_2(s-t_k) = d_o^2(s-t_k)/(s-t_k)^2 \text{ for } s = (t_{k+1}+t_k)/2, \; t_{k+1}$$

$$P_2(s-t_k) = 1 \text{ for } s = t_k \quad ,$$

and

$$P_3(s-t_k) = \lim_{t \downarrow s} \left\{ \frac{[\zeta(s)-z]\dot{\rho}(s)-\rho\dot{\zeta}}{d_o^2(s)} \right\} \text{ for } s = t_k, \; \frac{t_{k+1}+t_k}{2}, \; t_{k+1} \quad .$$

$\mathfrak{N}$ and $\mathfrak{D}$ are approximated by

$$\mathfrak{N}(k,i) \approx h_j(k) \sum_{\ell=1}^{LM} w_\ell (s_\ell)^i P_1(s_\ell)/[P_2(s_\ell)]^{\frac{1}{2}} \quad ,$$

and

$$\mathfrak{D}(k,i) \approx h_j(k) \sum_{\ell=1}^{LM} w_\ell (s_\ell)^i P_1(s_\ell)P_3(s_\ell)/[P_2(s_\ell)]^{\frac{1}{2}} \quad .$$

X. MISCELLANEOUS MATHEMATICAL NOTES

Once the matrix A and the vector E have been generated, the matrix equation,

$$Az = E \quad , \quad (31)$$

must be solved for the vector $z = (z_1, z_2, \ldots, z_{NT})^T$. This problem can be solved by an iterative technique or by a direct method such as Gaussian elimination. For a Neumann problem, an iterative technique is usually faster than Gaussian elimination; for a Dirichlet problem, the opposite is true; and for a mixed problem, there is no fixed rule. Only Gaussian elimination is used in the four programs discussed here.

The error analysis that follows is directly applicable only to program (i). However, the results apply to each of the programs. Let z be the vector in Eq. (31). Define $z_A(s) = \sum_{k=1}^{NT} \theta_k(s) z_k$,

$$f_A(s) = \sum_{k \text{ with } t_k \epsilon S_D} \theta_k(s) f(t_k) \quad , \text{ and } \quad g_A(s) = \sum_{k \text{ with } t_k \epsilon S_N} \theta_k(s) g(t_k) \quad .$$

The approximate solution, $u_1(x,y)$, to the stated problem is

$$u_1(x,y) = \frac{1}{2\pi}\left[\int_{S_N} g_A(s)\ln(\tfrac{1}{r})ds + \int_{S_D} z_A(s)\ln(\tfrac{1}{r})ds - \int_{S_N} z_A(s)\,\frac{\partial}{\partial \nu}\ln(\tfrac{1}{r})ds - \int_{S_D} f_A(s)\,\frac{\partial}{\partial \nu}\ln(\tfrac{1}{r})ds\right] .$$

If the boundary S consists only of line segments, then all integrals can be evaluated exactly. In this case, we have

$$|u(x,y)-u_1(x,y)| \le \left[\frac{1}{2\pi}\left|\int_{S_N}(g-g_A)\ln(\tfrac{1}{r})ds\right| + \left|\int_{S_D}(\tau-z_A)\ln(\tfrac{1}{r})ds\right| + \left|\int_{S_N}(\tau-z_A)\frac{\partial}{\partial \nu}\ln(\tfrac{1}{r})ds\right| + \left|\int_{S_D}(f-f_A)\frac{\partial}{\partial \nu}\ln(\tfrac{1}{r})ds\right|\right] \le [1+\|A^{-1}\|]O(h^3) \quad .$$

Here we assume the existence of bounded third derivatives to get $|g-g_A| \le O(h^3)$, $|f-f_A| \le O(h^3)$, and $|\tau-z_A| \le |\tau-\tau_A| + |\tau_A-z_A| \le [1+\|A^{-1}\|]O(h^3)$. For the Neumann problem with a different discretization, it can be shown that $\|A^{-1}\|$ is bounded independent of the number of points NT, if NT is large enough. For mixed problems and Dirichlet problems, numerical studies suggest that $\|A^{-1}\| \le O(1/h)$, except for a special case to be discussed later. Thus, using the scheme explained above, we can expect $O(h^2)$ accuracy. However, the order of accuracy is arbitrary, and we can use whatever order is convenient. When the boundary S does not consist of line segments, there is also an error due to the approximation of the integrals in Eq. (21). However, as long as we do not have cusps, or as long as h_j, divided by the minimum radius of curvature on S_j, remains small for $j = 1, 2, \ldots, NDCT$, this error remains negligible.

One might ask about the uniqueness of a solution, τ, to the system of Eqs. (11). What follows has been proved only for the x-y plane. If the region G is homogeneous and S has no corners, then Eq. (11) becomes

$$\tau(t) + \frac{1}{\pi}\int_S \tau(s)K(s,t)ds = E(t) \quad \text{for } t\epsilon S_N \quad ,$$

and

$$\frac{1}{\pi}\int_S \tau(s)K(s,t)ds = E(t) \quad \text{for } t\epsilon S_D \quad . \qquad (11\text{-}A)$$

Kellner[12] has studied the homogeneous problem associated with Eq. (11-A), i.e., $E(t) \equiv 0$. He has shown that if $S_D \neq 0$, then there is a nontrivial solution if and only if the transfinite diameter of G is equal to 1. See Hille[13] for a definition of transfinite diameter. Thus, we can expect the matrix A in Eq. (31) to be singular if, and only if, G has transfinite diameter 1. This problem does not occur very often, but it does occur with the unit circle. The difficulty can be eliminated by scaling the region G so that it does not have transfinite diameter 1. We can numerically compute the transfinite diameter of a given region G using programs (i) and (iii), by simply scaling the region G so that the matrix A is singular.

XI. PROGRAM STRUCTURE

We will attempt to explain how the four routines are programmed. Associated with each routine are a number of auxiliary subprograms. For instance, the subprogram LAPLDDC uses the subprograms:

(1) BC
(2) LAPLOT
(3) ECSW
(4) FN
(5) QV
(6) BDRZ
(7) GRADFN
(8) QG
(9) REGNSEL
(10) ROWSTOR
(11) ECW
(12) ER
(13) ECRD .

To use LAPLDRS, one needs all of the above except (8) and (1), plus the two subprograms:

(14) ELLINT

and

(15) ELLINT2 .

To use LAPLACE, one needs the first eight subprograms and program (13). LAPLARS needs (2) to (7) and (13) to (15). Although these 15 subprograms have the same name, they vary slightly depending on which of the four programs they are associated with.

Before we discuss the actual program structure, a description of the general storage scheme is in order. Almost all the variables mentioned below are placed in common storage and used by several subroutines. Most of the storage discussed here is used to define the boundary. In the four programs, the variables are named as though they lay in the x-y plane. This makes it rather difficult to follow programs (ii) and (iv), because all the z-coordinates are called x-coordinates and all the r-coordinates are called y-coordinates. However, it does mean that most of the FORTRAN statements are common to all four programs.

We will discuss the storage scheme for programs (i) and (ii) first. Let S_i be the ith boundary section given as input, and let S_i have the parametric representation $\{[x_i(t),y_i(t)]|t\epsilon(0,d_i)\}$ with respect to arc length. Define KT_i to be the number in the ninth field of the ith boundary data card, that is columns 74 to 76. The following variables have one entry for each boundary section.

DC(i) = d_i = length of the boundary section,

LT(i) = eighth data field of boundary data card, i.e., column 72,

HD(i) = h_i = DC(i)/(KT$_i$-1),

KV(i+1) = KV(i) + KT$_i$, [KV(1) = 0].

The following variables are used to store the coordinates of boundary points.

$$X[j+KV(i)] = x_i[(j-1)h_i] \text{ for } j=1,2,\ldots,KT_i ,$$

$$Y[j+KV(i)] = y_i[(j-1)h_i] \text{ for } j=1,2,\ldots,KT_i ,$$

$$XN[j+KV(i)] = dy_i[(j-1)h_i]/dt \text{ for } j=1,2,\ldots,KT_i ,$$

$$YN[j+KV(i)] = dx_i[(j-1)h_i]/dt \text{ for } j=1,2,\ldots,KT_i ,$$

$$G[j+KV(i)] = \frac{1}{2}\left[\frac{dx_i(t)}{dt}\frac{d^2y_i(t)}{dt^2} - \frac{dy_i(t)}{dt}\frac{d^2x_i(t)}{dt^2}\right]$$

evaluated at $t = (j-1)h_i$ for $j = 1, 2, \ldots, KT_i$,

$$XI[j+KV(i)] = x_i[(j-3/2)h_i] \text{ for } j=2,3,\ldots,KT_i ,$$

$$YI[j+KV(i)] = y_i[(j-3/2)h_i] \text{ for } j=2,3,\ldots,KT_i ,$$

$$XIN[j+KV(i)] = \frac{dy_i[(j-3/2)h_i]}{dt} \text{ for } j=2,3,\ldots,KT_i,$$

$$YIN[j+KV(i)] = \frac{dx_i[(j-3/2)h_i]}{dt} \text{ for } j=2,3,\ldots,KT_i.$$

Before the unknown boundary values are computed, the variables D[j+KV(i)] for j = 1, 2, ..., KT_i contain the known boundary values corresponding to the points {X[j+KV(i)],Y[j+KV(i)]}, and the variable F is the same as the vector E in Eq. (20). After the unknown boundary values have been computed, the variables

$$D[j+KV(i)],\ F[j+KV(i)] \text{ for } j = 1, 2, \ldots, KT_i$$

contain the normal derivative and the potential, respectively, for the points {X[j+KV(i)],Y[j+KV(i)]}. To illustrate this let i = 1, KT_1 = 5 and assume that $S_1 = \{[\cos(t),\sin(t)]|t\epsilon(0,2\pi)\}$ is the unit circle. Figure 14 shows how the variables correspond to this problem. At every point [X(j),Y(j)], the vector [XN(j),-YN(j)] is the unit exterior normal vector to S. The points [XI(j),YI(j)] and the vector [XIN(j), -YIN(j)] satisfy the same relation. Between each pair of boundary values there is one set of boundary points that are used only to define the boundary better. This choice is arbitrary; there could be none or whatever number is desired.

The variable NDC is the total number of boundary sections. For programs (i) and (ii) it is the same as the variable NDCT defined earlier. The variable N = KV(NDC+1). For programs (i) and (ii) it is the same as the variable NT defined in Sec. III.

Now consider programs (iii) and (iv). The main difference is that in this case the boundary data for one specified subregion only is stored for use at any given time. The information for any subregion is stored in exactly the same way as the whole region is stored for programs (i) and (ii). Thus, NDC con-

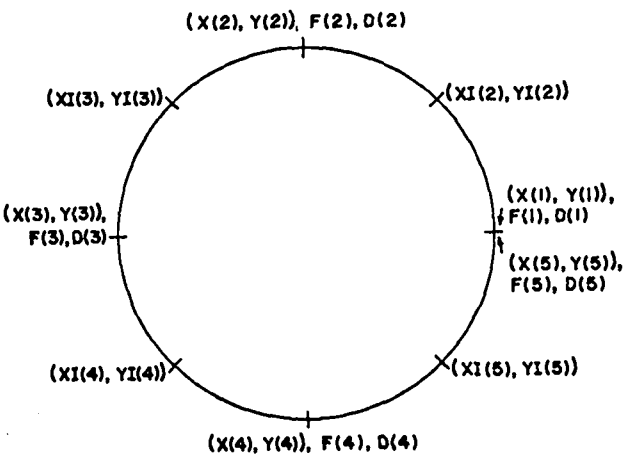

Fig. 14. Distribution of points on a circle.

tains the number of boundary sections for the selected subregion, and [X(1),Y(1)] is the first boundary point of the first boundary section of the selected subregion. The subroutine REGNSEL stores the information for the selected subregion. The indexed boundary data in the labeled common ZLAP are subregion oriented and are stored by REGNSEL. The other labeled common storage remains fixed throughout the computation.

The solution in any subregion is dependent on the boundary data from all of the subregions. Hence, there must be some variables concerned with the problem as a whole. The variable NRE is the total number of subregions. The subregions are numbered consecutively, 1, 2, . . . , NRE. The variable ND(J) for J = 1, 2, . . . , NRE gives the number of boundary sections in the Jth subregion. The variable $ICH(J) = \sum_{K=1}^{J-1} ND(K)$ for J = 2, 3, . . . , NRE, and ICH(1) = 0. ICH(J) is the number of boundary sections preceding the Jth subregion. The variable NDCT is the number of boundary sections for the whole problem. If the boundary section $S_I \notin C^2$, then the variable IC(I) = 0. See Sec. VI for the definition of C^2. If $S_I \in C^2$, then IC(I) is the number of the common boundary section that is the same as S_I but with opposite direction. The variable RA(I) is related to the σ value. If $S_I \notin C^2$, then RA(I) = 1, and if $S_I \in C^2$, then $RA(I) = -\sigma_{IC(I)}/\sigma_I$.

The variable NT is the number of unknown boundary values for the whole problem. Before the unknown boundary values are computed, the indexed variable FT corresponds to the vector E in Eq. (20). After the unknown boundary values have been computed, they are stored in FT. The way the boundary values are stored is probably not so satisfactory as it could be. If $S_I \in C^1$, then the first 2 KT_1 words of FT are used to store the unknown boundary values on S_1. All of the computed unknown values for the potential precede those for the normal derivative. If $S_1 \notin C^1$, then the first KT_1 words on FT are used to store the unknown boundary values on S_1. If $S_2 \in C^2$, then the unknown boundary values for S_2 are already stored in FT. If $S_2 \in C^1$, then the next $2KT_2$ words of FT are used to store the unknown boundary values on S_2. If $S_2 \notin C$, then the next KT_2 words of FT are used to store the unknown boundary values on S_2. The unknown boundary values for the remaining boundary sections are stored in the same manner. The variable IE(I) is used as a pointer to tell where in FT the unknown boundary values for S_I are stored. If $S_I \notin C^2$, then IE(I) is the number in FT of the first unknown boundary value for S_I. If $S_I \in C^2$, then IE(I) is the number +1 in FT of the last boundary value on $S_{IC(I)}$. This is because the boundary values for S_I can be derived from those of $S_{IC(I)}$, except that they are in reverse order. The variable $IX(L) = 1 + \sum_{i=1}^{L-1} KT_i$ for L = 2, 3, . . . , NDCT, and IX(1) = 1. This variable is similar to KV, except that KV is defined only for subregions. The variable KSEL has the number of the subregion which was last selected.

Most of the following discussion is applicable to all four programs. There are a number of other variables in common storage which perform various jobs. Most of these variables are initialized in FN and used in QV. We will not discuss them. The other variables and their functions are as follows. BZ is a logical variable used to indicate when the unknown boundary values have been computed. BZ = F if, and only if, the unknown boundary values have been computed. The variable is used by the plot routine to determine whether anything but the boundary should be plotted. The variable BZ is also used by FN to tell whether the approximate solution is desired, or merely the integrals in Eqs. (21) or (26). The variables CN(I) and CD(I) for I = 1, 2, . . . , NT are used by the routine FN to store the integrals in Eqs. (21) and (26). The indexed variable Q is used to store the integrals in Eqs. (22) and (27). Assuming that one is using program (ii), for instance, a call to QV(I) will cause the following numbers to be stored in Q:

$Q(J) = \tilde{D}(I,J-1)$ for J = 1, 2, 3

$Q(J+3) = \tilde{N}(I,J-1)$ for J = 1, 2, 3.

See Eq. (27) for the definition of $\tilde{D}$ and $\tilde{N}$. The variable Q is also used by the gradient routine to store similar integrals. The logical variable RI is used to indicate whether a point is, or is not, in the region G. After a call to FN(S,T), the variable RI = T if, and only if, (S,T) $\in$ G. If RI = F, then FN is set equal to -0. The variable RI is used by the plot routine, but it is in common storage and is available to a user.

The subroutine FN has the formal parameters (S,T). To make these two variables available to QV, they are put in common storage; S is stored in V, and T is stored in W. Assume that one makes a call

to FN, and that the point (S,T) is very close to the boundary. By finding the point $(\tilde{S},\tilde{T})$ on the boundary closest to (S,T) one can approximate

$$u(S,T) \approx u(\tilde{S},\tilde{T}) - \left[(S-\tilde{S})^2+(T-\tilde{T})^2\right]^{\frac{1}{2}} \frac{\partial u(\tilde{S},\tilde{T})}{\partial \nu} \quad . \quad (32)$$

This approximation assumes that $(\tilde{S},\tilde{T})$ is not at a reentrant corner, in which case the approximation is similar but more involved. The variables DP, ME1, and FB are all used in the approximation of Eq. (32). The variable DP is used both as an indicator and as a storage location for the minimum distance. Whenever $DP \neq 10^{100}$, then Eq. (32) has been used by QV to compute the solution. The routine QV uses DP to store the minimum distance to the boundary if Eq. (32) is used. The approximation is stored in the variable FB. The variable FB = 0. if, and only if, $(S,T) \notin G$. The variable ME1 is used to determine whether or not the point $(\tilde{S},\tilde{T})$ is at a corner. Corners can occur only at the ends of boundary sections.

To determine if a given point (S,T) is in G, one must first determine whether G is an exterior or exterior region. The variable AR is a discrete approximation to

$$\int_S y(t) \frac{dx(t)}{dt}\, dt \quad . \quad (33)$$

The integral is minus the area enclosed by S. See Ref. 2, pp. 311-314. If G is an interior region, then AR is approximately minus the area of G. If G is an exterior region, then AR is the area of the complement of G. Even though S need not be closed in the z-r plane, the approximation is still valid because the z-axis is a boundary in our generalized sense. The indexed variable ARR is used by programs (iii) and (iv). ARR(i) is the integral of Eq. (33) on the subregion G_i. This indexed variable is needed for these two programs because of the possibility of having both interior and exterior subregions in one problem.

The logical variable BQ is used as an indicator by the subroutine FN to determine whether or not the subroutine QV encountered case (a) of Sec. VIII. BQ = T if, and only if, case (a) was applicable. If BQ = T, then the integrals of Eq. (21) were computed directly in QV, and if BQ = F, then the integrals of Eq. (22) were computed in QV.

Most of the above variables are dimensioned. The problem size is limited by the dimensions of these variables and by the size of the matrix A which is NT x NT. The variable FT must be of at least NT length. The variables F, D, CD, CN, G, X, Y, XN, YN, XI, YI, XIN, and YIN must have dimensions as large as the number of approximation points for the largest subproblem. For programs (i) and (ii) they must be as large as the total number of approximation points. The variables HD, DC, LT, IC, IX, IE, and RA must all have dimensions at least NDCT, and KV must have the dimension NDCT + 1. The variables ND, ARR, and ICH must have dimensions at least NRE. Finally, there are some variables, namely FF, GM, AL, BE, R, B, GA1, and KT, all used in REGNSEL, that must have dimensions as large as those of KV. These variables are equivalenced, and some care is necessary when changing their dimensions.

We will now discuss the individual subprograms. BC is a real function used to read current data cards and set up branch cuts if they exist. Probably more important, the subprogram gives the user a simple way to use superposition of solutions. BC is exactly the same for programs (i) and (iii). The function $\tan^{-1}(y-yc/x-xc)$ with $-\pi < \tan^{-1} \leq \pi$ has a branch cut starting at the point (xc,yc) and running parallel to the x-axis to $x = -\infty$. The program BC merely adds one of these functions for each current source. The variable EC corresponds to current and determines the discontinuity across the branch cuts. The variables XC and YC correspond to the point (xc,yc). The variable NBC is the total number of branch cuts. The subprogram has three entry points. Entry point BC evaluates the branch cut functions if they exist, and entry point BCN evaluates the derivative of the branch cut functions if they exist. Entry point BCR reads current data cards if they exist.

The subroutine LAPLOT plots equipotential curves or gradient curves. The program also plots the boundary. The program has four entry points, the purpose of each of which is explained in Sec. IV. The plot routine has only slight differences for each of the four programs, the main difference being the use of the gradient function in programs (i) and (iii) as opposed to programs (ii) and (iv). See the routine GRADFN below.

The variable LM in LAPLOT is used to count the number of curves that have been plotted and is set equal to zero on each entry into the subroutine. The

variables XS and YS contain the starting point for the curve being processed. The variable HH contains the step size used to construct the curves. The sign of HH determines the direction of travel along the equipotential or gradient curve. When one constructs the curves, the plot goes from the point [XP(1), YP(1)] to the point [XP(2),YP(2)]. The gradient values corresponding to [XP(1),YP(1)] are always stored in FXO and FYO. If the gradient vector at [XP(1),YP(1)] dotted with the gradient vector at [XP(2),YP(2)] is negative, the curve turns more than 90° in one step and probably does not make sense. If this is the case, the curve is discontinued.

The film plotter has a resolution of one part in 1024 for both the x and y directions. By the time the grid lines and the scale have been drawn in, the resolution is down to one part in 840 for the equipotential and gradient curves. For the equipotential curves, it was arbitrarily decided that the computed resolution in the rectangle derived from XMIN, YMIN, XMAX, and YMAX would be one part in 1680. Thus, any point [XP(2),YP(2)] computed to be on a given equipotential curve will have at most an error of (YMAX -YMIN)/1680 in the y-coordinate and of (XMAX-XMIN)/ 1680 in the x-coordinate. This choice does not always give the full resolution of the plotter, but it is a compromise between computation time and the best possible resolution.

Construction of the equipotential curves is easy. Assume that FO is the value of the equipotential, and that [XP(1),YP(1)] was the last point to be plotted. Because [XP(1),YP(1)] is not always exactly on the equipotential curve, let F2 be the value of the potential at [XP(1),YP(1)]. Also, let FX and FY be the values of the gradient at [XP(1),YP(1)]. The first guess for the next point on the curve is

$$\begin{aligned} XP(2) &= XP(1)-HH*FY/(FX^2+FY^2)^{\frac{1}{2}}+(FO-F2)*FX/(FX^2+FY^2) \\ YP(2) &= YP(1)+HH*FX/(FX^2+FY^2)^{\frac{1}{2}}+(FO-F2)*FY/(FX^2+FY^2) \end{aligned} \quad . \tag{34}$$

The first term is a step of distance HH perpendicular to the direction of the gradient vector. The second term is a step parallel to the gradient vector, and is designed to correct for errors in [XP(1),YP(1)]. If the point [XP(2),YP(2)] is not close enough to the equipotential curve, then another estimate is made by moving in the direction of the gradient vector at [XP(2),YP(2)]. Such corrections are made until the point either is close enough to the curve or does not improve satisfactorily, in which case the curve is terminated.

The construction of gradient curves is slightly different from that of the equipotential curves. There is no direct way to determine how close a given point is to an actual curve. In LAPLOT the gradient curves are approximated by using a simple Runge-Kutta scheme on the autonomous system of ordinary differential equations

$$\frac{dx(t)}{dt} = u_x[x(t),y(t)] \quad ,$$

$$\frac{dy(t)}{dt} = u_y[x(t),y(t)] \quad .$$

See Hildebrand's[14] formulas 6.15.15 and 6.15.16. In terms of the variables used in LAPLOT, the method for any given step is

$$\begin{aligned} XP(2) &= XP(1) + \left(\frac{HH}{2}\right)\Big(u_x[XP(1),YP(1)] + u_x\{XP(1) \\ &\quad +HHu_x[XP(1),YP(1)],YP(1)+HHu_y[XP(1),YP(1)]\}\Big) \quad , \\ YP(2) &= YP(1) + \left(\frac{HH}{2}\right)\Big(u_y[XP(1),YP(1)] + u_y\{XP(1) \\ &\quad +HHu_x[XP(1),YP(1)],YP(1)+HHu_y[XP(1),YP(1)]\}\Big) \quad . \end{aligned} \tag{35}$$

The subroutine ECSW is simply a routine to read and write rows of a matrix. It does the bookkeeping on the size of the matrix and the storage location of various elements. It also stores some variables that are seldom used. The routine was written to give the programs the capability to use external storage. The subprogram is the same for programs (i) and (ii). Similarly, for programs (iii) and (iv) the subprogram is the same.

The subprograms FN and QV together perform one function. For programs (i) and (iii), they compute the integrals in Eq. (21) and for programs (ii) and (iv) they compute the integrals in Eq. (26). In both cases, the routine FN combines the integrals with the appropriate boundary values to give the approximate solution, if it is desired. For programs (i) and (iii), the routines QV are identical as they also are for programs (ii) and (iv). For each of the four programs, the routines FN are slightly dif-

ferent. For programs (i) and (iii), the routine QV usually computes the integrals in Eq. (22), and the routine FN combines the integrals to give the integrals in Eq. (21). However, in case (2) of Sec. VIII, the routine QV computes the integrals in Eq. (21) directly. For programs (ii) and (iv), the routine QV always computes the integrals in Eq. (27), and FN combines the integrals to give the integrals in Eq. (26). The routine FN does two other things. First, it sets up variables used in QV. Second, it combines the computed integrals to determine whether the point used in the calculation is inside the region G. The routine QV performs one other computation. If the given point at which the approximate solution is desired happens to be near the boundary, then QV computes the approximate solution using the values of u and $\partial u/\partial \nu$ at the nearest point on the boundary.

Now let us consider how the routine FN computes the integrals in Eq. (21) from the integrals in Eq. (22). For simplicity assume that t_k is the start of the boundary section $S_{j(k)}$. The function j(k) is defined in Sec. VII. From the definition of $\theta_k(s)$, it follows that

$$\theta_k(s) = \begin{cases} \left[s-t_k-h_{j(k)}\right]\left[s-t_k-2h_{j(k)}\right]/2h_{j(k)}^2 & \text{for } s\in(t_k, t_{k+2}) \\ 0 \quad \text{otherwise.} \end{cases}$$

Hence, on (t_k,t_{k+1}) it is true that

$$\theta_k(s) = \left[(s-t_k)-h_{j(k)}\right]\left[(s-t_k)-2h_{j(k)}\right]/2h_{j(k)}^2 = 1 - \frac{3(s-t_k)}{2h_{j(k)}} + \frac{(s-t_k)^2}{2h_{j(k)}^2} .$$

On (t_{k+1},t_{k+2}), we have

$$\theta_k(s) = (s-t_{k+1})\left[(s-t_{k+1})-h_{j(k)}\right]/2h_{j(k)}^2 = \frac{(s-t_{k+1})^2}{2h_{j(k)}^2} - \frac{(s-t_{k+1})}{2h_{j(k)}} .$$

Hence,

$$\int_S \theta_k(s) \ln\left[\frac{1}{r(s,x,y)}\right]ds = \int_{t_k}^{t_{k+1}} \theta_k(s)\ln\left[\frac{1}{r(s,x,y)}\right]ds + \int_{t_{k+1}}^{t_{k+2}} \theta_k(s)\ln\left[\frac{1}{r(s,x,y)}\right]ds$$

$$= N(k,0) - \frac{3N(k,1)}{2h_{j(k)}} + \frac{N(k,2)}{2h_{j(k)}^2} + \frac{N(k+1,2)}{2h_{j(k)}^2} - \frac{N(k+1,1)}{2h_{j(k)}} .$$

The other integral in Eq. (21) is treated exactly the same. The integrals for the axially symmetric case are also treated the same way.

If we call the routine FN in order to compute the approximate solution at some point (v,w), then FN determines if the point (v,w) is in the region G, or, in the case of programs (iii) and (iv), if it is in the previously selected subregion. This is done in one of two ways. Assume first that G is an interior region in the x-y plane, and that S has no common boundaries. Then

$$\int_S \frac{\partial}{\partial \nu} \ln[r(s,v,w)]ds = \begin{cases} 0 \text{ if } (v,w) \notin G \cup S, \\ 2\pi \text{ if } (v,w) \in G. \end{cases} \qquad (36)$$

If (v,w) is not close to the boundary, then the integral above is a by-product of the computation of the approximate solution. The case when (v,w) is close to the boundary is discussed later. If G is an exterior region, then the result in Eq. (36) is exactly opposite. If the region G has subregions, the above result holds on each subregion. If the point (v,w) $\in$ G, but it is not in the selected subregion when one calls FN, then the routine FN selects each of the other subregions until it finds the one containing (v,w). For the z-r plane the method is the same except that a different kernel is used in Eq. (36).

The variable DIC is used to store the integral in Eq. (36). The variable ADIC is used to store the integral of the absolute value of the kernel. DIC is compared to ADIC to determine whether or not the integral in Eq. (36) is approximately zero.

The QV subroutine for programs (i) and (iii) is completely different from the QV subroutine for programs (ii) and (iv). Most of the basic theory behind these two subroutines is covered in Secs. VIII and IX. Here we discuss only the case in which we want the approximate solution at the point (v,w), and (v,w) is close to the boundary. This section of the program is the same for the two QV subroutines. Let (XE,YE) be the point on the boundary closest to (v,w), and let (XP,YP) be the unit tangent vector at (XE,YE). Under the assumptions on the boundary, if (XE,YE) is not at a corner, then the vector (XE-v,YE-w) is parallel to the vector (YP,-XP). Possibly the direction

is opposite, if (v,w) is not in G. By taking the dot product of the two vectors, we can determine whether (v,w) ∈ G for this case. If (v,w) ∈ G, we approximate

$$u(v,w) \approx u(XE,YE)-\left[(XE-v)^2+(YE-w)^2\right]^{\frac{1}{2}} \frac{\partial u}{\partial \nu}\,(XE,YE) \quad . \tag{37}$$

If (v,w) ∈ G, and (XE,YE) is at a corner, then the corner is reentrant. In this case, we assume that any singularities at the corner have been subtracted out. Using this assumption, we can compute the derivative of the solution in any direction by using the two values of $\partial u/\partial \nu$ at the corner. To compute the approximate solution, we simply use Eq. (37) with the derivative term replaced by the derivative in the direction (XE-v,YE-w).

For programs (i) and (ii), the routine BDRZ reads the boundary data cards, puts the data in a usable form, and generates boundary points upon request. For programs (iii) and (iv), BDRZ generates only boundary points. The BDRZ routines for programs (i) and (ii) are almost identical, as is also true for programs (iii) and (iv). We will discuss only the BDRZ routine associated with programs (i) and (ii) because the other BDRZ routine is simply a subset of the one to be discussed. The routine BDRZ has two entry points. Entry point BDRZ reads one boundary data card and puts the information in a usable form. Whenever a -0 is encountered in a data field, the routine assumes that the field is blank. When the routine reads a blank card, it sends a message to stop reading cards. When using simplified boundary data, the entry point BDRZ in some sense replaces the subroutine BDRY. BDRZ gives exactly the same information for boundary sections with simplified boundaries as BDRY does for generalized boundary data.

The information needed to construct the simplified boundary sections is stored in seven-word blocks, that is, seven words are used for each boundary section. Generating a line segment or a complete circle is a straightforward task using a boundary data card. However, a circular arc is more difficult, and will be explained. Suppose we have read in the points (X1,Y1), (X2,Y2), and (X3,Y3). First solve for the center point, called (AL,BE) here, of the circular arc. Because the three points all lie on the circle, we have

$$(AL-X1)^2 + (BE-Y1)^2 = R^2,$$

$$(AL-X2)^2 + (BE-Y2)^2 = R^2 \ ,$$

and

$$(AL-X3)^2 + (BE-Y3)^2 = R^2 \ ,$$

where R is the radius of the circle. Subtracting the first equation from both of the other two, we get the system of linear equations

$$2AL(X1-X2)+2BE(Y1-Y2) = X1^2+Y1^2-X2^2-Y2^2 \ ,$$

$$2AL(X1-X3)+2BE(Y1-Y3) = X1^2+Y1^2-X3^2-Y3^2 \ .$$

These can easily be solved for (AL,BE). Substituting AL and BE back in the first equation, for instance, we can compute R. The parametric form for the circuler arc is

$$X(T) = R\cos(D*T/R+GA1) + AL \ ,$$

$$Y(T) = R\sin(D*T/R+GA1) + BE \ .$$

The variable GA1 must satisfy

$$X1 = R\cos(GA1) + AL \ ,$$

and

$$Y1 = R\sin(GA1) + BE \ ,$$

which is equivalent to

$$GA1 = TAN^{-1}[(Y1-BE)/(X1-AL)] \ .$$

The variable D is the orientation and is either +1 or -1. The computation of D is involved. Define the complex numbers Z2 and Z3 by

$$Z2 = \frac{(X2-AL)+i(Y2-BE)}{(X1-AL)+i(Y1-BE)} \ ,$$

and

$$Z3 = \frac{(X3-AL)+i(Y3-BE)}{(X2-AL)+i(Y2-BE)} \ .$$

Thus, arg(Z2) is the angle on the circle from (X1,Y1) to (X2,Y2), and arg(Z3) is the angle from (X2,Y2) to (X3,Y3). We assume here that $-\pi < \arg(\cdot) < \pi$. It follows then that D has the same sign as the argument that has the smaller absolute value. Equiv-

alently, D has the same sign as the imaginary part of the complex number whose argument has the smaller absolute value. The length of the circular arc is easy to compute.

For programs (i) and (iii), the subroutines GRADFN and QG perform approximately the same functions when one computes the gradient as the subroutines FN and QV do when one computes the potential. FN and QV, however, will handle more difficult situations. The gradient routine breaks down very near the boundary. Also, it will not handle the round-off difficulties described in case (c) of Sec. VIII. However, these problems do not occur often.

The subroutine QG approximates integrals of the form

$$\int_{t_k}^{t_{k+1}} (t-t_k)^j \frac{\partial^2}{\partial x \partial \nu} \ln[r(t,x,y)]dt \quad \text{for } j = 0,1,2 \quad ,$$

$$\int_{t_k}^{t_{k+1}} (t-t_k)^j \frac{\partial}{\partial x} \ln[r(t,x,y)]dt \quad \text{for } j = 0,1,2 \quad , \tag{38}$$

$$\int_{t_k}^{t_{k+1}} (t-t_k)^j \frac{\partial^2}{\partial y \partial \nu} \ln[r(t,x,y)]dt \quad \text{for } j = 0,1,2 \quad ,$$

and

$$\int_{t_k}^{t_{k+1}} (t-t_k)^j \frac{\partial}{\partial y} \ln[r(t,x,y)]dt \quad \text{for } j = 0,1,2 \quad .$$

The three integrals corresponding to the first line in Eq. (38) are stored in Q(1), Q(2), and Q(3). The next three are stored in Q(4), Q(5), and Q(6), and so forth, down to Q(12). In QG, one approximates the integrals above by explicitly evaluating integrals of the form

$$QJ = \int_0^h \frac{t^J dt}{p(t)} \quad \text{for } J = 0, 1, 2, 3, 4 \quad ,$$

and (39)

$$Q2J = \int_0^h \frac{t^J dt}{p^2(t)} \quad \text{for } J = 0, 1, 2, 3, 4 \quad .$$

Here p(t) is a polynomial of degree two used to approximate $r^2(t,x,y)$. Explicit formulas for the integrals Q2J for J = 0, 1, 2, 3, 4 are given on pp. 65-66 of Ref. 15. The integrals QJ for J = 0, 1, 2, 3, 4 can be found in almost any integral table. The integrals in Eq. (39) are combined to give a rational fraction approximation to the integrals in Eq. (38). The approximation is similar to case (c) of Sec. VIII.

The subroutine GRADFN completely treats what corresponds to case (a) of Sec. VIII. Assuming that the computation is being done on the interval (t_k, t_{k+2}), the decision to treat the integrals as in case (a) or case (c) of Sec. VIII is determined by whether or not

$$r(t,x,y) \geq 3h_j(k) \text{ for } t = t_k, \frac{(t_k+t_{k+1})}{2}, t_{k+1}, \frac{(t_{k+1}+t_{k+2})}{2}, t_{k+2} \quad .$$

The subroutine REGNSEL has two entry points. Entry point SELINIT reads all of the boundary data cards and puts the data in a usable form. This part of the program is similar to entry point BDRX of BDRZ. To a large extent, the programming is the same. SELINIT has a few more wrinkles because it must handle common boundaries. For this entry point, there is only one difference between programs (iii) and (iv). Program (iii) has a provision for reading current data cards. Entry point REGNSEL sets up all of the boundsry data for a requested subregion. This entry point for program (iv) differs from that for program (iii) in that program (iv) does not allow negative values of y.

Subroutine ROWSTOR is simply a program to store the rows of the matrix A given the integrals in Eqs. (21) or (26). The integrals are transmitted through common storage in the indexed variables CN and CD. The routine also computes a row of the vector E of Eq. (20) and stores it in the indexed variable FT.

Subroutine ECW has two entry points. Entry point ECW writes one word into a specified location of the matrix A. Entry point ECA1 adds one word to a specified element of the matrix A. The real function ER reads one specified word from the matrix A and stores it in ER.

ECRD is a routine to utilize storage which can be core storage, extended core storage, disk, drum, or what have you. The program presently uses core storage or extended core storage, but it can easily be modified to use whatever storage is available. The storage is mostly for the matrix A. There are two entry points to the subroutine. Entry point ECRD reads a specified number of words starting at

a given location. Entry point ECWR writes a specified number of words starting at a given location.

ELLINT is a subprogram to evaluate the two elliptic functions E(M) and K(M). If $M \leq 0.81$, a quadratic interpolation from tabular values is used. The table has 416 points for each of the two functions. It has special values outside both of the end points so that no special technique need be used at the ends. If $M > 0.81$, then the routine uses formulas 17.3.34 and 17.3.36 of Ref. 9 to approximate the functions. There are two versions of this subprogram, one in machine language and the other in FORTRAN. The machine language version is faster and should be used whenever possible because the timing of this routine is critical.

ELLINT2 is a subprogram to evaluate the functions K_1, Q_2, E_1, and Q_4 of Eq. (29). Depending on the parameters in the calling sequence, the routine evaluates K_1 and E_1 or Q_2 and Q_4. This routine is written in FORTRAN.

Finally, let us consider how some of the variables in common storage can be used for output calculations. Assume that program (iii) is being used, and that all the labeled common storage is available. Suppose S_J is a boundary section in the subregion J1. The following statements could be used to give a Simpson's rule approximation to $\int_{S_J} \partial u/\partial \nu \, dS$.

```
      CALL REGNSEL(J1)
      I1 = J - ICH(J1)
      L1 = KV(I1) + 2
      L2 = KV(I1 + 1)
      QUAD = D(L1-1)
      DO 1 M = L1, L2, 2
1     QUAD = QUAD + 4.*D(M) + 2.*D(M+1)
      QUAD = HD(I1)*(QUAD-D(L2))/3.
```

The first statement ensures that the right subproblem is stored in ZLAP. The second gives the number of the section S_J in the subregion J1. The other statements follow immediately from the definitions of the variables. Simpson's rule is about the highest order integration scheme that makes sense, considering the way the boundary values are computed.

REFERENCES

1. W. Sternberg and T. Smith, The Theory of Potential and Spherical Harmonics, 2nd Ed., The University of Toronto Press, Toronto, 1952.
2. B. Budak, A. Samarskii, and A. Tikhonov, A Collection of Problems on Mathematical Physics, Macmillan, New York, 1964.
3. J. Hayes, "The Laplace FORTRAN CODE," Los Alamos Scientific Laboratory Report LA-4004, 1968.
4. W. Smythe, Static and Dynamic Electricity, 3rd Edition, McGraw-Hill, New York, 1950.
5. J. Whiteman, "Treatment of Singularities in a Harmonic Mixed Boundary Value Problem by Dual Series Methods," Quart. J. Mech. Appl. Math. 21, 41 (1968).
6. D. Schultz, "Two Test Cases for the Numerical Solution of Harmonic Mixed Boundary Value Problems," Mathematics Research Center, U. S. Army, Technical Summary Report No. 900, 1968.
7. L. Fox, Numerical Solution of Ordinary and Partial Differential Equations, Addison-Wesley, Reading, Massachusetts, 1962.
8. R. Courant, Differential and Integral Calculus, Vol. I, 2nd Ed., Interscience, Glasgow, 1937.
9. National Bureau of Standards, Handbook of Mathematical Functions, U. S. Government Printing Office, Washington, D. C., 1964.
10. E. Whittaker and G. Watson, A Course of Modern Analysis, 4th Ed., Cambridge University Press, Cambridge, England, 1952.
11. A. Stroud and D. Secrest, Gaussian Quadrature Formulas, Prentice-Hall, Englewood Cliffs, N. J., 1966.
12. R. Kellner, Private Communication, 1969.
13. E. Hille, Analytic Function Theory, Vol. II, Blaisdell Publishing Company, Waltham, Massachusetts, 1962.
14. F. B. Hildebrand, Introduction to Numerical Analysis, McGraw-Hill, New York, 1956.
15. I. Ryshik and I. Gradstein, Tables, Veb Deutscher Verlag der Wissenschaften, Berlin, 1957.