The Development of a Program for the Plotting of accurate Contours of any second degree Surface from its Cartesian Equation

by

WILLIAM WRIGHT, M.A., B.SC.

Thesis submitted for the degree of
Master of Science of the University
of St. Andrews

(Oct. 1970)

DECLARATION

I declare that the following thesis is a record of research work carried out by me, that the thesis is my own composition, and that it has not been presented in application for a higher degree previously.

WILLIAM WRIGHT

CERTIFICATE

I certify that William Wright has satisfied the conditions of the Ordinance and Regulations and is thus qualified to submit the accompanying thesis in application for the degree of Master of Science.

A. J. COLE

Acknowledgments

I should like to thank my supervisor, Professor A. J. Cole, for his guidance and advice throughout the investigation. I am also greatly indebted to Dr. R. Erskine and the operators for their support in the computer room, to Mrs. G. Taylor for typing the manuscript of the thesis and to Mr. T. McQueen for the printing of the final result.

The development of a program for the plotting of accurate contours of any second degree surface from its Cartesian equation.

INTRODUCTION

1. Computer-Plotter

The plotter attached to the I.B.M. 360 computer has ten fundamental movements which are linear; four can be regarded as parallel to the sides of a square and are of length .01 in., four as parallel to the diagonals and of length (.01*SQRT2) in., whilst the remaining two are perpendicular to the plane and are used for lowering or raising the stylus. All other movements are composite, being a succession of those simple steps. The plotting plane can therefore be regarded as a grid surface with squares of side .01 in., whose vertices constitute the set of all possible positions of the stylus on the plane, and the proper linking of selected elements of the set is the essence of the tracing of contours. It is the combination of computer and plotter which makes this possible.

2. Plot Routines

The plotting plane is, in effect, the first quadrant of a Cartesian plane of reference. The movement of the stylus of the plotter is controlled by plotting routines which are called, as with any subprogram, by name and associated parameters, the first parameter being an integer selected from a set of integers which constitute a code. Before plotting, a plot routine of nine parameters is called.
This routine, according to parameters chosen, effectively sets the stage, attending to such matters as axes and scale.
An example of this initialising call is CALL PLOT (1,10.,20., 10.,1.,0.,10.,10.,1.), which would have both axes drawn, the maximum and minimum values of X and Y, (10.,20.) and (0.,10.) respectively, referred to the extremities of the axes, and the axes marked in units. The drawing of the axes, which is often unnecessary and always time consuming, could be omitted by substituting 201 for 1 as the integer parameter.
Other calls appearing frequently in this paper are CALL PLOT (99) which raises the stylus; CALL PLOT (90,X,Y) which results in the stylus moving, in whatever position it happens to be, up or down, to the point (X,Y) and assuming the down position; and CALL PLOT (98,X,Y) whereby the stylus is raised, moves to (X,Y) and remains raised.

The following program and the resulting plot illustrate the use of these calls.

PROGRAM 1

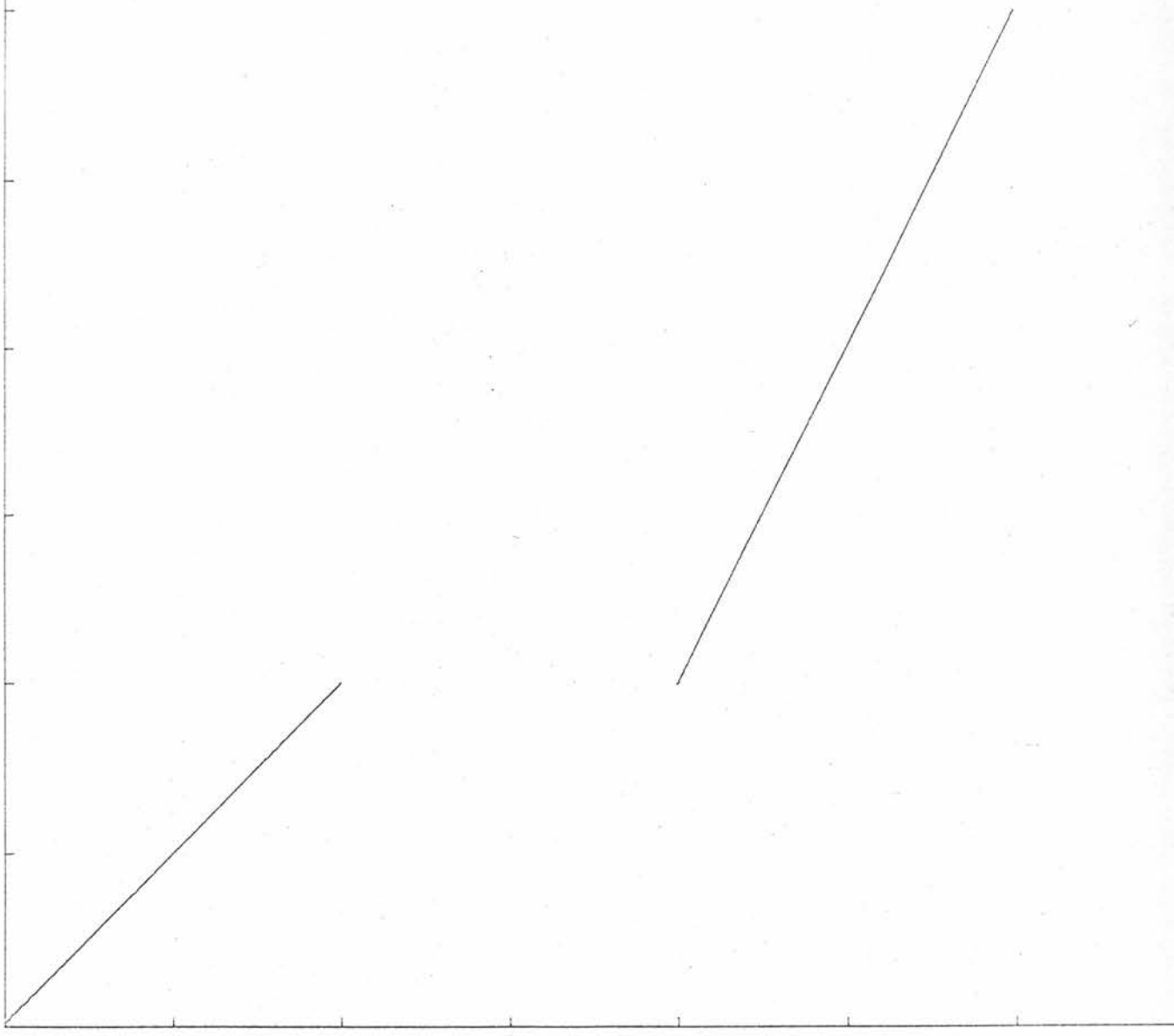
```
c     Illustrative program using plot routines
      CALL PLOT (1,0.,10.,10,1.,0.,10.,10.,1.)
      CALL PLOT (90,0.,0.)
      CALL PLOT (90,2.,2.)
      CALL PLOT (98,4.,2.)
      CALL PLOT (90,4.,2.)
      CALL PLOT (90,6.,6.)
      CALL PLOT (98,0.,0.)
c     The preceding call causes the stylus to return to the origin
c     in the up position
      CALL PLOT (7)
      STOP
      END
```

CALL PLOT (7) is not essential to the program, but acts as a separator between adjacent plots

The plot resulting from this program is shown on next page.

FIGURE 1

EFFECTS OF THE INITIALISING CALL, AND OF ROUTINES, PLOT (90,X,Y) AND PLOT (98,X,Y), AS USED IN PROGRAM 1.



PART I

1. The Plane

The plotting of accurate contours of a second degree surface is essentially that of drawing, in the first quadrant of the Cartesian system, plane curves which are as close as the plotter permits to the ideal curve as defined by an algebraic equation. Those curves will be obtained by calling plot routines, which have the effect of the plotter linking in succession pairs of grid vertices, whose co-ordinates are the two variables of an equation associated with the equation of the original surface. Those vertices are sufficiently close together to give the impression of a smooth curve. Although an analysis of the contour problem for the plane is not necessary for the creation of a program for the contours of a second degree surface, yet it is described in this paper, as it was this analysis which played a considerable part in leading to the solution of the second degree problem. It provided many of the ideas and techniques used in later programs, and those ideas and techniques, along with much of the method, can be clearly demonstrated in the simpler setting of the plane. The contours of the plane are, of course, straight lines, and the next two sections are devoted to the writing of programs to link the given equation of a line with its ultimate trace in the plotting quadrant. This part of the paper is completed by the inclusion of the program for the contours of any plane.

2. The Straight Line

This is a unique type of curve, parts of it being obtained quite simply by two calls of the routine, PLOT(90,X,Y), where X and Y satisfy the equation of the line, but to obtain completely that part of it which lies within the quadrant, one would require to calculate the co-ordinates of its points of intersection with the boundary of the quadrant before calling the routine. This can be done quite easily, but the method does not lend itself to extension, and, for present purposes, is of little value. There are later in the project requirements for the tracing of curves of varying radii of curvature, for plotting the totality of the curve which is within the quadrant and for avoiding plot routine calls which refer to points outwith the quadrant. Because of these future requirements, a method of scanning and plotting has been adopted for the straight line, whereby X increases by defined increments from XMIN to XMAX, with the corresponding values of Y being calculated. The range of the scan, along with the insertion in the program of the condition Y(YMAX-Y) not less than 0, ensures that all possible pairs of points, relevant to a plotting routine, have been obtained, and their joining gives the complete line within the quadrant. This procedure lends itself to extension to higher curves and is shown and tried in the next section, where, beginning with the equation of a particular line, a program is written to produce that part of the line which lies within the plotting quadrant.

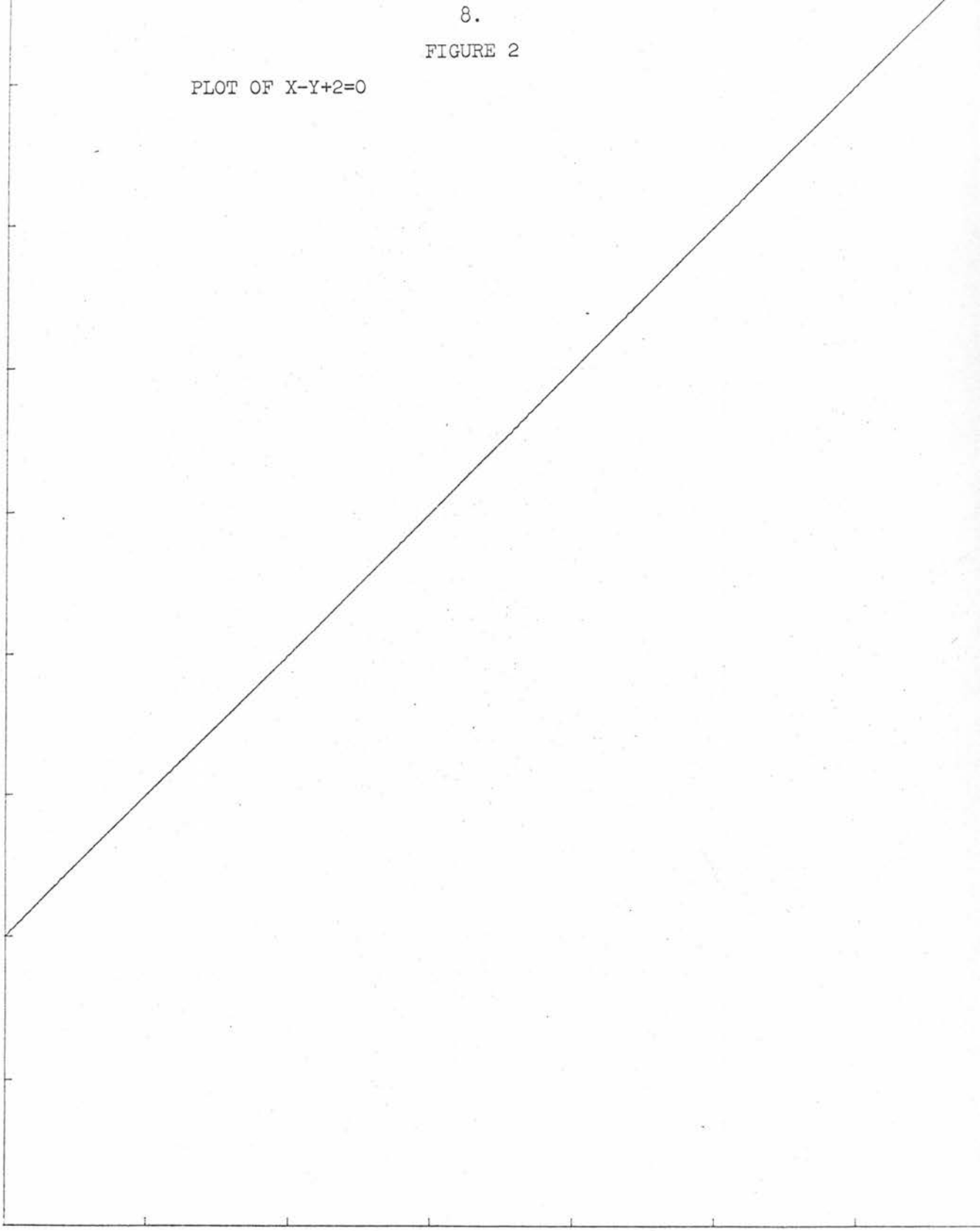
3. PROGRAM 2

```
c.  Illustration  of scanning technique  and boundary conditions
c.  within the program for the plotting of X-Y+2=0
      CALL PLOT (1,0.,10.,10.,1.,0.,10.,10.,1.)
      CALL PLOT (99)
      X = 0.
   11 IF (X.GT.10.) GO TO 22
c.  This conditional statement is used for control to pass from
c.  the loop to the termination of the program when the maximum
c.  value of X has been reached, and along with the second
c.  conditional statement ensures that the co-ordinates refer
c.  to points within the quadrant.
      Y = X + 2.
      IF (Y * (10.-Y).LT.0.) GO TO 20
      CALL PLOT (90,X,Y)
   20 X = X+ .2
      GO TO 11
   22 CALL PLOT (98,0.,0.)
      CALL PLOT (7)
      STOP
      END
```

The trace which is the result of this program is shown in figure 2.

FIGURE 2

PLOT OF X-Y+2=0



Whilst the loop in program 2, containing both the computation of the ordinate and the associated plotting, is effective, it is inefficient and, more important, is unsuitable in such cases as those where there may be gaps in the curve. In general, all calculations of ordinates associated with a complete arc should be done before calling on the plotting routine. This will require a reservation of core storage which will appear in subsequent programs.

The investigation of the straight line was continued by writing a program for the general equation. A read statement was included for the parameters which define the line, and as will be seen in the program, program 3, one of those parameters determine which of two routines will be followed in the execution of the program.

PROGRAM 3

4. General Case $Ax+By+C=0$

In planning a program for the line $Ax+By+C=0$, the analysis gives:-

$B \neq o$, $y = (-Ax-C)/B$ or

$B = o$, $x = -C/A$. The two cases correspond to the separate routines.

Further, when $B \neq o$, it is economical to use a statement function such as $F(x) =(-Ax-C)/B$, and a possible program is the following

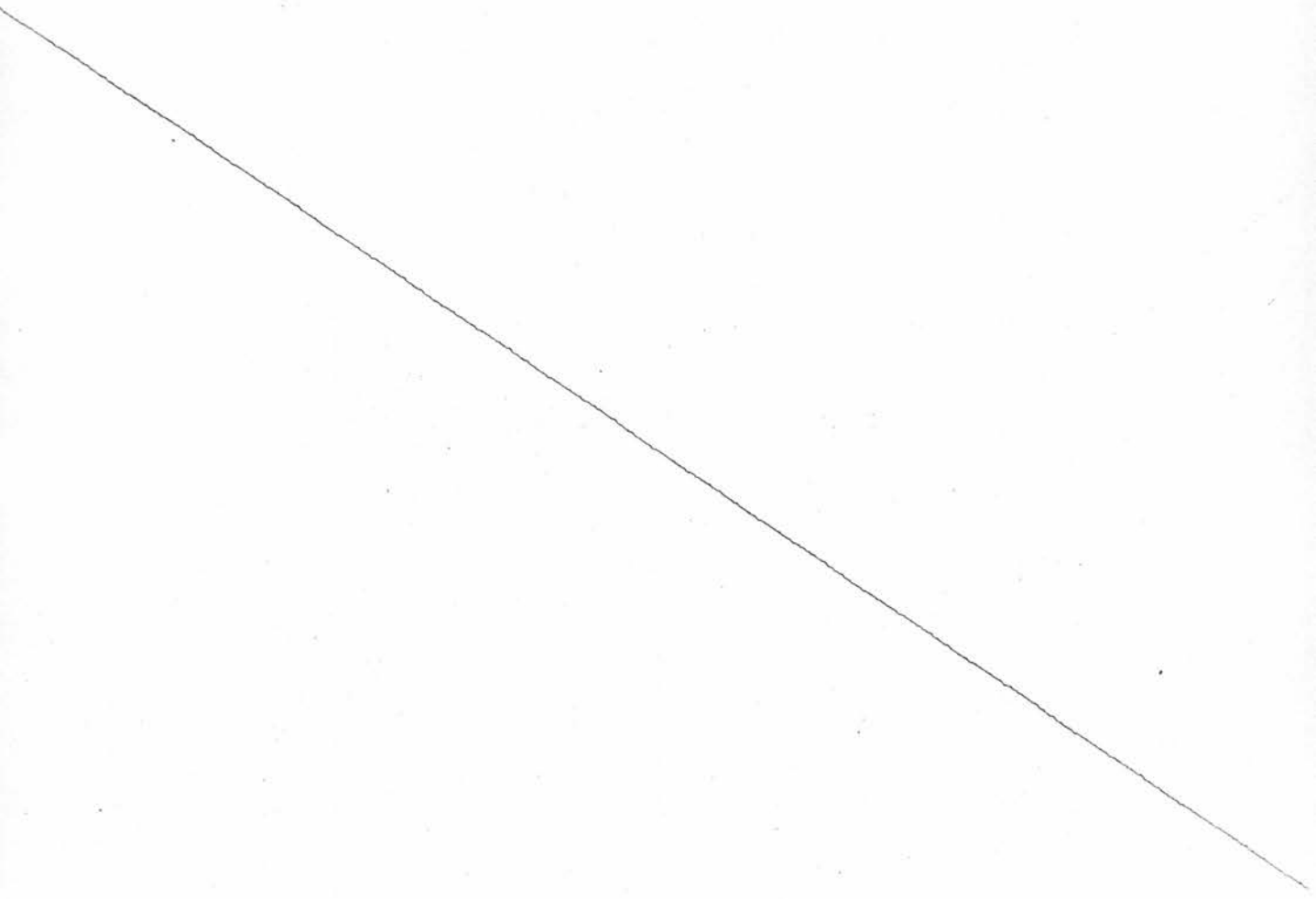
PROGRAM 3

```
      F(X) = (-A*X-C)/B
      DIMENSION P(60), Q(60)
      CALL PLOT (201,0.,10.,10.,1.,0.,10.,10.,1.)
      CALL PLOT  (99)
      READ (5,12) A, B, C
12    FORMAT    (3F5.1)
      IF (B.EQ.0.) GO TO 40
      X = 0.
      J = 0
24    IF (X.GT.10.) GO TO 36
      Y = F(X)
      IF (Y*(10.-Y).LT.0.) GO TO 35
C VALUES WILL NOW BE STORED
```

```
      J = J+1
      P(J) = X
      Q(J) = Y
   35 X = X + .2
      GO TO 24
   36 DO 37 N = 1,J
   37 CALL PLOT (90,P(N),Q(N))
      GO TO 50
   40 X = -C/A
C  This is the equation of a line parallel to the Y-AXIS
C  and is drawn by joining the two points of intersection
C  with the horizontal boundaries.
      CALL PLOT (90,X,0.)
      CALL PLOT (90,X,10.)
   50 CALL PLOT (98,0.,0.)
      CALL PLOT (7)
      STOP
      END
```

FIGURE 3

PLOT OF AX+BY+C=0 WHERE A,B and C ARE
READ AS 2.0,3.0 and -24.0 respectively



5 ## CONTOURS OF PLANES

The ideas now introduced were used when writing a program for plotting the contours of any plane. Taking the general equation of the plane as Ax+By+Cz+D=0, the equation of the contour on the Cartesian plane corresponding to a height, K say, is AX + BY + (CK+D) = 0 - the equation of a straight line. The output will therefore be a family of straight lines depending on the various values assigned to K, and in the actual program the six values 0, 1, 2, 3, 4, 5 were so assigned. The pattern of the program is a loop which is traversed once for each value of Z and is shown on program 4.

PROGRAM 4

```
C.        A GENERAL PROGRAM FOR CONTOURS OF PLANES
            F(X,Z) = (-D-A*X-C*Z)/B
            DIMENSION  P(120), Q(120)
     77     CALL PLOT (201,0.,20.,10.,1.,0.,20.,10.,1.)
C  XMAX     AND YMAX HAVE BEEN ASSIGNED VALUE 20
C  FOR GREATER RANGE
             CALL PLOT (99)
             READ (5,22)  A,B,C,D
     22      FORMAT (4 F4.1)
             Z = 0.
             IF (B.EQ.0.)  GO TO 33
```

```
C B= 0 and A = 0 ARE USED TO BRANCH TO END OF PROGRAM
      DO  66  N = 1, 6
      X = 0.
      J = 0
   23 IF (X.GT.20.) GO TO 50
      Y = F (X,Z)
      IF (Y*(20. - Y).LT.0.)  GO TO 24
      J = J + 1
      P(J) = X
      Q(J) = Y
   24 X = X + .2
      GO TO 23
   50 DO  60  K = 1,J
   60 CALL PLOT (90,P(K),Q(K))
      CALL PLOT (99)
   66 Z = Z + 1.
      CALL PLOT (7)
      GO TO 77
   33 IF (A.EQ.0.) GO TO 100
C THIS IS THE SECOND CONDITION WHICH EFFECTS EXIT FROM PROGRAM
      DO  88  N = 1,6
      X = (-D-C*Z)/A
      IF (X * (20.-X).LT.0.)  GO TO 88
      CALL PLOT (90,X,0.)
      CALL PLOT (90,X,20.)
      CALL PLOT (99)
   88 Z = Z + 1.
      CALL PLOT (7)
      GO TO 77
  100 STOP
      END
```

CONTOURS OF PLANES FIGURE 4A

A = 3.0, B=4.0, C=10.0 and D = -60.0

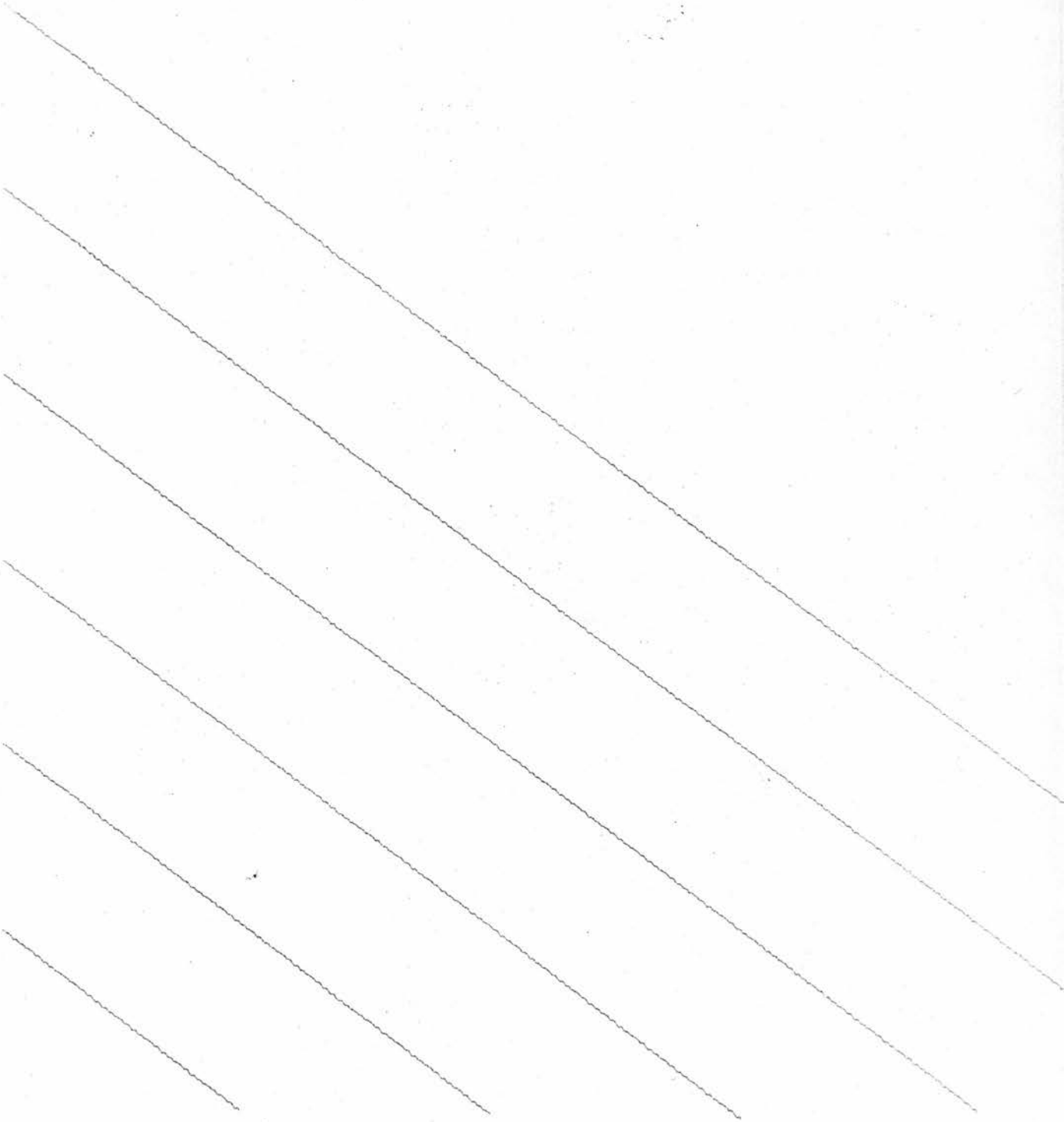


FIGURE 4B

A = 0, B = 1.0, C = 2.0 and D = -16.0

FIGURE 4C

A = 1.0, B = 0., C = 2.0, D = -16.

PART 2

The Second Degree Surface

1. The main objective was the creation of a program which would result in the plotter tracing, for any second degree surface, all relevant contours and parts of contours which might be in the quadrant. To discover what was required for this purpose, it was decided to write a tentative program, based on previous programs, for a second degree surface, whose contours were known to be completely within the quadrant. So, against a setting of XMIN=YMIN=0 and XMAX=YMAX=XL=YL=10., the surface Z=(X-5)**2/5.+(Y-5)**2/2. was chosen with the intention of tracing the contours corresponding to Z=1, 2, 3, 4 and 5. The contours are, of course, ellipses with centre (5,5) and axes parallel to the axes of co-ordinates.

From the original equation the two following expressions were derived for y viz

```
Y = 5.+SQRT(50Z-10X**2+100X-250)/5
```

and

```
Y = 5 - SQRT(50Z-10X**2+100X-250)/5
```

Therefore, for each value of Z, there appeared to be two branches which together would constitute the required contour. It was expected that this early production would be crude, but that it would, nevertheless, provide information helpful towards programming for the accurate contour.

PROGRAM 5

```
C     TENTATIVE PROGRAM FOR CONTOURS OF THE SPECIALLY CHOSEN
C     SURFACE   2XX+5YY-20X-50Y-10Z+175 = 0
C     THE WRITE STATEMENTS IN THE PROGRAM ARE EXPECTED TO GIVE
C     INFORMATION WHICH WILL BE HELPFUL WITH LATER PROGRAMMING.
C     Q(X,Z) IS THE DISCRIMINANT OF THE TWO PREVIOUS EQUATIONS
      Q(X,Z)=50.*Z-10.*X*X+100.*X-250.
         DIMENSION   XX(100), Y(100), YY(100)
         CALL PLOT   (201,0.,10.,10.,1.,0.,10.,10.,1.)
         Z = 1.
         DO  70   KK=1,5
         X = 0.
      10 IF(Q(X,Z).GE.0.)  GO TO 20
C     THIS IS THE CONDITION FOR REAL ROOTS AND BRANCHING TO
C          STORAGE OF SUBSCRIPTED VARIABLES
         X = X+.2
         GO TO 10
      20 J = 1
         R = Q(X,Z)
         WRITE (6,15)  R
      15 FORMAT ('  ',F6.2)
      30 XX(J) = X
         IF (Q(X,Z).LT.0.)  GO TO 40
C     THIS CONDITIONAL STATEMENT PROVIDES FOR BRANCHING TO
C     PLOTTING ROUTINE WHEN STORAGE OF VARIABLES IS
C           COMPLETED
         Y(J) = 5.+SQRT (Q(X,Z))/5.
        YY(J) = 5.-SQRT (Q(X,Z))/5.
         X = X+.2
         J = J+1
         GO TO 30
```

```
40 CALL PLOT (98,XX(1),Y(1))
   J=J-1
   WRITE (6,25)J
25 FORMAT ('  ',I4)
   DO 50 L=1,J
50 CALL PLOT (90,XX(L),Y(L))
   CALL PLOT (98,XX(1),YY(1))
   DO  60  L=1,J
60 CALL PLOT (90,XX(L),YY(L))
70 Z=Z+1.
   CALL PLOT (98,0.,0.)
   CALL PLOT (7)
   STOP
   END
```

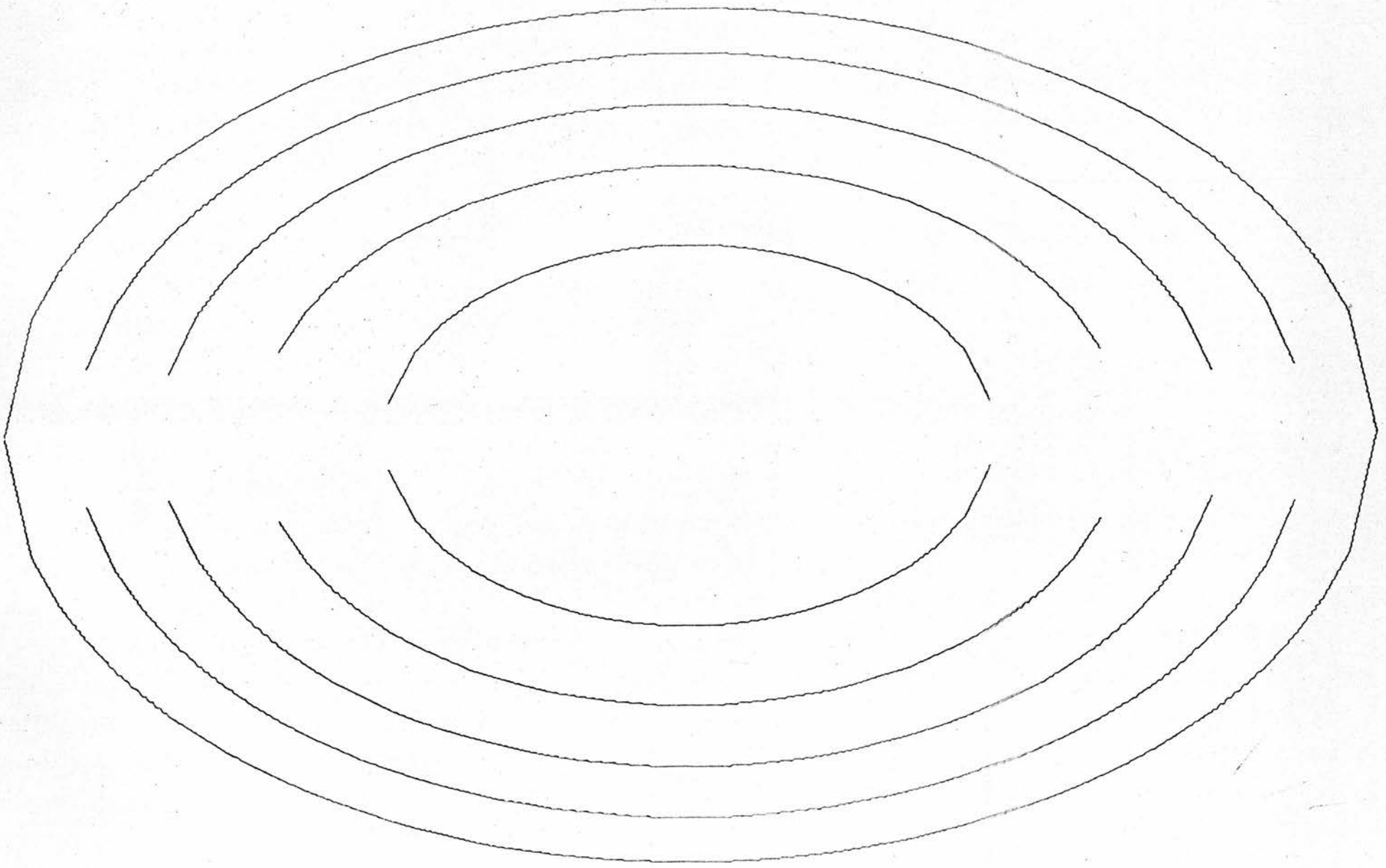
The output from this program was

```
Q = 1.60  10.0  5.6  6.4  0.0
J = 23     31   39    45   51
```

and the diagram shown in figure 5.

FIGURE 5

SET OF INCOMPLETE CONTOURS FROM TENTATIVE PROGRAM 5



2. A cursory glance confirms that the curves produced fall far short of being complete contours, but, as expected, useful information is available. The greater part of the curve is satisfactory, and the tracing procedure is suitable where |gradient| < 1. It is when this value is exceeded that discontinuities arise, and they become critical when |gradient| is great.
Those critical values occur near boundaries separating regions of real and complex roots. Two matters require particular attention. First, when |gradient| > 1 the change in y corresponding to an increment in X can be sufficiently great to give a polygon effect, when such a part of the curve is being traced, and, second, in the process of scanning, values corresponding to the end or steep parts of the curve will be omitted, thereby producing the gaps which appear in figure 5. By selecting smaller increments over such arcs, both the gap and the polygon effect should be reduced.
Finer scanning implies a greater number of values being calculated, and consequently greater storage demand. One needs to assess such demand as work proceeds.
At this stage of the development the immediate problem had become, not one of producing a set of contours, but that of drawing an accurate eclipse, given its equation. It was decided to concentrate on the inner contour.

The next program tried was developed from the previous one, but for the single contour, and included additional statements to meet the demands of finer scanning, when entering and leaving the curve, and also whenever the |gradient| exceeded one. Those additional statements are introduced in the succeeding paragraphs.

Additional Statements

When scanning, the increment used, in the interests of time and space, should not be unnecessarily fine, so it was considered right to continue with the increment 0.2 as the general increment and to introduce the finer increment .02, when either of the two sets of circumstances, already described, arose.

To measure the gradient, the finite difference formula, which has a general application, was taken and the conditional statements, -

```
IF (ABS( (Y(J)-Y(J-1))/(X(J)-X(J-1)).GT.1.)  GO TO 20
20  X = X+.02          :
```

and

```
IF (ABS( (Y(J)-Y(J-1))/(X(J)-X(J-1)).LT.1.)  GO TO 30
                       :
30  X = X+.2
```

were introduced as a means of branching to a change in increment, as demanded by the value of the gradient.

On the other hand, the change to finer scanning on entering the curve was achieved by a different procedure, which is shown in the following extract from the program (the complete program 6 is not shown, since it differs from program 5 only in the additional statements described here).

```
10    IF (Q(X).GE.0.)  GO TO 20
      X = X+.2
      GO TO 10
20    X=X-.2
24    X = X+.02
      IF (Q(X) .GE.0.)  GO TO 30
      GO TO 24
30    J = 1
```

The subtraction in statement 20 was necessary to allow a new approach to the curve, using the smaller increment, but the conditional statements of the previous paragraph were used to control the exit. The principle introduced in statements 20 and 24 proved to be of vital importance in the final solution of the main problem. The new program gave the plot shown in figure 6A, which shows little improvement on that of figure 5. The number of cycles of the loop had been doubled. The program was repeated with the measure of the fundamental plotter length, 0.1", being used as the smaller increment. This length is that of the smallest trace which the plotter can make. It was hoped that, by using this fine measure, the gap might be closed. There was some improvement, as can be seen by comparing figures 6A and 6B, but the result was disappointing and it now appeared at this stage that the method might have to be abandoned. The number of cycles had now trebled.

FIGURE 6A

RESULT OF PROGRAM 6 WITH INCREMENTS OF .2 and .02

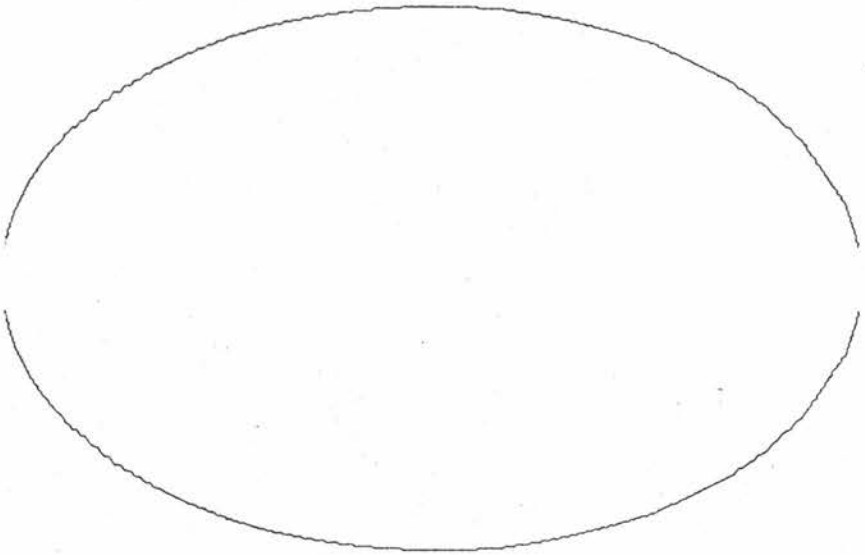
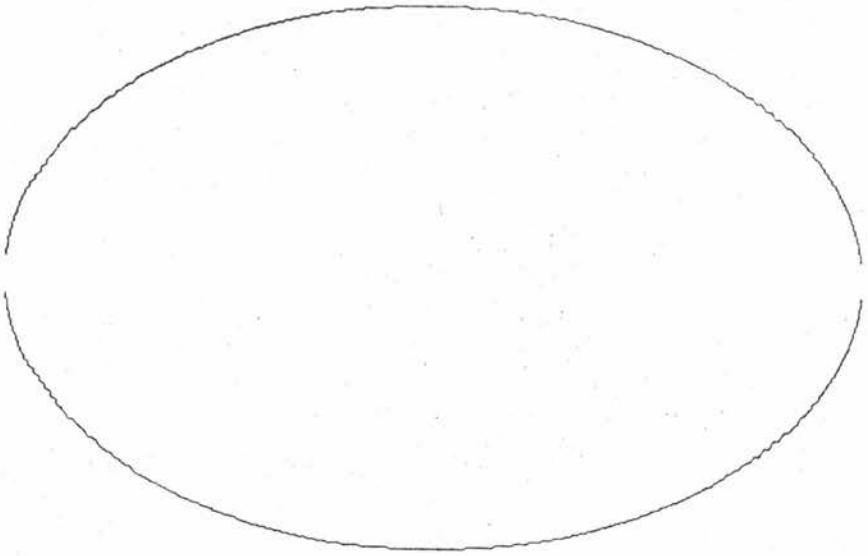


FIGURE 6B

RESULT OF PROGRAM 6 WITH INCREMENTS .2 and .01



3. An effective program

It had become clear that it was the value of the gradient that had greatest significance in determining a suitable increment. As their product is a rough measure of the change in Y produced by the increment in X, to avoid the polygon effect, there had to be several refinements of the increment as the |gradient| increased, and, for the sake of economy, the process would be reversed as the |gradient| diminished. However, with regard to the closing of the gaps, where the |gradient| was very large indeed, the method of refining the increment as |gradient| increased was eminently suitable when leaving the curve, but, as gradient values would not be available on entry to the curve from a region of complex roots, some other method had to be adopted for entry. This other method had to be consistent with the gradient - increment procedure already described, and was, in fact, a development of the subtract and add technique of program 6. When the development of this technique was first considered, it was thought that, with the increments decreasing by powers of ten, as shown in the following sequence of loops -

```
10    X=X+.1
      IF (Q(X).GT.0.)  GO TO 20
      GO TO 10
20    X=X-.1
25    X=X+.01
      IF (Q(X).GT.0.)  GO TO 30
30    X=X-.01
35    X=X+.001
      IF (Q(X).GT.0.)  GO TO 40
40    X=X-.001,
```

a program could evolve to meet the situation, but that it would be too costly in time and storage space

However, when it was realised that there could not be more than ten cycles in each loop, the solution of the main problem seemed to be a possibility. The plan was now to proceed by a method of trial and error, with increasing refinement of the increments used until either the branches merged or the ends were sufficiently close for the gaps to be closed satisfactorily by a joining routine. During this period of trial and error, several programs were run with increments ranging from .1 to .001, both for entry to the curve and for curve tracing. For curve tracing, each increment gradually became associated with a particular range of gradient rather than with a particular value. The delimiting values of these ranges bore some rough relationship to the powers of ten in ascending order, in contradistinction to the order of the increments. Satisfactory diagrams were still not being produced and it became clear that those delimiting values were of critical importance. Not only had they to be related to a particular increment, but the value of the gradient itself and its rate of change were pertinent to their selection. After some estimates and trials the values shown in the next program were selected, as likely to give satisfactory results and also provide some margin for error. The value of the ultimate increment was also reduced.

The gradient value had assumed a greater importance than had been apparent at the beginning. It had to be as accurate as possible, especially where the second derivative was high. Consequently, the finite difference formula for the gradient was replaced by the first derivative, which is accurate and easily calculated.

Gradually a program was developed which proved successful.

First program to give a continuous contour

Program 7

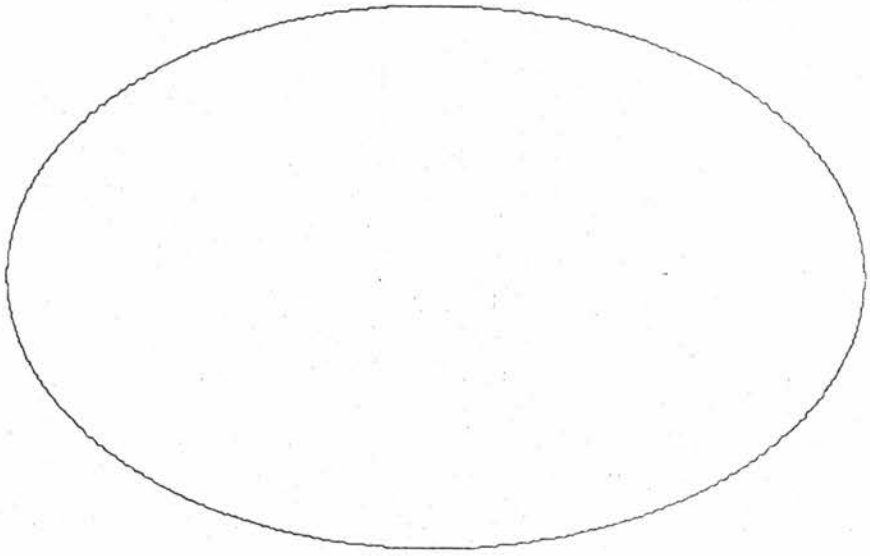
```
C   EQUATION OF INNER CONTOUR IN GENERAL  FORM IS
C                  2XX+5YY-20X-50Y+165=0
      Q(X) =-10.*X*X+100.*X-200.
C   THE FOLLOWING STATEMENT FUNCTION MEASURES THE GRADIENT
      D(X,Y)=(10.-2.*X)/(5.*Y-25.)
      DIMENSION  XX(250), Y(250), YY(250)
      CALL PLOT (201,0.,10.,10.,1.,0.,10.,10.,1.)
      X=0.
      J=0
 10   IF (Q(X).GE.0.)  GO TO 20
      X=X+.1
      GO TO 10
 20   X=X-.1
 25   X=X+.01
      IF(Q(X).GE.0.) GO TO 30
      GO TO 25
 30   X=X-.01
 35   X=X+.001
      IF (Q(X).GE.0.)  GO TO 40
      GO TO 35
 40   X=X-.001
 45   X=X+.0001
      IF (Q(X).GE.0.)  GO TO 50
      GO TO 45
 50   IF(Q(X).LT.0.)   GO TO 70
C   PREVIOUS STATEMENT TAKES EFFECT WHEN ROOTS BECOME
C   COMPLEX AND BRANCH LEADS TO PLOTTING ROUTING
      J=J+1
      XX(J)=X
       Y(J)=5.+SQRT(Q(X))/5.
      YY(J)=5.-SQRT(Q(X))/5.
C   THE NEXT THREE STATEMENTS SHOW THE DELIMITING VALUES
C   OF THE GRADIENT RANGES
```

```
      IF(ABS(D(X,Y(J))).GT.20.)  GO TO 65
      IF(ABS(D(X,Y(J))).GT.5.)   GO TO 60
      IF(ABS(D(X,Y(J))).GT.1.)   GO TO 52
      X=X+.1
      GO TO 50
   52 X=X+.01
      GO TO 50
   60 X=X+.001
      GO TO 50
   65 X=X+.0001
      GO TO 50
   70 WRITE (6,75) J
   75 FORMAT (' ',I4)
C    WRITE STATEMENTS GIVE SIZE OF ARRAYS
      CALL PLOT (98,XX(1),Y(1))
      DO 80  N=1,J
   80 CALL PLOT (90,XX(N),Y(N))
      CALL PLOT (98,XX(1),YY(1))
      DO 90 N=1,J
   90 CALL PLOT (90,XX(N),YY(N))
      CALL PLOT (98,0.,0.)
      CALL PLOT (7)
      STOP
      END
```

FIGURE 7A

ELLIPSE 2XX+5YY-20X-50Y+165=0

WITH INCREMENTS .1,.01,.001 and .0001



4. TESTING THE PROGRAM

For the ellipse 2XX+5YY-20X-50Y+165=0 with its associated statement functions -

Q(X) = -10XX+100X-200 and

D(X,Y) = (10-2X)/(5Y-25),

a program had now been written which had yielded a continuous curve, accurate within the limits of the computer-plotter combination. Its major axis was parallel to the direction of scan. To test the program, a greater length of high gradient arc was presented to the scanning ordinate by rotating the ellipse about its centre, first, into an intermediate position where its major axis was at an angle of PI/4 radians to the horizontal and, finally, to a position where it was vertical. The equation of the ellipse in the first position is

7XX+7YY-6XY-40X-40Y+180=0, and

the associated statement functions are

P(X) = (3X+20)/7

Q(X) = 400X-40XX-860 and

D(X,Y) = (14X-6Y-40)/(6X-14Y+40)

The expressions for the y- ordinates are

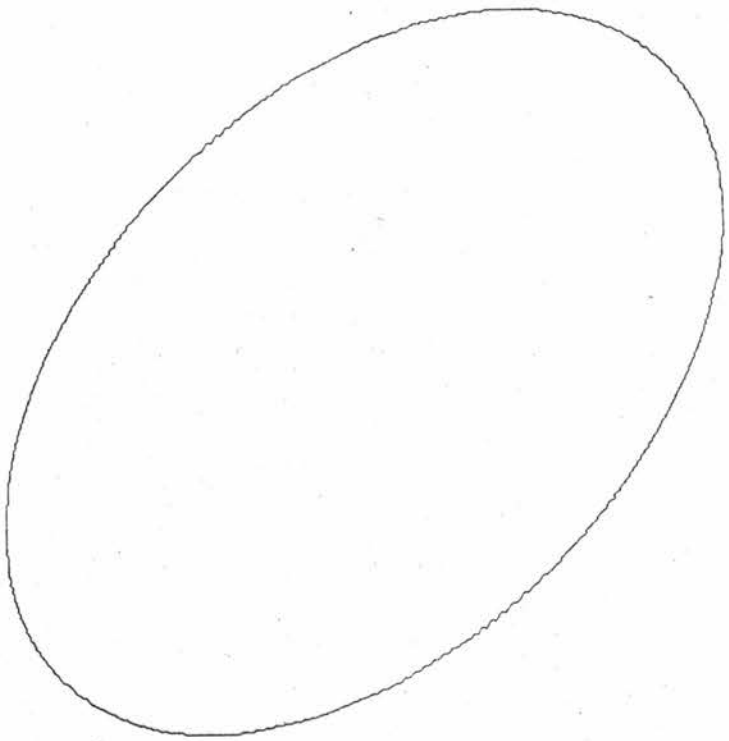
Y(J) = P(X)+SQRT(Q(X))/7.

and YY(J) = P(X)-SQRT(Q(X))/7.

The program was amended according and the satisfactory curve of figure 7B was the result.

FIGURE 7B

ELLIPSE 7XX+7YY-6XY-40X-40Y+180=0



With the ellipse in the final position the equations are

```
5XX+2YY-50X-20Y+165=0
Q(X)=-10XX+100X-230
D(X,Y)=(5X-25)/(10-2Y)
Y(J)=5+SQRT(Q(X))/2
```

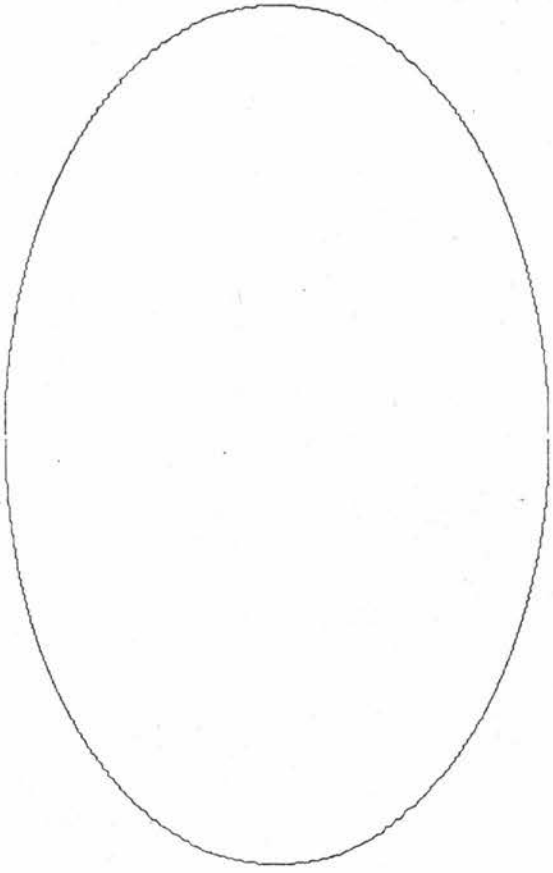
and `YY(J)=5-SQRT(Q(X))/2`

The resulting tracing is shown in figure 7C, and in this extreme position the re-appearance of the gaps between the branches can just be detected.

The count, not surprisingly, is over 300, and was anticipated in the dimension statement.

FIGURE 7C

GAPS JUST VISIBLE



The re-appearance of the gaps with the ellipse in the vertical position presented no problem. They are very small and disappeared with the further refinement of the increment, achieved by including in the program the following six statements

```
      IF (Q(X).GE.0.) GO TO 46
      GO TO 45
   46 X=X-.0001
   47 X=X+.0001
      IF (Q(X).GE.0.)  GO TO 50
      GO TO 47
```

However, as the curve in the neighbourhood of the gaps is now vertical, a simple joining routine would suffice.

The following statements constitute such a routine

```
      CALL PLOT (90,XX(J),YY(J))
      CALL PLOT (90,XX(J),Y(J))
      CALL PLOT (98,XX(1),Y(1))
      CALL PLOT (90,XX( 1 ),Y(1))
      CALL PLOT (90,XX(1),YY(1))
```

The program amended in either fashion produced good results as in figures 7D and 7E. Generally, a joining routine would not be required, but its inclusion might be desirable in extreme cases where the major axis is long relative to the minor, and is in a vertical position.

FIGURE 7D

GAPS CLOSED BY REFINING INCREMENTS

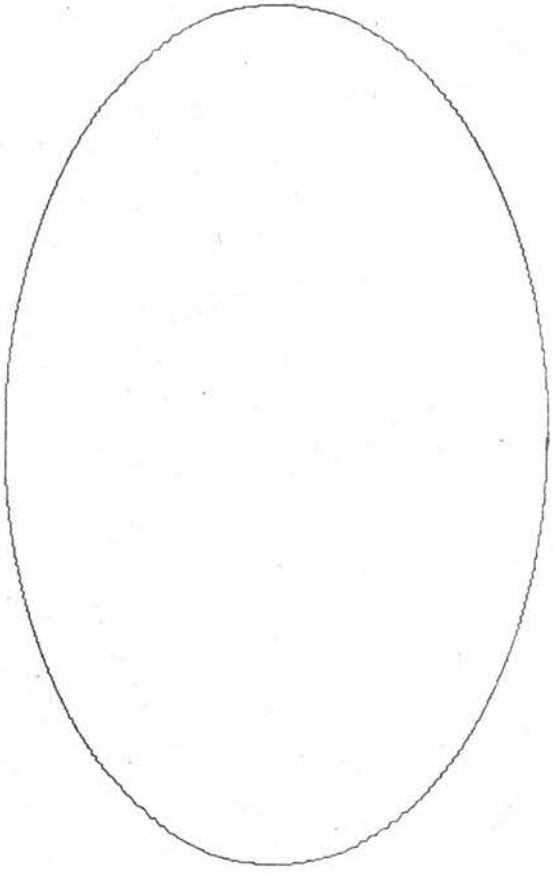
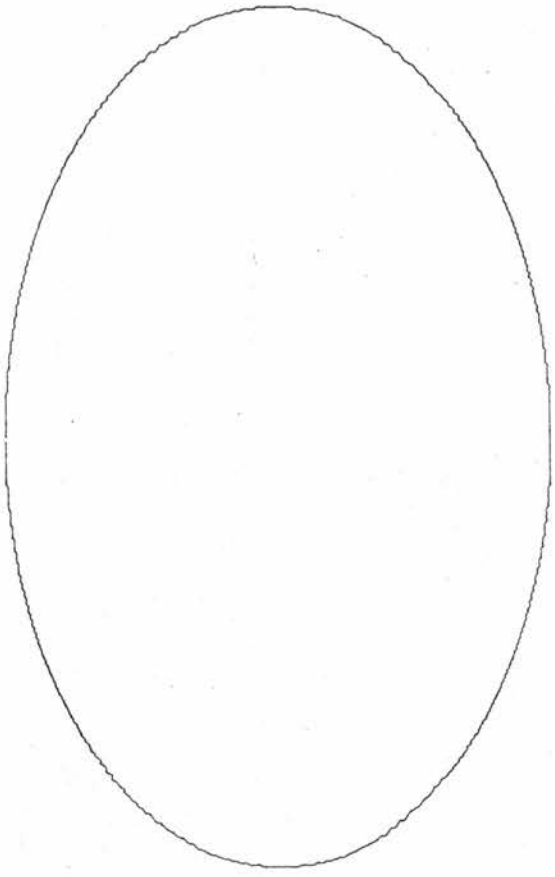


FIGURE 7E

GAP CLOSED USING JOINING ROUTINE



PROGRAM FOR A SET OF CONTOURS

As a program for the accurate tracing of ellipses had been developed, it was relatively a simple matter to add the statements to give the complete program for the tracing of a set of elliptical contours of a specially chosen second degree surface. The pattern is similar to that of program 4, written for the contours of the plane. Although the present program differs little from the previous one, it is written in full, as it was the main objective of this part of the investigation.

PROGRAM 8

```
C   FINAL PROGRAM FOR A SET OF CONTOURS OF THE SPECIALLY
C   CHOSEN SURFACE 2XX+5YY-20X-50Y-10Z+175=0
        Q(X,Z)=50.*Z-10.*X*X+100.*X-250.
        D(X,Y)=(10.-2.*X)/(5.*Y-25.)
        DIMENSION  XX(500),Y(500),YY(500)
        CALL PLOT (201,0.,10.,10.,1.,0.,10.,10.,1.)
        Z=1.
        DO  95  KK=1,5
        X=0.
        J=0
   10   IF (Q(X,Z).GE.0.)  GO TO 20
        X=X+.1
        GO TO 10
   20   X=X-.1
   25   X=X+.01
```

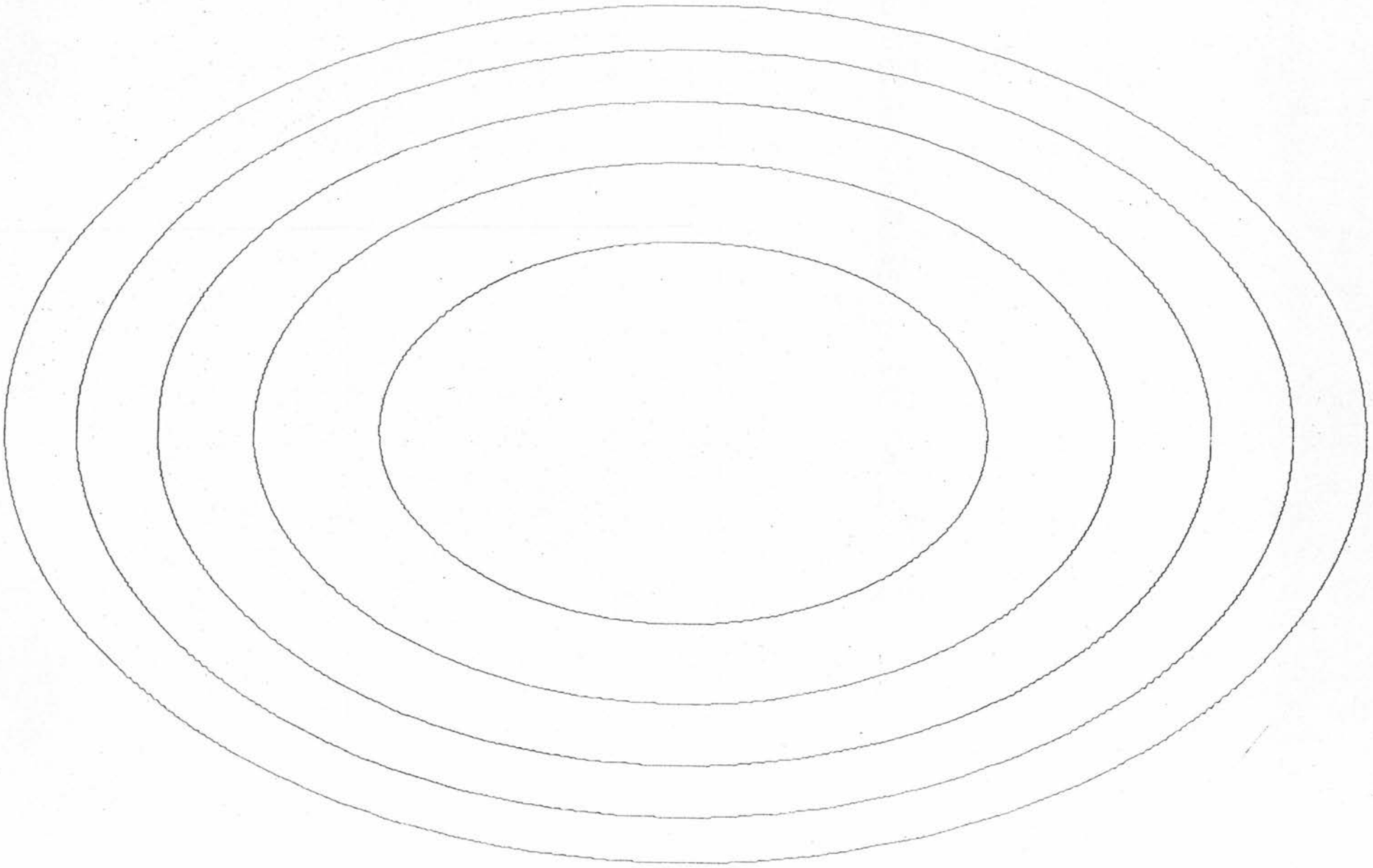
```
      IF (Q(X,Z).GE.0.)   GO TO 30
      GO TO 25
30    X=X-.01
35    X=X+.001
      IF (Q(X,Z). GE.0.)  GO TO 40
      GO TO 35
40    X=X-.001
45    X=X+.0001
      IF (Q(X,Z).GE.0.)  GO TO 50
      GO TO 45
50    IF (Q(X,Z).LT.0.)  GO TO 70
      J=J+1
      XX(J)=X
       Y(J)=5.+SQRT(Q(X,Z))/5.
      YY(J)=5.-SQRT(Q(X,Z))/5.
      IF (ABS(D(X,Y(J))).GT.20.)  GO TO 65
      IF (ABS(D(X,Y(J))).GT.5.)   GO TO 60
      IF (ABS(D(X,Y(J))).GT.1.)   GO TO 52
      X=X+.1
      GO TO 50
52    X=X+.01
      GO TO 50
60    X=X+.001
      GO TO 50
65    X=X+.0001
      GO TO 50
```

```
 70  CALL PLOT (98,XX(1),Y(1))
     DO 80   N=1,J
 80  CALL PLOT (90,XX(N),Y(N))
     CALL PLOT (98,XX(1),YY(1))
     DO  90    N=1,J
 90  CALL PLOT (90,XX(N),YY(N)
C    A JOINING ROUTINE FOLLOWS
     CALL PLOT (90,XX(J),Y(J))
     CALL PLOT (98,XX(1),Y(1))
     CALL PLOT (90,XX(1),Y(1))
     CALL PLOT (90,XX(1),YY(1))
 95  Z=Z+1.
     CALL PLOT (98,0.,0.)
     CALL PLOT (7)
     STOP
     END
```

The set of contours traced is shown in figure 8.

FIGURE 8

CONTOURS OF A SECOND DEGREE SURFACE



Program 8 assumes that all contours are continuous and completely within the plotting quadrant. There remained the original and more comprehensive problem of programming to detect and trace whatever contours or parts of contours might be in the quadrant. Its solution became the objective of the remaining part of the investigation, and is described in the next section.

PART 3

PART OR WHOLE OF CURVE OUTSIDE PLOTTING QUADRANT

1. In attacking the more general problem of programming to trace the contours of any second degree surface, the intention was to continue with the previous procedures of scanning, of the grading of increments and of relating the increments chosen to the numerical values of the gradients. In the tracing done previously, both branches of the contour had been completely within the plotting quadrant, and there had been a one to one correspondence between points on each branch and positive values of the discriminant of a quadratic equation. It therefore had been possible to deal with the two branches simultaneously.

There was no such symmetry in the new situation where an individual branch might be completely within the quadrant, partly within or entirely outside. It was necessary to evaluate and plot for each branch separately. However, the sequence of procedures was the same for each branch, and so the overall structure of the new program was that of a loop which had to be traversed twice. These traverses were effected by making use of the difference in the expressions for Y in the two branches when writing the program. Thus, if the equation associated with one branch is $Y = A + \sqrt{B}$, then the equation associated with the other will be $Y = A - \sqrt{B}$. Each equation can be represented by the form $Y = A + S\sqrt{B}$ where S is equal to 1 or -1. This form was

adopted in the program. S was initially assigned the value 1, and after the routine associated with the corresponding branch had been completed, the sign of S was changed. A test was then made to determine whether control should return to the beginning of the routine for the second traverse or proceed to the next statement. This pattern is demonstrated by the following selection of statements

```
          S = 1.
C  BEGINNING OF BRANCH ROUTINE
        5 X =
          .
          .
          .
          .
          .
          Y = A + S*SQRT(B)
          .
          .
          .
C  END OF BRANCH ROUTINE
       80 S = -S
       85 IF (S.LT.0.)  GO TO 5
          STOP
          END
```

Further, in the extended problem, the configuration within the plotting quadrant, if it existed, was unknown. It was therefore necessary to do a complete scan from 0 to 10 for each branch, and so X greater than 10 could be a condition for transfer to the next scan, to the end of the program or to a plotting routine if sets of co-ordinates had been stored.

Whole or parts of a branch might be missing, consequently, when real roots, which were relevant, were detected, the corresponding values of X and Y were stored in arrays as before. On reaching a boundary, including that between regions of real and complex roots, an exit was made to the plotting routine, and the corresponding arc was traced. Examples of those exits are shown in the next complete program, program 9, by those transfers to statement 90. Statement 90 introduces the routine which is as follows

```
90  CALL PLOT (98,XX(1),YY(1))
    DO 70    N = 1,J
70  CALL PLOT (90,XX(N),YY(N))
    IF (X.LT.10.)  GO TO 8
C   SCANNING IS RESUMED IF X IS LESS THAN 10
```

Program 9, a development of previous programs, and including the new ideas put forward in those recent paragraphs, was tested on the hyperbola, whose equation in standard form is

$$\frac{(X-5) * (X-5)}{2} - \frac{(Y-5) * (Y-5)}{5} = 1,$$

with the favourable result shown in figure 9A

PROGRAM 9

```
C   HYPERBOLA  5XX-2YY-50X+20Y+65 = 0
          Q(X)  = (5.*X*X-50.*X+115.)/2.
          YF(X) = 5.+SQRT(Q(X))*S
          D(X,Y) = (5.*(X-5.))/(2.*(Y-5.))
          DIMENSION  XX(2000), YY(2000)
          CALL PLOT (1,0.,10.,10.,1.,0.,10.,10.,1.)
          S = 1.
    5     X = 0.
    8     J = 0
   10     X = X+.1
          IF (X.GT.10.)  GO TO 100
          IF (Q(X).LT.0.)  GO TO 10
          Y = YF(X)
          IF (Y*(10.-Y).LT.0.)  GO TO 10
          X = X-.1
   11     X = X+.01
          IF (Q(X).LT.0.)  GO TO 11
          Y = YF(X)
          IF (Y*(10.-Y).LT.0.)  GO TO 11
          X = X-.01
   12     X = X+.001
          IF (Q(X).LT.0.)  GO TO 12
          Y = YF(X)
```

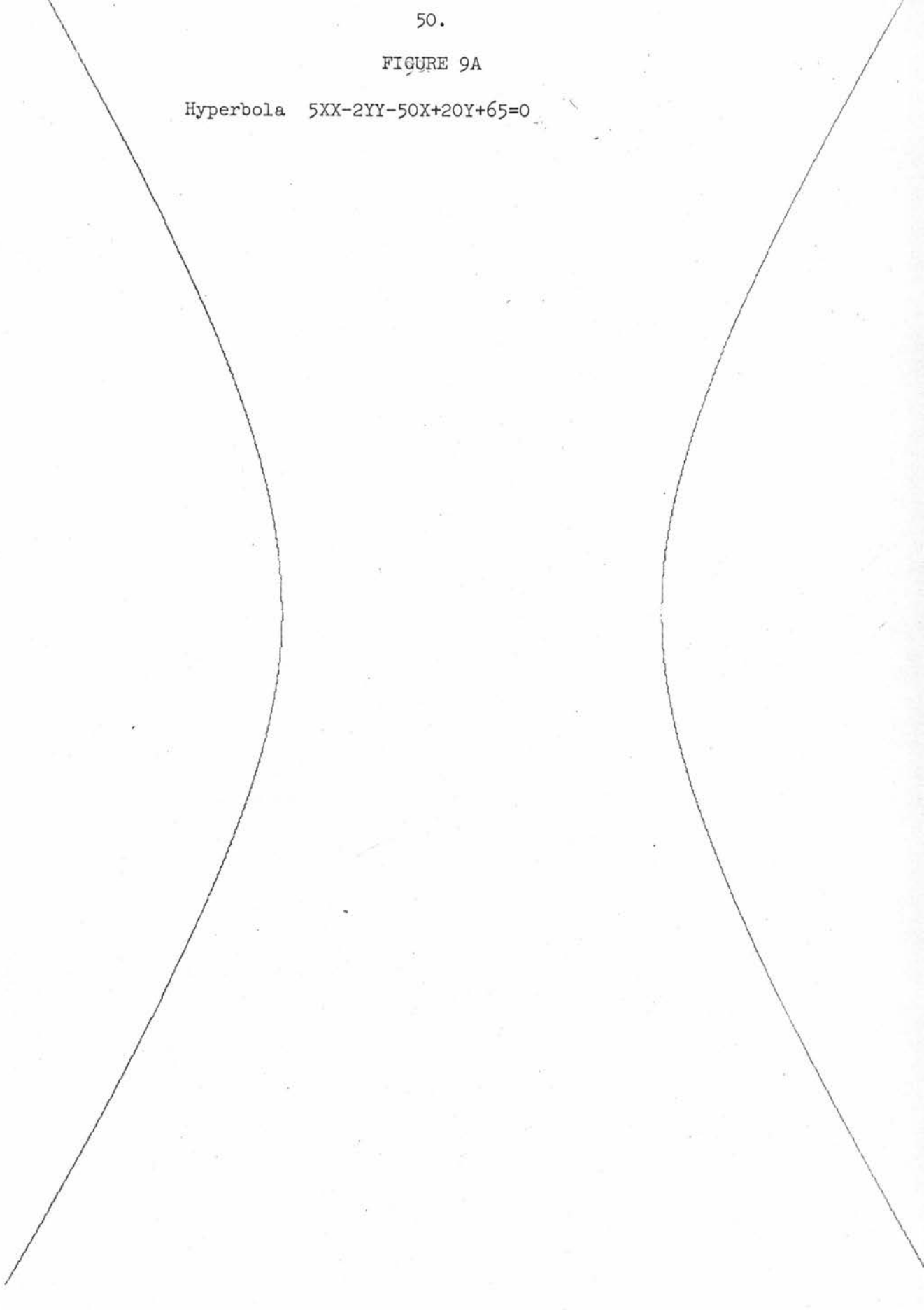
```
      IF (Y*(10.-Y).LT.0.)  GO TO 12
      X = X-.001
   13 X = X+.0001
      IF (Q(X).LT.0.)  GO TO 13
      Y = YF(X)
      IF (Y*(10.-Y).LT.0.)  GO TO 13
      X = X-.0001
   14 X = X+.00001
      IF (Q(X).LT.0.)  GO TO 14
      Y = YF(X)
      IF (Y*(10.-Y).LT.0.)  GO TO 14
C     THE '14' LOOP SUPPLANTS THE PLOTTING ROUTINE
C     VALUES ARE NOW ENTERED IN ARRAYS
   17 J = J+1
      XX(J)=X
      YY(J)=Y
      IF (ABS(D(X,Y)).GT.50.)  GO TO 47
      IF (ABS(D(X,Y)).GT.20.)  GO TO 45
      IF (ABS(D(X,Y)).GT.2.)  GO TO 35
      IF (ABS(D(X,Y)).GT.1.)  GO TO 25
   20 X = X+.1
      GO TO 50
   25 X = X+.01
      GO TO 50
   35 X = X+.001
      GO TO 50
```

```
 45 X = X+.0001
    GO TO 50
 47 X = X+.00001
C   THE THREE CONDITIONAL STATEMENTS WHICH FOLLOW CORRESPOND
C   TO THE THREE BOUNDARIES WHERE CONTROL IS PASSED
C   TO THE PLOTTING ROUTINE.
 50 IF (X.GT.10.)  GO TO 90
    IF (Q(X).LT.0.)  GO TO 90
    Y = YF(X)
    IF (Y*(10.-Y)).LT.0.)  GO TO 90
    GO TO 17
C   PLOTTING IS NOW INTRODUCED.
 90 CALL PLOT (98,XX(1),YY(1))
    DO 70   N=1,J
 70 CALL PLOT (90,XX(N),YY(N))
    IF (X.LT.10.)  GO TO 8
C   THIS CONDITIONAL STATEMENT ENSURES THAT EACH SCAN
C   IS COMPLETED
100 S = -S
    IF (S.LT.0.)  GO TO 5
    CALL PLOT (98,0.,0.)
    CALL PLOT (7)
    STOP
    END
```

The resulting diagram is shown in figure 9A

FIGURE 9A

Hyperbola 5XX-2YY-50X+20Y+65=0



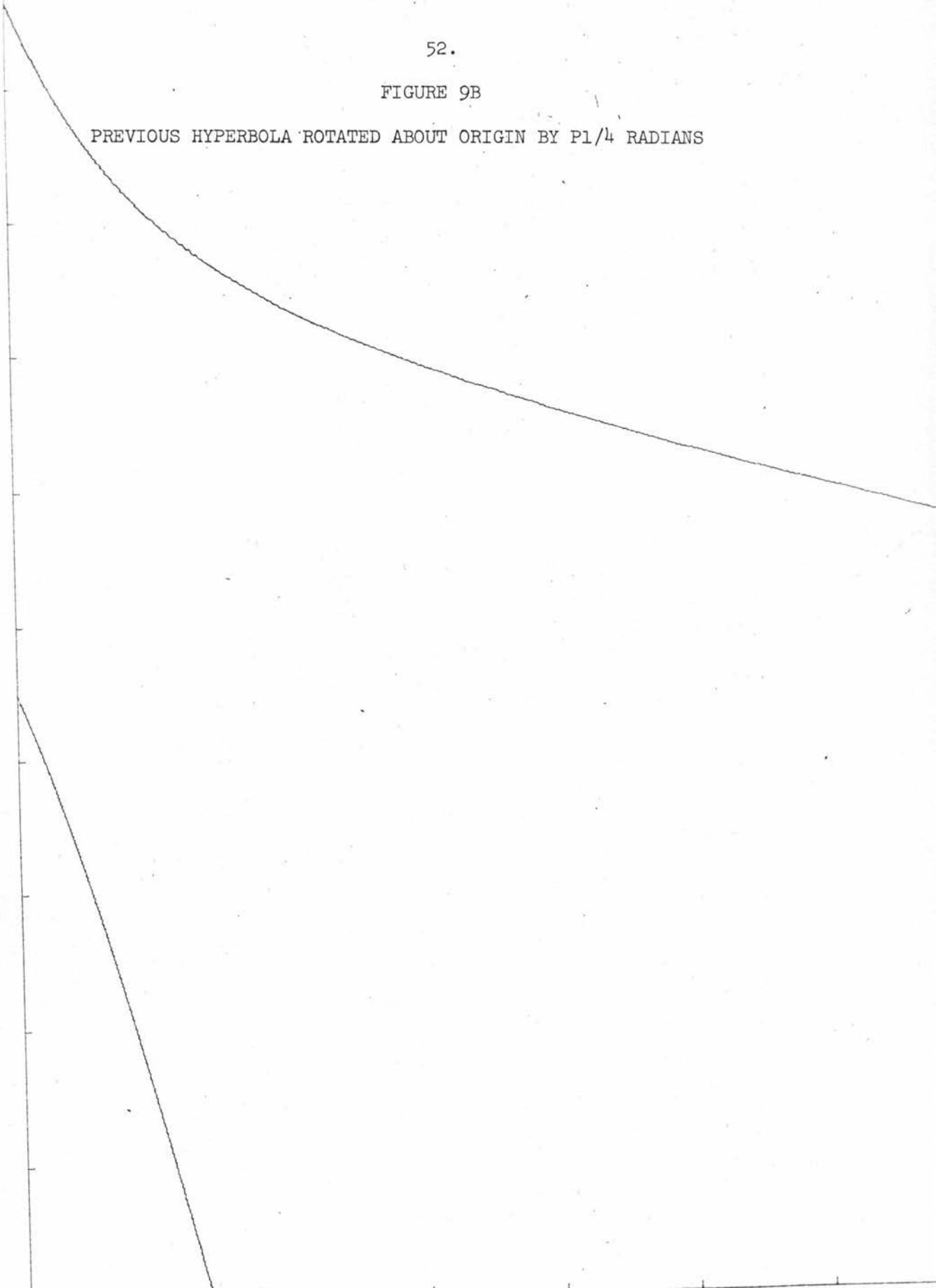
The branches of the hyperbola in figure 9A are bounded by 2 of the 3 types of boundary mentioned earlier. To test the program for the three types simultaneously, this hyperbola was considered as occupying a new position, obtained by rotating it about the origin through an angle of PI/4 radians. The hyperbola in the new position has the associated equations

```
3XX+14XY+3YY-70SQRT(2)X-30SQRT(2)Y+130=0
D(X,Y)=(6X+14Y-70SQRT(2)/(14X+6Y-30SQRT(2))
P(X) =     (15SQRT(2)-7X)/3
Q(X) =     10XX+15
  Y  =     P(X)+25SQRT(Q(X))/3
```

The program, as amended, produced the curve shown in figure 9B.

FIGURE 9B

PREVIOUS HYPERBOLA ROTATED ABOUT ORIGIN BY P1/4 RADIANS



2. The current program had been written for a known second degree equation, but without knowledge of the associated configuration It had now been tested and proved where the configuration was known All that remained was to modify it to fit the general equation. This was done quite simply by substituting general statement functions for those at the beginning of program 9, and by using them in the program where appropriate. It had also been necessary to introduce a read statement for the parameters of the equations.
The derivation of the functions follows from an analysis of the general equation. If the equation is

AXX+RXY+BYY+GX+FY+C=0,

D(X,Y) = -(2AX+RY+G)/(RX+2BY+F) and

Y = (-(RX+F)+S*SQRT((RX+F)**2-4B(AXX+GX+C)))/2B, if B≠0
or

Y = -(AXX+GX+C)/(RX+F), if B = 0

(The instance where RX+F=0 is irrelevant).
The first statements of the program were of the form

```
C  AXX+RXY+BYY+GX+FY+C=0
   D(X,Y) = -(2.*A*X+R*Y+G)/(R*X+2*B*Y+F)
   FN(X)  = R*X+F
   FF(X) = X*(A*X+G)+C
   Q(X) = FN(X)**2-4.*B*FF(X)
   YF(X)= (-FN(X)+S*SQRT(Q(X)))/(2.*B)
   YYF(X) = -FF(X)/FN(X)
```

```
    DIMENSION  XX(2000),YY(2000)
    CALL PLOT
    READ (5,7) A,R,B,G,F,C
 7  FORMAT
```

This general program was used with two sets of parameters, one for a circle with centre(5,15) and radius 13/2, and the other for a parabola with directrix X-Y-2=0 and focus (5,5). The tracings were done in a setting established by CALL PLOT (1,0.,10.,10.,.1.,0.,10.,10.,1.) and are shown in figures 9C and 9D

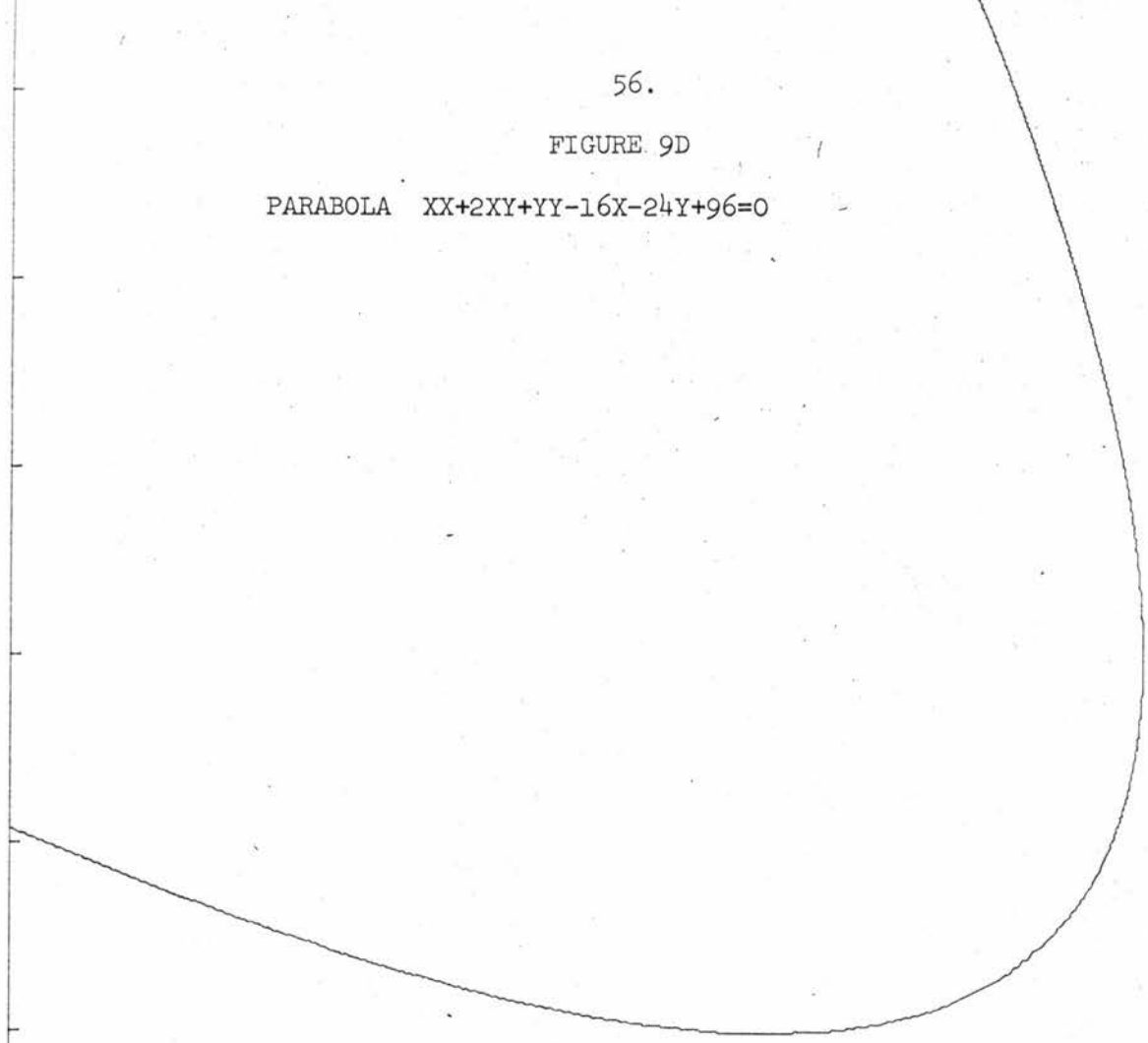
FIGURE 9C

CIRCLE, CENTRE(5,15), RADIUS 13/2.

A=4.,R=0.,B=4.,G=-40.,F=-120.,C=831.

FIGURE 9D

PARABOLA XX+2XY+YY-16X-24Y+96=0



3. A program had now been written to give the quadrant configuration for the general equation of the second degree in X and Y, when the parameters of the equation were read. To obtain the program which would result in the plotting of contours for any second degree surface, the goal of the whole investigation, the core of the previous program became the contents of a loop, and certain modifications were made depending on the contour constants. This program, the ultimate program, is written for the condition $B \neq 0$. A similar routine applies if $B = 0$ and to which control would pass under this contingency. The program was executed with the parameters having the following values -
$A = .04$, $R = .0$, $B = .0625$, $G = -0.96$, $F = -1.375$ and $C = 13.3225$, and for heights of 1, 2, 3, 4, 5 and 6. The complete configuration should be part of a set of six concentric ellipses around (12,11) whose major and minor axes are respectively parallel to the X and Y axes of co-ordinates. The output from this program, Program 10, giving the configuration for the plotting quadrant is shown in figure 10.

PROGRAM 10.

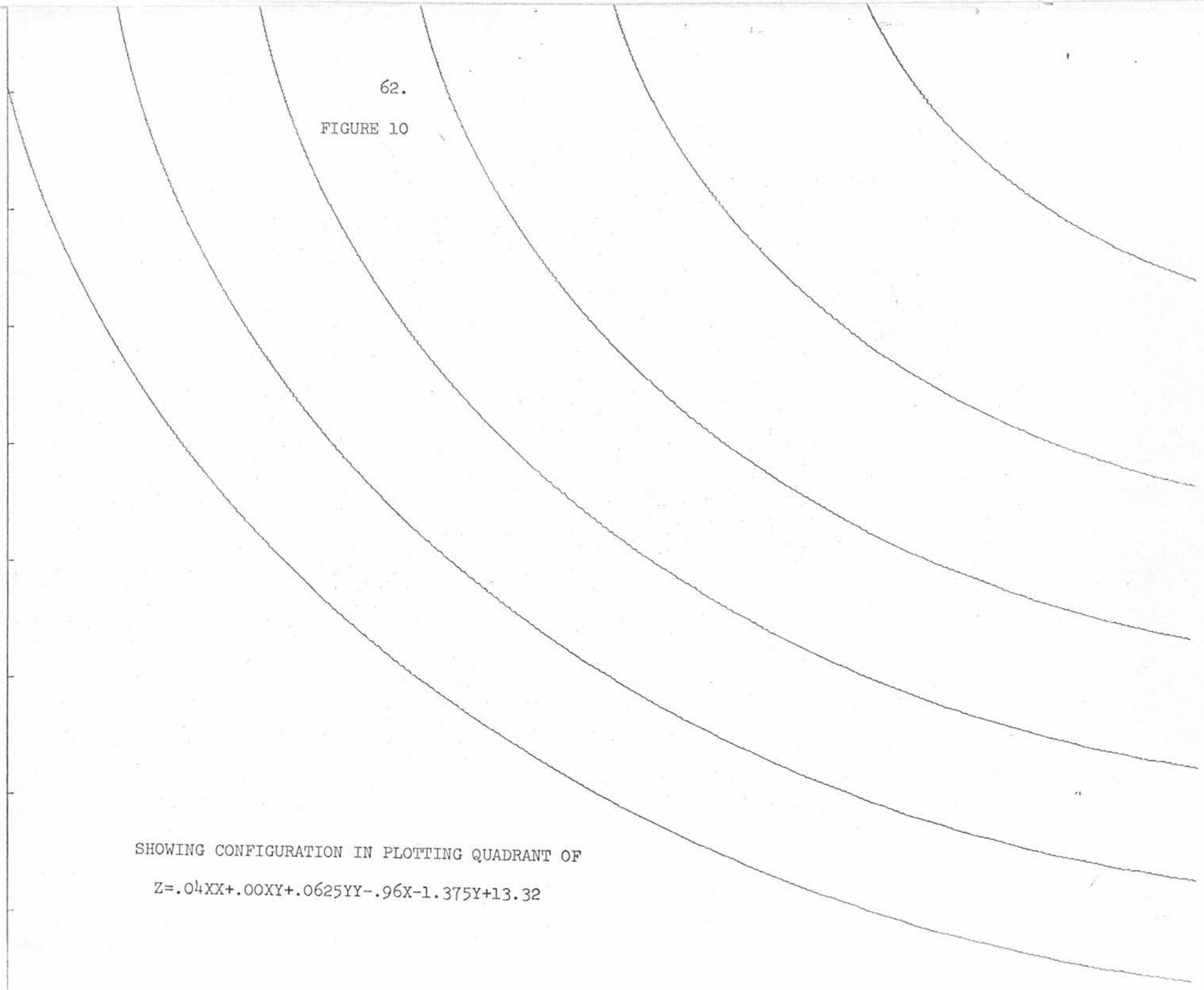
```
C   FOR CONTOUR CONFIGURATION, WITHIN PLOTTING QUADRANT,
C   OF ANY SECOND DEGREE SURFACE Z=AXX+RXY+BYY+GX+FY+C=0
C   CONTOURS CORRESPOND TO HEIGHTS FROM 1 THROUGH INTEGERS
C   TO 6.
C   B IS NOT EQUAL TO ZERO.
          D(X,Y)=-(2.*A*X+R*Y+G)/(R*X+2.*B*Y+F)
          FN(X)=R*X+F
          FF(X)=X*(A*X+G)+(C-Z)
          Q(X)=FN(X)*FN(X)-4.*B*FF(X)
          YF(X)=(-FN(X)+S*SQRT(Q(X)))/(2.*B)
          DIMENSION XX(2000),YY(2000)
          CALL PLOT(1,0.,10.,10.,1.,0.,10.,10.,1.)
          READ (5,7)A,R,B,G,F,C
        7 FORMAT(6F6.4)
C   FORMAT CHOSEN WILL DEPEND ON PARAMETERS
          Z=1.
          S=1.
          DO 200  K=1,6
        5 X=0.
        8 J=0
       10 X=X+.1
          IF(X.GT.10.)  GO TO 100
```

```
      IF(Q(X).LT.0.)  GO TO 10
      Y=YF(X)
      IF(Y*(10.-Y).LT.0.)  GO TO 10
      X=X-.1
11    X=X+.01
      IF(Q(X).LT.0.)  GO TO 11
      Y=YF(X)
      IF(Y*(10.-Y).LT.0.)  GO TO 11
      X=X-.01
12    X=X+.001
      IF(Q(X).LT.0.)  GO TO 12
      Y=YF(X)
      IF(Y*(10.-Y).LT.0.)  GO TO 12
      X=X-.001
13    X=X+.0001
      IF(Q(X).LT.0.)  GO TO 13
      Y=YF(X)
      IF(Y*(10.-Y).LT.0.)  GO TO 13
      X=X-.0001
14    X=X+.00001
      IF(Q(X).LT.0.)  GO TO 14
      Y=YF(X)
      IF(Y*(10.-Y).LT.0.)  GO TO 14
17    J=J+1
      XX(J)=X
```

```
      YY(J)=Y
      IF(ABS(D(X,Y)).GT.50.)  GO TO 47
      IF(ABS(D(X,Y)).GT.20.)  GO TO 45
      IF(ABS(D(X,Y)).GT.2.)  GO TO 35
      IF(ABS(D(X,Y)).GT.1.)  GO TO 25
20    X=X+.1
      GO TO 50
25    X=X+.01
      GO TO 50
35    X=X+.001
      GO TO 50
45    X=X+.0001
      GO TO 50
47    X=X+.00001
50    IF(X.GT.10.)  GO TO 90
      IF(Q(X).LT.0.)  GO TO 90
      Y=YF(X)
      IF(Y*(10.-Y).LT.0.)  GO TO 90
      GO TO 17
90    CALL PLOT (98,XX(1),YY(1))
      DO 70  N=1,J
70    CALL PLOT(90,XX(N),YY(N))
      WRITE(6,27) J
27    FORMAT('    ',14)
```

```
C   WRITE STATEMENT IS INFORMATIVE
      IF(X.LT.10.)  GO TO 8
100   S=-S
      IF(S.LT.0.)  GO TO 5
200   Z=Z+1.
      CALL PLOT(98,0.,0.)
      CALL PLOT(7)
      STOP
      END
```

FIGURE 10



SHOWING CONFIGURATION IN PLOTTING QUADRANT OF

$Z=.04XX+.00XY+.0625YY-.96X-1.375Y+13.32$

Concluding Remarks

1. With the writing of program 10 the development proposed in the title of this paper had been completed. Attention had been confined to the plotting quadrant adjacent to the origin (0.,0.), but there is no need for such restriction. The contours of any part of the surface can be traced. Program 10, however, would require to be modified to include two parameters which would correspond to the co-ordinates of a new origin.

2. The technique adopted for drawing second degree curves is essentially that of joining points on the curves linearly. The succession of joins constitute the curve. This technique is quite different from that of Pitteway (1), used for a similar purpose and described in the Computer Journal, Volume 10. It is applied to an ellipse which is entirely in the plotting quadrant.
Pitteway's method depends on a simple algorithm by Bresenham (2) which is used to draw the straight line between two points as a succession of fundamental movements of the plotter, and as efficiently as possible. This algorithm could have some bearing on a future development of the programs of this paper, and is given in some detail.

Bresenham takes the eight fundamental directions of the plotter to divide the plane into octants. The plane, as before, is regarded as a grid of small squares whose sides are equal to the fundamental plotter length of .01 in. and lengths are expressed in this unit. A moment's reflection will convince that the step between any two vertices can be associated with one of those octants, and that the join can be effected by the plotter using no more than two of the eight fundamental movements. There is a theoretical straight line between the vertices. When tracing, the stylus begins at the first vertex and throughout its path has a choice of two adjacent vertices. Under the control of the algorithm, it moves to that which is nearest the theoretical line and the resulting trace is the straight line in this context. If the differences in the co-ordinates of the two vertices being joined are given by the equations DX=U and DY=V, U and V will be integers and V/U will measure the gradient of the join. The algorithm is true for all integral values of U and V, but in the context of curve tracing U and V are numerically small. Its exact form depends on the associated octant. Should U and V be positive and U greater than V the associated octant will be the first, and the corresponding Fortran program for the I.B.M. plotter would be

```
      X = 0.
      Y = 0.
      V = 12.
      U = 15.
C     ARBITRARY VALUES ASSIGNED TO U AND V
      B = 2*V
      A = 2*U-B
      D = B-U
C     D IS PROPORTIONAL TO DIFFERENCE IN VERTEX DISTANCES
C     FROM LINE
      CALL PLOT (1,0.,100.,1.,100.,0.,100.,1.,100.)
      CALL PLOT (99)
      CALL PLOT (90,0.,0.)
   10 IF (D.LT.0.)  GO TO 20
      X = X+1.
      Y = Y+1.
      CALL PLOT (90,X,Y)
      D = D-A
      GO TO 30
   20 X = X+1.
      CALL PLOT (90,X,Y)
      D = D+B
   30 U = U-1.
```

```
C   PROGRAM WILL TERMINATE WHEN U IS NOT GREATER
C   THAN ZERO
          IF (U.GT.0.)  GO TO 10
          CALL PLOT (98,0.,0.)
          CALL PLOT (7)
               STOP
               END
```

A line of gradient V/U would be traced terminating after U movements.

The appropriate octant is determined by a Boolean test. Pitteway begins with the equation of the curve, say AXX+RXY+BYY+GX+FY+C=0, and takes the gradient of the tangent which is -(2AX+RY+G)/(RX+2BY+F). His intention is to draw, beginning from a point on the curve and using an algorithm based on that of Bresenham, the intercepts of a succession of straight lines whose gradients are obtained by substituting the co-ordinates of the relevant grid vertex in the expression for the gradient of the tangent to the curve. The intercepts are bounded by consecutive vertical lines of the grid. When the process is completed a curve has been drawn which is a 'best fit'. To obtain his algorithm from Bresenham's, considerable modifications have to be made. The constants V and U

become $-(2AX+RY+G)$ and $(RX+2BY+F)$ respectively. These are functions of two variables and their values have to be changed after each intercept is drawn. During the tracing of the intercepts octant changes occur. These require to be detected and provision has to be made in the algorithm for modifications consequent to those changes. Also, as the original terminating condition no longer applies, a new one has to be substituted. The algorithm in the end is quite elaborate and is executed before each intercept is drawn.

3. Bresenham and Pitteway are primarily concerned with keeping processing time to a minimum. The aim in the present paper, if it can be properly described, may have been the solution of the problem by the creation of programs which depended on familiar analysis. As with Bresenham, points on curves are joined, and it is possible that the procedures so far used could be developed to advantage, influenced by his algorithm. In particular, programs for the tracing of contours of surfaces of higher degree than the second might be written.

4. Possible Developments

This paper describes the development of a program aimed at producing sets of contours of any second degree surface, when its equation is given in the Cartesian form. In the process a program was written for the contours of the plane.

These latter contours constituted a set of simple straight lines, and the former, a set of conic sections. It was the problem of programming for the tracing of these conic sections or whatever part of them lay in the plotting quadrant which provided the greatest challenge of the investigation; and it soon became clear that progress in programming for the contours of different types of surfaces would be dependent on the extent to which development had been previously done on the tracing of curves on the plane. It would therefore be well to examine such work, as described in this paper, and to consider how it might be the beginning for further progress in this direction.

Reference has been made to Pitteway's Algorithm for the tracing of a complete ellipse within the plotting quadrant, and the curves drawn in the present thesis are also of the second degree, both developed in a Cartesian setting. Hence when one thinks of further development one's thoughts are directed to a choice of two possibilities: one, of programming for the tracing of parametric curves, that is, curves where the co-ordinates are given in terms of a third parameter, such as $X = g(t)$ and $Y = f(t)$; and the other, for the contours of surfaces of higher degree than the second.

Before discussing these possibilities, it would be worth

pointing out that the programs in the paper, such as programs 7 and 9, contain within them all that is required for the tracing of curves where Y is given explicitly as a function of X, provided that the evaluation of the function does not involve operations and functions outwith the scope of the high level language and of the computer. Polynomial functions, functions involving square roots, trigonometric functions, those functions contained in the computer library and even the programmer's own functions would be allowed. This asset could be regarded as a valuable by-product of the original investigation, and its use in scientific inquiry could be helpful.

Within this context it might also be opportune to take up the reference to greater efficiency in future programming, under the influence of the Bresenham Algorithm, which was made on page 67.

When the straight line is being traced, under the control of the algorithm, only fundamental plotter steps are made and, what is important, one of two steps will always be made after each execution of the algorithm. There will be no void execution.

On the other hand, in the programs used here, to have the lines drawn plot routines, such as PLOT (90,X,Y), are called and, although the grading of increments as described should reduce

the possibility, there may be a number of ineffective calls resulting from neither of the co-ordinates, X nor Y, having changed by the minimum of ·005 necessary for a new fundamental step. Accordingly the introduction of conditional statements, so that X or Y will have changed by this minimum amount before calling a plot routine, would increase efficiency.

The use of a step by step plot routine would lead to still greater efficiency.

Parametric Curves

Returning to the first of the two possible major developments mentioned earlier, we see that in Pitteway's Algorithm the value of the gradient of the curve at each stage in its tracing determines the nature of the next geometric step. The gradient is of paramount importance and the calculation of its value a necessity. When the equation of the curve is given in parametric form, $X = g(t)$ and $Y = f(t)$, the value of its gradient is given by the formula

$$DY/DX = (DY/DT)/(DX/DT)$$

and is therefore easily obtained when functions g and f are differentiable. (In certain circumstances numerical methods would be suitable.)

It would therefore appear that, whatever difficulties might otherwise arise, this algorithm does lend itself to the

development of curve tracing from parametric equations.

On the other hand, the fact that the procedures in this thesis are based on a Cartesian equation might suggest that adaptation of the programs to parametric equations would be involved or impossible. However, on considering the matter in greater detail, it would be accepted that corresponding values of X and Y are more easily obtained from the parametric equations and that, as already shown in the previous paragraph, the calculation of the gradient is quite straightforward; but beyond that there exist two major difficulties.

The first arises from the scanning technique which was adopted to detect any part of the curve which might lie within the plotting quadrant. Here the argument ranges from XMIN to XMAX by a variety of increments, and when this range has been completed one can be sure of having located all that is relevant to that part of the branch which is about to be drawn. The determination of the corresponding range or ranges in the parametric situation would generally be involved. However, in cases where the parametric limits (TMIN,TMAX) are given and the plotting area defined, this criticism would not apply.

But it is the second difficulty which would be more troublesome. It arises in the following way. In the development of the accurate plotting of contours success depended on the possibility of linking an acceptable delta Y, through the

gradient of the curve, with an appropriate increment in X. Thus was obtained what might be described as a mutually dependent ordered trio, which resulted in the system of X-increments, so much a part of the structure of the programs. The introduction of a third parameter into the equations does not destroy this relationship, which can be established through the new differential, but the relevant part of the programming becomes elaborate, and it would appear that the logical development of present programs lies more with the polynomial surface and its associated Cartesian equation. An approach to this latter problem is described in the remaining paragraphs.

Contours of Polynomial Surfaces
of Higher Degree than the Second

For these surfaces the form of the equation is $Z = f(X,Y)$ where f is a polynomial function. To obtain the equation of a particular contour, Z becomes a constant, K say, giving $f(X,Y) = K$.

The problem now would be to write a program to give the complete contour configuration, such that the structure of the program would be quite general. In this paper the problem has been solved for surfaces of the second degree by the writing of program 10. A brief examination of this program shows that the overall pattern with reference to plotting routines, exits to

these routines, and to selecting increments according to gradient value, is suitable; that the method of obtaining the gradient through partial differentiation will still be effective, but that the problem of obtaining corresponding values of X and Y will certainly be new and probably difficult. For this purpose, in program 10, use was made of the formula for the solution of quadratic equations; unfortunately there is no corresponding general formula and a new way would need to be sought.

The relevant equation has already been given as $f(X,Y) = K$ and can be considered in the form $g(X,Y) = 0$. In the scanning process X assumes a series of known values and the corresponding values of Y have to be calculated. Here these values of Y are the real roots of the equations obtained by substituting the values of X in $g(X,Y) = 0$; and so the problem is reduced to that of solving an equation in one unknown, say $F(Y) = 0$, where F is a polynomial.

The mathematician might be tempted to link the investigation completely with the Theory of Equation, but, in a computational environment, there will be a subroutine which will supply the real roots of such an equation, when called. The subroutine might be named POLYR and the adoption of this routine would be a major change in the programs of this paper.

As the functions and curves are continuous, each complete branch of the contour will be determined by a particular set of

roots, which in turn corresponds to one of the real roots returned by POLYR on a particular call. Further, each real root from POLYR belongs to a particular set, and might be used to identify the set. Some system of identification of sets (branches) is required, and this might be achieved by arranging the roots according to size, as each set of roots maintains its rank relative to another. With some such system it is tempting to proceed in the old way, assigning a value for X, calculating the Y, storing and plotting. The contours could certainly be completed, but the excessive demand on the computer would be prohibitive.

Fortunately it appears that there may be another way, which is straightforward, economical and open to extension. The principles underlying this method can be demonstrated, without loss of generality, with reference to the cubic equation in the following way.

As before the functions and curves are continuous, excluding infinity. When called, the subroutine would return one or three values.

Initially a call would be made, an identifying routine executed and the smallest root might be selected. Thus a point on an identified branch could be plotted. The gradient could then be calculated from the formula, the appropriate X-increment chosen and, using Taylor's Series, the next Y determined (I would

expect the first approximation to suffice, but this could be investigated). This sequence would be continued until the gradient passed through the infinite. It would then be necessary to apply an elementary test for a turning point as against a point of inflexion. If a turning point had been reached, the appropriate delta X would be subtracted, and a second call would be made on POLYR. The value, one rank higher than that associated with the previous set, would be taken as a starting point for the next branch. The previous procedures would be continued until the next infinite gradient was met, when there would be a third call on the subroutine and a repetition of the sequence. Clearly the method does not depend on the order of the polynomial and is quite general.

Many of the techniques of the programs of this paper would still apply. What is new is the use of POLYR, the need for a method of identifying the set of roots which is associated with each branch, the calculation of Y from Taylor's Series and the infinite gradient marking the beginning and end of branches.

Research might begin with the surface of second degree and then proceed to those of higher degree. Further attention would need to be given to such matters as scanning techniques, turning points and approximations, but it does seem to be a way on which one could embark with some degree of confidence.

References

(1) Computer Journal Volume 10 1967-68
M.L.V. Pitteway Algorithm for drawing ellipses or hyperbolae with a digital plotter.

(2) I.B.M. Systems Journal 1965 Volume 4. Number 1.
J.E. Bresenham Algorithm for computer control of a digital plotter.