AECL 7614

c 1

AECL-7614

ATOMIC ENERGY OF CANADA LIMITED

L'ÉNERGIE ATOMIQUE DU CANADA LIMITÉE

SLACINPT – A FORTRAN PROGRAM THAT GENERATES BOUNDARY DATA FOR THE SLAC GUN CODE

SLACINPT – Programme FORTRAN engendrant des données limites pour le code SLAC

W.L. MICHEL and J.D. HEPBURN

Chalk River Nuclear Laboratories Laboratoires nucléaires de Chalk River

Chalk River, Ontario

March 1982 mars

ATOMIC ENERGY OF CANADA LIMITED

SLACINPT - A FORTRAN PROGRAM THAT GENERATES BOUNDARY DATA FOR THE SLAC GUN CODE

W.L. Michel and J.D. Hepburn

Accelerator Physics Branch
Research Company
Chalk River Nuclear Laboratories
Chalk River, Ontario K0J 1J0

March 1982

AECL-7614

L'ENERGIE ATOMIQUE DU CANADA, LIMITEE

SLACINPT - Programme FORTRAN engendrant des données limites pour le code SLAC

par

W.L. Michel et J.D. Hepburn

Résumé

Le programme FORTRAN SLACINPT a été écrit pour simplifier la préparation des données limites pour le code à crayon lumineux SLAC. Dans le SLACINPT, la limite est décrite par une série de lignes droites ou de segments d'arc. A partir de là, le programme engendre les données individuelles de limite requises comme données d'entrée via le code SLAC.

Département de physique des accélérateurs
Société de recherche
Laboratoires nucléaires de Chalk River
Chalk River, Ontario KOJ 1JO

Mars 1982

AECL-7614

ATOMIC ENERGY OF CANADA LIMITED

SLACINPT - A FORTRAN PROGRAM THAT GENERATES BOUNDARY DATA FOR THE SLAC GUN CODE

W.L. Michel and J.D. Hepburn

ABSTRACT

The FORTRAN program SLACINPT was written to simplify the preparation of boundary data for the SLAC gun code. In SLACINPT, the boundary is described by a sequence of straight line or arc segments. From these, the program generates the individual boundary mesh point data, required as input by the SLAC gun code.

Accelerator Physics Branch
Research Company
Chalk River Nuclear Laboratories
Chalk River, Ontario K0J 1J0

March 1982

AECL-7614

Table of Contents

Table of Contents (continued)

Table of Contents (continued)

SLACINPT - A FORTRAN PROGRAM THAT GENERATES BOUNDARY DATA FOR THE SLAC GUN CODE

W.L. Michel and J.D. Hepburn

Accelerator Physics Branch
Atomic Energy of Canada Limited
Research Company
Chalk River Nuclear Laboratories
Chalk River, Ontario K0J 1J0

1. Introduction

1.1 General Comments

The SLAC gun code[1] was written to calculate electron trajectories in electrostatic and magnetostatic fields, and is now in use at many laboratories. In it, Poisson's equation is solved by finite difference equations using boundary conditions specified by inputting the type and position of the boundary.

In use of this code at CRNL, the somewhat cumbersome boundary input method was found to be awkward. Accordingly, a program SLACINPT was written to augment the SLAC gun code, providing an easy-to-use, accurate boundary data generator. SLACINPT is executed immediately preceding the SLAC gun code; absolutely no changes are required in the gun code for successful use of both programs.

This report covers three topics: a brief description of the SLAC gun code boundary input method, a full description of the use of SLACINPT, and a description of the SLACINPT program. Thus it serves the dual role of user's manual and programmer's guide.

1.2 SLAC Gun Code Input

A detailed description of the SLAC gun code and of its boundary data input method is not necessary here; however, a brief outline is given to provide a basis for discussion of SLACINPT.

The SLAC gun code requires a closed, continuous and singly connected boundary made up of two line segment types: 1) Dirichlet boundaries representing metal surfaces on which the potential is known and 2) Neumann boundaries representing gaps between surfaces that must be chosen so that the normal component of the field is zero. Dirichlet boundaries can lie anywhere on the problem grid while Neumann boundaries are restricted to intersect mesh points, either along horizontal or vertical mesh lines or at 45° to mesh lines. Figure 1 shows a typical problem boundary. Note that the example shown is the same as the example used in the SLAC gun code report[1]. Figure 2 shows the output plot produced by the SLAC gun code using, as input, the boundary data generated by SLACINPT for the problem in Figure 1. (SLACINPT input data for this case are described in section 2.4.)

For purposes of SLACINPT, two special cases of the Neumann boundary are defined to be "transition" boundaries. The first case, used for dielectric surfaces such as ceramic materials, is described in reference 1. In it, the boundary of the surface is put in and then overlaid with difference equation coefficients for the solution of Poisson's equation. The second case is used for a boundary segment completely transparent to the Poisson equation solution and is needed to satisfy the requirement that the problem boundary be closed and singly connected.

For the remainder of this report, the following definitions apply: a "boundary point" is any point lying on the actual boundary line, and a "boundary mesh point" is a mesh point lying inside the problem space (i.e., inside the boundary) and within one mesh unit of the boundary line.

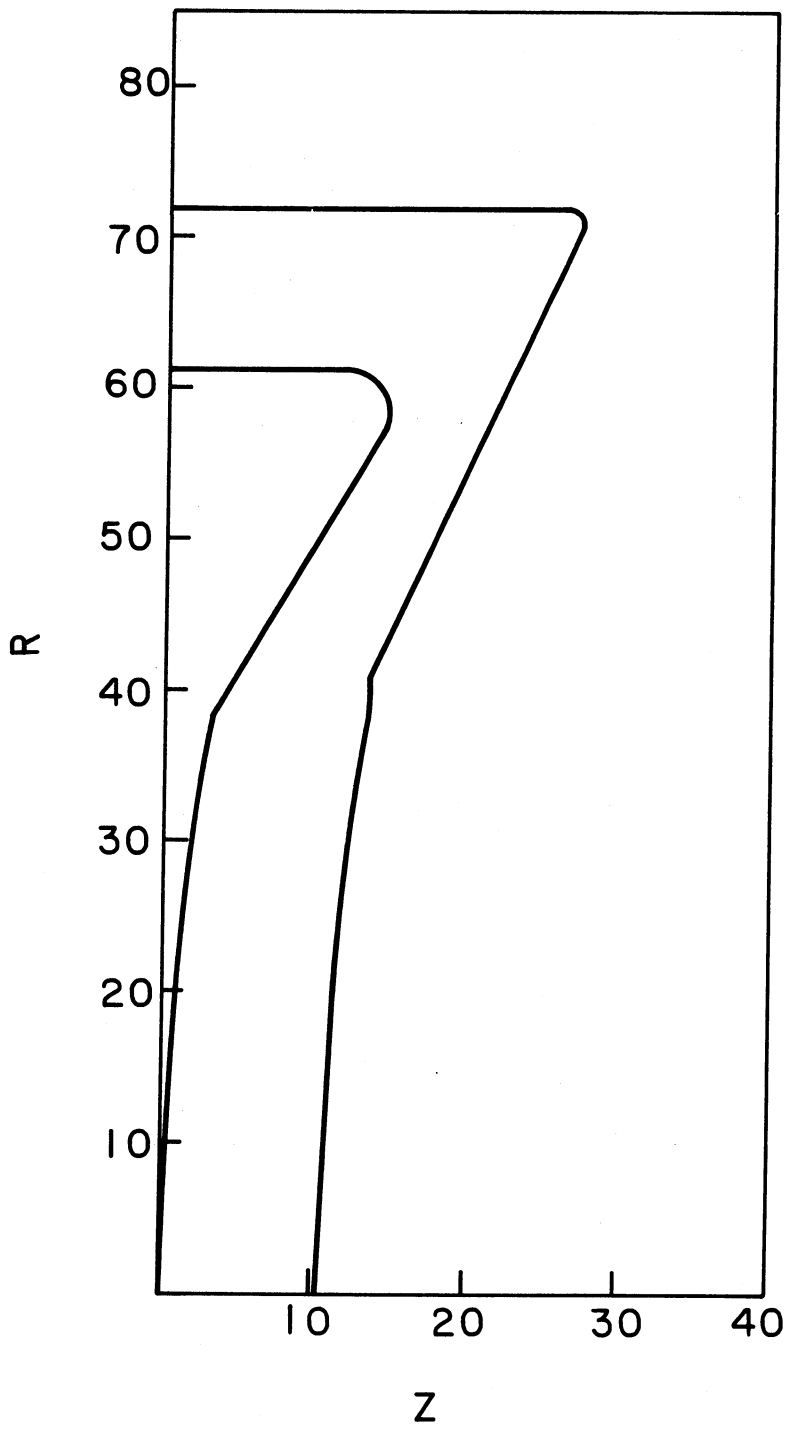

Figure 1 User's sketch of a boundary problem.

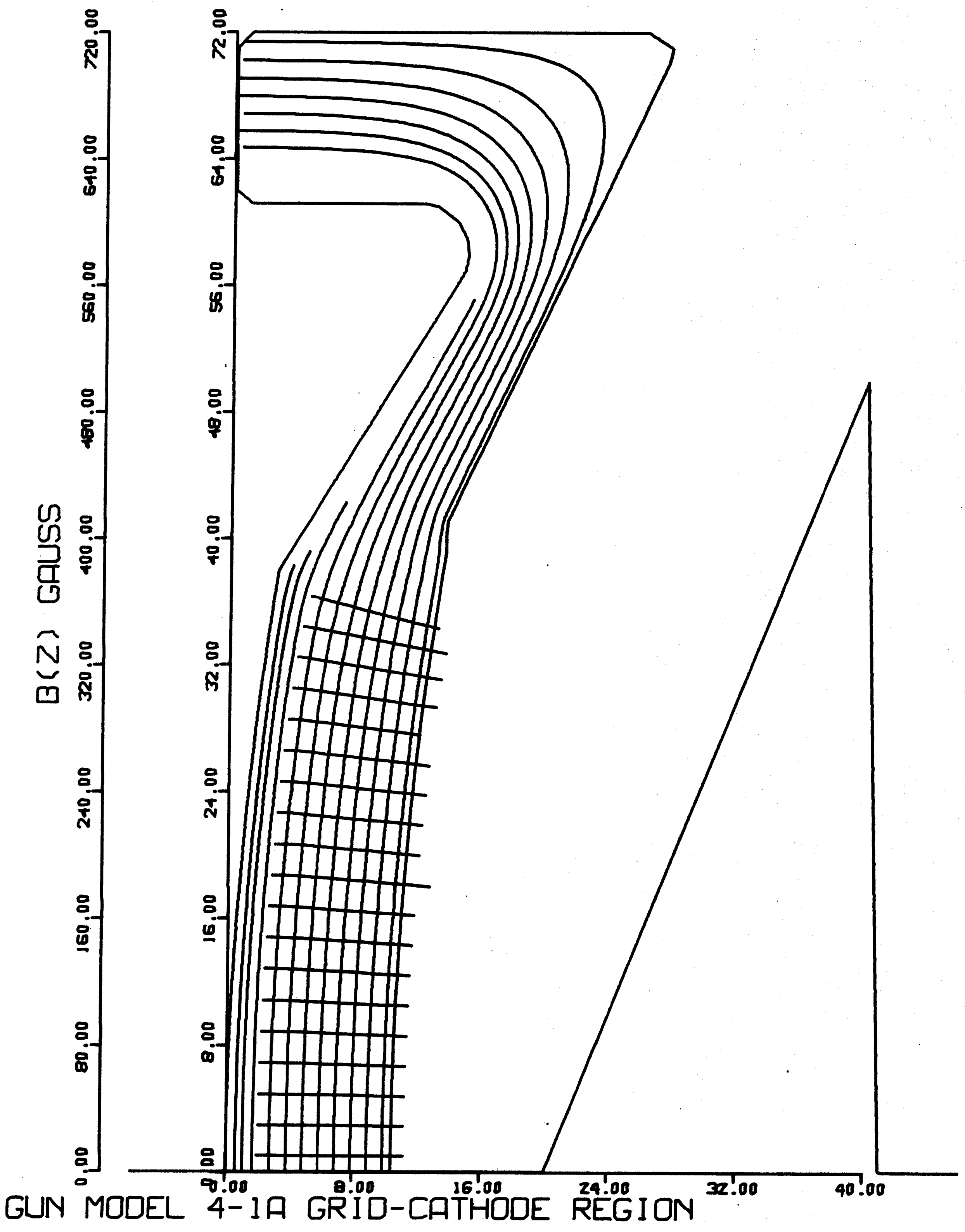

Figure 2 Output plot produced by SLAC gun code for boundary problem.

The boundary data required by the SLAC gun code consists of boundary mesh points, each specified by a set of five numbers, defined as follows:

N - an integer corresponding to a particular value in an array of potentials, identifying the potential of the adjacent boundary. For boundary mesh points on a Neumann or transition boundary and more than one mesh unit from a Dirichlet boundary, the potential number is not significant and is normally set to zero.

R,Z - integers specifying the radial and axial coordinates of the boundary mesh point.

DR,DZ - real numbers specifying the distances from the boundary mesh point to the boundary line in the radial or axial direction. DR and DZ can be positive or negative. If greater than one mesh unit, the distance is not relevant and is defined to be 2.0.

The SLAC gun code requires the following conventions: for a Dirichlet (metal) boundary, neither DR nor DZ can be 0.0; for a Neumann boundary, at least one of DR and DZ must be 0.0; and for a transition boundary, DR=DZ=2.0. Thus, for Dirichlet boundaries, boundary mesh points are never boundary points, while for Neumann boundaries, boundary mesh points are always boundary points. The program analyzes problem space in terms of rows (i.e., lines of R = constant) and columns (i.e., lines of Z = constant). Every row and column crossing the problem space must have $|DZ| < 1.0$ or $|DR| < 1.0$, respectively, at its end points. In practice this means no row or column can consist of a single boundary mesh point.

For solution of Poisson's equation, boundary mesh points must be one mesh unit apart. If input boundary mesh points are not adjacent, then the gun code tries to fill in the missing points by using a fitting routine. In practice, the user must have a good sketch of the boundary system on graph paper. From this he must determine the boundary mesh points and estimate the DR and DZ values. This estimate must be fairly accurate so that the internal interpolation can produce a smooth and accurate representation of the actual boundary system. This usually requires a large number of data cards, especially if the boundary system consists of several curved boundaries. This input method is tedious and often the desired accuracy is difficult to achieve. Iterative design changes are equally tedious and inaccurate.

1.3 SLACINPT

The program SLACINPT simplifies the method of boundary specification, as compared with the SLAC gun code. The boundary system is again drawn on graph paper and consists of a continuous sequence of straight line or arc segments. The user then has to determine only:

(1) the coordinates of the starting point of the first segment

(2) the coordinates of the end point of each segment
(Since the end point of each segment is also the starting point of the next segment, only the end point coordinates need be entered by the user.)

(3) the radius and approximate coordinates of the centre of an arc
(The program calculates the centre of each arc from its end points and radius and then uses the approximate centre location input by the user to select the correct centre from the two possible centres.)

(4) the coordinates of the first boundary mesh point.

In practice, a boundary system consisting of N segments can be completely defined by (N+1) data cards. A clockwise progression is assumed. SLACINPT uses this simplified boundary specification to accurately determine all boundary mesh points enclosed by the boundary system.

2. Problem Data Preparation

2.1 Job Control Deck

Table I shows the structure of the job control card deck, excluding the user's input data, as run on the CRNL CDC 6600 computer using the NOS/BE 1.3 operating system.

Table I

User's Job Control Deck

Deck Contents	Comments
WLMSLAC,B0231-02017,T25,I030.	- for job control card and user identification
ATTACH,LGO,SLACINPT,ID=WLM5J.	- load binary file for SLACINPT
LGO.	- execute SLACINPT
IFE,FILE(INFILE,AS)=TRUE,NOGUN.	- SLACINPT output stored on INFILE, or error exit
REWIND,INFILE.	- tape rewind
ATTACH,A,GUN5,ID=HUTCHEON.	- load binary file for SLAC gun code (GUN5)
A,INFILE.	- execute GUN5, using INFILE as input
ENDIF,NOGUN.	- error exit
7/8/9	

If SLACINPT is terminated because of errors detected by the operating system or because of errors detected by SLACINPT, the file INFILE will not be created and the SLAC gun code will not be attached or executed. If SLACINPT was error free, INFILE will exist and the SLAC gun code, GUN5, will be attached and executed.

SLACINPT is written in FORTRAN IV language for CDC computers. The binary deck occupies about 29500 (decimal) words during execution, and never takes more than 4 seconds on the CDC 6600 for problems with up to 800 boundary mesh points.

2.2 Input Data Deck Structure

All of the data required by the SLAC gun code program, except the boundary specification data, is prepared as described in the gun code instructions. This data is then divided into three parts - the section preceding the gun code boundary data, the boundary data itself and the section following it. All of the problem data is then input to SLACINPT in the following order: the first section of the gun code data (terminated by a bFLAG card), the SLACINPT boundary data, and then the third section of the gun code data (again terminated by a bFLAG card). SLACINPT then calculates the boundary data required by the SLAC gun code, assembles a complete input file for the gun code, and stores the file as INFILE.

2.3 Guide for Boundary Segment Specifications

2.3.1 General Comments

While SLACINPT has been made as general as possible, some restrictions do exist. These are:

(1) The number of boundary segments must not exceed 150.

(2) The number of potential values must not exceed 9.
(3) Arc to arc, horizontal to horizontal, and vertical to vertical sequences of boundary segments are not permitted.
(4) Each segment must cross at least 2 mesh lines.

2.3.2 Boundary Specification Requirements

To generate the boundary mesh point data, SLACINPT requires the following information in its boundary specification input:

(1) the R and Z coordinates of the starting boundary point,
(2) the R and Z coordinates of the end point for each segment,
(3) the R and Z coordinates of the first boundary mesh point,
(4) the shape of the boundary segment (straight line, clockwise arc (CW) or counter-clockwise arc (CCW)),
(5) the type of boundary segment (Dirichlet, Neumann or transition), and
(6) the potential number for that boundary segment.

2.3.3 Boundary Segment Restrictions

Boundary segments must conform to certain restrictions imposed by the SLAC gun code, as mentioned briefly before. Adherence to these restrictions is vital, and they are listed below.

(1) Horizontal Neumann or transition boundary segments must be on a Z mesh line.
(2) Horizontal Dirichlet boundary segments cannot be on a Z mesh line.
(3) Vertical Neumann or transition boundary segments must be on an R mesh line.
(4) Vertical Dirichlet boundary segments cannot be on an R mesh line.

(5) Sloped Neumann or transition boundary segments must have a slope $= \pm 1.0$.

(6) Arc boundary segments cannot be defined as Neumann or transition boundary segments.

2.3.4 Segment End Point Restrictions

As a result of the boundary segment restrictions, the segment end points must conform to the following restrictions:

(1) For a crossover from a Neumann or transition boundary to a Neumann or transition boundary, the end point must be at a mesh point.

(2) For a crossover from a Neumann or transition boundary to a Dirichlet boundary (or vice versa), the end point cannot be at a mesh point. If the Neumann or transition segment is vertical, the end point must be on an R mesh line, while for a horizontal Neumann or transition segment, the end point must be on the Z mesh line. If the Neumann or transition boundary segment is sloped, the end point cannot be on either mesh line but must have the fractional part of R equal to the fractional part of Z.

(3) For a crossover from a Dirichlet boundary segment to a Dirichlet boundary segment, the end point cannot be on any mesh line.

2.4 Data Card Description

Table II shows the structure of the complete input data deck for the previously shown test case. The bFLAG cards inform SLACINPT of the end of SLAC gun code data sections. See reference 1 for a description of these. A card-by-card description of the boundary data section follows:

Table II

Computer Printout of User's Input

```
                              DATE OF RUN:  82-02-02
                              TIME OF RUN:  13.44.04.
=================================================================


                         COPY OF USER INPUT TO SLACINPT
                          ------------------------------
     INJECTION GUN MODEL 4-1A GRID-CATHODE REGION
 $INPUT1
  RLIM=72,ZLIM=40,POTN=4,POT=0.0,5000.0,0.0,0.0,MI=1,MAGSEG=1,
 $END
 $INPUT2
  Z1=20,Z2=40,Z3=20,BC=0.0,25.0,0.0,0.0,0.0,0.0,0.0,
 $END
 FLAG
 1 1
 0.0 0.01 0 1
-111 37.99 2.99 256.99 0.0 257.0
014  56.01 14.01
114  61.2 12.1 3.2 58.0 11.2
014 61.2 0.0
000 71.99 0.0
012 71.99 26.6
-112 70.25 27.40 1.25 70.8 26.25
012 40.8 13.5
012 38.8 13.5
112 0.0 10.4 246.61 0.0 257.0
000 0.0 0.01
200
 $INPUT5
  IZ1=1,IZ2=2,IZS=10,RAD=257,RMAX=37.99,UNITIN=0.01,SPC=0.0,
 $END

 FLAG



-----------------------------------------------------------------
```

Card 1 N1 N2 - these two integers control the printout produced by SLACINPT

N1=0: suppresses printout of boundary segment information

N1=1: produces printout of the boundary segment information, as shown in Table III

N2=0: suppresses printout of boundary mesh point data

N2=1: produces printout of boundary mesh point data, as shown in Table IV

Card 2 PY(1,1),PX(1,1) - the R and Z coordinates (real numbers) of the starting boundary point of the first boundary segment

BPR(1),BPZ(1) - the R and Z coordinates (integer numbers) of the first boundary mesh point

Card 3 to (N+1) - N is the number of segments
- one card required per segment

IJK - a signed 3 digit integer code to describe
(1) the shape of the boundary segment
(2) the type of boundary segment (Dirichlet, Neumann or transition)
(3) the potential number of the segment

Table III

Computer Printout of Boundary Segment Information

```
THE TOTAL NUMBER OF SEGMENTS IS:          11
THE NUMBER OF ARC SEGMENTS IS:             4

THE NUMBER OF TRANSITION BOUNDARIES IS:    0
THE NUMBER OF NEUMANN BOUNDARIES IS:       2
THE NUMBER OF DIRICHLET BOUNDARIES IS:     9

SEG  BOUNDARY  POT  SEGMENT            INITIAL           END            ARC CENTRE      RADIUS
 #     TYPE     #    TYPE            R        Z       R        Z       R        Z


 1   DIRICHLET  1   CW  ARC         0.00      .01   37.99     2.99   -1.05   257.00    256.990
 2   DIRICHLET  4   SLOPE=  1.6352 37.99     2.99   56.01    14.01
 3   DIRICHLET  4   CCW ARC        56.01    14.01   61.20    12.10   58.05    11.54      3.200
 4   DIRICHLET  4   HORIZONTAL     61.20    12.10   61.20     0.00
 5   NEUMANN    0   VERTICAL       61.20     0.00   71.99     0.00
 6   DIRICHLET  2   HORIZONTAL     71.99     0.00   71.99    26.60
 7   DIRICHLET  2   CW  ARC        71.99    26.60   70.25    27.40   70.78    26.27      1.250
 8   DIRICHLET  2   SLOPE=  2.1187 70.25    27.40   40.80    13.50
 9   DIRICHLET  2   VERTICAL       40.80    13.50   38.80    13.50
10   DIRICHLET  2   CCW ARC        38.80    13.50    0.00    10.40    -.18   257.01    246.610
11   NEUMANN    0   HORIZONTAL      0.00    10.40    0.00      .01
```

Table IV

Sample from Computer Printout of Boundary Mesh Point Data

POINT #	POTN	R	Z	DR	DZ
1	1	0	1	2.0000	-.9900
2	1	1	1	2.0000	-.9840
3	1	2	1	2.0000	-.9741
4	1	3	1	2.0000	-.9603
5	1	4	1	2.0000	-.9426
6	1	5	1	2.0000	-.9210
7	1	6	1	2.0000	-.8955
8	1	7	1	2.0000	-.8661
9	1	8	1	2.0000	-.8328
10	1	9	1	2.0000	-.7957
11	1	10	1	2.0000	-.7546
12	1	11	1	2.0000	-.7096
13	1	12	1	2.0000	-.6607
14	1	13	1	2.0000	-.6080
15	1	14	1	2.0000	-.5513
16	1	15	1	2.0000	-.4907
17	1	16	1	2.0000	-.4261
18	1	17	1	2.0000	-.3577
19	1	18	1	2.0000	-.2853
20	1	19	1	2.0000	-.2091
21	1	20	1	2.0000	-.1289
22	1	21	1	.5133	-.0447
23	1	22	2	2.0000	-.9566
24	1	23	2	2.0000	-.8646
25	1	24	2	2.0000	-.7687
26	1	25	2	2.0000	-.6688
27	1	26	2	2.0000	-.5649
28	1	27	2	2.0000	-.4571
29	1	28	2	2.0000	-.3453
30	1	29	2	2.0000	-.2296
31	1	30	2	.8899	-.1099
32	1	31	3	2.0000	-.9862
33	1	32	3	2.0000	-.8585
34	1	33	3	2.0000	-.7269
35	1	34	3	2.0000	-.5912
36	1	35	3	2.0000	-.4515
37	1	36	3	2.0000	-.3079
38	1	37	3	2.0000	-.1602
39	4	38	3	.0064	-.0039
40	4	39	4	.6416	-.3923
41	4	40	5	2.0000	-.7808
42	4	41	5	.2768	-.1693
43	4	42	6	.9120	-.5577
44	4	43	7	2.0000	-.9462
45	4	44	7	.5472	-.3346
46	4	45	8	2.0000	-.7231
47	4	46	8	.1824	-.1115
48	4	47	9	.8176	-.5000
49	4	48	10	2.0000	-.8885
50	4	49	10	.4528	-.2769
51	4	50	11	2.0000	-.6654
52	4	51	11	.0880	-.0538
53	4	52	12	.7232	-.4423
54	4	53	13	2.0000	-.8307
55	4	54	13	.3584	-.2192
56	4	55	14	.9936	-.6077
57	4	56	15	2.0000	-.9961

I=0:	segment is a straight line
I=+1:	segment is a counter-clockwise arc
I=-1:	segment is a clockwise arc
J=0:	segment is a Neumann boundary
J=1:	segment is Dirichlet boundary
J=2:	segment is a transition boundary
$K \leq 9$:	the potential number of the boundary segment; K=0 for a Neumann or transition boundary segment

PY(I,2),PX(I,2) - the R and Z coordinates (real numbers) of the end point of the Ith boundary segment

R,CY(I),CX(I) - the radius and user's estimate (real numbers) of the centre coordinates for an arc boundary segment. This centre estimate need not be accurate since SLACINPT calculates the actual coordinates and only uses the user's estimate to select the correct centre from the two possible centres.

Card (N+3) - indicates the end of boundary input
- any value $\geq$ 200 can be used
- value must be 999 if special boundary data follows

2.5 Running Successive Problems

When running successive problems, the SLAC gun code requires that the input data deck of the second problem immediately follows the last card of the first problem. A parameter, SAVE in namelist $INPUT5, is used to signal the gun code whether any relationship exists between the two problems. SAVE can have the values 0, 1 or 2.

If SAVE=0 is specified, each problem has its own unique input data deck, including boundary specifications. FLAG cards are required before each boundary data block and at the end of each problem.

If SAVE=1 is specified, the second problem will use the same boundary conditions as the first problem, so boundary input cards are not needed for the second problem. FLAG cards are only needed before the boundary data block of the first problem and at the end of the last problem using this boundary data.

If SAVE=2 is specified, the gun code will use the output from one problem in the next problem with new input data, including new boundary conditions. FLAG cards in the input deck are positioned as for a SAVE=0 problem.

2.6 Special Boundary Conditions

General Neumann and other special boundary conditions such as dielectric surfaces may be used in the SLAC gun code. The presence of special boundary data is signaled by using special numbers to terminate the normal boundary data.

SLACINPT recognizes the end of normal boundary by a code number ($IJK \geq 200$) and then copies this number to the end of the boundary data generated for the gun code by SLACINPT.

3. Program Description

3.1 Overall Program Description

In describing the operation of SLACINPT, various subroutines and variables are named - a short description of each subroutine is given in section 3.2 and a definition of the most common variables in section 3.3.

SLACINPT creates a file, INFILE, which is used by the SLAC gun code as input. INFILE is identical in structure to the original input to the SLAC gun code except that INFILE now contains all of the boundary mesh point data for the boundary system - the gun code does not have to do any "fitting" to fill in any missing boundary mesh points.

Upon entering the main routine, SLACBND, subroutine COPINPT is called to copy all of the user's input data to the printout (as shown in Table II). Next, the first section of the user's input is read and saved for later use. The subroutine SEGPTS is then called to read and decode the boundary segment data and program control returns to SLACBND where a printout of the boundary information is produced if the user has selected N1=1. An example of a printout is shown in Table III.

Before starting the boundary mesh point data generation, the subroutine VALID is called to validate the boundary segments according to the rules specified in sections 2.3.3 and 2.3.4. Any error condition detected produces an error message and terminates the job.

The program then proceeds to generate the boundary mesh point data starting with the first segment and the initial boundary point's coordinates. The pointers I, J and N are referred to often in this description and are defined as follows: I is the current segment pointer, N is the next segment pointer and J is the current boundary mesh point pointer. The boundary mesh point data generated will consist of three parts:

(1) BPR(J),BPZ(J) - the mesh coordinates of the Jth boundary point

(2) DR(J),DZ(J) - the distances from the Jth boundary mesh point to the boundary surface along the R and Z mesh line, respectively

(3) IBPT(J) - the potential number of the Jth boundary mesh point

SLACBND calls on one of a number of subroutines, depending on the current boundary segment shape, to generate the boundary mesh points.

For a straight line segment, one of four subroutines is called:

(1) HORZ - for horizontal segments

(2) VERT - for vertical segments

(3) POSSLOP - for positive sloped segments

(4) NEGSLOP - for negative sloped segments

Each subroutine will then determine the type of crossover to the next segment. For crossovers from one straight line segment to another straight line segment, the boundary mesh point data is generated within one of these subroutines (i.e., each straight line to straight line crossover has its own program section within the subroutine).

If the crossover is from a straight line to an arc segment, then the straight line segment subroutine calls one of the following subroutines:

(1) HORZARC - called by HORZ for a horizontal to arc crossover

(2) VERTARC - called by VERT for a vertical to arc crossover

(3) POSVARC - called by POSSLOP for a positive slope to arc crossover

(4) NEGVARC - called by NEGSLOP for a negative slope to arc crossover

Each one of these subroutines is divided into two parts to treat crossovers to CW or CCW arcs separately. Each part is further subdivided depending on the quadrant position, QP, of the arc segment starting point and the direction of the current straight line segment. The quadrant Q (or QP) = 1 is defined to be the lower right hand quadrant of a circle; quadrants 2, 3 and 4 are the other quadrants in a counter-clockwise direction. A more detailed definition of Q and QP is found in section 3.2.

For crossovers from an arc segment to straight line segment, there are two sets of subroutines available: one set for CW arcs and the other for CCW arcs. The CCW arc to straight line crossover subroutines are: ARCHORZ for crossovers to horizontal segments, ARCVERT for crossovers to vertical segments, ARCPOSV for crossovers to positively sloped segment and ARCNEGV for crossovers to negatively sloped segments. The CW arc to straight line crossover subroutines are

NARCHRZ, NARCVRT, NARCPOS and NARCNEG - they treat the crossovers from a CW arc to horizontal, vertical, positively sloped or negatively sloped segments, respectively.

The subroutine selected depends on the combination of the arc direction, ID(I), and the next segment identifier, N. Each of the arc to straight line crossover subroutine functions in a similar manner. The quadrant, QP, in which the arc segment end point lies is determined as well as the quadrant, Q, in which the current boundary point lies. As long as the current boundary point is not in the same quadrant as the arc segment end point (i.e., $Q \neq QP$), the crossover subroutine calls the subroutine ARC to generate the boundary mesh point data. When Q=QP, the program jumps to sections within the crossover subroutine depending on the current boundary point's quadrant and the direction of the next segment.

In section 4.1, the methods used to generate the DR(J) and DZ(J) values for the current boundary mesh point BPR(J),BPZ(J) are described in some detail. In general, the method consists of determining the intersection points made on the boundary surface by the R and Z mesh lines through a boundary mesh point and then calculating the directed distance from the boundary mesh point to the intersection point. If there is no intersection with the current or the next segment, the distance along the mesh line is arbitrarily set to 2.0.

If a boundary mesh point seems acceptable (i.e., $|DR(J)|$ and/or $|DZ(J)| < 1.0$), program control passes to the subroutine RESULT. This subroutine performs some more adjustments on the DR(J) and DZ(J) values so that these values conform to the requirements of the gun code. These modifications are described in section 4.2. Another function of RESULT is to determine the potential number, IBPT(J), for each boundary mesh point. The method of determining the potential number is detailed in section 4.3.

After adjusting the DR(J) and/or DZ(J) values and finding the potential number IBPT(J), the boundary mesh point data is printed if the user has selected this option (N2=1). An example of this printout is given in Table IV. RESULT then determines the mesh coordinates of the next possible boundary mesh point using the method described in section 4.4. Program control then returns to the calling subroutine to try and generate the data for this new boundary mesh point. When the subroutine detects that the new point has crossed over into the region of the next boundary segment, program control returns to SLACBND to select the next appropriate segment subroutine. When the last segment becomes the current segment, then the first segment is considered to be the next segment to close the boundary system loop. Boundary mesh point data generation terminates when the program detects that all the points for the last segment have been generated.

After determining all the boundary mesh point data, SLACBND checks the points for duplication. If any but the first and last point have the same R and Z coordinates, an error message is printed and the program is terminated. If the first and last point have the same coordinates, then the first point assumes the boundary mesh point data for the last point and the last point is discarded.

If the boundary mesh point data generation was successful, the first section of the user's input data (read and saved at beginning of the run) is written to INFILE. SLACBND then writes all of the boundary mesh point data (potential numbers, R and Z coordinates of the boundary mesh points and the DR(J) and DZ(J) values) to the file INFILE. The boundary data is terminated by 200 0 0 0.0 0.0 (or 999 0 0 0.0 0.0 if special boundary cards follow) as required by the SLAC gun code. The third section of the gun input data is then read and written to the file INFILE. The SLACINPT program then terminates. Some error checking is done by the various routines and a description of error messages produced is found in section 3.4.

3.2 Parameter Definitions

This section is a glossary for the more commonly used variable names in the program.

BPR(J),BPZ(J)	- the R and Z coordinates of the Jth boundary mesh point
CY(I),CX(I)	- R and Z coordinates of the center if the Ith segment is an arc - on input, user specifies approximate coordinates, program then calculates actual coordinates
DR(J)	- directed distance along the R mesh line from the Jth boundary mesh point to the boundary segment surface
DZ(J)	- directed distance along the Z mesh line from the Jth boundary mesh point to the boundary segment surface
I	- current segment pointer
IBS(I)	- boundary segment type identifier - 0 for Neumann, 1 for a Dirichlet and 2 for a transition
ICRR	- a pointer to the current or the next boundary segment intersected by the R mesh line through the current boundary mesh point - has the value I or N

ICRZ	- a pointer to the current or the next boundary segment intersected by the Z mesh line through the current boundary mesh point - has the value I or N
ID(I)	- defines the direction for an arc boundary segment = 0 for a straight line = +1 for a CCW arc = -1 for a CW arc
ITC=IT(I)	- current segment shape identifier = 0 current segment is an arc = +1 current segment is horizontal with increasing Z = -1 current segment is horizontal with decreasing Z = +2 current segment is vertical with increasing R = -2 current segment is vertical with decreasing R = +3 current segment has positive slope with increasing R = -3 current segment has positive slope with decreasing R = +4 current segment has negative slope with increasing R = -4 current segment has negative slope with decreasing R
ITN=IT(N)	- next segment shape identifier - can assume the same values as specified for ITC
J	- current boundary mesh point pointer
K	- an integer equal to $\|KK\|$

KK	- a signed integer used to select the algorithm for determining the coordinates of the next boundary mesh point - a positive value indicates a continuation of data generation using the current subroutine - a negative value causes a return to the calling routine
N	- next segment pointer
NA	- number of arc segments
NCT	- number of transition segments
NMTS	- number of Dirichlet segments
NNEU	- number of Neumann segments
NPOTN	- number of potential numbers
NS	- total number of segments
N1	- controls printout of segment specification = 0: no printout requested = 1: printout requested
N2	- controls printout of boundary data generated = 0: no printout requested = 1: printout requested

PY(I,1),PX(I,1)-	the R and Z coordinates of the starting point of the Ith segment
PY(I,2),PX(I,2)-	R and Z coordinates of the end point of the Ith segment
Q	- defines the quadrant (with respect to the current arc segment centre) in which the current boundary mesh point is located - Q=1 if BPR(J) $<$ CY(I), BPZ(J) $\geq$ CX(I) - Q=2 if BPR(J) $\geq$ CY(I), BPZ(J) $\geq$ CX(I) - Q=3 if BPR(J) $\geq$ CY(I), BPZ(J) $<$ CX(I) - Q=4 if BPR(J) $<$ CY(I), BPZ(J) $<$ CX(I)
QP	- defines the quadrant (with respect to an arc centre) of the end point of a current arc segment or the starting point of the next line segment - has same values as Q
R	- radius of an arc segment
RAD(I)	- an array containing the radii of the arc segments (=0 if a segment is not an arc)
SEGP(I)	- defines the potential number of segment I
SG	- an integer, +1 or -1, defining the direction of the current boundary segment - its value is dependent on the current segment shape identifier

- for straight line segments
  - SG = +1 when ITC = 1, 2, 3 or 4
  - SG = -1 when ITC = -1, -2, -3 or -4
- for all arc segments (ITC=0)
  - SG = ID(I) if the Z coordinate of the current boundary mesh point is $\geq$ Z coordinate of the arc centre (Q=1 or 2)
  - SG = -ID(I) if the Z coordinate of the current boundary mesh point is < Z coordinate of the arc centre (Q=3 or 4)

SLP(I)	- slope for a sloped line segment I

3.3 Description of Program Routines

3.3.1 Program SLACBND

SLACBND is the main program and performs the following functions:

(1) copies the input data deck to the printout,

(2) copies the first section of the user's input to the file INFILE,

(3) reads the boundary specification data and decodes the data using the subroutine SEGPTS,

(4) prints the boundary specification data, if requested, using the subroutine PRISEGS,

(5) uses the subroutine VALID to check the validity of the boundary specification data,

(6) selects the appropriate boundary segment subroutines and generates the boundary data,

(7) validates the generated data and then writes the data to the file INFILE, and

(8) copies the third section of the user's input to the file INFILE.

3.3.2 Subroutine COPINPT

The subroutine COPINPT prints the date and time of the run and then the input data deck.

3.3.3 Subroutines READS, WRITES

The subroutine READS is called by SLACBND to read the first and third section of the input data into a scratch file. The first section is always read, whereas the third section is read only if the boundary mesh point data generation was successful (i.e., SLACINPT did not terminate because of input or boundary mesh point errors).

The subroutine WRITES is called to write the contents of the scratch file to INFILE.

3.3.4 Subroutine SEGPTS

The subroutine SEGPTS is called by the program SLACBND to read and decode the user's boundary specification. It performs the following tasks:

(1) assigns a pointer, I, to each segment,
(2) determines each boundary segment shape and then assigns a value to the segment shape identifier, IT(I), and the arc direction pointer, ID(I), for an arc segment,
(3) determines the accurate center for an arc segment using the subroutine CENTR,
(4) calculates the slope, SLP(I), for any sloped line segments,
(5) determines the boundary segment type (Neumann, Dirichlet or transition) and assigns the value 0, 1 or 2 to the boundary segment type identifier, IBS(I),
(6) determines the potential number, POTN(I), for each boundary segment.

3.3.5 Subroutine PRISEGS

The subroutine PRISEGS is called by SLACBND to print the boundary specification if the user requested it by setting printout control parameter N1=1. Table III shows the printout for the problem shown in Figure 1.

3.3.6 Subroutine VALID

The subroutine VALID is called by SLACBND to validate the boundary specifications read by the subroutine SEGPTS. If any specifications are invalid according to the conditions described in sections 2.3.3 and 2.3.4 then an error condition exists and an appropriate error message is printed and the job is terminated.

3.3.7 Subroutine CENTR

The subroutine CENTR is called by SEGPTS to calculate the centre of an arc given the arc end point's coordinates, the arc radius and a guess at the centre's coordinates. Using the end point and the radius, CENTR finds the coordinates of the two possible centres. The required centre is then the calculated centre that is closer to the estimated centre.

3.3.8 Subroutine RESULT

The subroutine RESULT is called by the various segment subroutines to

(1) make final adjustments to the DR(J) and DZ(J) distances (section 4.2),

(2) determine the potential number, IBPT(J), for the current boundary mesh point (section 4.3),

(3) print the boundary mesh point data if the user requested it by setting the printout control parameter N2=1 (see Table IV), and

(4) determine the coordinates of the next boundary mesh point (section 4.4).

3.3.9 Subroutines SUBPOSV, SUBNEGV

The subroutine SUBPOSV is called by subroutine POSSLOP to generate the boundary mesh point data when both the R and Z mesh lines through the boundary mesh point intersect the current positively sloped line segment. Similarly, SUBNEGV is called by the subroutine NEGSLOP if the current segment has a negative slope. The subroutines in turn call RESULT to further treat the boundary data and generate the coordinates of the next boundary mesh point. Control then returns to the calling program.

3.3.10 Subroutines SUBDZ, SUBDR

The subroutines SUBDZ and SUBDR are used to calculate the DR(J) and DZ(J) distances from the current boundary mesh point to a sloped line segment along the R and Z mesh lines, respectively.

3.3.11 Subroutine HORZ

The subroutine HORZ is called by SLACBND if the current segment is a horizontal line segment. The subroutine is divided into various sections, each of which treats a particular type of crossover region to the next section. The crossover section selected is dependent on the current segment's direction indicator, SG, and the next segment's shape identifier, ITN. For the case of ITN=0 (next segment is an arc), HORZ calls the subroutine HORZARC to generate the boundary mesh point data. The subroutine returns to SLACBND if HORZ determines that the current boundary mesh point has crossed over into the next segment's region.

3.3.12 Subroutine HORZARC

The subroutine HORZARC is called by the routine HORZ to treat the crossovers from a horizontal line segment to an arc segment. The subroutine is divided into two sections: for crossovers to either a CW or a CCW arc segment. Each section is further subdivided into four parts to treat the crossovers that can occur in the four possible quadrants. After generating a boundary mesh point, RESULT is called to further treat the boundary data and determine the coordinates of the next boundary mesh point. Control returns to SLACBND (via HORZ) when HORZARC detects that the next boundary mesh point has crossed over into the next segment's region.

3.3.13 Subroutines VERT, VERTARC

The subroutines VERT and VERTARC are similar in their functions to the routines HORZ and HORZARC, except that they treat crossovers from a vertical boundary segment to the next segment.

3.3.14 Subroutine POSSLOP

The subroutine POSSLOP is called by SLACBND to generate boundary mesh point data for boundary mesh points along a positively sloped segment. Each crossover to the next segment is treated separately, depending on the direction of the current segment and the shape of the next segment. If the next segment is an arc, the subroutine POSVARC is called to generate the boundary mesh points. If the R and Z mesh lines through the current boundary mesh point both intersect the current positively sloped segment, the subroutine SUBPOSV is called to generate the data. If only one of the mesh lines intersects the current segment, the DR(J) or DZ(J) value is calculated by calling the subroutine SUBDR or SUBDZ, respectively. The distance along the other mesh line is then found within the subroutine. RESULT is then called to further treat the boundary data, and the coordinates of the next boundary mesh point are determined. When POSSLOP detects that the next point has crossed over into the next segment's region, control returns to SLACBND.

3.3.15 Subroutine POSVARC

The subroutine POSVARC is called by POSSLOP to treat the special case of crossovers from a positively sloped segment to an arc segment. The CW and CCW arc segments are again handled by separate sections, and each section is further broken up into smaller parts to treat the boundary mesh point with regard to the crossover quadrant, QP, determined from the current segment's end point.

After the boundary mesh point data has been determined, RESULT is called for further data treatment and the coordinates of the next boundary mesh point are determined. When POSVARC detects that the next boundary mesh point has crossed into the next segment's region, control returns (via POSSLOP) to SLACBND.

3.3.16 Subroutines NEGSLOP, NEGVARC

The subroutines NEGSLOP and NEGVARC are similar in their function to the subroutines POSSLOP and POSVARC, respectively, except that they treat crossovers from a negatively sloped line segment.

3.3.17 Subroutine ARC

The subroutine ARC is called by one of the subroutines that treat the crossovers from an arc segment to a straight line segment.

It is only called when the current boundary mesh point is not located in the same quadrant as the current arc segment's end point. ARC calls on the subroutines ROOTR (or ROOTZ) to find the intersection points (RT1 and RT2) made with the arc segment by the R and Z mesh lines through the current boundary mesh point. DR(J) or DZ(J) then is the distance from the current boundary mesh point to one of the intersection points, RT1 or RT2, as calculated by ROOTR (or ROOTZ). Which intersection point is used will depend on ID(I), the direction of the current arc segment and on Q, the quadrant pointer for the current boundary mesh point.

The subroutine is used to treat both CW and CCW segments. For CCW arc segments, the boundary mesh point will always be on the "outside" of the arc and the R or Z mesh lines need not always intersect the arc - in this case, DR(J) or DZ(J) is set equal to 2.0. For CW arc segments, however, the boundary mesh point is always "inside" the arc and the R and Z mesh lines must always intersect the arc. If ROOTR or ROOTZ do not find any intersection points, an error condition exists, an appropriate error message is issued and the job is terminated. After determining the boundary mesh point data, RESULT is called for further data treatment and to determine the coordinates of the next boundary mesh point. Program control then returns to the calling subroutine to find the quadrant pointer for the new boundary mesh point.

3.3.18 Subroutine ARCHORZ

The subroutine ARCHORZ is called by SLACBND to calculate the boundary mesh point data for crossovers from a CW arc segment to a horizontal straight line segment.

ARCHORZ first determines the quadrant pointers, QP and Q. If Q and QP are not equal, the current boundary mesh point is not located in the same quadrant as the arc segment's end point (i.e., the point is not in the crossover quadrant) and the subroutine ARC is called to generate the boundary mesh point data and the coordinates of the next boundary mesh point. If Q and QP are equal (i.e., the point lies in the crossover quadrant), the boundary mesh point data is generated by one of four subroutine parts. Which part is used is determined by the combination of Q, the quadrant pointer for the boundary mesh point, and ITN, the shape identifier of the horizontal segment. After calculating the boundary mesh point data, RESULT is called for further data treatment and to determine the coordinates of the next boundary mesh point.

The subroutine will then determine the new quadrant pointer, Q, for the new boundary mesh point. Each of the four parts of ARCHORZ checks to see if the boundary mesh point has crossed over into the horizontal segment's region - this causes program control to return to SLACBND.

3.3.19 Subroutines ARCVERT, ARCPOSV, ARCNEGV

The subroutines ARCVERT, ARCPOSV and ARCNEGV are similar in function to the subroutine ARCHORZ except that they treat crossovers to vertical, positively sloped, and negatively sloped segments, respectively.

3.3.20 Subroutines NARCHRZ, NARCVRT, NARCPOS, NARCNEG

The subroutines NARCHRZ, NARCVRT, NARCPOS and NARCNEG are called by SLACBND to treat crossovers from a CW arc segment to a horizontal, vertical, positively sloped, or negatively sloped segment, respectively. Each subroutine is similar in its function to its CCW equivalent. The major difference is that for all CW arc segments, the R and/or Z mesh lines through the boundary mesh point must have an intersection point with the arc segment - if none is found, an error condition exists.

3.3.21 Subroutine QUAD

The subroutine QUAD is called by one of the arc-to-straight line crossover subroutines to determine the quadrant pointer, Q, for the current boundary mesh point.

3.3.22 Subroutine QEND

The subroutine QEND is called by one of the arc to straight line (or vice versa) subroutines to determine the crossover quadrant pointer, QP, for the starting point of an arc segment (or the end point of a straight line segment).

3.3.23 Subroutine ROOTZ

The subroutine ROOTZ may be called by any of the subroutines involving arc segments to find the intersection points made with an arc by the Z mesh line through the boundary mesh point. The intersection points are found by simply substituting the R coordinate of the current boundary mesh point in the equation of the circle that has the same radius and centre as the arc segment involved. If there are real intersection points, the indicator RT will have a non-negative value and the intersection points RT1 and RT2 are determined. If the intersection points are imaginary, then the indicator RT will have a negative value upon return to the calling program.

3.3.24 Subroutine ROOTR

The subroutine ROOTR is similar to ROOTZ except that it is used for the R mesh lines.

3.4 Error Messages

SLACINPT is able to detect some errors in the boundary mesh data produced during a program run. Any error detected will produce an appropriate error message and will terminate any further program execution. The messages produced are all self-explanatory and the user should be able to quickly identify the problem area. Any error will cause job termination.

4. Calculation Descriptions

4.1 DR(J) and DZ(J) Calculations

The boundary system of a given problem can be represented by two types of line segments:

(1) a straight line (horizontal, vertical or sloped), and
(2) an arc CW (clockwise) or CCW (counter-clockwise).

The radial, R, and the axial, Z, mesh lines through the boundary mesh point at BPR(J),BPZ(J) can intersect the boundary surface in such a way that

(1) both mesh lines intersect the current boundary segment, Figure 3(a), or
(2) one mesh line intersects the current boundary segment while the other mesh line intersects the next boundary segment, Figure 3(b), or
(3) one mesh line intersects the current boundary segment while the other mesh line does not intersect the current or the next boundary segment, Figure 3(c).

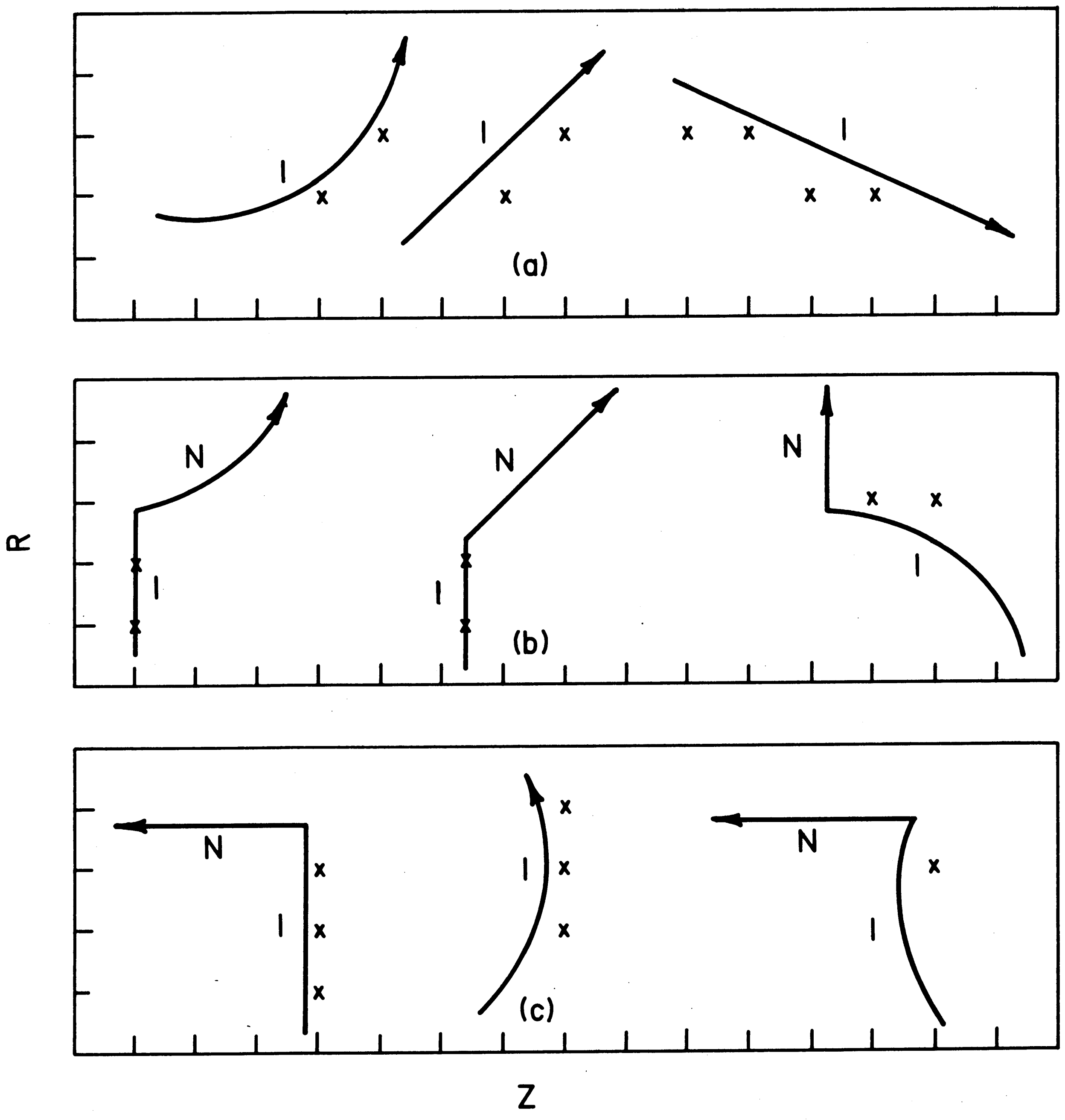

Figure 3 Examples of possible boundary-to-boundary mesh point configurations. I denotes current boundary segment, N denotes next boundary segment, and x denotes boundary mesh point.

(a) R and Z mesh lines intersect the same boundary segment.

(b) R and Z mesh lines intersect different boundary segments.

(c) Only one mesh line intersects the current or next boundary segment.

Conditions (2) and (3) define a crossover region. When calculating the DR(J) and DZ(J) distances, any one of the five calculation methods described in sections 4.1.1 to 4.1.5 can be used, depending on the intersected boundary segment shape.

4.1.1 Method 1: Intersected Segment is Horizontal

DR(J) = PY(M,1) - BPR(J), ICRR = M
where M=I for the current segment, or
M=N for the next segment.

DZ(J) and ICRZ are calculated using method 2, 3 or 4 if the Z mesh line through the current boundary mesh point intersects the next boundary segment.

4.1.2 Method 2: Intersected Segment is Vertical

DZ(J) = PX(M,1) - BPZ(J), ICRZ=M
where M=I for the current segment, or
M=N for the next segment.

DR(J) and ICRR are calculated using method 1, 3 or 4 if the R mesh line through the current boundary mesh point intersects the next boundary segment.

4.1.3 Method 3: Intersected Segment is a Sloped Line

For the distance along the Z mesh line,

DZ(J) = PX(M,1) + [BPR(J) - PY(M,1)]/SLP(M) - BPZ(J)
ICRZ = M

where M=I if distance is to the current segment, or
M=N if distance is to the next segment.

For the distance along the R mesh line

DR(J) = PY(M,1) + SLP(M) * [BPZ(J) - PX(M,1)] - BPR(J)
ICRR = M
where M=I if distance is to the current segment, or
M=N if distance is to the next segment.

4.1.4 Method 4: Intersected Segment is an Arc

Each arc segment is a section of the circle defined by the arc segment end points, the radius, and the centre calculated in subroutine CENTR. If a mesh line intersects this circle, it does so at two points - only one of which is the "correct" intersection point used to find the distance DR(J) and DZ(J).

Each boundary mesh point lies in one of the four quadrants, Q. If a mesh line intersects an arc segment, then distances DR(J) and DZ(J) will depend on Q. Using the known BPR(J) or BPZ(J) and the subroutines ROOTR or ROOTZ, the program determines the two intersection points, RT1 and RT2 (with RT1 $\geq$ RT2) on the circle.

For the R mesh line intersecting a CW or CCW arc, the distance DZ(J) is given by

DZ(J) = RT1 - BPZ(J) when Q is 1 or 2, and by
DZ(J) = RT2 - BPZ(J) when Q is 3 or 4.
ICRZ = M
where M=I if distance is to the current segment, or
M=N if distance is to the next segment.

For the Z mesh line intersecting a CW or CCW arc, distance DR(J) is given by

DR(J) = RT2 - BPR(J) when Q = 1 or 4, and by

DR(J) = RT1 - BPR(J) when Q = 2 or 3.

ICRR = M

where M=I if the distance is to current segment, or

M=N if the distance is to next segment.

If for an arc, a mesh line does not intersect the arc, then the roots, RT1 and RT2, are imaginary. For a CCW arc this is acceptable and the DR(J) or DZ(J) would be set to 2.0. For a CW arc, the boundary mesh point must be "inside" the circle and the mesh line must intersect the circle - imaginary roots for RT1 or RT2 constitute an error condition.

4.1.5 Method 5: Mesh Line Does Not Intersect the Current or Next Boundary Segment

If there is no intersection along the Z mesh line, DZ(J) = 2.0 and ICRZ = N. If there is no intersection along the R mesh line, DR(J) = 2.0 and ICRR = N.

4.2 Modifications of DR(J) and DZ(J) by RESULT

Any $|DR(J)|$ or $|DZ(J)|$ value $\geq$ 1.0 is set to 2.0 by RESULT. In addition, RESULT will make further modifications (depending on the boundary shapes and types involved), so that the DR(J) and DZ(J) values conform to the SLAC gun code requirements.

4.2.1 $|DR(J)| \geq 1.0$ and/or $|DZ(J)| \geq 1.0$

For all situations, RESULT will set DR(J) and/or DZ(J) = 2.0.

4.2.2 DR(J) = 0.0 and DZ(J) = 0.0

If the R and Z mesh lines intersect the same Neumann boundary segment, no changes are made.

If the R and Z mesh lines intersect the same transition boundary segment, RESULT sets DR(J) = DZ(J) = 2.0.

If the R and Z mesh lines intersect the same Dirichlet boundary segment, RESULT sets DR(J) = ± 0.01 and DZ(J) = ± 0.01, with the sign dependent on the direction from the boundary mesh point to the boundary surface along the R and Z mesh lines.

If the R and Z mesh lines intersect different segments (not possible for crossovers involving Dirichlet boundaries), RESULT will set the DR(J) and/or DZ(J) = 2.0 for the transition boundary segment involved.

4.2.3 DR(J) ≠ 0.0, DZ(J) = 0.0

This situation is only possible if the boundary mesh point is on the current boundary segment, which must be a vertical Neumann or transition. The next segment may be any type. If the vertical segment is a transition boundary, RESULT sets DR(J) = 2.0.

4.2.4 DR(J) = 0.0, DZ(J) ≠ 0.0

This situation is only possible if the boundary mesh point is on the current boundary segment, which must be a horizontal Neumann or transition. The next segment may be any type. If the horizontal segment is a transition boundary, RESULT sets DZ(J) = 2.0.

4.2.5 DR(J) $\neq$ 0.0, DZ(J) $\neq$ 0.0

This situation is only possible if the segments involved are Dirichlet boundaries. Any $|DR(J)|$ or $|DZ(J)| \geq 1.0$ is set to 2.0 by RESULT.

4.3 Determination of Potential Number, IBPT(J)

The potential number for a boundary mesh point is used by the SLAC gun code to select a potential value for the point from a potential array. In general, the potential number of a boundary mesh point is the same as the potential number of the line segment, SEGP(M), where M = ICRZ or ICRR, depending on the intersected segment. The potential number will depend on the values DR(J) and DZ(J), and on the intersected boundary segment types.

If the intersected segment pointers are different, i.e., ICRZ $\neq$ ICRR, then the potential number depends on the DR(J) and DZ(J) values as well as the intersected boundary segment types.

4.3.1 $|DZ(J)| < 1.0$ and $|DR(J)| \geq 1.0$

The potential number is the same as the potential number for the intersected boundary segment ICRZ.

4.3.2 $|DZ(J)| \geq 1.0$ and $|DR(J)| < 1.0$

The potential number is the same as the potential number for the intersected boundary segment ICRR.

4.3.3 $|DZ(J)| < 1.0$ and $|DR(J)| < 1.0$

(1) If ICRZ = ICRR, then the potential number is the same as the potential number of the intersected boundary segment ICRZ.

(2) If ICRZ ≠ ICRR, there are three possibilities

(i) both segments have the same potential number: the potential number of the boundary mesh point is the same as the current segment's,

(ii) one segment is a Dirichlet boundary type: the potential number is the same as the one for the Dirichlet boundary segment, and

(iii) one segment is a Neumann and the other is a transition: the potential number is always zero.

4.4 Coordinates of Next Possible Boundary Mesh Point

The position of the next possible boundary mesh point depends on the current DR(J) and DZ(J) values, and on the current and next segment shapes. For current boundary mesh point at BPR(J), BPZ(J), there are four possible combinations of DR(J) and DZ(J)

(i) $|DR(J)| \geq 1.0$ and $|DZ(J)| \geq 1.0$
(ii) $|DR(J)| \geq 1.0$ and $|DZ(J)| < 1.0$
(iii) $|DR(J)| < 1.0$ and $|DZ(J)| \geq 1.0$
(iv) $|DR(J)| < 1.0$ and $|DZ(J)| > 1.0$

For case (i), the directed DR(J) and DZ(J) are not acceptable and the current boundary mesh point is invalid. The coordinates of a new boundary mesh point are then determined in terms of the invalid point's coordinates and the current value for SG. This new point is always determined within the current segment's subroutine.

For cases (ii), (iii) and (iv), the directed distances are valid and the boundary mesh point data is saved (and printed if requested by the user) by the subroutine RESULT. The next boundary mesh point's coordinates are then determined in terms of the last point's coordinates and the current value for SG.

4.4.1 Determination of Segment Direction Indicator, SG

Before describing the individual cases, the method of determining SG, the segment direction indicator, is described. In all cases, SG will always be either +1 or -1, depending on the current segment type.

(1) For straight line segments, SG depends only on the R and Z direction of the segment.

(i) HORIZONTAL SEGMENT

in +Z direction: SG = +1
in -Z direction: SG = -1

(ii) VERTICAL SEGMENT

in +R direction: SG = +1
in -R direction: SG = -1

(iii) POSITIVELY SLOPED SEGMENT

in +R and +Z direction:	SG = +1
in -R and -Z direction:	SG = -1

(iv) NEGATIVELY SLOPED SEGMENT

in +R and -Z direction:	SG = +1
in -R and +Z direction:	SG = -1

(2) For an arc segment, SG depends on the quadrant position (Q) of the current boundary mesh point and the CW or CCW direction of the arc.

Q	CW arc	CCW arc
1 or 2	SG = -1	SG = +1
3 or 4	SG = +1	SG = -1

4.4.2 Determination of Next Boundary Mesh Point Coordinates

4.4.2.1 CASE I $|DR(J)| \geq 1.0$ and $|DZ(J)| \geq 1.0$

Since a boundary point can not have both $|DR(J)|$ and $|DZ(J)| \geq 1.0$, the point is shifted by 1 mesh unit along either the R or the Z mesh line using either equation A or B.

A) BPR(J) = BPR(J) + SG

B) BPZ(J) = BPZ(J) - SG

A) is selected if the current segment is horizontal, positively sloped, or an arc segment (CW or CCW) for which the Q is 1 or 3.

B) is selected if the current segment is vertical, negatively sloped, or an arc segment (CW or CCW) for which the Q is 2 or 4.

The boundary mesh point pointer, J, remains unchanged.

4.4.2.2 CASE II $|DR(J)| \geq 1.0$ and $|DZ(J)| < 1.0$

The next possible boundary mesh point has pointer J=J+1 and coordinates given by

$$BPR(J) = BPR(J-1) + SG$$

$$BPZ(J) = BPZ(J-1)$$

4.4.2.3 CASE III $|DR(J)| < 1.0$, $|DZ(J)| \geq 1.0$

The next possible boundary mesh point has pointer J=J+1.

If Q=2 or 3 for the current boundary mesh point, or if the current boundary segment has a negative slope, the coordinates of the next possible boundary mesh point are given by

$$BPR(J) = BPR(J-1)$$

$$BPZ(J) = BPZ(J-1) + SG$$

For all other situations, the coordinates are given by

BPR(J) = BPR(J-1)

BPZ(J) = BPZ(J-1) + SG

4.4.2.4 CASE IV $|DR(J)| < 1.0$ and $|DZ(J)| < 1.0$

(1) R and Z mesh lines intersect the current segment.

(i) current segment is a positively sloped segment or an arc segment with Q = 1 or 3

BPR(J) = BPR(J-1)

BPZ(J) = BPZ(J-1) + SG

(ii) current segment is a negatively sloped segment or an arc segment with Q = 2 or 4

BPR(J) = BPR(J-1) + SG

BPZ(J) = BPZ(J-1)

(2) R mesh line intersects the current segment, and
Z mesh line intersects the next segment.

(i) straight segment to straight segment crossover
(a) SG*ITN>0

BPR(J) = BPR(J-1) + SG

BPZ(J) = BPZ(J-1)

(b) SG*ITN<0

BPR(J) = BPR(J-1) - SG

BPZ(J) = BPZ(J-1)

(ii) arc segment to straight segment crossover

(a) Q = 2 or 4

BPR(J) = BPR(J-1) + SG

BPZ(J) = BPZ(J-1)

(b) Q = 1 or 3

BPR(J) = BPR(J-1) - SG

BPZ(J) = BPZ(J-1)

(iii) straight segment to arc segment crossover

BPR(J) = BPR(J-1) - SG

BPZ(J) = BPZ(J-1)

(3) R mesh line intersects the next segment, and
Z mesh line intersects the current segment.
For all types of crossovers

BPR(J) = BPZ(J-1)

BPZ(J) = BPZ(J-1) + SG

The above four cases hold true for most of the boundary mesh points generated. But there are some situations that require special considerations, all of them involving crossovers. Section 5 discusses some typical crossover situations and the special treatment required.

5. Some Typical Crossover Examples

Figure 4 shows some examples of crossovers from an arc or sloped segment to a straight line segment. In discussing these cases the following definitions apply:

I is the current segment number
N is the next segment number
A is the last valid boundary mesh point generated
B is the current possible boundary mesh point
C, D are other mesh points.

For the type of crossover shown in Figs. 4(a) and 4(b), SLACINPT calculates three values for the current possible boundary mesh point:

(1) DZ(J), the distance to the segment I,
(2) DR(J), the distance to the current segment or to the next segment, and
(3) TZ, the distance to the next segment.

After mesh point A has been validated as a boundary mesh point, RESULT has put the next possible boundary mesh point at B. SLACINPT finds that mesh point B is not within one mesh unit of the current or of the next segment. Normally SLACINPT would discard B and make mesh point C the next possible boundary mesh point. But as is seen in Fig. 4(b), mesh point C would then be a single point row, i.e., it is less than one mesh unit from two different boundary

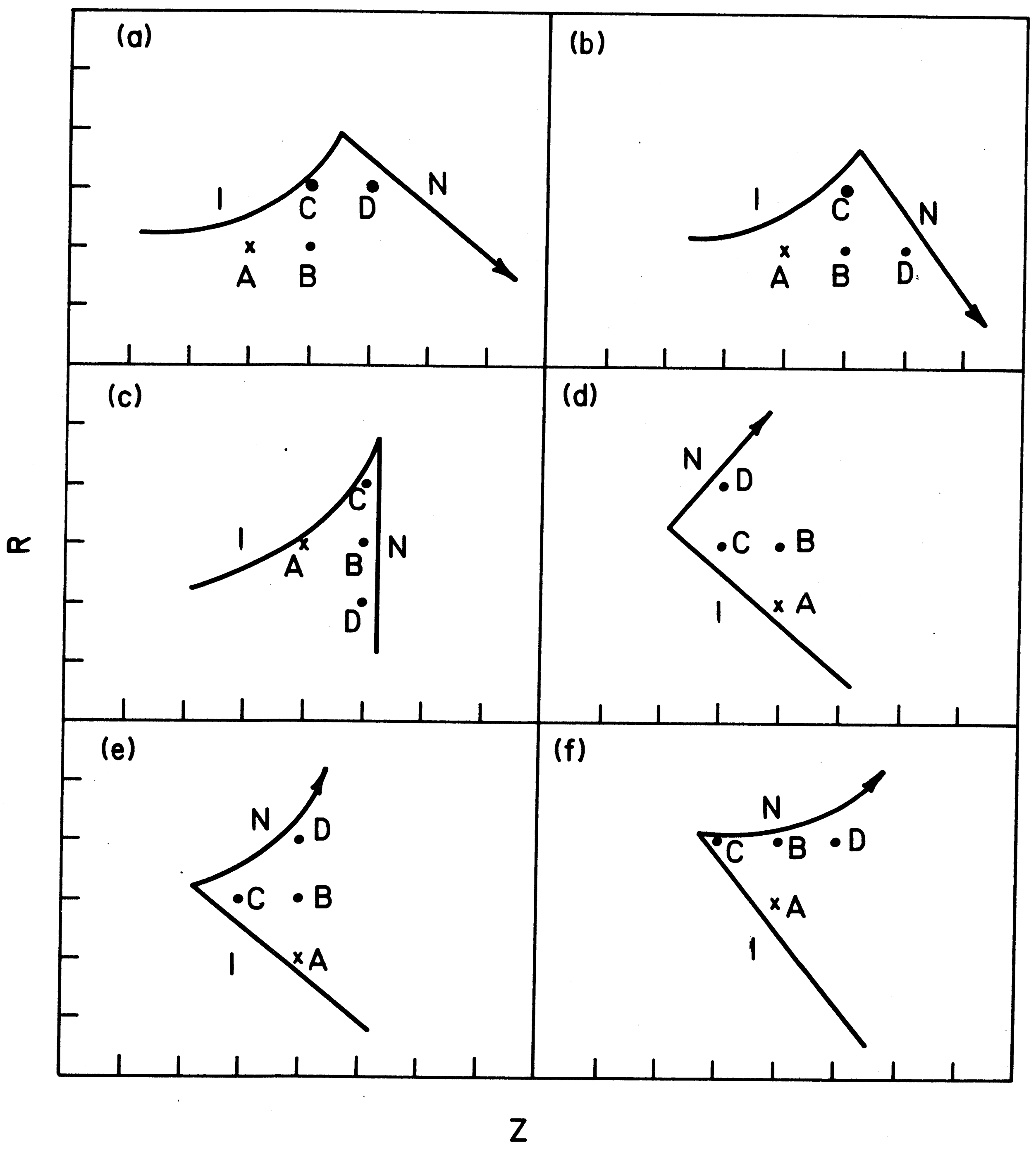

Figure 4 Examples of typical crossovers. I is the current segment number, N is the next segment number, A is the last valid boundary mesh point generated, B is the current possible boundary mesh point and C and D are other mesh points.

segments in the Z direction and the SLAC gun code considers this as an error condition. SLACINPT tests mesh point C before discarding B. In Fig. 4(a), the $|DZ(J)|$ for mesh point C is $<$ 1.0 while the $|TZ|$ is $\geq$ 1.0 and point C becomes a valid boundary mesh point and B is discarded. In Fig. 4(b), for C the $|DZ(J)|$ is $<$ 1.0 and the $|TZ|$ is also $<$ 1.0 - mesh point C cannot be a boundary mesh point. SLACINPT then forces B to become a valid boundary mesh point by setting DR(J) = $\pm$ 0.999, the sign depending on the direction to the boundary. RESULT then selects D as the next possible boundary mesh point.

Figure 4(c) shows a slightly different crossover situation. RESULT has selected mesh point B as the next possible boundary mesh point. Both $|DZ(J)|$ and $|DR(J)|$ for B are $\geq$ 1.0 but $|TZ|$ is $<$ 1.0. B is therefore a boundary mesh point since it is within one mesh unit of the next boundary segment and SLACINPT sets DZ(J) = TZ. The SLAC gun code requires that the end points of a column of boundary mesh points have the $|DR(J)| <$ 1.0 - a condition not fulfilled by point B. SLACINPT therefore sets the DR(J) = $\pm$ 0.999, the sign depending on the direction to the boundary. B is now a valid boundary mesh point and RESULT selects D as the next possible boundary mesh point.

Figures 4(d) and 4(e) are similar to 4(a) and 4(b), respectively, except they are concerned with the R direction. In Fig. 4(e), mesh point C cannot be a boundary mesh point since it is a single point column - an error condition for the SLAC gun code. SLACINPT therefore forces mesh point B to be the boundary mesh point by setting its DZ(J) = $\pm$ 0.999, the sign depending on the direction to the boundary. Figure 4(f) is similar to 4(c) except here mesh point B is a row end point with $|DZ(J)| \geq$ 1.0 - SLACINPT makes B a valid boundary mesh point by setting DZ(J) = $\pm$ 0.999, the sign depending on the direction to the boundary.

6. Summary

The program SLACINPT has been used extensively at CRNL, both on test cases and on many problems that use the SLAC gun code. Solutions for special-case situations (some of which have been described in sections 3, 4 and 5 of this report) have been introduced as the need arose; at present the program can cope with virtually all reasonable electrode configurations. Should difficulties arise, however, the user can make minor modifications to the problem boundary shape to resolve the difficulty.

Combining the new code SLACINPT with the existing SLAC gun code has created a versatile program package capable of solving a great variety of electrostatic electrode and charged particle trajectory design problems in a fast, accurate and efficient manner.

7. Acknowledgements

The authors wish to acknowledge the help of R.M. Hutcheon for useful discussions on the program and this report, and for providing feedback on use of the program.

8. References

(1) W.B. Hermannsfeldt, "Electron Trajectory Program", Stanford Linear Accelerator Center, Report SLAC-226 (1979).

ISSN 0067 - 0367

ISSN 0067 - 0367

1082–82