MATHEMATICS NOTES

NOTE 26

# BRUT

## A System of Subroutines for the Generation of Contours

July, 1972

Terry L. Brown
The Dikewood Corporation

Abstract

Subroutine BRUT is a multi-purpose code that finds and plots any number of specified contours in a given domain defined by a two-dimensional array. Plots can be produced on microfilm or Calcomp.

## I INTRODUCTION

In many instances in the investigation of EMP simulation and phenomenology contour graphs have been and continue to be used to condense much information into one graph. This has been demonstrated in the theoretical studies of the TORUS[1, 2], ARES[3, 4], ALECS and ATLAS[5] simulators as well as high altitude phenomenology.[6] Contour plots have been made of such quantities as fields in time and space, error deviation, uniformity, defraction and field and potential mappings. The field and magnetic potential or stream function mapping might prove to be especially interesting if an approach is taken as described in SSN 148[7] as opposed to that in SSN 21.[8] For example, by defining a complex potential function

$$w(\zeta) = u(x, y) + iv(x, y)$$
$$\zeta = x + iy \tag{1}$$

and allowing a complex mode function to be defined as

$$\vec{g}_o(x, y) + i\vec{h}_o(x, y) = \nabla w(\zeta) \tag{2}$$

readily gives the electric field components by

$$g_o(\zeta) = ih_o(\zeta) = \frac{\partial w(\zeta)}{\partial \zeta} \tag{3}$$

The field and potential graph are obtained by simply plotting contours of fixed values of the u and v functions. The subroutine described here, having been useful in some past studies, may prove helpful in the future regardless of the conceptual approach.

## II GENERAL

Subroutine BRUT is a sure-fire, brute-force method for calculating and plotting contours in a specified domain. Given a doubly dimensioned array defining a surface and the desired values of contours, this routine will calculate the coordinates (x,y) of each contour and plot it in a region determined by the user. Any number of contours, positive or negative, can be requested but aesthetic spacing is the user's responsibility. Since a complete search is made vertically and horizontally through the array for each contour no problem exists from multi-valued functions in x or y. Likewise, disconnected contours in the region of interest prove to be of no consequence.

This subprogram exists mainly as three distinct and separate subroutines. The first routine finds the contours as (x,y) data points, the second sequentially orders this data into an assemblage of continuous coordinates and the last is a plotting routine that graphs the ordered data. A short sorting routine called by the ordering subroutine is included.

Subroutine BRUT finds the points for a specified contour by first holding x constant and looking through the y's to determine if that particular contour value has been encountered. That is, for a given contour C

$$C_x \equiv (x_i, y_i) \approx \left\{ \Big|_{x_i = x_n} \; y_m \le y_i \le y_{m+1} \right\} \qquad \begin{array}{l} m = 1,2,3,\ldots,M \\ n = 1,2,3,\ldots,N \\ i \le 500 \end{array} \qquad (4)$$

If so, a linear interpolation is used to achieve the approximate y. The x is then incremented for another search ($x_i = x_{i+1}$) until the domain of x is exhausted. An analogous search

$$C_y \equiv (x_j, y_j) \approx \left\{ \Big|_{y_j = y_m} \; x_n \le x_j \le x_{n+1} \right\} \qquad \begin{array}{l} m = 1,2,3,\ldots,M \\ n = 1,2,3,\ldots,N \\ j \le 500 - i \end{array} \qquad (5)$$

is then made across the array holding y constant and incrementing x. Theoretically speaking then, the maximum distance between valid points (x, y) in the union of these two sets ($C_x$ and $C_y$) would be

$$\Delta_d = (\Delta_x^2 + \Delta_y^2)^{1/2}$$

where

$$\Delta_x = |x_n - x_{n+1}|, \quad \Delta_y = |y_n - y_{n+1}| \qquad (6)$$

It is assumed, of course, that the radius of curvature $R_c >> \Delta_d$. Obviously, the finer the grid, the more accurate (and smoother) the graph.

This generally discontinuous set of points generated by BRUT is then arranged by subroutine ORDER to allow a continuous line to be drawn. It is the responsibility of subroutine ORDER to determine the correct sequence of points along a given contour. This is accomplished, generally, by first sorting on the abscissa values of a contour and examining the points to determine if any points are within an appropriate distance ($\Delta_d$) of a given point (($x_i, y_i$) i = 1,2,...,N). When more than one point satisfies the criterion a look ahead feature chooses the closest point and orders accordingly. Not all starting and ending points are connected and in these circumstances a gap is left which is $\leq \Delta_d$. Subroutine ORDER may be called several times for one contour if BRUT senses points that have not been ordered. This would happen, for instance, if the particular curve appeared as completely disjointed line segments in the graphing region. It might be noted that fewer than four points are not plotted.

The last subroutine, D4, is a general-purpose linear-linear graphing routine. This routine is called from BRUT at the outset to initiated the plot. With this call, the region of interest in which the contours will be plotted is drawn and scaled. Thereafter, D4 is called from ORDER to plot, as overlays, all the curves of the various contours. It is impossible to tell in advance how many calls to D4 will be initiated from ORDER, therefore the terminating call for the plotting routine is made from BRUT. An option that might be of interest is the possibility of inserting a COMMON/ GOOP/··· card in the calling program in which the arrays SAVEX and SAVEY have been filled. Making this addition and changing the number of points to be plotted in the first call to D4 (card BRT4) would allow the user to draw some initial line or boundary pertaining to his problem. Figure 1A shows an example of an image boundary included by this method.

There are four arrays whose dimensions might need to be altered if the dimensions of A (the array defining the surface) are large. These arrays are SAVEX, SAVEY, PX and PY. An error message will be written if overflow occurs. When this happens an attempt is made to order and plot the data to that point at which overflow occurred. A general rule of thumb is to have these arrays dimensioned about 2(M+N) where M and N are the dimensions of A. Of course if a contour is very long as compared to the perimeter, for example like that in Fig. 1B, such that Eq. 4 or Eq. 5 would be satisfied many times for a given $x_i$ or $y_i$ then an increase will be necessary.

It is sometimes convenient to truncate a surface on which cross-sectional contours are desired. This can be done; however, the set of contour values should be judiciously chosen. Much time could be wasted, for example, by trying to connect a plane region of points.

If the calling routine already utilizes another plotting package such as GRAPH or CURVES, then subroutine D4 could be deleted and the existing routine used. Precautions should be taken to insure that the call located in BRUT (card BRT4) is an initiating or dummy plot, all calls in ORDER are overlays and the last call in BRUT terminates the graphing. This substitution should be fairly straightforward for microfilm. However, if Calcomp plots are desired, the change may prove to be more challenging depending upon how the new graphing package positions itself between overlays.

## III OPERATION

Subroutine BRUT will produce graphs on either microfilm or Calcomp. Plot 29 is needed from the DAFWL library to produce film while plot 20 is used for Calcomp. In the program header card of the calling routine the FILMPL file must be declared for microfilm and TAPE10 file, along with the appropriate tape request, for Calcomp. Also, for the case of Calcomp, the card

CALL PLOT (0.,0.,40)

should be added to the calling program just before completion. An output file is also required.

Storage requirements for subroutine BRUT (plus the necessary library package) is approximately $4700_8$ words exclusive of the A array. About $3300_8$ words are needed for just BRUT and ORDER if some other plotting package is already present.

The time requirements of subroutine BRUT depend on several things. Obviously, the number of contours wanted and the size of the A array are the major factors in determining time consumption. Another factor to consider is the length of the contours. Even though the search time for any two contours is the same, the time required to order the points is different if one contour is longer than the other. Looking for non-existent contours certainly must be avoided if time is of the essence.

For a handle on the timing of this approach consider the graph in Fig. 2. In this example, the dimension of the A array was 100 by 80. Fourteen contours were requested and a total of 1197 points were found and approximately that many plotted. The time required for this more or less "typical" run was just under 7.5 CPU seconds. This was the time lapse between the call to BRUT and the time control was transferred back to the calling program. A complete breakdown of the number of points found for each contour value is given in Table 1, though the value of the individual contour is unimportant. The ordering time is also listed per contour in Table 1. All storage and timing approximations are based upon the AFWL CDC 6600 computer at Kirtland Air Force Base.

To use subroutine BRUT the calling program must furnish the standard FORTRAN statement

CALL BRUT (A, M, N, P, CU, NV, I, J)

The arguments of which are defined as follows

A - the two-dimensional array (M by N) defining the surface

M - dimension of A in the y direction
N - dimension of A in the x direction } $A(M,N) \equiv (x_N, y_M)$

P - this argument is a dimensioned array (must be dimensioned 6 by the calling routine) and contains the following information defining the graphing parameters

P(1) - minimum x represented on the graph

P(2) - maximum x represented on the graph

P(3) - minimum y represented on the graph

P(4) - maximum y represented on the graph

P(5) - scaling factor for x (the length between tic marks)

P(6) - scaling factor for y [under most conditions P(5) = P(6) and since the frame size is about 10 x 10 then (scaling factor) x (the integral length) < 10]

CU - the array containing values of the desired contours

NV - the number of contours in the CU array

I - the integral length of the graph

J - the integral height of the graph

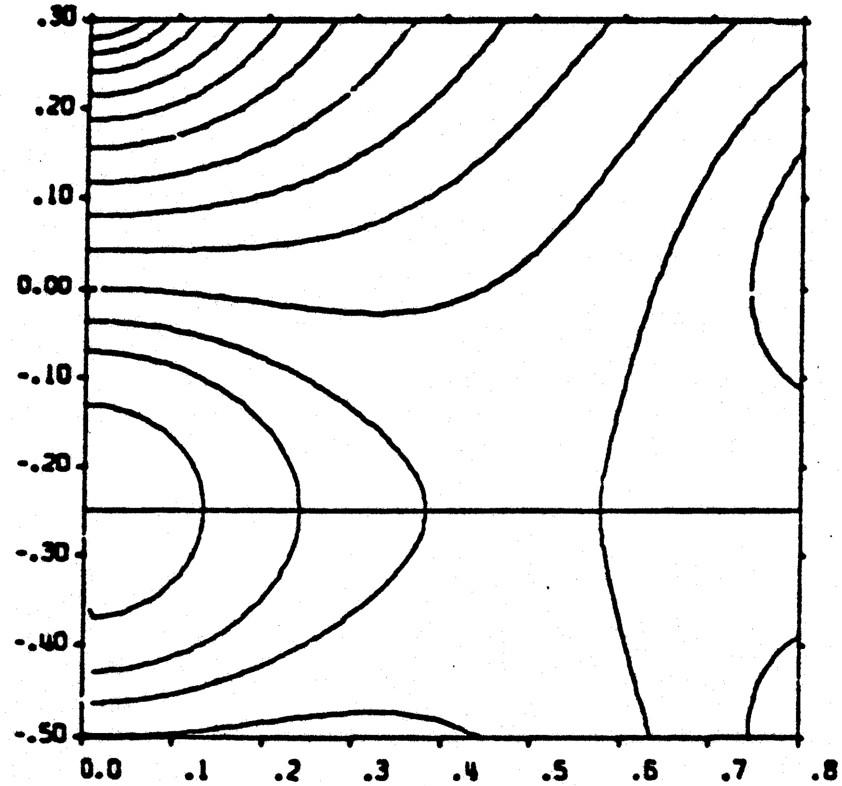
Note that the array A is loaded by columns. That is, A(m,n) is the $m^{th}$ element in the $n^{th}$ column in the M by N array.

An example of the "raw" output from BRUT is given in Figs. 2, 3A and 3B. These plots were produced directly from the computer generated microfilm. The plot in Fig. 2 will be discussed in more detail later in connection with time analysis. The technique used in BRUT has been utilized in several different versions on a variety of problems. A few graphs have been included as examples of Calcomp output (Fig. 4) and microfilm output (Fig. 5). These last four graphs are included only as a representation of the capabilities of the technique presented here.

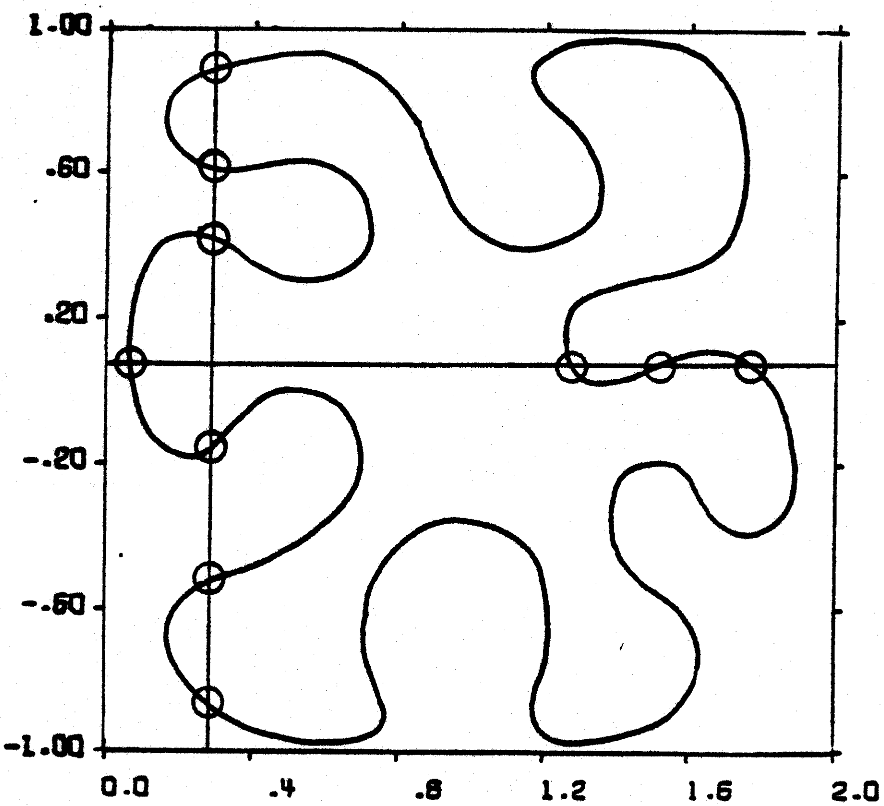
Along with the advantages of this subroutine some of the limitations might be noted. First, the curves on the graph are not labeled. This forces an examination of the data or some other trick to determine the value of each curve. Secondly, since much of the pertinent information for the printout is contained deep within subroutine ORDER, altering the coding to cause printout to occur in the user's program may present a problem. It can be seen that sometimes contours do not quite connect. This may be of little consequence however since individual graphs can be "retouched" and for masses of graphs, e.g., movies, completely ignored.

## IV SUMMARY

The subroutine presented here, though simple in approach, has a distinct advantage. Basically, it can be said that if the contour exists BRUT will find it. This is better in many instances than a code that follows a contour, say using a first derivative method, once that contour has been found. Finding an isolated contour can be as difficult as following it. And the ability to get the job done sometimes is as important as the finesse with which the task was accomplished.




A. Image plane artificially added




B. Multivalued contour in x and y resulting in contour length $>> 2(M+N)$

Figure 1. Examples of Programming Considerations

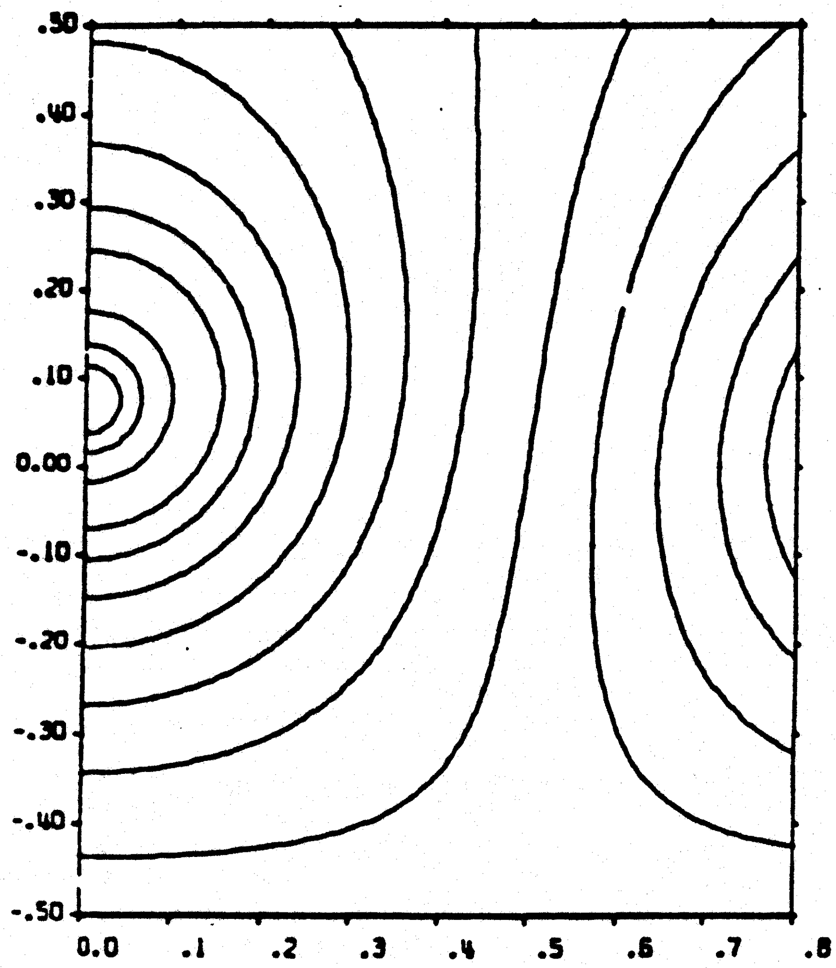
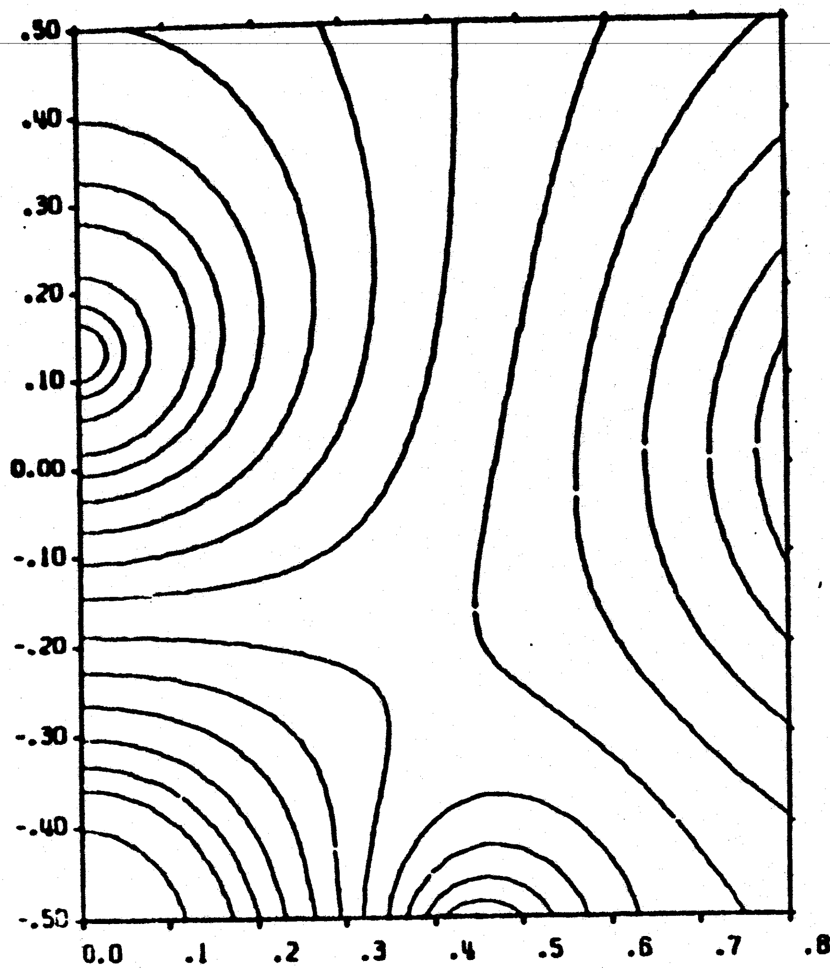


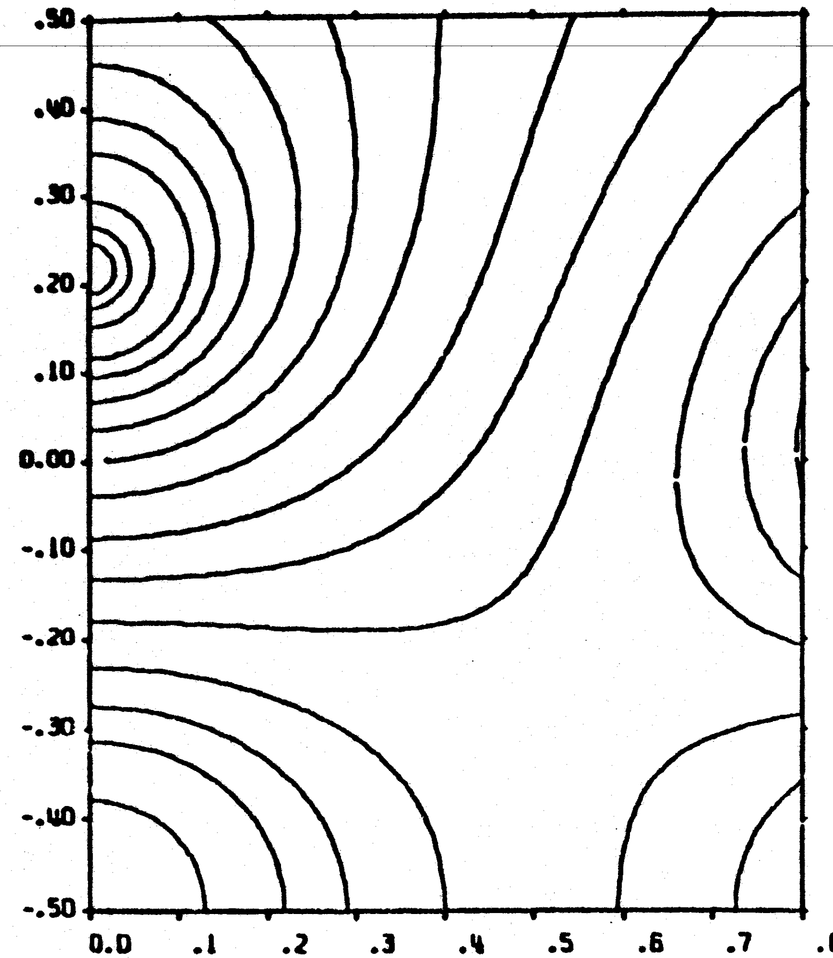

Figure 2. Microfilm Output from BRUT of a "Typical Run"

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Contour value | .1 | .16 | .25 | .4 | .5 | .63 | .8 | 1. | 1.25 | 1.6 | 2.0 | 2.5 | 3.2 | 4 |
| No. of pts. found | 16 | 26 | 39 | 64 | 78 | 99 | 129 | 121 | 129 | 155 | 139 | 101 | 63 | 34 |
| Order time | .024 | .042 | .070 | .138 | .184 | .263 | .401 | .316 | .292 | .370 | .344 | .241 | .128 | .059 |

Table 1. Contour Values, Time Expended and Number of Points Found in the Graph in Figure 2.

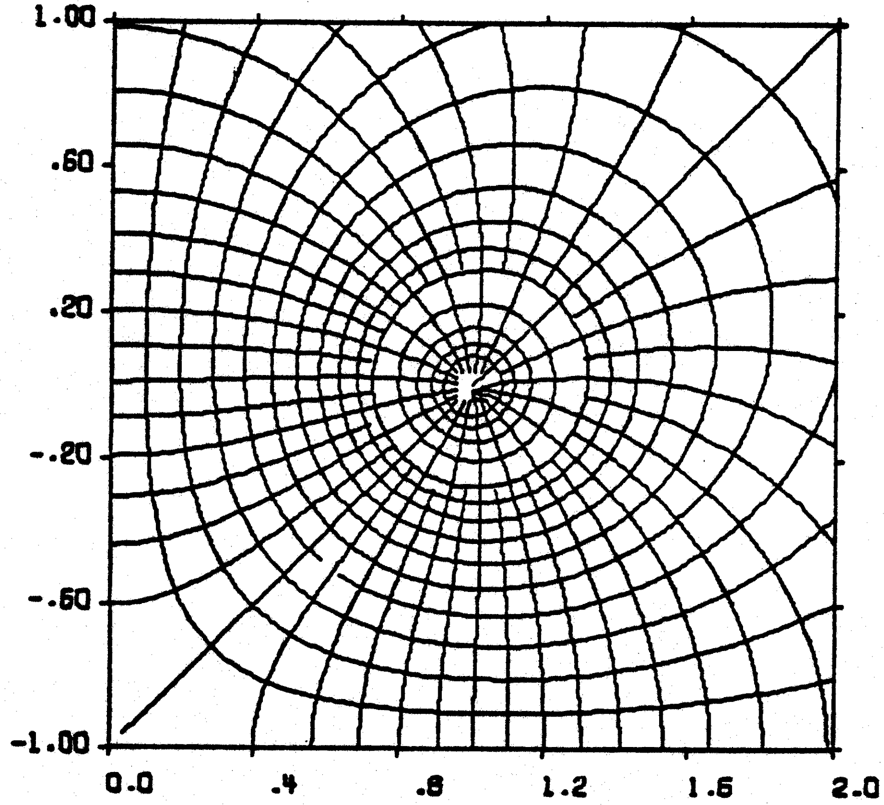



A. Unaltered microfilm output

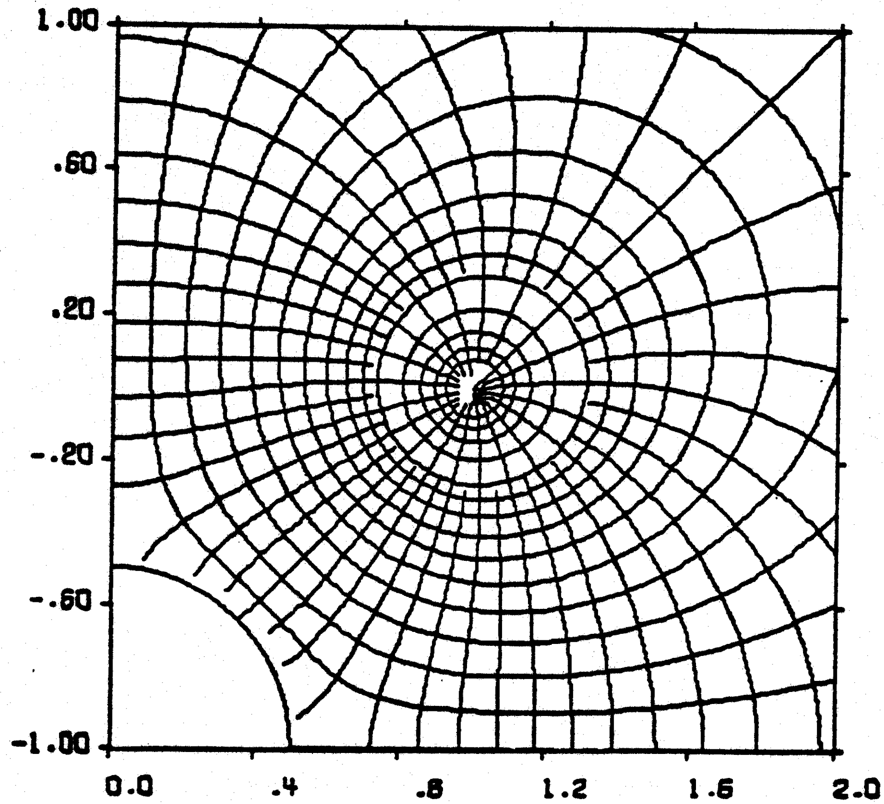



B. Unaltered microfilm output

Figure 3. Examples of Quality That Can Be Expected from Subroutine BRUT

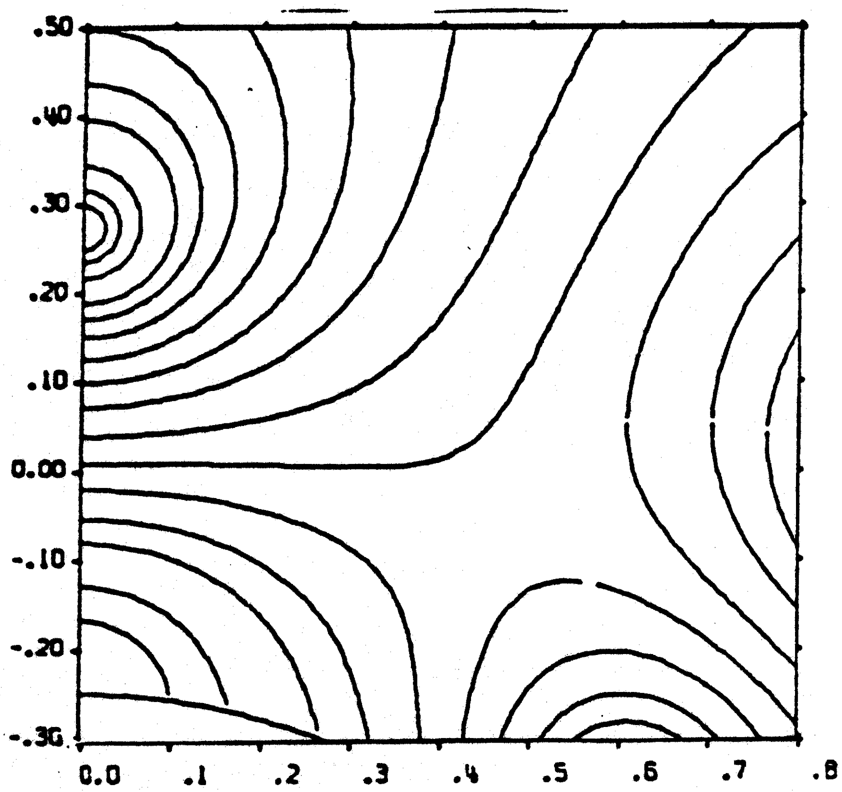



A. Unaltered Calcomp output

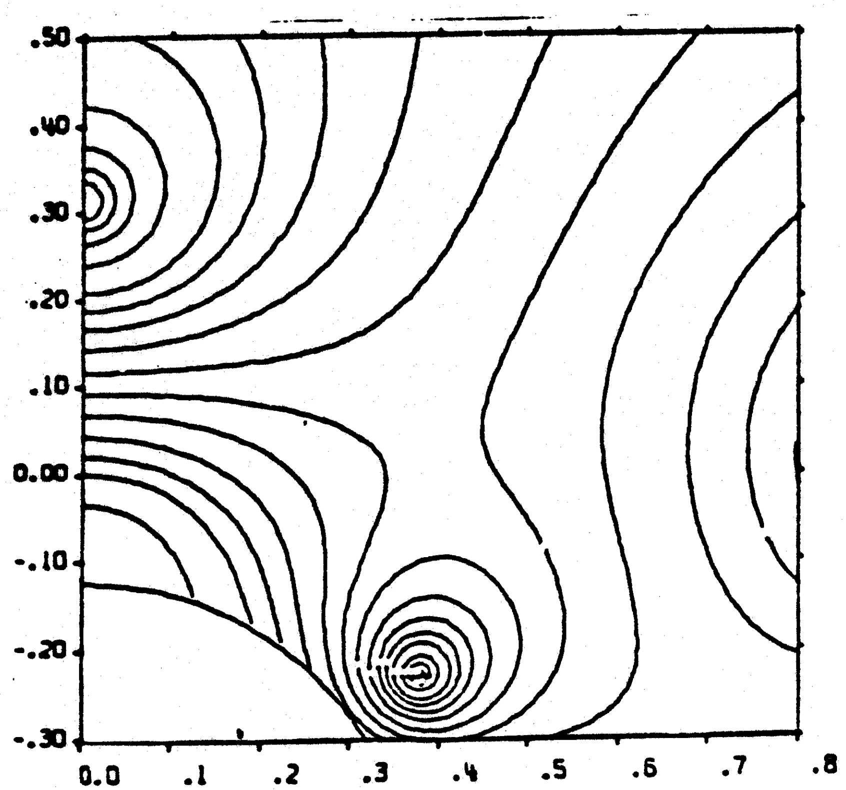



B. Unaltered Calcomp output

Figure 4. Field and Potential Plots Made With a Version of BRUT




B. Unaltered microfilm output




A. Unaltered microfilm output

Figure 5. Contour Plots Made With Variation of BRUT

## REFERENCES


1. Baum, C. E., Sensor and Simulation Note 112, Low-Frequency Magnetic Field Distribution for a Simulator with the Geometry of a Half Toroid Joined to the Surface of a Medium with Infinite Conductivity, July 1970.

2. Sancer, M. I., and A. D. Varvatsis, Sensor and Simulation Note 123, Low-Frequency Magnetic Field Distribution of a Half Toroid Simulator Joined to a Finitely Conducting Ground: Modified Ground Connections, February 1971.

3. Higgins, D. F., Sensor and Simulation Note 128, The Diffraction of an Electromagnetic Plane Wave by Interior and Exterior Bends in a Perfectly Conducting Sheet, January 1971.

4. Higgins, D. F., ARES MEMOS 2, Diffraction at the Junction Between the Wave Launching Section and the Working Volume of the ARES Facility, April 1971.

5. Higgins, D. F., Sensor and Simulation Note 155, The Effect of a Perfectly Conducting Ground Plane with a Symmetrically Located Semi-cylindrical Hump on the Impedance and Field Distribution of a Two-wire Transmission Line, June 1972.

6. Marks, J. A., Program HAPS, a Two-Dimensional Computer Code to Calculate Electromagnetic Fields Resultant from High Altitude Nuclear Detonations, March 1972. DC-TN-1214-8

7. Baum, C. E., Sensor and Simulation Note 148, General Principles for the Design of ATLAS I and II, Part V: Some Approximate Figures of Merit for Comparing the Waveforms Launched by Imperfect Pulser Arrays onto TEM Transmission Lines, May 1972.

8. Baum, C. E., Sensor and Simulation Note 21, Impedances and Field Distributions for Parallel Plate Transmission Line Simulators, June 1966.

## Appendix A

## Listing of Subroutine BRUT

```
      SUBROUTINE BRUT (A, MM, NN, P, CU, NV, IL, J)
      DIMENSION A(MM, NN), CU(NV), P(6)
      COMMON /GOOP/ DX, DY, K, SAVEX(500), SAVEY(500)
      CALL D4 (P(1), P(2), P(3), P(4), IL, J, P(5), P(6), 1, SAVEX, SAVEY, 1, -1)
      DY=(P(4)-P(3))/FLOAT(MM-1)
      DX=(P(2)-P(1))/FLOAT(NN-1)
      NOMM1=MM-1
      NONM1=NN-1
      DO 45 I=1, NV
      K=0
      DO 5 N=1, NN
      DO 5 M=1, NOMM1
      IF ((A(M, N)-CU(I))*(A(M+1, N)-CU(I)).GT.0.) GO TO 5
      K=K+1
      IF (K.GT.500) GO TO 20
      SAVEX(K)=FLOAT(N-1)*DX+P(1)
      SAVEY(K)=FLOAT(M-1)*DY+P(3)+(DY/(A(M+1, N)-A(M, N)))*(CU(I)-A(M, N))
    5 CONTINUE
      DO 10 M=1, MM
      DO 10 N=1, NONM1
      IF ((A(M, N)-CU(I))*(A(M, N+1)-CU(I)).GT.0.) GO TO 10
      K=K+1
      IF (K.GT.500) GO TO 20
      SAVEX(K)=FLOAT(N-1)*DX+P(1)+(DX/(A(M, N+1)-A(M, N)))*(CU(I)-A(M, N))
      SAVEY(K)=FLOAT(M-1)*DY+P(3)
   10 CONTINUE
      PRINT 15, CU(I), K
   15 FORMAT (1H0/1H0, 20X, F10.6, 9H  CONTOUR/33X, 13HNO. OF PTS. =, I4)
      IF (K.LT.4) GO TO 45
      GO TO 30
   20 PRINT 25, CU(I)
   25 FORMAT (1X, 23H*** ARRAY OVERFLOW FOR , F10.6, 9H CONTOUR)
   30 CALL ORDER (P)
      DO 40 MISUM=1, 3
      LOAD=0
      DO 35 JOY=1, K
      IF (SAVEX(JOY).GE.1.E5) GO TO 35
      LOAD=LOAD+1
      SAVEX(LOAD)=SAVEX(JOY)
      SAVEY(LOAD)=SAVEY(JOY)
   35 CONTINUE
      IF (LOAD.LT.5) GO TO 45
      K=LOAD
   40 CALL ORDER (P)
   45 CONTINUE
      CALL PLOT(FLOAT(IL)*P(5)+3.6, 0., -3)
      RETURN
      END
```

```
      SUBROUTINE ORDER (P)
      COMMON /GOOP/ DX, DY, IPLOT, SAVEX(500), SAVEY(500)
      DIMENSION PX(500), PY(500), P(6)
      V=DX*DX+DY*DY
      CALL SORT (SAVEX, IPLOT, SAVEY)
      IMAX=0
      TEMPX=SAVEX(1)
      TEMPY=SAVEY(1)
      K6=0
      IFD=0
      I=1
      IP=1
      GO TO 10
 5    IP=2
      I=JHOLD
      PX(1)=SAVEX(I)
      PY(1)=SAVEY(I)
 10   PX(IP)=SAVEX(I)
      PY(IP)=SAVEY(I)
 15   KOUNT=0
      DO 50 J=1, IPLOT
      IF (SAVEX(J).GT.1.E5) GO TO 50
      SX=SAVEX(J)-SAVEX(I)
      SY=SAVEY(J)-SAVEY(I)
      DEL=SX*SX+SY*SY
      IF (DEL.LT.1.E-12) GO TO 50
      KEEP=0
      IF (DEL.GT.V) GO TO 30
      KK=J+6
      KK=MIN0(KK, IPLOT)
      DELT=DEL
      KEEP=-1
      IF (J.EQ.IPLOT) GO TO 30
      KB=J+1
      DO 25 K=KB, KK
      SXC=SAVEX(K)-SAVEX(I)
      SYC=SAVEY(K)-SAVEY(I)
      DELC=SXC*SXC+SYC*SYC
      IF (DELC.LT.1.E-12) GO TO 25
      IF (DELC-DELT) 20, 25, 25
 20   DELT=DELC
      KEEP=K
 25   CONTINUE
 30   IF (KEEP) 80, 35, 75
 35   IF (IFD) 45, 40, 45
 40   JHOLD=J
 45   IFD=IFD+1
 50   KOUNT=KOUNT+1
      IF (KOUNT.EQ.IPLOT.AND.IFD.NE.0) GO TO 55
      IF (IP.GT.4) CALL D4 (P(1), D, P(3), M, M, P(5), P(6), IP, PX, PY, 2, -1)
```

```
      IF (K6.EQ.1) GO TO 100
      IF (IP.EQ.IPLOT) GO TO 95
      IMAX=IMAX+1
      IF (IMAX.LT.(IPLOT+5)) GO TO 15
      TSX=PX(IP)-TEMPX
      TSY=PY(IP)-TEMPY
      DEL=TSX*TSX+TSY*TSY
      IF (DEL.LT.V) GO TO 110
      GO TO 100
   55 CONTINUE
      IF (K6.NE.0) GO TO 60
      PRINT 105, (PX(L), PY(L), L, L=1, IP)
      SAVEX(IP)=SAVEX(IP)+1.E6
      IF (IP.GT.4) CALL D4 (P(1), D, P(3), D, M, M, P(5), P(6), IP, PX, PY, 2, -1)
   60 IF (K6) 70, 70, 65
   65 K6=0
      TSX=PX(2)-TEMPX
      TSY=PY(2)-TEMPY
      DEL=TSX*TSX+TSY*TSY
      IF (DEL.GT.V) GO TO 100
      PX(1)=TEMPX
      PY(1)=TEMPY
      GO TO 100
   70 K6=1
      GO TO 5
   75 J=KEEP
   80 IP=IP+1
      PX(IP)=SAVEX(J)
      PY(IP)=SAVEY(J)
      SAVEX(I)=SAVEX(I)+1.E6
      IF (J-JHOLD) 90, 85, 90
   85 IFD=0
      JHOLD=0
   90 I=J
      IF (IP-IPLOT) 15, 95, 95
   95 TSX=PX(IP)-TEMPX
      TSY=PY(IP)-TEMPY
      DEL=TSX*TSX+TSY*TSY
      IF (DEL.LE.V) GO TO 110
  100 PRINT 105, (PX(I), PY(I), I, I=1, IP)
  105 FORMAT(23X, 1HX, 14X, 1HY/(15X, 2E15.5, I4))
      CALL D4 (P(1), D, P(3), D, M, M, P(5), P(6), IP, PX, PY, 2, -1)
      SAVEX(I)=SAVEX(I)+1.E6
      RETURN
  110 IP=IP+1
      PX(IP)=TEMPX
      PY(IP)=TEMPY
      GO TO 100
      END
```

```
      SUBROUTINE D4 (XMIN, XMAX, YMIN, YMAX, IL, IH, SX, SY, NPTS, X, Y, KIND, LAST)
      DIMENSION X(NPTS), Y(NPTS)
      DATA IFT, JFT, FX, FY/4HF6.1, 4HF6.2, 0.6, 0.59/
      IF (KIND-1) 10, 15, 5
 5    IF (KIND-2) 10, 90, 10
 10   RETURN
 15   IF (IO-2) 20, 25, 20
 20   CALL PLOTS (TB, TB, 10)
      IO=2
 25   CALL PLOT (FX, FY, 3)
      REALH=IH
      REALL=IL
      SCALEX=(XMAX-XMIN) /REALL
      SCALEY=(YMAX-YMIN)/REALH
      RSCALX=1./SCALEX
      RSCALY=1./SCALEY
      DO 85 I=1, 4
      GO TO (30, 35, 30, 35), I
 30   NN=IH+1
      GO TO 40
 35   NN=IL+1
 40   DO 85 N=1, NN
      REALN=N
      GO TO (45, 55, 65, 75), I
 45   R=REALN-1.
      CALL PLOT (-.05+FX, R*SY+FY, 2)
      CALL PLOT (FX, R*SY+FY, 2)
      YNUM=R*SCALEY+YMIN
      IF (ABS(YNUM).LE.1.E-10) YNUM=0.
      RR=(REALN-1.)*SY-.03
      CALL NUMBER (-.6+FX, RR+FY, .10, YNUM, 0., JFT)
      CALL PLOT (FX, R*SY+FY, 3)
      IF (N-NN) 50, 85, 50
 50   CALL PLOT (FX, REALN*SY+FY, 2)
      GO TO 85
 55   R=REALN-1.
      RR=REALH+.05
      CALL PLOT (R*SX+FX, RR*SY+FY, 2)
      CALL PLOT (R*SX+FX, REALH*SY+FY, 2)
      IF (N-NN) 60, 85, 60
 60   CALL PLOT (REALN*SX+FX, REALH*SY+FY, 2)
      GO TO 85
 65   R=REALL+.05
      RR=REALH-REALN+1.
      CALL PLOT (R*SX+FX, RR*SY+FY, 2)
      CALL PLOT (REALL*SX+FX, RR*SY+FY, 2)
      IF (N-NN) 70, 85, 70
 70   CALL PLOT (REALL*SX+FX, (RR-1.)*SY+FY, 2)
      GO TO 85
 75   R=REALL-REALN+1.
```

```
      CALL PLOT (R*SX+FX, -.05+FY, 2)
      CALL PLOT (R*SX+FX, FY, 2)
      XNUM=R*SCALEX+XMIN
      IF (ABS(XNUM).LE.1.E-10) XNUM=0.
      RR=R*SX-.25
      CALL NUMBER (RR+FX, -.25+FY, .10,XNUM, 0., IFT)
      CALL PLOT (R*SX+FX, FY, 3)
      IF (N-NN) 80, 85, 80
   80 CALL PLOT ((R-1.)*SX+FX, FY, 2)
   85 CONTINUE
   90 CALL PLOT ((X(1)-XMIN)*RSCALX*SX+FX, (Y(1)-YMIN)*RSCAL*SY+FY, 3)
      DO 95 I=1, NPTS
      XX=(X(I)-XMIN)*RSCALX
      YY=(Y(I)-YMIN)*RSCALY
   95 CALL PLOT (XX*SX+FX, YY*SY+FY, 2)
      IF (LAST.LT.0) RETURN
      CALL PLOT ((REALL+3.)*SX+FX, FY, -3)
      RETURN
      END
```

```
      SUBROUTINE SORT (KEY, NUM, KEY1)
      INTEGER KEY(NUM), T, KEY1(NUM), T2
5     IF (NUM.LT.2) RETURN
      I=1
10    I=I+I
      IF (I.LE.NUM) GO TO 10
      M=I-1
15    M=M/2
      IF (M.LT.1) RETURN
      K=NUM-M
      DO 25 J=1, K
      I=J
20    IM=I+M
      IF (KEY(I).LE.KEY(IM)) GO TO 25
      T2=KEY1(I)
      T=KEY(I)
      KEY1(I)=KEY1(IM)
      KEY(I)=KEY(IM)
      KEY1(IM)=T2
      KEY(IM)=T
      I=I-M
      IF (I.GE.1) GO TO 20
25    CONTINUE
      GO TO 15
      END
```