NRL Memorandum Report 2335

A "French Curve" Computer Plotting Subroutine

LAMONT V. BLAKE

Radar Geophysics Branch
Radar Division

September 1971

NAVAL RESEARCH LABORATORY
Washington, D.C.

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) Naval Research Laboratory Washington, D. C. 20390	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP ---

3. REPORT TITLE

A "FRENCH CURVE" COMPUTER PLOTTING SUBROUTINE

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

A final report on one phase of the problem; work is continuing

5. AUTHOR(S) (First name, middle initial, last name)

Lamont V. Blake

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
September 1971	36	0

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
NRL Problem 53R02-55 b. PROJECT NO. RF-151-402-4011 c. d.	NRL Memorandum Report 2335 9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)

10. DISTRIBUTION STATEMENT

Approved for public release; distribution unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	Department of the Navy (Office of Naval Research) Washington, D.C. 20360

13. ABSTRACT

A computer subroutine has been developed which will result in a smooth-curve plot through a given set of data points, similar to the plot that a draftsman would produce using a French curve. The method is especially useful when it is desired to plot a smooth curve through computed points and the computation of each point is fairly lengthy, so that excessive computer time would be used if a smooth curve were plotted by direct computation of an adequate number of points. It can also be used to plot a curve through experimental data points if the measurements are sufficiently precise -- that is, if there is not a significant amount of random error in the measurements. The method produces a curve that exactly passes through each of the given data points, rather than giving a "least-squares best fit" curve. The plotting algorithm is described and examples of its use are given.

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Digital computer Machine plotting						

CONTENTS

ABSTRACT

A computer subroutine has been developed which will result in a smooth-curve plot through a given set of data points, similar to the plot that a draftsman would produce using a French curve. The method is especially useful when it is desired to plot a smooth curve through computed points and the computation of each point is fairly lengthy, so that excessive computer time would be used if a smooth curve were plotted by direct computation of an adequate number of points. It can also be used to plot a curve through experimental data points if the measurements are sufficiently precise -- that is, if there is not a significant amount of random error in the measurements. The method produces a curve that exactly passes through each of the given data points, rather than giving a "least-squares best fit" curve. The plotting algorithm is described and examples of its use are given.

PROBLEM STATUS

This is a final report on one phase of the problem; work is continuing on other phases.

AUTHORIZATION

NRL Problem R02-55
Project RF-151-402-4011

A "FRENCH CURVE" COMPUTER PLOTTING SUBROUTINE

INTRODUCTION

The usual method of plotting a smooth curve with a digital plotter is to compute points sufficiently close together so that the plotter pen traces an apparently curving line, even though it actually moves in short straight-line segments. This is illustrated by Fig. 1, in which points on five semicircles have been plotted with different spacings of the points. In the lowest semicircle, the points are separated by $\pi/8$ radians ($22\frac{1}{2}^{\circ}$). In each successive higher semicircle, the plotted points are separated by, respectively, $\pi/16$, $\pi/32$, $\pi/64$, and $\pi/128$ radians. It is evident that the first and second "semicircles" appear to consist of 8 and 16 straight-line segments, as they actually do. The third, fourth, and fifth semicircles, however, appear to be smoothly curving, although in actuality they consist of, respectively, 32, 64, and 128 straight-line segments. Thus, the longest straight-line segment that can be plotted and still give the appearance of a smooth curve is one that is approximately $\pi/32$ radians of a circular arc, or about 0.1 of the radius of curvature of any curve.*

In other words, if a sufficient number of points of a curve are plotted with a digital plotter so that the intervals are equal to or smaller than 0.1 times the radius of curvature, the resulting plot will appear to be a smooth curve. If the computation of each point is very lengthy, however, to compute an adequate number of points may require an excessive amount of computer time.

When a situation of this nature existed before computers and machine plotters were commonly available, the procedure employed was to compute enough points to define the curve approximately and then connect the points with a smooth curve, using the draftsman's "French curve" as a plotting aid. The purpose of this report is to describe a computer algorithm which in a sense mimics this technique. That is, if a relatively limited number of points on a curve is computed and the coordinates of these points are input to a subroutine based on this algorithm, the result will be a machine-plotted smooth curve similar to one that would have been plotted by using a French curve. Moreover, the resulting curve will exactly pass through each one of the computed points.

Figures 2, 3, and 4 illustrate the idea of this procedure. In Fig. 2, 17 equally spaced points on a sine curve at intervals of $\pi/8$ radians were computed, and the resulting "curve" was plotted in the usual way with a machine plotter, so that the 17 points are simply connected by straight-line segments. The computed points are identified by the small circles. It is apparent that the result is not a smooth curve. In Fig. 3, 101 equally

---

*There is probably also some dependence on the absolute lengths of the segments, but the angular effect is believed to be the dominant factor.

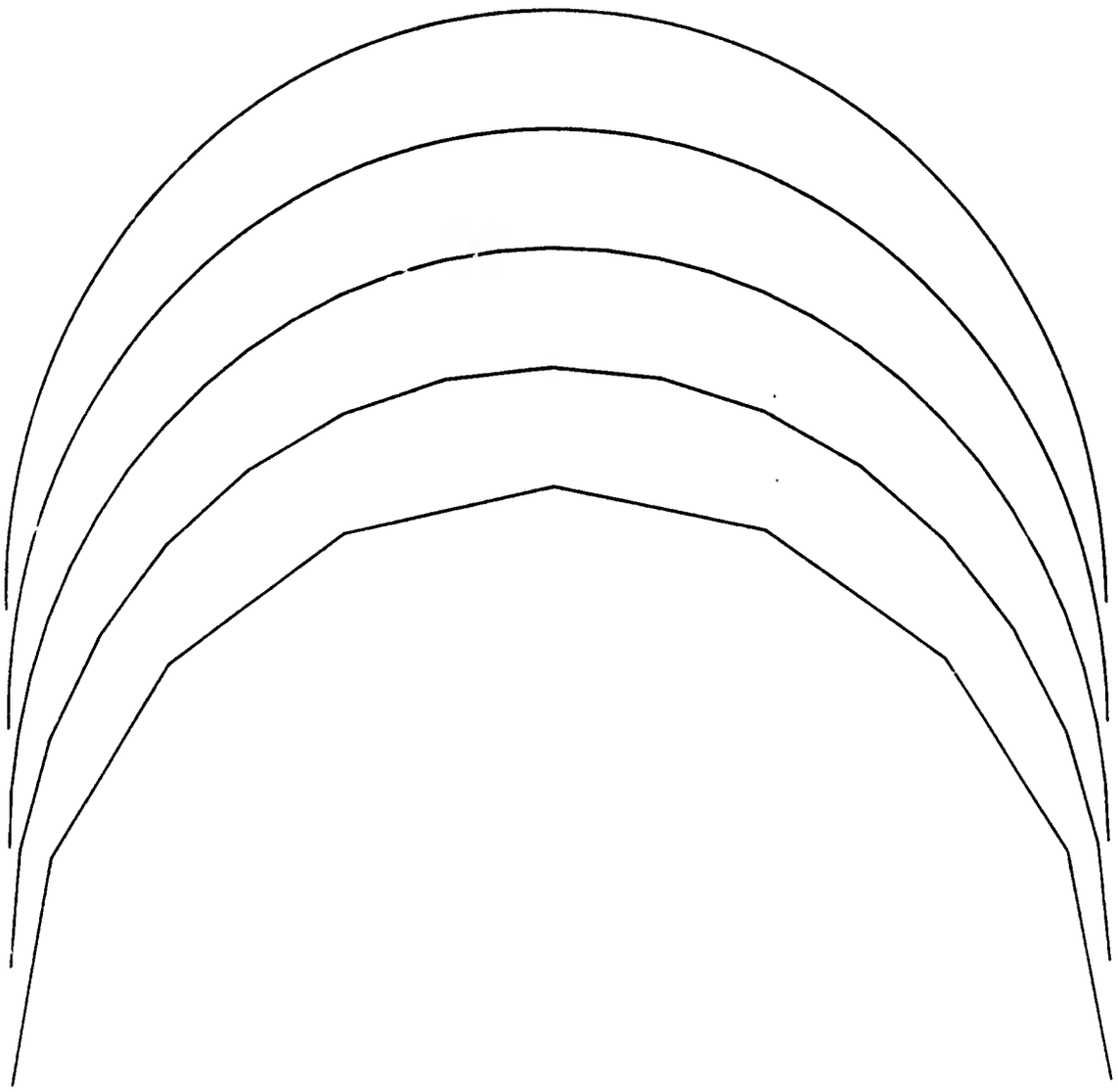

Fig. 1. Computed points on a semicircle connected by straight line segments. The successive plots, from bottom to top, have 9, 17, 33, 65, and 129 equispaced points, so that the straight lines subtend arcs of $\pi/8$, $\pi/16$, $\pi/32$, $\pi/64$, and $\pi/128$ radians.

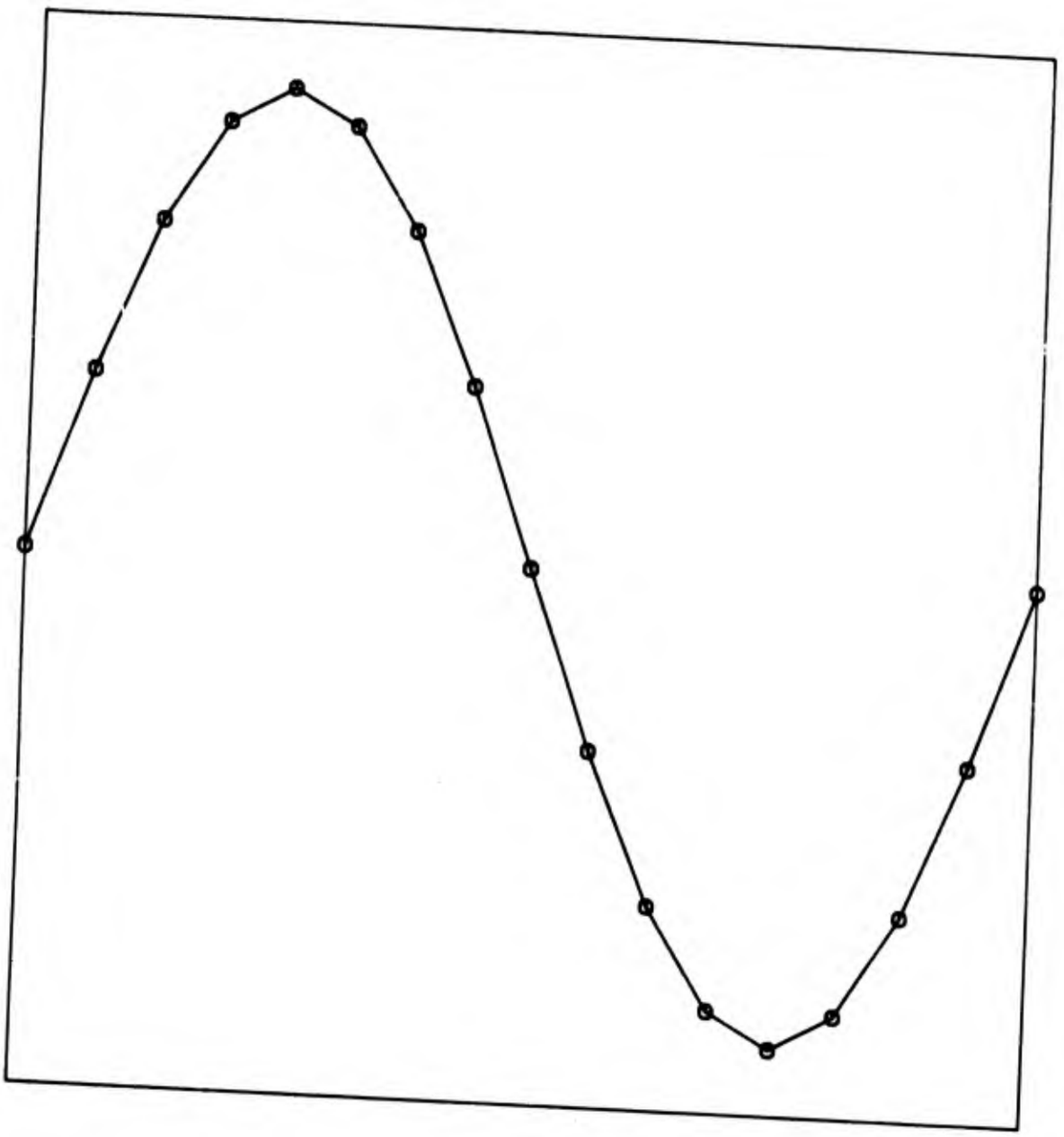

Fig. 2. Computed points on a sine curve connected by 16 straight line segments. The 17 computed points are identified by the small circles. Their separation on the abscissa scale is $\pi/8$ radians.

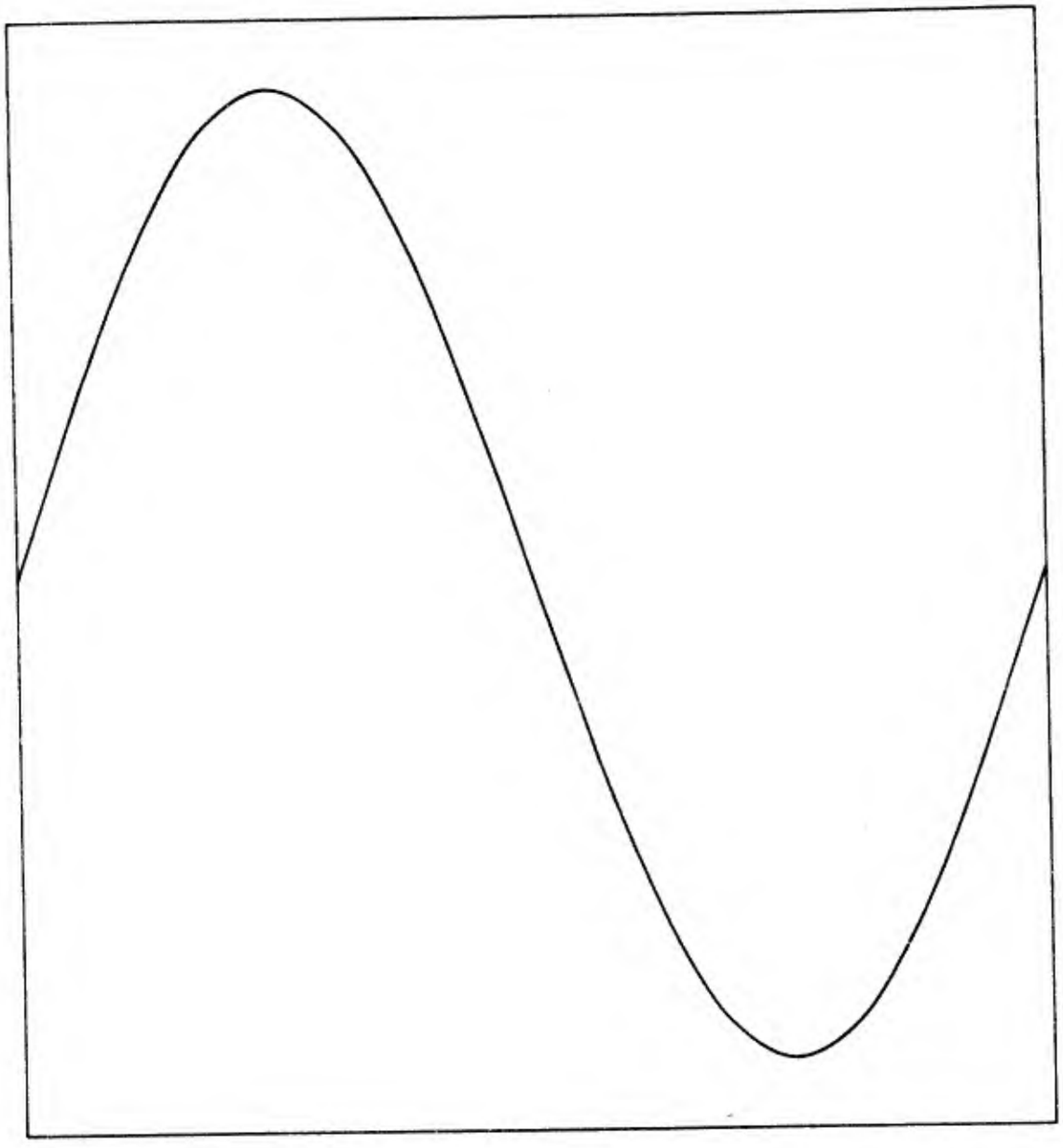

Fig. 3. Computed points on a sine curve connected by 100 straight line segments. The points are separated on the abscissa scale by $\pi/50$ radians.

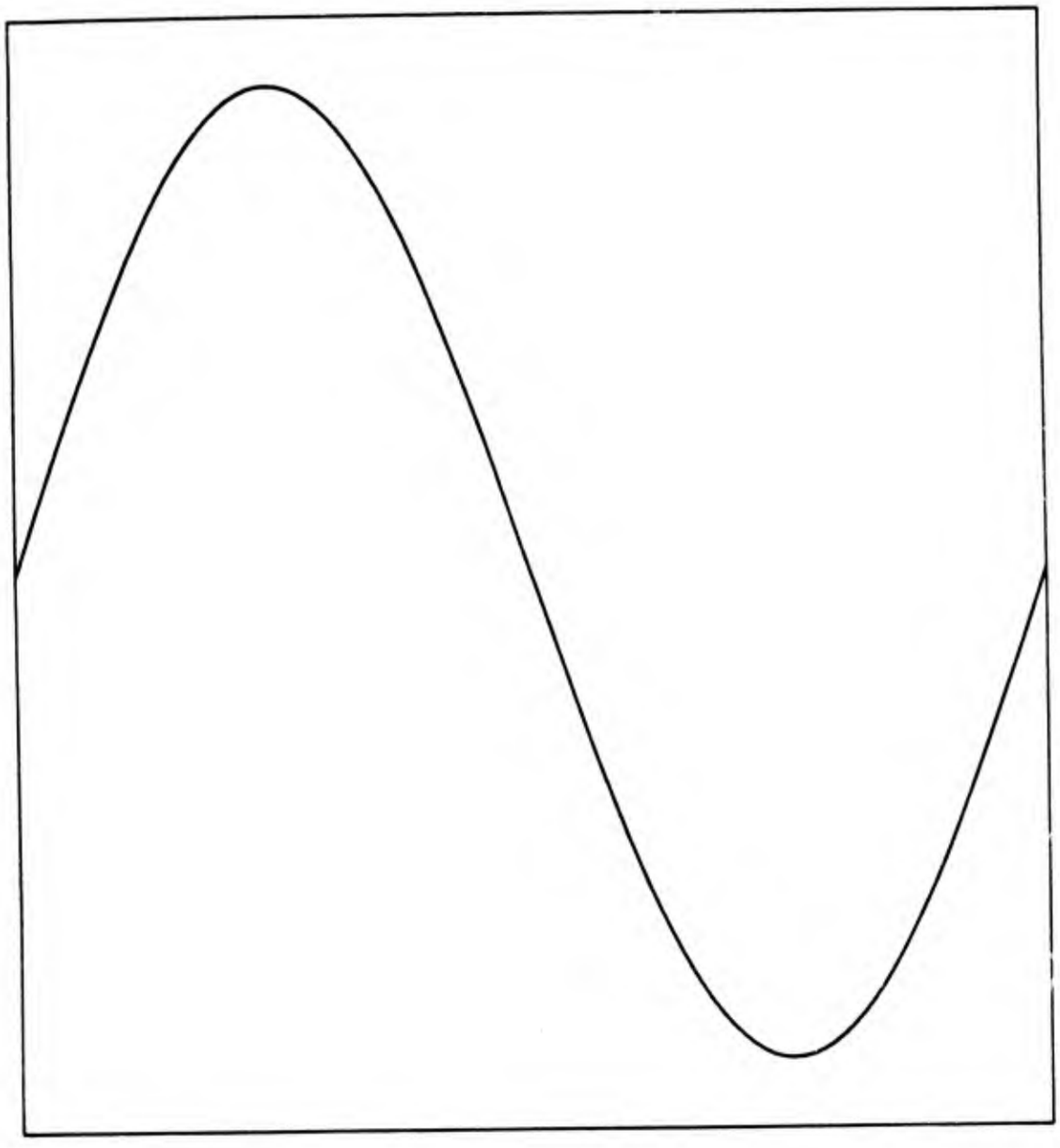

Fig. 4. Sine curve plotted using 17 computed points, as in Fig. 2, but with additional interpolated points obtained by the "French curve" method.

spaced points of the same curve were computed and plotted in exactly the same way (100 intervals of $.02\pi$ radians each). The result is a smooth sine curve. In this case, the computation of 101 points is not a lengthy procedure (it takes only about 12 milliseconds on the NRL CDC-3800 computer), and so there is no real need to employ any "French curve" technique. However, in Fig. 4 the curve was plotted using the technique to be described, using the same 17 sine-curve points as in Fig. 2. As can be seen, it is difficult to see any difference between Figs. 3 and 4. Actually, the correspondence is not perfect, but it is adequate for most purposes; and it can be made as good as desired by increasing the number of computed points supplied to the French-curve subroutine.

This curve plotting technique should not be confused with methods of plotting a "best fit" curve through a set of experimental data points that have random deviations from the "true" function, due to errors of measurement. The usual method of plotting in that case is to find the polynomial that gives the best fit to the points in the least-squares sense. The resulting curve does not necessarily pass exactly through any of the data points. The method being discussed here is applicable, rather, to plotting a curve through calculated points so that the resulting curve passes exactly through each point. The purpose of the method is to be able to plot a smooth curve, using a digital plotter, when the available number of calculated points is too few to provide a smooth curve otherwise. Obviously, some judgment is required in its use, since in many instances the preferred method will be simply to calculate more points using the exact equations or function. The method to be described is useful when the exact calculation is so lengthy that an inordinate amount of computer time would be required for the direct method. This situation arises frequently in calculations of complex physical phenomena. The method can also be used with experimentally measured data points if they define a smooth curve with sufficient accuracy -- that is, if the random measurement errors are sufficiently small.

ALGORITHM

If a set of n points is to be connected by a smooth curve, one obvious method is to find the $(n-1)^{th}$-degree polynomial that exactly passes through these points and use it to compute the coordinates of, say, 10 n points for the plotter. However, even with a digital computer, this is not very practical if n is larger than about 20. Moreover, the resulting curve may not be at all what is desired; even though it passes exactly through each point, it may swing wildly above and below the curve that a draftsman would plot by the French curve technique, or which would be sketched "free hand." However, the method that has been devised does not have this difficulty.

This method will be described for a set of n points through which the curve is to be drawn, with coordinates (x_1, y_1), (x_2, y_2), $x_3, y_3)$,... $x_n, y_n)$. For each adjacent pair of these points, a set of four simultaneous equations is written. Two of these equations are of the form:

$$y_i = a_1 + a_2 x_i + a_3 x_i^2 + a_4 x_i^3 \tag{1}$$

and the other two are of the form:

$$\left(\frac{dy}{dx}\right)_{x = x_i} = a_2 + 2a_3 x_i + 3a_4 x_i^2 \tag{2}$$

For example, for plotting the curve between the 7th and 8th points of the set (i = 7,8), the four equations are solved for the coefficients a_1, a_2, a_3, and a_4 in the usual manner; thus:

$$a_1 = \frac{\begin{vmatrix} y_7 & x_7 & x_7^2 & x_7^3 \\ y_8 & x_8 & x_8^2 & x_8^3 \\ \left(\frac{dy}{dx}\right)_{x = x_7} & 1 & 2x_7 & 3x_7^2 \\ \left(\frac{dy}{dx}\right)_{x = x_8} & 1 & 2x_8 & 3x_8^2 \end{vmatrix}}{\begin{vmatrix} 1 & x_7 & x_7^2 & x_7^3 \\ 1 & x_8 & x_8^2 & x_8^3 \\ 0 & 1 & 2x_7 & 3x_7^2 \\ 0 & 1 & 2x_8 & 3x_8^2 \end{vmatrix}} \tag{3}$$

and similarly for a_2, a_3, and a_4.

The coefficients thus found are used in the equation

$$y = a_1 + a_2 x + a_3 x^2 + a_4 x^3 \tag{4}$$

to plot the curve y(x) between the points x_7 and x_8. By repeating

this process for all the pairs of points, the entire curve is plotted. If the same slopes are used at the common points in each successive operation of this type, the successive curve segments will have slope continuity; and, of course, the complete curve will exactly pass through all of the specified points.

The one aspect of this procedure not yet explained is how to find the slopes $\left(\frac{dy}{dx}\right)_{x = x_i}$. Several "reasonable" methods were tried, and the one that worked best will now be described.

Figure 5 shows a set of three successive points of the set through which a curve is to be plotted, designated (x_{i-1}, y_{i-1}), (x_i, y_i), and (x_{i+1}, y_{i+1}). The two lines joining these three points are drawn; their slopes are tan α_1 and tan α_2. The slope used in Eq. (2) at point (x_i,y_i) is then tan α, where α is the average of the angles α_1 and α_2; that is,

$$\alpha = (\alpha_1 + \alpha_2)/2. \tag{5}$$

The slope to be used at (x_i,y_i) in the previously described procedure is then:

$$\left(\frac{dy}{dx}\right)_{x = x_i} = \tan \alpha. \tag{6}$$

The angles α_1 and α_2 are of course given by

$$\alpha_1 = \tan^{-1}\left[\frac{y_i - y_{i-1}}{x_i - x_{i-1}}\right], \tag{7}$$

$$\alpha_2 = \tan^{-1}\left[\frac{y_{i+1} - y_i}{x_{i+1} - x_1}\right] \tag{8}$$

It is to be noted that the slope <u>angles</u> are averaged, rather than the slopes themselves, to find the slope to be used in the simultaneous equations. This is necessary because, of course, the slopes are very nonlinearly related to the angles, and averaging the slopes would, in effect, give a greater weight to a steep slope than to a shallow one.

Although this type of weighting would obviously not be desirable, some other weighting schemes were considered, based on the relative distances separating the pairs of points. However, for general use, the simple averaging of the angles seemed to give the best results.

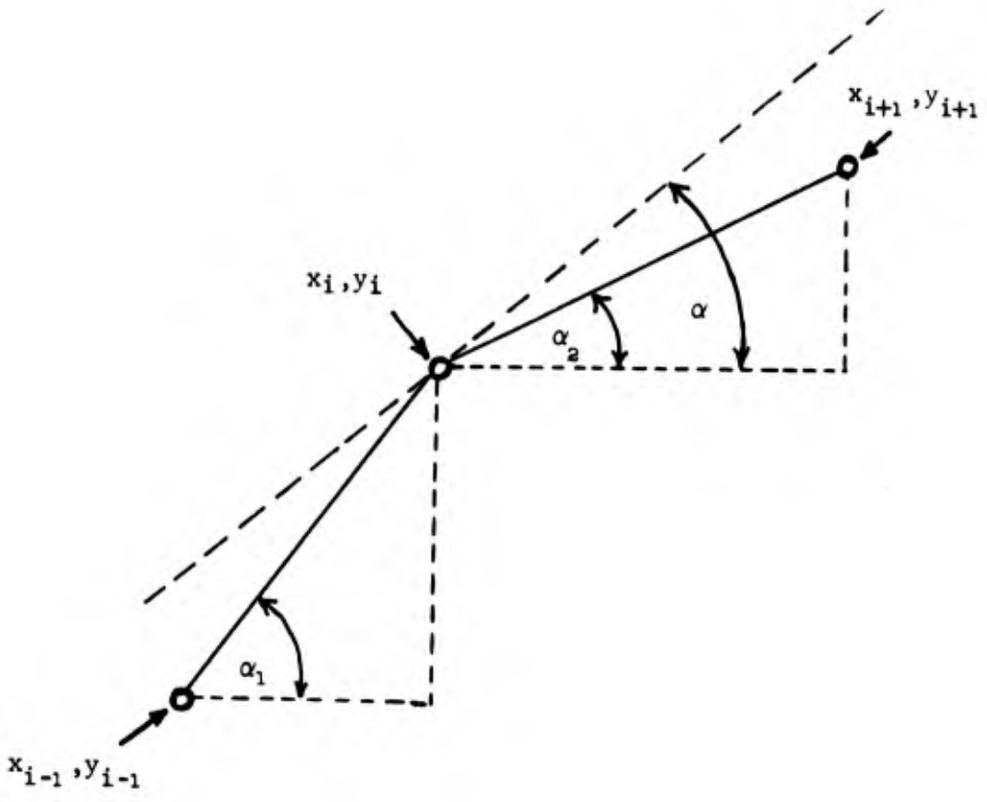

Fig. 5. Three successive data points, showing how the slope of the curve at the center point is calculated.

The procedure described can be used for plotting all except the first and last segments of the curve. The slope at the first and last points cannot, of course, be determined in this way. Therefore, a slightly different procedure is used in plotting the first and last segments of each curve. Here only three simultaneous equations are available. For example, for the first segment there are two equations of the type of Eq. (1), with i = 1 and i = 2, and one of the type of Eq. (2), with i = 2. These are solved for the three coefficients of a quadratic equation

$$y = a_1 + a_2 x + a_3 x^2, \tag{9}$$

which is used as the plotting equation in the first interval of the curve. The same procedure is used in the last interval, except that the two equations of the type of Eq. (1) are for i = n-1 and i = n and the equation of the type of Eq. (2) is for i = n-1.

It is evident that this procedure is a fairly good mathematical representation of the use of a French curve in drafting. When the coefficients of the interpolating equation have been found, a sufficient number of points can be calculated and plotted to produce a smooth curve. The question then arises, how many such points should be plotted? The answer is based on the information derived from Fig. 1. The radius of curvature is calculated in the interval to be plotted at 3 points -- the end points and the midpoint, from the formula:

$$R_c = \frac{\left[1 + \left(\frac{dy}{dx}\right)^2\right]^{3/2}}{d^2y/dx^2} \tag{10}$$

and the additional formulas:

$$\left(\frac{dy}{dx}\right)_{x = x_i} = a_2 + 2a_3 x_i + 3a_4 x_i^2 \tag{11}$$

$$\left(\frac{d^2y}{dx^2}\right)_{x = x_i} = 2a_3 + 6a_4 x_i \tag{12}$$

Then the interval δs desired between plotted points is set equal to 0.1 times the smallest of the three calculated radii of the curvature. Finally, this interval δs is converted to an x-axis increment δx by

the formula:

$$\delta x = \frac{\delta s}{\sqrt{1 + (dy/dx)^2}} \tag{13}$$

with dy/dx evaluated at the same point at which the smallest radius of curvature was calculated. The actual calculation of the points to be plotted is then done in a DØ loop in which x is incremented by δx the correct number of times to plot through the interval between the points, e.g., from x_7 to x_8 in the example considered in Eq. (3).

COMPARISON WITH SOME ALTERNATIVE METHODS

The method that has been described was the result of an evolutionary process in which a number of other ideas were tried and discarded, over a period of several years. Some of these ideas worked fairly well in many applications, but failed in certain special cases. A compairson of two other methods with the method just described will now be made for a special case which reveals the shortcomings of the other methods. This comparison will also illustrate some of the features of the recommended method, and some techniques for its most efficient use.

The plotting problem to be considered is illustrated by Fig. 6, in which 8 points with equal x-axis separations are designated by the small circles; they are connected by straight lines. The problem is to connect these 8 points by a smooth curve -- one without sharp corners. The first 4 points all have the same y value, say y_1, while the last 4 points are all of another y value, say y_2, with $y_2 > y_1$.

Figure 7 shows the result of fitting a single $7^{\underline{th}}$-degree polynomial to these 8 points, illustrating the "wild swings" that result.

Figure 8 was plotted through the same 8 points using a method which was tried and discarded in the process of arriving at the method finally adopted. It has some features in common with the $(n-1)^{\underline{th}}$-degree-polynomial method (for plotting a curve through n points), and it also has some features in common with the French-curve method finally adopted. In this method the points are taken in sets of three at a time, and a system of four simultaneous equations is solved for the four coefficients of a cubic interpolating equation. Three of these equations are position equations, using the coordinates of the three points. The fourth equation is a slope equation to insure that the plotted curve

segment will have continuity of slope with the preceding segment. The coefficients thus found are used for plotting only between the first pair of these three points, and for finding the slope to be used, via Eq. (2), in the set of simultaneous equations for the next set of three points. The essential difference between this method and the method finally adopted is that the slopes in this method are determined by the projection of the interpolating polynomial through the next pair of given points, rather than by the angle-averaging method described in the next section. As is seen, this works well enough in the left-hand portion of the curve, but it results in appreciable "overshoot" when there is an abrupt change from a steep slope to a shallow slope in the straight-line plot (Fig. 6).

Figure 9 is the result of fitting this same set of points with a curve using the average-angle (French curve) method which has been described. As is seen, the curve is well-behaved, with only a slight deviation from the desired trajectory at the corners of the straight-line plot.

A method of minimizing this slight deviation is shown in Figs. 10-13. These plots correspond to those of Figs. 6-9, respectively, except that the initial set of points has been slightly modified, as shown by Fig. 10. Instead of 8 points equally spaced along the x axis, the inner 6 points are now clustered around the corners of the curve so as to define the desired behavior somewhat more closely. As seen by Fig. 11, the $7^{\underline{th}}$-degree-polynomial fit is even wilder than before (Fig. 7). In fact, the entire curve annot be shown with a reasonable amount of scale reduction; it is "off scale" by a large amount. (The rectangular border in this plot was, before reduction, the same size as the borders in the other figures.) The method of projecting a cubic equation through sets of 3 points, Fig. 12, also works less well here, with even more pronounced overshoot. But the French curve method, Fig. 13, works very well. The dip and overshoot at the corners of the curve are minimized.

The curves of Figs. 6, 7, 10, and 11 exhibit a noticeable amount of unsmoothness in the sharply curving regions. In these plots, the technique of adjusting the number of points plotted per interval by calculating the radius of curvature was not followed; instead, an arbitrary number of points (10 per interval) was plotted, and this number turned out to be inadequate. However, this fact does not affect the comparisons being made; therefore, these curves were not replotted with a greater number of plotted points, to make them look smooth. Figures 9 and 13, in which the radius-of-curvature technique was used, are seen to be adequately smooth in appearance.

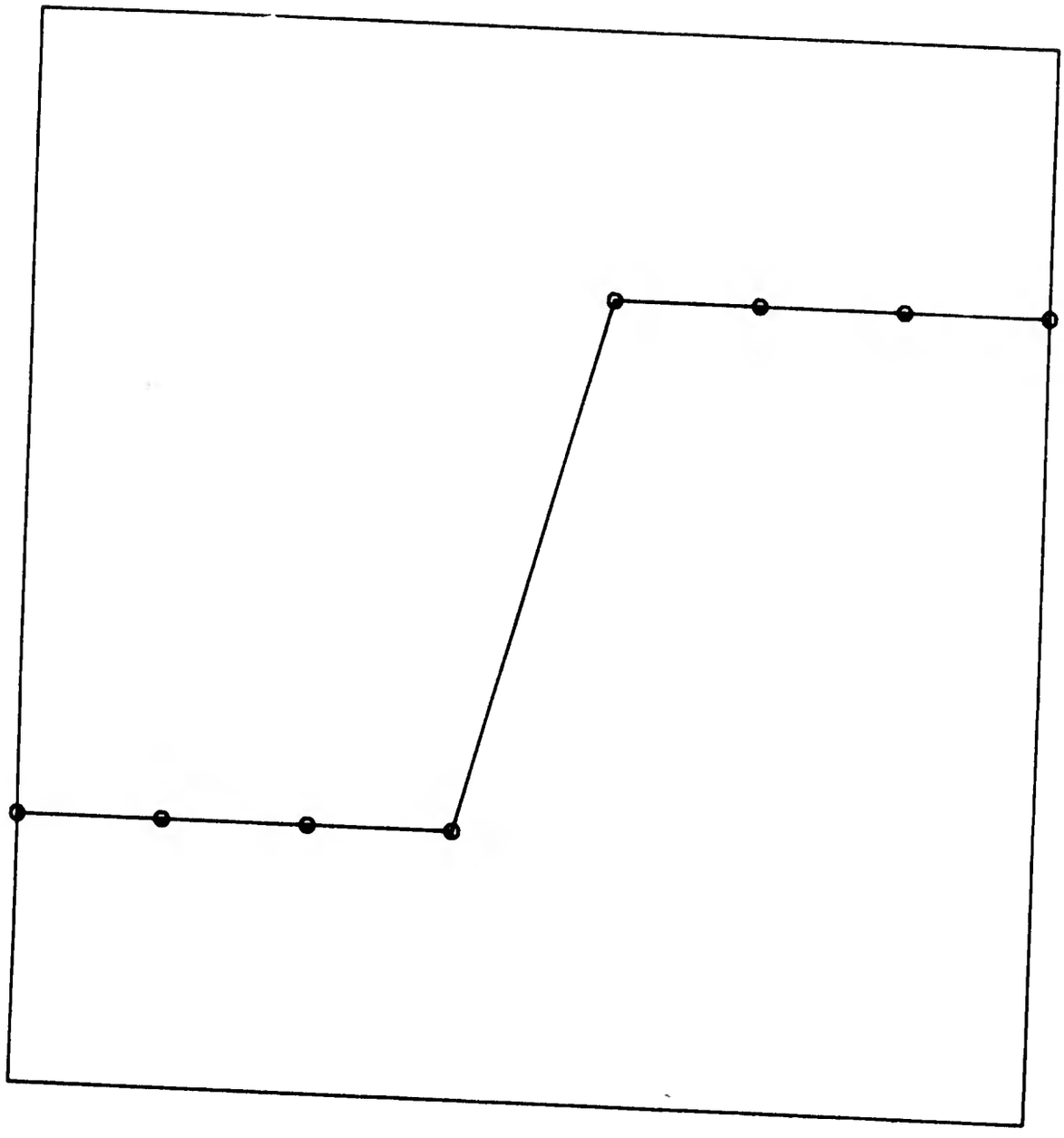

Fig. 6. Plotting problem used to illustrate difficulties encountered with some methods. The 8 given data points, connected here by straight lines, are identified by the small circles. The points are equi-spaced on the abscissa (x) axis; the first four points have a constant y value, and the remaining four points have another constant y value.

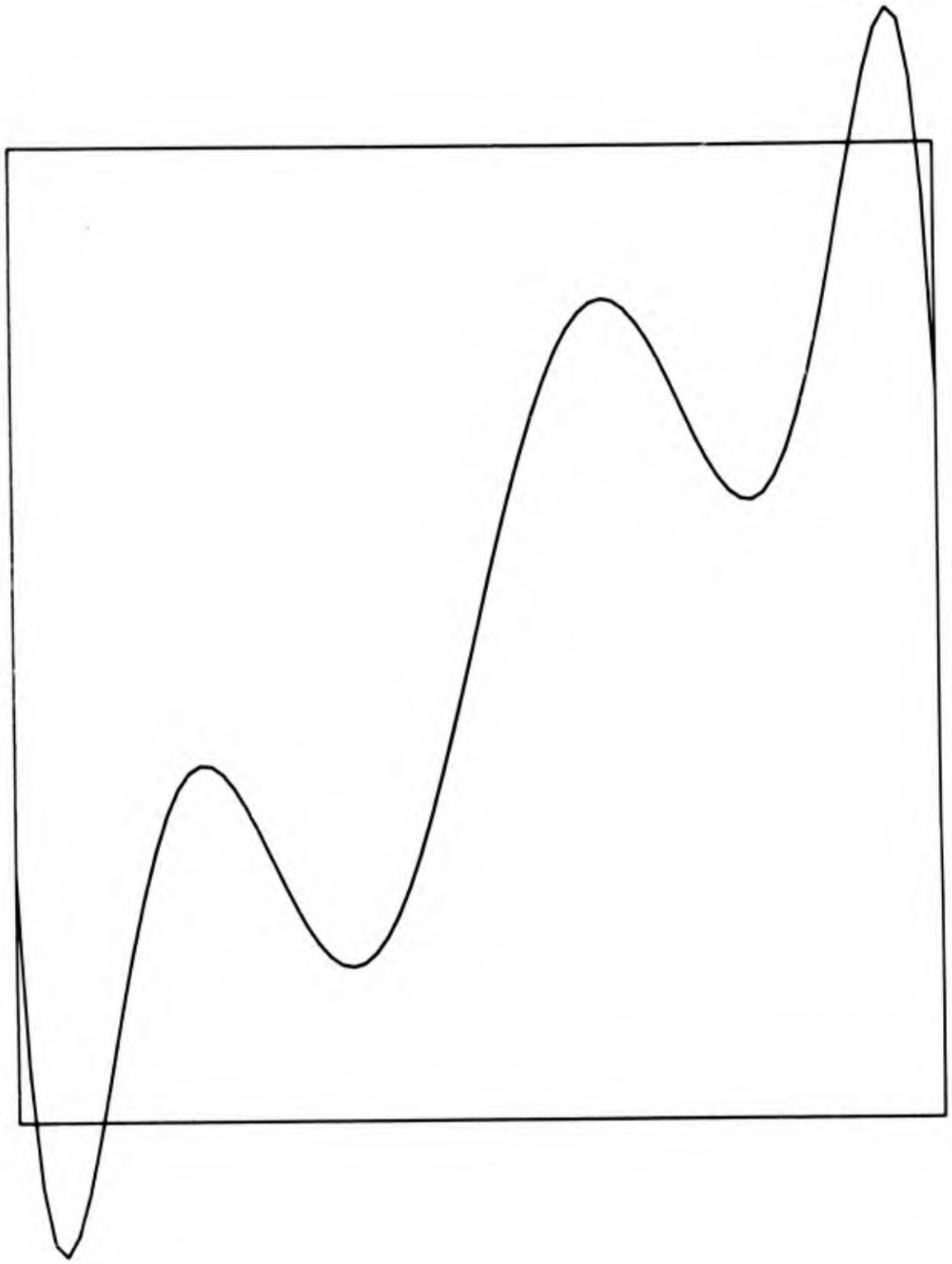

Fig. 7. The 8 points shown in Fig. 6 connected by a 7th-degree-polynomial curve.

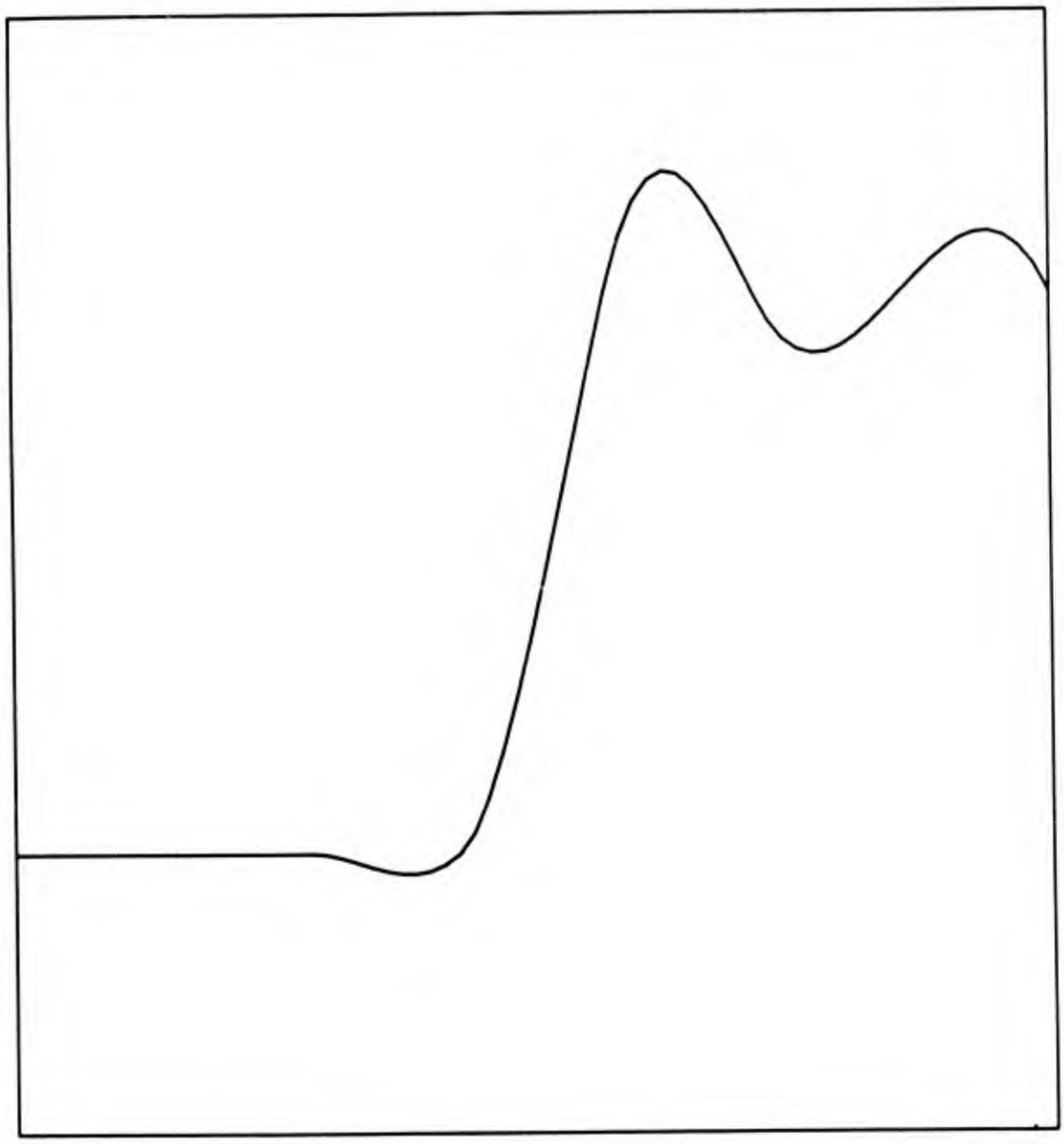

Fig. 8. The 8 points shown in Fig. 6 connected by a modification of the finally adopted French curve method.

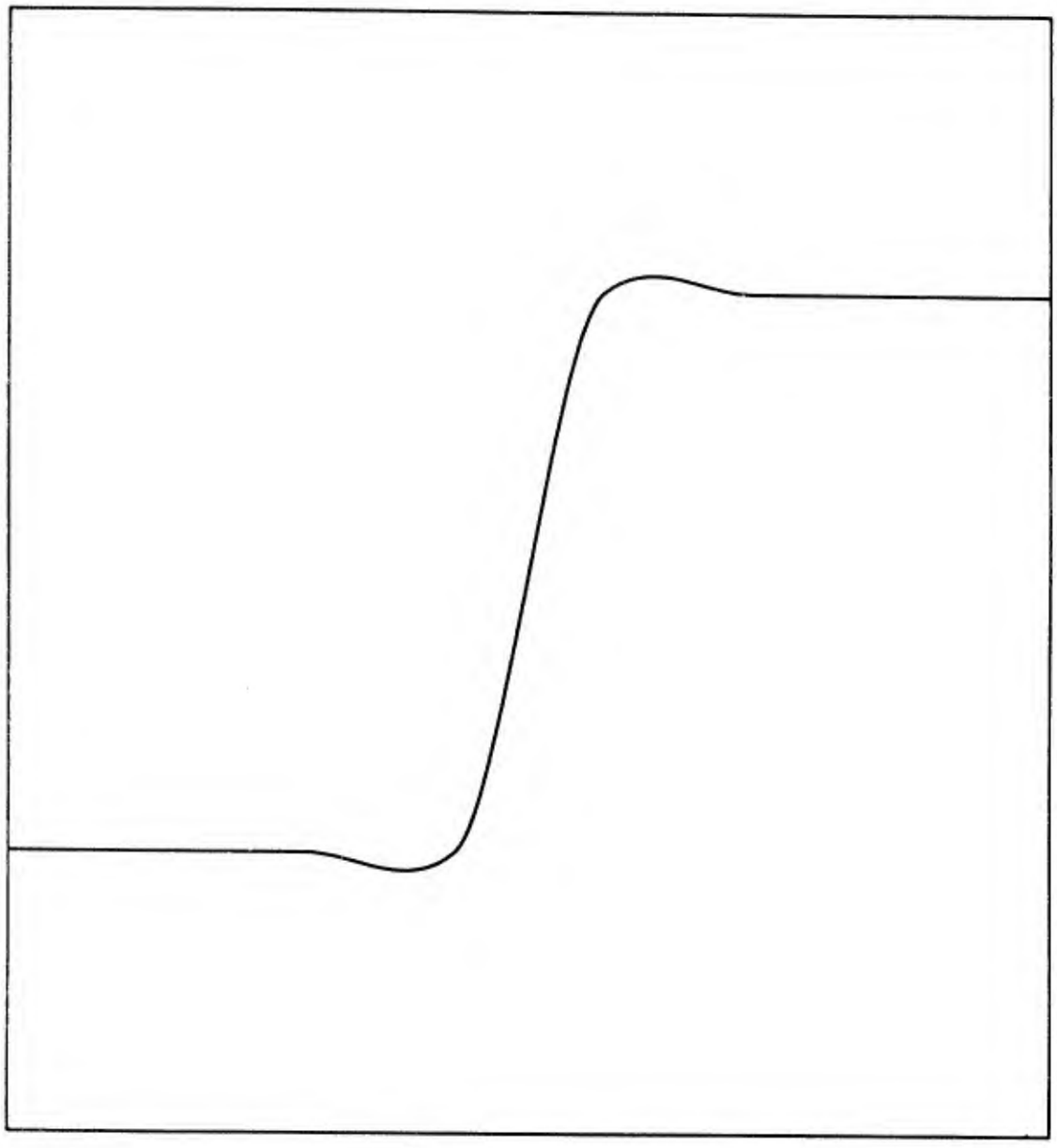

Fig. 9. The 8 points shown in Fig. 6 connected using the French curve method.

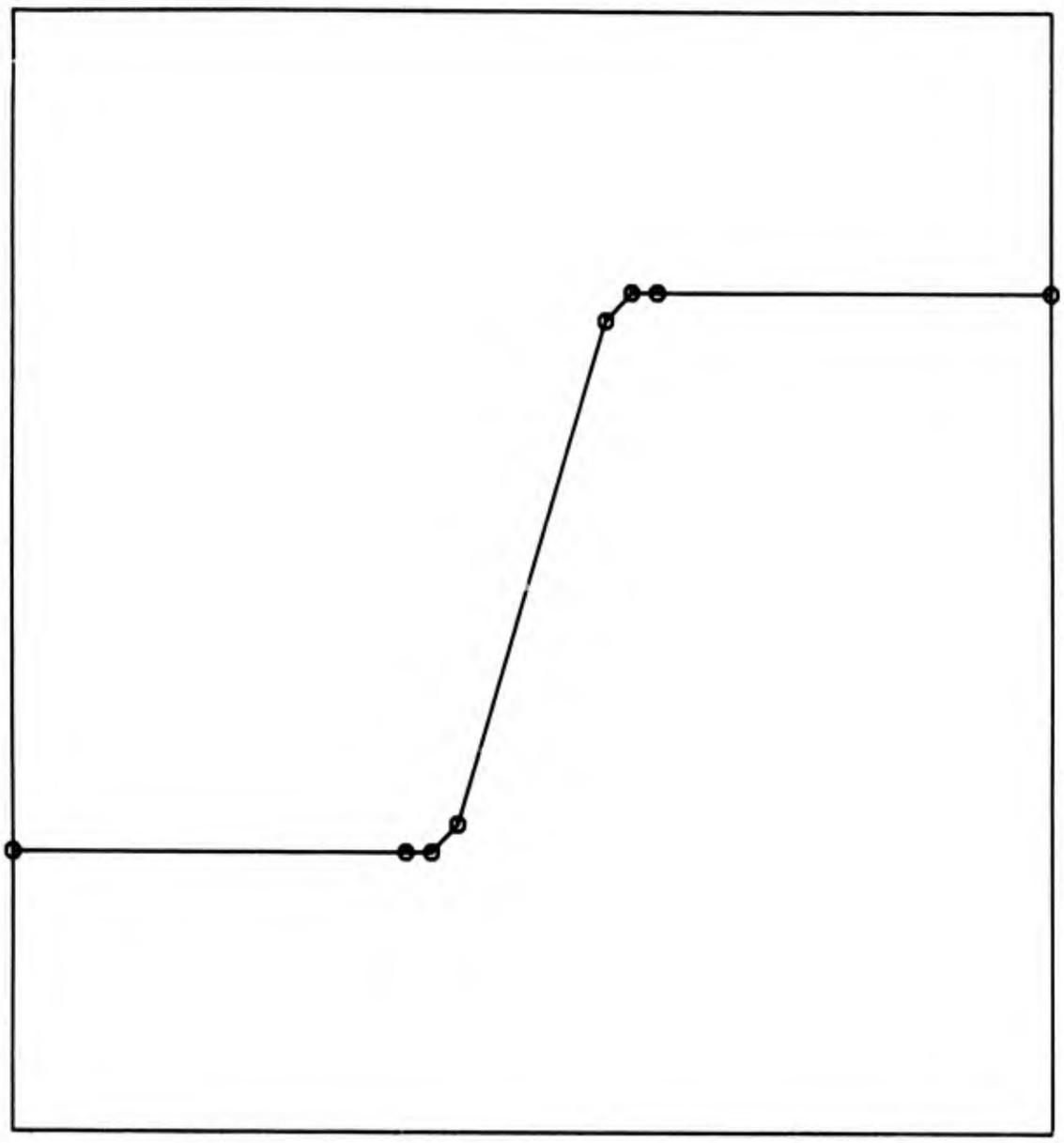

Fig. 10. A modification of the plotting problem of Fig. 6, in which the interior points are clustered at the turning points instead of being equispaced on the x axis.

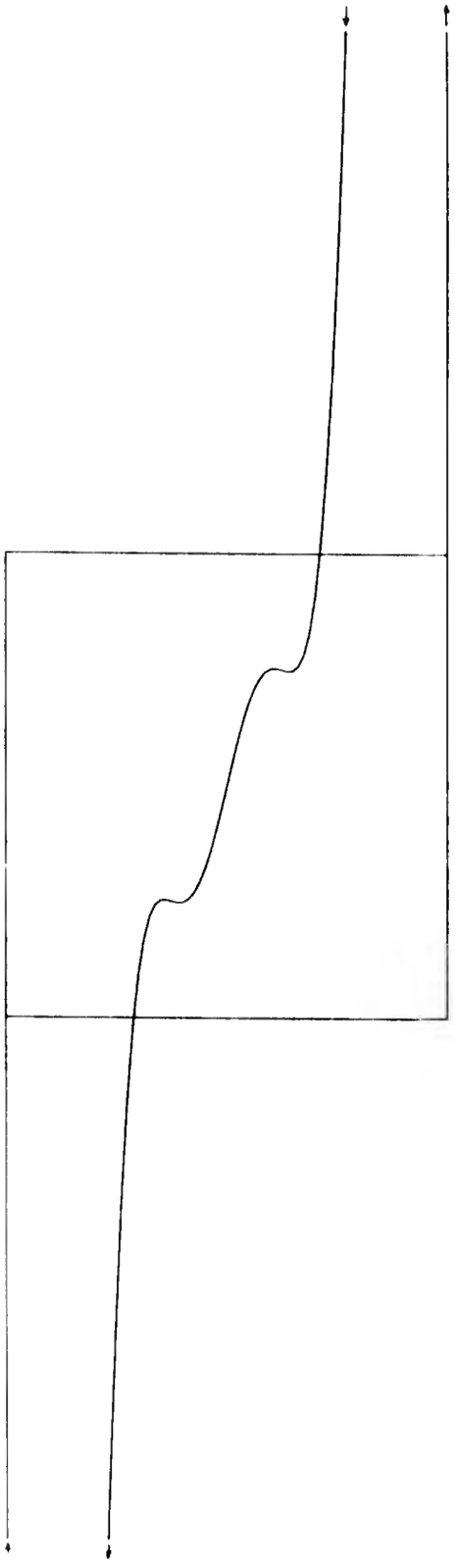

Fig. 11. The 8 points of Fig. 10 connected by a $7^{\underline{th}}$-degree-polynomial curve. Because of the wild off-scale swings of the curve, the plot is shown in reduced size, but even so the entire curve is not included.

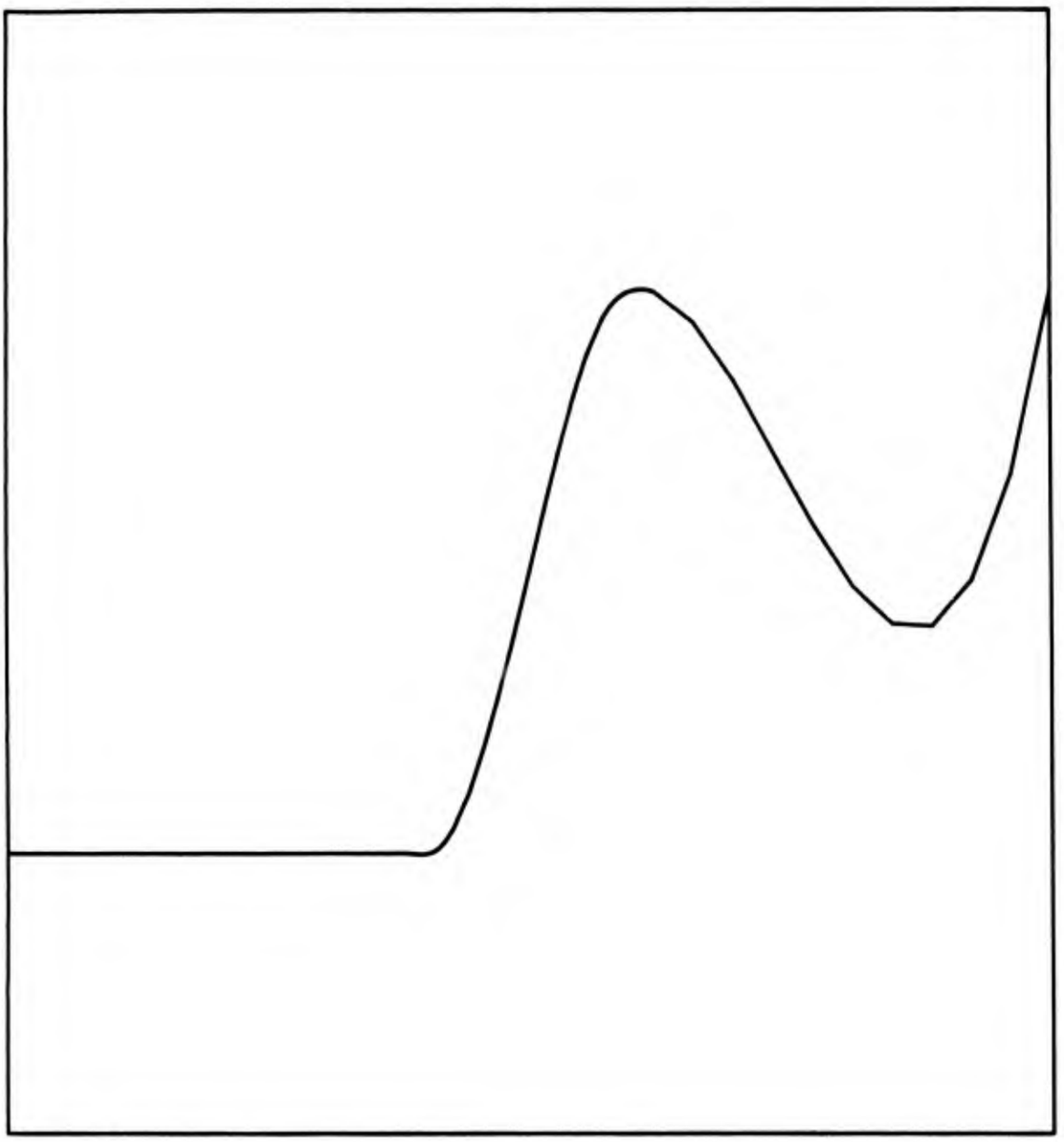

Fig. 12. The 8 points of Fig. 10 connected by the modified French curve method which was also used in Fig. 8.

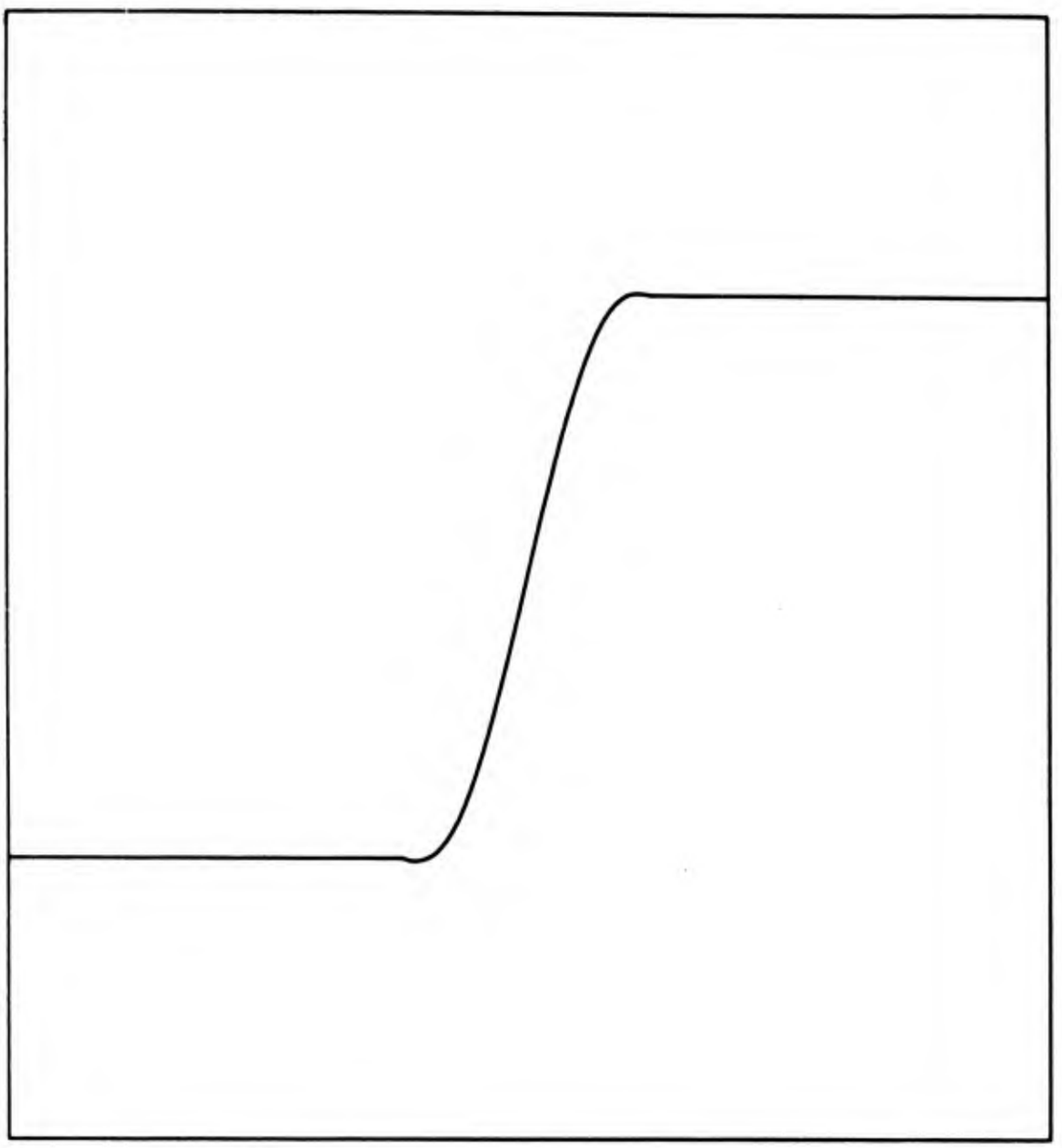

Fig. 13. The 8 points of Fig. 10 connected using the French curve method.

As the last example illustrates, it is not necessary that the initial (given) set of points be equally spaced on the x axis of the plot; and, in fact, it is often advantageous to have them more densely clustered where the curve undergoes sharp changes of slope.

Another good rule that has been learned by experience in the use of this algorithm is that when a curve has a symmetrical extremum or extrema, as in the sine curve, Figs. 2-4, one of the specified points should, if possible, be exactly at each extremum, and there should be two adjacent points at equidistant x-axis intervals from the extremum. This insures that the curve will have zero slope, as it should, at the extremum. When there is a "corner" with two adjoining nearly straight portions, as in Fig. 6, a perfect fit is not possible; but an acceptable curve can be produced by the point-clustering method of Fig. 10.

FORTRAN SUBROUTINES

The French-curve plotting method has been implemented by a Fortran plotting subroutine named FRCURV. A listing of this subroutine and some auxiliary subroutines follows, as written for the NRL CDC-3800 computer. The parameters of Subroutine FRCURV are:

XX -- the array of the x-coordinates of the points to be plotted (inches), ordered from the smallest to the largest;

YY -- the array of y-coordinates (inches);

YMIN -- the y-coordinate at the bottom of the plotting area (inches);

YMAX -- the y-coordinate at the top of the plotting area (inches);

NP -- the number of points specified for each curve;

NC -- the total number of curves to be plotted.

As the parameter NC implies, the subroutine is written so that a family of curves can be plotted in one call to FRCURV, by establishing a two-dimensional array of values of the x and y coordinates. The arrays XX and YY, therefore, each have two subscripts, the first going to NP and the second going to NC. Thus, for example, if a set of 5 curves is to be plotted with 20 points per curve, the program that calls FRCURV must contain a declarative statement reading:

```
DIMENSION XX(20,5), YY(20,5)
```

The first subscripts of the points in the XX and YY arrays determine the order of plotting of the points in each curve, while the second subscripts determine the order of plotting the curves. (If there is only one curve to be plotted, XX and YY can nevertheless be doubly subscripted and dimensioned, with the second subscripts equal to 1.) If a "single" curve is to be plotted, but it contains actual discontinuities, the successive sections of the discontinuous curve can be treated as separate curves in setting up the XX and YY arrays. However, if the number of points (NP) in the successive sections are not all the same, then each curve section must be plotted by a separate call to FRCURV, with NC = 1 on each call.

The minimum value of XX must be no less than the leftmost coordinate of the plotting area, and the maximum value of XX must be no greater than the rightmost coordinate, if it is desired to plot the curves only within the plotting area. However, values of YY above or below the prescribed plotting area can be used. As will be described, Subroutine LIMITS will cause the curve to be discontinued (automatically) wherever it goes above or below the prescribed plotting area defined by YMIN and YMAX.

The coordinates XX and YY must define a single-valued function YY(XX). If they do not, a diagnostic statement to that effect will be printed, and the program will be terminated. The curves will be plotted up to that point.

The solutions of the systems of simultaneous equations in Subroutine FRCURV are obtained by calls to Subroutine MATALG, which was obtained from the NRL CDC-Co-op Program Library. Its Co-op Identification Number is F2-CODA-MATALG. The subroutine was written by Sanford Elkin of Control Data Corporation, in December 1963, as a modification of F2-CODA-MATEQ.

Provision is made in Subroutine FRCURV, by calling Subroutine LIMITS, for discontinuing the plotting whenever the curve goes outside the prescribed upper and lower limits of the plotting area, specified by the parameters YMIN and YMAX. These parameters are in inches with respect to the origin of coordinates established for the plotter. A listing of Subroutine LIMITS also follows. The algorithm used in this subroutine will be described in another report. It in turn employs another subroutine named INTRSECT, also listed below, which computes the intersection of the curve with the border of the plotting area, so that when the curve goes beyond the limits of this area, the plot is made exactly to the border of the area rather than to the nearest computed points.

Subroutine RADCRV computes the radius of curvature for each segment of the plotted curve, sampled at its endpoints and midpoint, then sets the plotting interval equal to 0.1 times the least of these radii. It computes the x-axis increment according to Eq. (13).

An example of the use of Subroutine FRCURV, which in turn calls Subroutines MATALG, RADCRV, and LIMITS, is given in the following listing of a program named TEMPLØT. The resulting plot is shown in Fig. 14. This is an actual plot for some calculated atmospheric noise-temperature data. The data points are assumed to have been punched on data cards which are read as the initial step of the program. The program for calculating these points is quite lengthy, so that it would be quite expensive to calculate enough points to plot a smooth curve directly.

Program TEMPLØT as listed below has been skeletonized to eliminate statements not relevant to the use of Subroutine FRCURV. The removed statements are those that result in plotting the coordinate grid and labeling the axes and the curves. The calls to PLØTS and STØPPLØT are the required initializing and terminating calls for use of the on-line plotter of the NRL CDC-3800 computer.

```
      PRØGRAM TEMPLØT
      DIMENSIØN P(254), X(48,7), Y(48,7)
      CALL PLØTS (P,254,18)
      READ 1, (((X(I,J),Y(I,J)), I=1,48), J=1,7)
      CALL FRCURV (X, Y, 0., 8., 48, 7)
      CALL STØPPLØT
1     FØRMAT (10F8.3)
      END
```

The third and fourth parameters in the call to FRCURV establish zero as the lower limit on the plotted values of Y, and 8 inches as the upper limit; these are the lower and upper boundaries of the coordinate grid.

Listings of Subroutines FRCURV, RADCRV, LIMITS, INTRSECT, and MATALG follow.

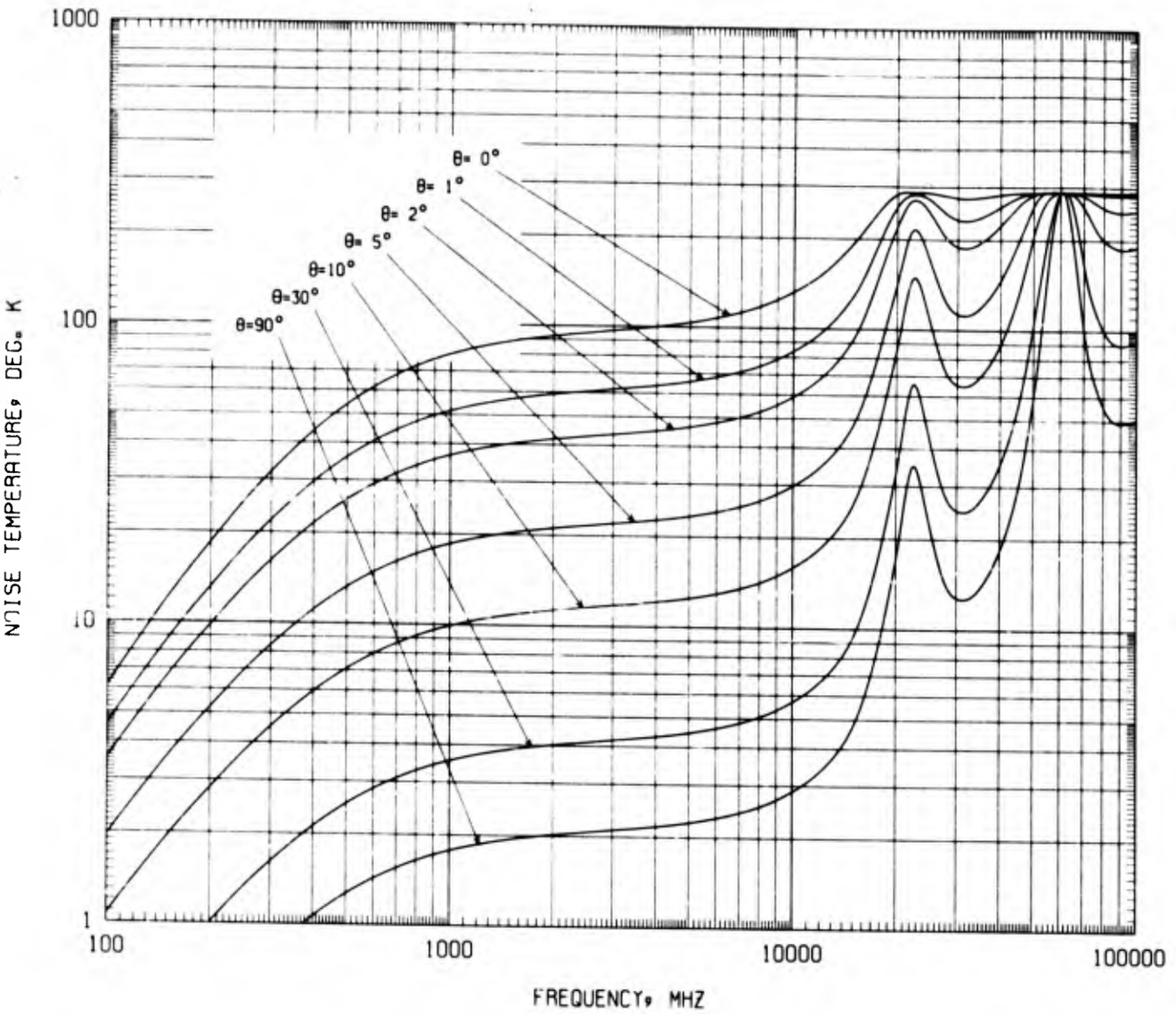

Fig. 14. A multiple curve plot made using Subroutine FRCURV. The 7 curves each have 48 computed points which are not equally spaced on the x axis; there are clusters of points in the regions of greater curvature. These curves represent actual computations of tropospheric noise temperature as a function of radio frequency and elevation angle of the observing antenna located at the earth's surface, in a log-log coordinate system. The entire plot including the coordinate grid and the labeling was done by the machine plotter (Gerber Model 875).

```
      SUBROUTINE FRCURV(XX,YY,YMIN,YMAX,NP,NC)
C
C     FRENCH CURVE PLOTTING SUBROUTINE.  THIS VERSION USES TWO POSITION
C     EQUATIONS AND TWO SLOPE EQUATIONS.  SLOPE AT EACH SPECIFIED POINT
C     OF THE CURVE IS THE AVERAGE OF THE SLOPES OF THE LINES JOINING
C     THE POINT WITH THE TWO ADJACENT POINTS.
C
C     ADDITIONAL SUBROUTINES REQUIRED -- MATALG, LIMITS, INTRSECT,
C     RADCRV.   SUBROUTINES EXCEPT MATALG WRITTEN BY L. V. BLAKE,
C     NRL CODE 5370.  COMPLETED JUNE 1971.
C
      DIMENSION AR(4,4),AY(4,4),AM(4,4),XX(NP,NC),YY(NP,NC)
      COMMON/MNTP/ MTP
C
C     THIS COMMON DECLARATION AND THE STATEMENT MTP=1 RESULTS IN
C     MINIMIZING THE NUMBER OF POINTS ACTUALLY PLOTTED, WHEN THE
C     PLOTTING IS DONE FOR THE GERBER PLOTTER, USING THE SPECIAL
C     PLOTTING SUBRCUTINES DEVELOPED FOR THIS PLOTTER AT NRL.
C
      DATA (NCALL=0)
      Y(Z)=AY(1,1)+Z*(AY(2,1)+Z*(AY(3,1)+Z* AY(4,1)))
      MTP=1
      NCALL = NCALL +1
      JMAX=NP-2
      AM(4,2)=1.
      DO 1 K=1,NC
      IF (XX(2,K).LE.XX(1,K)) 480,481
  480 JJ=1
      KK=K
      GO TO 500
  481 AR(1,1)=AR(2,1)=AR(3,2)=AR(4,2)=1.
      ILIM=0
      DO 2 I=1,2
      AY(I,1)=YY(I,K)
      DO 2 L=2,4
      AR(I,L)=AR(I,L-1)*XX(I,K)
    2 AM(I,L)=AR(I,L)
      A1=ATANF((YY(2,K)-YY(1,K))/(XX(2,K)-XX(1,K)))
      A2=ATANF((YY(3,K)-YY(2,K))/(XX(3,K)-XX(2,K)))
      AV = TANF((A1+A2)*.5)
      AY(3,1)=AV
      AR(3,1)=0.
      AR(3,3)=AM(4,3)=2.*XX(2,K)
      AM(4,4)=3.*XX(2,K)**2
      CALL MATALG(AR,AY,3,1,0,DT,4)
      DXT=XX(2,K)-XX(1,K)
      AY(4,1)=0.
      CALL RADCRV(AY(2,1),AY(3,1),AY(4,1),XX(1,K),XX(2 ,K),DXT,DX)
      IN=DXT/DX + .9
      DEL=DXT/IN
      XI=XX(1,K)
      CALL LIMITS( XX(1,K),YY(1,K),ILIM,YMIN,YMAX)
      DO 3 II=1,IN
      XI=XI+DEL
      YI=Y(XI)
    3 CALL LIMITS(XI,YI)
```

```
      SUBROUTINE RADCRV(AY2,AY3,AY4,X1,X2,DXT,DX)
C
C     SUBROUTINE COMPUTES RADIUS OF CURVATURE (FUNCTION RC) OF THIRD-
C     DEGREE POLYNOMIAL FOR WHICH COEFFICIENTS OF NON-CONSTANT TERM ARE
C     AY2, AY3, AND AY4, IN THE ABSCISSA INTERVAL FROM X1 TO X2.
C     CURVATURE IS COMPUTED AT ENDS AND CENTER OF INTERVAL, AND LEAST
C     OF THESE THREE VALUES IS FOUND.  THEN THE ABSCISSA INTERVAL DX IS
C     FOUND WHICH IS AN APPROPRIATE PLOTTING INTERVAL FOR PLOTTING A
C     SMOOTH CURVE., BASED ON ARC-LENGTH EQUAL TO 0.1 TIMES THE RADIUS
C     OF CURVATURE.
C
      RC(XM)=ABSF(((1. + SLSQ)**1.5)/DENOM)
      SLS(XM)= (AY2 + 2.*AY3*XM + 3.*AY4*XM*XM)**2
      DEN(XM)=2.*AY3 + 6.*AY4*XM
      XAV=(X1+X2)*.5
      DENOM=DEN(X1)
      IF (DENOM .EQ. 0.) DENOM = 1.E-45
      SLSQ=SLS(X1)
      RCM=RC(X1)
      SLSQA=SLSQ
      DENOM=DEN(XAV)
      IF (DENOM .EQ. 0.) DENOM = 1.E-45
      SLSQ=SLS(XAV)
      RC2=RC(XAV)
      IF (RC2.LT.RCM) 1,2
    1 RCM=RC2
      SLSQA=SLSQ
    2 DENOM=DEN(X2)
      IF (DENOM .EQ. 0.) DENOM = 1.E-45
      SLSQ=SLS(X2)
      RC3=RC(X2)
      IF (RC3.LT.RCM) 3,4
    3 RCM=RC3
      SLSQA=SLSQ
    4 DD=.1*RCM
      DX=SQRTF(DD**2/(1.+SLSQA))
      IF (DX .GT. DXT) DX=DXT
      END
```

```
      SUBROUTINE LIMITS(X,Y,ILIM,YMIN,YMAX)
C     MODIFICATION OF JUNE 18, 1971, WITH ILIM, YMIN, AND YMAX IN PARA-
C     METER LIST RATHER THAN IN COMMON STATEMENT.
C     NOTE--  ILIM SHOULD BE SPECIFIED AS AN INPUT PARAMETER ONLY WHEN
C     ILIM=0.  ILIM = 1 AND ILIM = 2 ARE GENERATED INTERNALLY IN THE
C     SUBROUTINE.  NEVER USE A NUMERAL FOR ILIM IN THE CALL STATEMENT --
C     ALWAYS USE A FORTRAN IDENTIFIER.
C     ILIM=0 SIGNIFIES STARTING NEW CURVE.
C     ILIM=1 SIGNIFIES Y WAS OFF SCALE ON PRECEDING CALL TO LIMITS.
C     ILIM=2 OR GREATER SIGNIFIES PRECEDING Y VALUE WAS WITHIN LIMITS.
      GO TO (1,2) ILIM
    1 IF (Y .GT. YMAX .OR. Y .LT. YMIN) 10,11
   10 XLAST=X
      YLAST=Y
      IF (ILIM.EQ.0) 50,51
   50 IF(Y.GT.YMAX) 52,53
   52 Y0=YMAX
      GO TO 51
   53 Y0=YMIN
   51 ILIM=1
      RETURN
   11 IF (ILIM .EQ. 0) 15,16
   15 CALL PLOT (X,Y,3)
      GO TO 17
   16 IF (ILIM .EQ. 1) 12,13
   12 CALL INTRSECT(XLAST,YLAST,X,Y,0.,Y0,1.,Y0,X0,Y0)
      CALL PLOT (X0,Y0,3)
   13 CALL PLOT(X,Y,2)
   17 XLAST=X
      YLAST=Y
      ILIM=2
      RETURN
    2 IF (Y .LT. YMIN) 20,21
   20 CALL INTRSECT(XLAST,YLAST,X,Y,0.,YMIN,1.,YMIN,X0,Y0)
      CALL PLOT (X0,Y0,2)
      ILIM=1
      XLAST=X
      YLAST=Y
      RETURN
   21 IF (Y .GT. YMAX) 30,31
   30 CALL INTRSECT(XLAST,YLAST,X,Y,0.,YMAX,1.,YMAX,X0,Y0)
      CALL PLOT (X0,Y0,2)
      XLAST=X
      YLAST=Y
      ILIM=1
      RETURN
   31 CALL PLOT (X,Y,2)
      XLAST=X
      YLAST=Y
      ILIM=2
      END
```

```
      DO 4 J=2,JMAX
      J1=J+1
      J2=J+2
      IF (XX(J1,K).LE.XX(J,K)) 488,499
  488 JJ=J
      KK=K
      GO TO 500
  499 AY(1,1)=YY(J,K)
      AY(2,1)=YY(J1,K)
      AY(3,1)=AV
      A1=ATANF((YY(J1,K)-YY(J,K))/(XX(J1,K)-XX(J,K)))
      A2=ATANF((YY(J2,K)-YY(J1,K))/(XX(J2,K)-XX(J1,K)))
      AV = TANF((A1+A2)*.5)
      AY(4,1)=AV
      AR(1,1)=AR(2,1)=AR(4,2)=1.
      AR(3,1)=AR(4,1)=0.
      DO 5 IJ=2,4
      AR(1,IJ)=          AM(2,IJ)
      AR(3,IJ)=          AM(4,IJ)
    5 AR(2,IJ)=AM(2,IJ)=AR(2,IJ-1)*XX(J+1,K)
      AR(4,3)=AM(4,3)=2.*XX(J1,K)
      AR(4,4)=AM(4,4)=3.*XX(J1,K)**2
      CALL MATALG(AR,AY,4,1,0,DT,4)
      DXT=XX(J1,K)-XX(J,K)
      CALL RADCRV(AY(2,1),AY(3,1),AY(4,1),XX(J,K),XX(J1,K),DXT,DX)
      IN=DXT/DX + .9
      DEL=DXT/IN
      DO 6 IL=1,IN
      XI=XI+DEL
      YI=Y(XI)
    6 CALL LIMITS(XI,YI)
    4 CONTINUE
      J1=JMAX+1
      J2=NP
      IF (XX(J2,K).LE.XX(J1,K)) 550,551
  550 JJ=J1
      KK=K
      GO TO 500
  551 AY(1,1)=YY(J1,K)
      AY(2,1)=YY(J2,K)
      AY(3,1)=AV
      AR(1,1)=AR(2,1)=AR(3,2)=1.
      AR(3,1)=0.
      AR(1,2)=AM(2,2)
      AR(1,3)=AM(2,3)
      AR(2,2)=XX(J2,K)
      AR(2,3)=XX(J2,K)**2
      AR(3,3)= 2.*XX(J1,K)
      CALL MATALG(AR,AY,3,1,0,DT,4)
      AY(4,1)=0.
      DXT=XX(NP,K)-XX(J1,K)
      CALL RADCRV(AY(2,1),AY(3,1),AY(4,1),XX(J1  ,K),XX(NP,K),DXT,DX)
      IN=DXT/DX + .9
      DEL=DXT/IN
      DO 7 IZ=1,IN
      XI=XI+DEL
```

```
      YI=Y(XT)
    7 CALL LIMITS(XI,YI)
    1 CONTINUE
      RETURN
  500 PRINT 501, NCALL
      PRINT 502
      PRINT 503,JJ,KK
C     IF X-DIRECTION REVERSAL OF CURVE OCCURS, PLOT WILL BE OBTAINED TO
C     THAT POINT, BUT EXECUTION OF PROGRAM WILL THEN BE TERMINATED.
      CALL STOPPLOT
      STOP
  501 FORMAT (/////*   MESSAGE FROM SUBROUTINE FRCURV, ON *,I5,*-TH CALL
     1,*)
  502 FORMAT(*   THE ARRAY OF POINTS TO BE PLOTTED CONTAINS A REVERSAL O
     1F X-DIRECTION AT THE *)
  503 FORMAT(3X,I5,*-TH COORDINATE OF THE *,I3,*-TH CURVE. THIS IS NOT P
     1ERMISSIBLE. EXECUTION TERMINATED AT THIS POINT.*)
      END
```

```
      SUBROUTINE RADCRV(AY2,AY3,AY4,X1,X2,DXT,DX)
C
C     SUBROUTINE COMPUTES RADIUS OF CURVATURE (FUNCTION RC) OF THIRD-
C     DEGREE POLYNOMIAL FOR WHICH COEFFICIENTS OF NON-CONSTANT TERM ARE
C     AY2, AY3, AND AY4, IN THE ABSCISSA INTERVAL FROM X1 TO X2.
C     CURVATURE IS COMPUTED AT ENDS AND CENTER OF INTERVAL, AND LEAST
C     OF THESE THREE VALUES IS FOUND.  THEN THE ABSCISSA INTERVAL DX IS
C     FOUND WHICH IS AN APPROPRIATE PLOTTING INTERVAL FOR PLOTTING A
C     SMOOTH CURVE., BASED ON ARC-LENGTH EQUAL TO 0.1 TIMES THE RADIUS
C     OF CURVATURE.
C
      RC(XM)=ABSF(((1. + SLSQ)**1.5)/DENOM)
      SLS(XM)= (AY2 + 2.*AY3*XM + 3.*AY4*XM*XM)**2
      DEN(XM)=2.*AY3 + 6.*AY4*XM
      XAV=(X1+X2)*.5
      DENOM=DEN(X1)
      IF (DENOM .EQ. 0.) DENOM = 1.E-45
      SLSQ=SLS(X1)
      RCM=RC(X1)
      SLSQA=SLSQ
      DENOM=DEN(XAV)
      IF (DENOM .EQ. 0.) DENOM = 1.E-45
      SLSQ=SLS(XAV)
      RC2=RC(XAV)
      IF (RC2.LT.RCM) 1,2
    1 RCM=RC2
      SLSQA=SLSQ
    2 DENOM=DEN(X2)
      IF (DENOM .EQ. 0.) DENOM = 1.E-45
      SLSQ=SLS(X2)
      RC3=RC(X2)
      IF (RC3.LT.RCM) 3,4
    3 RCM=RC3
      SLSQA=SLSQ
    4 DD=.1*RCM
      DX=SQRTF(DD**2/(1.+SLSQA))
      IF (DX .GT. DXT) DX=DXT
      END
```

```
      SUBROUTINE LIMITS(X,Y,ILIM,YMIN,YMAX)
C     MODIFICATION OF JUNE 18, 1971, WITH ILIM, YMIN, AND YMAX IN PARA-
C     METER LIST RATHER THAN IN COMMON STATEMENT.
C     NOTE--  ILIM SHOULD BE SPECIFIED AS AN INPUT PARAMETER ONLY WHEN
C     ILIM=0.  ILIM = 1 AND ILIM = 2 ARE GENERATED INTERNALLY IN THE
C     SUBROUTINE.  NEVER USE A NUMERAL FOR ILIM IN THE CALL STATEMENT --
C     ALWAYS USE A FORTRAN IDENTIFIER.
C     ILIM=0 SIGNIFIES STARTING NEW CURVE.
C     ILIM=1 SIGNIFIES Y WAS OFF SCALE ON PRECEDING CALL TO LIMITS.
C     ILIM=2 OR GREATER SIGNIFIES PRECEDING Y VALUE WAS WITHIN LIMITS.
      GO TO (1,2) ILIM
    1 IF (Y .GT. YMAX .OR. Y .LT. YMIN) 10,11
   10 XLAST=X
      YLAST=Y
      IF (ILIM.EQ.0) 50,51
   50 IF(Y.GT.YMAX) 52,53
   52 Y0=YMAX
      GO TO 51
   53 Y0=YMIN
   51 ILIM=1
      RETURN
   11 IF (ILIM .EQ. 0) 15,16
   15 CALL PLOT (X,Y,3)
      GO TO 17
   16 IF (ILIM .EQ. 1) 12,13
   12 CALL INTRSECT(XLAST,YLAST,X,Y,0.,Y0,1.,Y0,X0,Y0)
      CALL PLOT (X0,Y0,3)
   13 CALL PLOT(X,Y,2)
   17 XLAST=X
      YLAST=Y
      ILIM=2
      RETURN
    2 IF (Y .LT. YMIN) 20,21
   20 CALL INTRSECT(XLAST,YLAST,X,Y,0.,YMIN,1.,YMIN,X0,Y0)
      CALL PLOT (X0,Y0,2)
      ILIM=1
      XLAST=X
      YLAST=Y
      RETURN
   21 IF (Y .GT. YMAX) 30,31
   30 CALL INTRSECT(XLAST,YLAST,X,Y,0.,YMAX,1.,YMAX,X0,Y0)
      CALL PLOT (X0,Y0,2)
      XLAST=X
      YLAST=Y
      ILIM=1
      RETURN
   31 CALL PLOT (X,Y,2)
      XLAST=X
      YLAST=Y
      ILIM=2
      END
```

```
      SUBROUTINE INTRSECT(X1,Y1,X2,Y2,XA,YA,XB,YB,X0,Y0)
C     THIS SUBROUTINE FINDS THE INTERSECTION POINT X0,Y0 OF TWO STRAIGHT
C     LINES. WHEN TWO POINTS ON EACH LINE ARE GIVEN. THE TWO POINTS ON
C     ONE OF THE LINES ARE X1,Y1 AND X2,Y2.  THE TWO POINTS ON THE OTHER
C     LINE ARE XA,YA AND XB,YB. THE INTERSECTION POINT NEED NOT LIE
C     BETWEEN THESE GIVEN POINTS.
      IF (ABS(X2-X1) .LT. 10.E-306) 1, 2
    1 IF (ABS(XB-XA) .LT. 10.E-306) GO TO 99
      X0 = X1
      S = (YB-YA)/(XB-XA)
      Y0 = S*(X0-XA) + YA
      RETURN
    2 IF (ABS(XB-XA) .LT. 10.E-306) 3, 4
    3 IF (ABS(X2-X1) .LT. 10.E-306) GO TO 99
      X0 = XA
      S = (Y2-Y1)/(X2-X1)
      Y0 = S*(X0-X1) + Y1
      RETURN
    4 S1 = (YB-YA)/(XB-XA)
      S2 = (Y2-Y1)/(X2-X1)
      IF (S1 .EQ. 0.) GO TO 5
      IF (ABS(S2-S1) .LT. 10.E-306) GO TO 99
      RTO = S2/S1
      Y0 = (Y2+S2*(XA-X2)-YA*RTO)/(1.-RTO)
      X0 = (Y0-YA)/S1 + XA
      RETURN
    5 IF (S2 .EQ. 0.) GO TO 99
      X0= (YA-Y2)/S2 + X2
      Y0=YA
      RETURN
   99 PRINT 100, X1, Y1
      PRINT 101,     X2,Y2,XA,YA,XB,YB
      X0 = 0.
      Y0 = 0.
  100 FORMAT (3X,64HCALL TO INTRSECT ABORTED. LINES PARALLEL, NO INTERSE
     1CTION. X1 = ,E10.2,6H Y1 = ,E10.2)
  101 FORMAT(3X,
     2 6H XA = ,E10.2,6H YA = ,E10.2,6H XB = ,E10.2,6H YB = ,E10.2 //)
      END
```

```
      SUBROUTINE MATALG(A,X,NR,NV,IDO,DET,NACT)
C
C     SUBROUTINE WRITTEN BY SANFORD ELKIN, CONTROL DATA CORPORATION,
C     DEC. 1963, AS A MODIFICATION OF PREVIOUS SUBROUTINE F2-COMA-MATEQ.
C
      DIMENSION A(NACT,NACT),X(NACT,NACT)
      IF(IDO) 1,2,1
    1 DO 3 I=1,NR
      DO 4 J=1,NR
    4 X(I,J)=0.0
    3 X(I,I)=1.0
      NV=NR
    2 DET=1.0
      NR1=NR-1
      DO 5 K=1,NR1
      IR1=K+1
      PIVOT=0.0
      DO 6 I=K,NR
      Z=ABSF(A(I,K))
      IF(Z-PIVOT) 6,6,7
    7 PIVOT=Z
      IPR=I
    6 CONTINUE
      IF(PIVOT) 8,9,8
    9 DET=0.0
      RETURN
    8 IF(IPR-K) 10,11,10
   10 DO 12 J=K,NR
      Z=A(IPR,J)
      A(IPR,J)=A(K,J)
   12 A(K,J)=Z
      DO 13 J=1,NV
      Z=X(IPR,J)
      X(IPR,J)=X(K,J)
   13 X(K,J)=Z
      DET=-DET
   11 DET=DET*A(K,K)
      PIVOT=1.0/A(K,K)
      DO 14 J=IR1,NR
      A(K,J)=A(K,J)*PIVOT
      DO 14 I=IR1,NR
   14 A(I,J)=A(I,J)-A(I,K)*A(K,J)
      DO 5 J=1,NV
      IF(X(K,J)) 15,5,15
   15 X(K,J)=X(K,J)*PIVOT
      DO 16 I=IR1,NR
   16 X(I,J)=X(I,J)-A(I,K)*X(K,J)
    5 CONTINUE
      IF(A(NR,NR)) 17,9,17
   17 DET=DET*A(NR,NR)
      PIVOT=1.0/A(NR,NR)
      DO 18 J=1,NV
      X(NR,J)=X(NR,J)*PIVOT
      DO 18 K=1,NR1
      I=NR-K
      SUM=0.0
```

```
      DO 19 L=I,NR1
   19 SUM=SUM+A(I,L+1)*X(L+1,J)
   18 X(I,J)=X(I,J)-SUM
      END
```