

GENERATING THREE DIMENSIONAL CUTTER PATHS
FOR
AN XY OR XZ CONTOUR MILLING MACHINE

by

ASHOK N. KABADI

B.E., University of Bombay, 1976

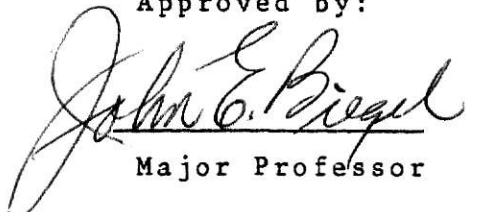
A MASTER'S THESIS

submitted in partial fulfillment of the
requirements for the degree

MASTER OF SCIENCE

Department of Industrial Engineering
KANSAS STATE UNIVERSITY
Manhattan, Kansas
1981

Approved by:


Major Professor

SPEC
COLL
LD
2668
.T4
1981
K32
C.2

TABLE OF CONTENTS

	Page
ACKNOWLEDGEMENT	iii
LIST OF FIGURES	iv
LIST OF TABLES	vii
CHAPTER 1. INTRODUCTION	1
1.1 BRIEF HISTORY OF NUMERICALLY CONTROLLED MACHINES	2
1.2 INTRODUCTION TO THE PROJECT	3
CHAPTER 2. LINES IN A PLANE AND IN SPACE	9
2.1 LINES IN A PLANE	10
2.2 LINES IN SPACE	35
CHAPTER 3. REGULAR CURVES IN A PLANE	49
3.1 CIRCULAR ARCS	50
3.2 PARABOLIC ARCS	56
3.3 ELLIPTIC ARCS	61
3.4 HYPERBOLIC ARCS	67
CHAPTER 4. BLENDING OF REGULAR CURVES IN A PLANE	71
4.1 ELLIPTIC ARC BLENDING	72
4.2 BLENDING OF TWO CIRCULAR ARCS	79
4.3 PARABOLIC BLENDING	84
CHAPTER 5. MOTION IN A PLANE GIVEN BY AN EQUATION	93
5.1 CURVES GIVEN BY AN EQUATION : $Y=F(X)$	94
5.2 CURVES GIVEN BY AN EQUATION : $X=F(Y)$	95
5.3 CURVES GIVEN BY AN EQUATION : $F(X,Y)=0$	100
CHAPTER 6. CURVES IN SPACE	105
6.1 CURVES GIVEN BY THE INTERSECTION OF TWO SURFACES	106
6.2 BEZIER CURVES	109

**THIS BOOK
CONTAINS
NUMEROUS
PAGES WITH
THE ORIGINAL
PRINTING ON
THE PAGE BEING
CROOKED.**

**THIS IS THE
BEST IMAGE
AVAILABLE.**

APPENDICES

1.	LINE SUBROUTINE FOR LINES IN A PLANE : CASE 1	122
	LINES IN A PLANE : CASE 2 AND CASE 3	124
	LINE SUBROUTINE FOR LINES IN SPACE	126
2.	CIRCLE	129
	PARABOLA	130
	ELLIPSE	131
	HYPERBOLA	132
3.	DERIVATION OF INPUT SLOPE LIMIT FOR AN ELLIPSE IN STANDARD POSITION WITH CENTER ON Y-AXIS	134
	DERIVATION OF THE EQUATION OF AN ELLIPSE IN STANDARD POSITION WITH CENTER ON Y-AXIS	136
	DERIVATION OF SLOPES m_1 AND m_2 FOR CIRCULAR ARC BLENDING	138
	CIRCULAR ARC BLENDING	140
	PARABOLIC BLENDING	141
4.	EQUATION OF A CURVE : $Y=F(X)$	145
	EQUATION OF A CURVE : $X=F(Y)$	146
	EQUATION OF ANY CURVE IN A PLANE	147
5.	CURVES IN SPACE	150
	BEZIER CURVE FOR 3-D OR 2-D	152
	REFERENCES	154

ACKNOWLEDGEMENT

The author is indebted to Professor John E. Biegel for his invaluable help and guidance, rendered so generously, during the course of this research.

LIST OF FIGURES

Figure	Page
1.1 An outline of the main programs and the subroutines	8
2.1 Possibilities of reaching the end point of a line in a plane	11
2.2 Lines in a plane	13
2.2a Case 1	
2.2b Case 2	
2.2c Case 3	
2.3 Algorithm for lines in a plane : Case 1	14
2.4 Flow diagram for lines in a plane : Case 1	17
2.5 Lines in a plane	19
2.6 Maximum deviation for lines in a plane	21
2.7 Lines in a plane	24
2.7a Case 2	
2.7b Case 3	
2.8 Algorithm for lines in a plane : Case 2	26
2.9 Flow diagram for lines in a plane : Case 2	27
2.10 Algorithm for lines in a plane : Case 3	30
2.11 Flow diagram for lines in a plane : Case 3	32
2.12 Examples of Cases 2 and 3	34
2.13 Lines in a plane : $TL=1$	36
2.14 Lines in a plane : $TL \neq 1$	37
2.15 Practical application of Cases 2 and 3.	38
2.16 Distance of a point in space to a line in space	40
2.17 Algorithm for lines in space	42
2.18 Flow diagram for lines in space	45
2.19 Lines in space	49

Figure

3.1a	Circular arc of length $r\theta$	51
3.1b	Chordal method	
3.1c	Tangential method	
3.1d	Secantial method	
3.2	Generation of a motion along a circular arc	53
3.3	Flow diagram : An arc of a circle	55
3.4	Arcs of circles : Case 1	57
3.5	Arcs of circles : Cases 2 and 3	58
3.6	Flow diagram for a parabola	60
3.7	Parabolas	62
3.8	Flow diagram for an ellipse	64
3.9	Ellipse inclined at an angle i to the horizontal	65
3.10	Ellipses	66
3.11	Flow diagram for a hyperbola	69
3.12	Hyperbolic arcs	70
4.1	Input conditions for elliptic arc design	75
4.2	Alternative to Fig.4.1 for elliptic arc design	76
4.3	Examples of input constraints for elliptic blending	77
4.4	Acceptable range of the input slope, m_c	80
4.5	Input data used in searching for solution for two blended	82
	Circular arcs	
4.6	Parabolic blending	86
4.7	Geometric relationships : $r(t)$ and $s(t)$	88
4.8	Parabolic blending for example in the text	92
5.1	Flow diagram : Equation of curve $Y=F(X)$	96

Figure

	Page
5.2 Curves in a plane : $Y=F(X)$	97
5.3 Curves in a plane : $Y=F(X)$	98
5.4 Flow diagram : Equation of curve $X=F(Y)$	99
5.5 Flow diagram for generalized curve in a plane	102
5.6 Curves in a plane : Any Equation : $F(X,Y)=0$	103
5.7 Curves in a plane : Any equation : $F(X,Y)=0$	104
6.1 Flow diagram for curves in space	110
6.2 Curves in space projected on a plane	112
6.3 Nomenclature for Bézier curves	113
6.4 Bézier polygon for cubics	116
6.5 Flow Diagram for Bézier curves	119
6.6 Bézier curves	120

LIST OF TABLES

Table	Page
1.1 Specifications of the numerically controlled machine	6
1.2 Specifications of the numerically controlled machine	7
2.1 Maximum error for lines in a plane	23
6.1 The results of an example for a cubic curve	117

CHAPTER 1
INTRODUCTION

**THIS BOOK
CONTAINS
NUMEROUS PAGES
WITH THE ORIGINAL
PRINTING BEING
SKEWED
DIFFERENTLY FROM
THE TOP OF THE
PAGE TO THE
BOTTOM.**

**THIS IS AS RECEIVED
FROM THE
CUSTOMER.**

INTRODUCTION

1.1 BRIEF HISTORY OF NUMERICALLY CONTROLLED MACHINES

In the late 1940s John T. Parsons proposed that a method of automatic machine control be developed to guide a milling cutter that would generate a smooth curve. In 1949 the U.S. Air Force commissioned the servomechanisms laboratory at the Massachusetts Institute of Technology to develop a workable NC system based on Parsons' concept.

Later, scientists and engineers at M.I.T. selected perforated paper tape as the communication medium and initially built a two-axis point-to-point system which positioned a drilling head over the hole location. As the research continued, a more sophisticated continuous path milling machine was produced. Independent machine tool builders have subsequently developed the sophisticated systems that are currently available.

Thus the evolution of numerically controlled machines may be divided into three distinct periods :

1. From 1942 to 1949 : the appearance of the first small-or-medium sized contour-milling machines.
2. From 1950 to 1960 : the appearance and development of "point-to-point" drilling/boring machines of all sizes.
3. After 1960 : the simultaneous development of "point-to-point" and contouring machines; the former being, nevertheless, in the majority.

A change in overall philosophy began in the 1970s, when Numerical control was then viewed as part of a larger concept; viz. computer-aided-manufacturing(CAM). CAM encompasses not

only NC but production control and monitoring, materials management and scheduling. The emphasis on the use of computers in the manufacturing process has spawned new forms of Numerical control: Computer Numerical Control(CNC) and Direct Numerical Control(DNC).

Numerical control is not a kind of machine tool but a technique for controlling a wide variety of machines. Numerical control performs best where other forms of specialized automation fail. NC is a system that can interpret a set of prerecorded instructions in some symbolic format; it can cause the controlled machine to execute the instructions, and then can monitor the results so that the required precision is maintained.

1.2 INTRODUCTION TO THE PROJECT

The objective of this project is to make a two-axis numerically controlled contouring milling machine into a three-axis computer controlled contouring milling machine by transferring the control of the table and the turret to a micro-computer.

1.2.1 About the milling machine

The machine referred to in this project is a three-axis, vertical turret, numerically controlled milling machine, distributed by Pratt & Whitney as the Tape-mate Series C. The detailed specifications of this machine are given in Tables 1.1 and 1.2. As depicted in Table 1.2, it can be seen that the machine has the X-Y or X-Z axes as the simultaneously controllable axes. The motions along +X, -X and +Y, -Y are obtained by the movement of the table and the motions along +Z and -Z are obtained by the

Table 1.1 Specifications of the milling machine.

SPECIFICATIONS

MACHINING	Metric	Inch
Drilling mild steel max. dia.	20 mm	3/4"
Tapping	3 - 16 mm	1/8 - 5/8" Dia.
Milling mild steel	16 mm	5/8" Dia. @0.4 cu. in./min.
Aluminum	16 mm	5/8 Dia. @1.0 cu. in./ min.

TABLE

Work surface	610 x 460 mm	24" x 18"
Table travel (X-Y)	500 x 380 mm	20" x 15"
Least command increment	0.01 mm/pulse	0.0002"/pulse
Rapid traverse	6000 mm/min.	250 IPM
Feed rate (Inf. var.)	1 - 2000 mm/min.	0.05 - 80 IPM
Jog feed rates (24 feeds)	1 - 2000 mm/min.	0.05 - 80 IPM
Positioning accuracy	±0.05 mm/300 mm	±0.002"/12"
Repeatability	±0.01 mm	±0.0005"
Max. wt. on table	150 Kg	330 lbs.

SPINDLE

Spindle travel	220 mm	8.5"
Distance from spindle nose to table	130 - 350 mm	5 - 13.5"
Least command increment	0.01 mm/pulse	0.0002"/pulse
Rapid traverse	4000 mm/min	150 IPM
Feedrate (Inf. var.)	1-2000 mm/min	0.05 - 80 IPM
Jog feed rates (24 Feeds)	1-2000 mm/min	0.05 - 80 IPM
Positioning accuracy	±0.05 mm	±0.002"
Repeatability	±0.01 mm	±0.0005"
AC spindle motor 3 speed	1,2,2 H.P.	
Low speed range	280,560,1120 RPM	
High speed range	840,1680,3360 RPM	
High - Low speed change	Manual belt change	
Sequential tool magazine capacity	7 tools	
Standard Arbors: #6 Jacobs taper arbor	3 Ea.	
#3 Morse taper arbor	4 Ea.	

SIZE

Floor space	71" x 55.2" x 70.5" high
Machine weight	4500 lbs.

Table 1.2 Specifications of the milling machine

Specifications

TRAVEL CAPACITIES

Longitudinal - X axis	20"
Traverse - Y axis	15"
Vertical - spindle travel	8-1/2"

TRAVEL RATES

Rapid traverse - X & Y axes	250 in./min.
- Z axis	150 in./min.
Feed rate range — X, Y, Z	0.05 - 100 in./min.

TABLE

Work surface	24" x 18"
Maximum weight on table	330 lbs.
T-slots - (4) on 3-15/16" centers	7/16" wide

POSITIONING ACCURACIES*

Positioning per axis - X and Y overall	±0.002"
Repeatability per axis - X and Y overall	±0.0005"
Type slide drive - X, Y & Z axes.. Ball lead screw w/DC servo motor	

SPINDLE

Horsepower	1, 2, 2
Spindle drive motor	3 speed
Speed selection	6 total
Low speed pulleys - tape selection of (3)	280-560-1120 rpm
High speed pulleys - tape selection of (3)	840-1680-3360 rpm
Tool magazine capacity	7 tools - max.
Distance, spindle nose to table top (min/max)	5" to 13-1/2"
Drawbar tool retention	

MACHINING CAPACITY

Drilling - mild steel	3/4" Dia. max.
Tapping - mild steel	to 5/8" Dia. max.
Milling rate - mild steel	0.4 cu. in. per min. (5/8" dia. cutter max.)
- aluminum	1.0 cu. in. per min. (5/8" dia. cutter max.)

CONTROL

Input resolution	0.0005"
------------------	---------

MACHINE WEIGHT (Approx.) 4500 lbs. net — 5000 lbs. gross

*Under no load conditions.

Accessories

Spindle arbor with #3 Morse taper.
Milling arbor with 5/8" collet.
Spindle arbor with 1JT (Jacobs taper mount)
Spindle arbor with 6JT (Jacobs taper mount)
Spindle arbor with 3JT (Jacobs taper mount)
Spindle arbor with 7/8" - 20 UNF mount for tapping attach.

Standard Equipment

3-axis control with 0.0005" input.
Manual, MDI and tape control.
Large scale integrated circuitry combined with microprocessor.
Solid state interface.
300 CPS photoelectric tape reader.
EIA/ISO tape format.
Tape tumble box (100' cap.).
Keyboard manual data input with command & position readout.
Linear interpolation for 2 axes (X-Y, X-Z).
Circular interpolation for 2 axes (X-Y).
Incremental programming.
Manual feed rate override 0—200%.
Buffer storage.
Point-to-point positioning, X, Y and Z.
Tool length compensation (8).
Zero reset.
Axis inversion.
Fixed machining cycles.
Direct feed rates selected by tape or MDI.
Dwell cycle.
Inch-metric switchable.
Axes offsets (4).
Floating zero digital readout X, Y, Z.
Diagnostic alarm system.

Options

Absolute and incremental programming.
Part program storage and editing (66 ft.) with search and display.
Flood coolant.
Auto transformer (5KVA).

Specifications, prices and equipment subject to change without notice.



movement of the vertical turret.

1.2.2 Problem

The specifications of the machine depicted in the Table 1.1 reveal that the machine has a linear interpolation capability in the X-Y and X-Z planes, and a circular interpolation capability in the X-Y plane. This imposes limitations on the operating characteristics of the machine as it does not have a circular interpolation capability in the X-Z plane and does not have simultaneous two-axis control in the Y-Z plane. Obviously, as currently controlled, this machine cannot be used for generating three dimensional cutter paths.

Hence the problem is to make the two-axis numerically controlled milling machine more versatile by using a micro-computer to generate the cutter paths available from a three-axis (X,Y,Z) machine.

1.2.3 Methodology

The method adopted for carrying out the research consists of

- i) developing algorithms for generating various motions in a plane and in space and
- ii) developing computer programs in the BASIC language which will be checked with a plot on a digital plotter interfaced with Exidy Sorcerer micro-computer.

For lines in a plane and in space, the algorithm will generate the incremental steps to be taken along the X,Y and Z axes starting from one end of a line until the end point of that line is reached. To generate the motions along a curve, straight

line approximations will be made. Algorithms were developed to generate the following motions :

1. Lines in a plane and in spaceChapter 2
2. Regular curves in a planeChapter 3
(This includes circles, parabolas, ellipses and hyperbolas).
3. Blending of regular curves in a plane.Chapter 4
(This includes circular arc blending, parabolic blending and elliptic blending).
4. Curves given by an equation in a plane.Chapter 5
5. Space curves.Chapter 6

To check the programs for lines in space, a line at an angle of 45° to the X and Y axes is considered to be the Z axis.

An outline of the main programs and the subroutines that will be used in this project is shown in Fig.1.1.

**THIS BOOK
CONTAINS
NUMEROUS PAGES
WITH DIAGRAMS
THAT ARE CROOKED
COMPARED TO THE
REST OF THE
INFORMATION ON
THE PAGE.**

**THIS IS AS
RECEIVED FROM
CUSTOMER.**

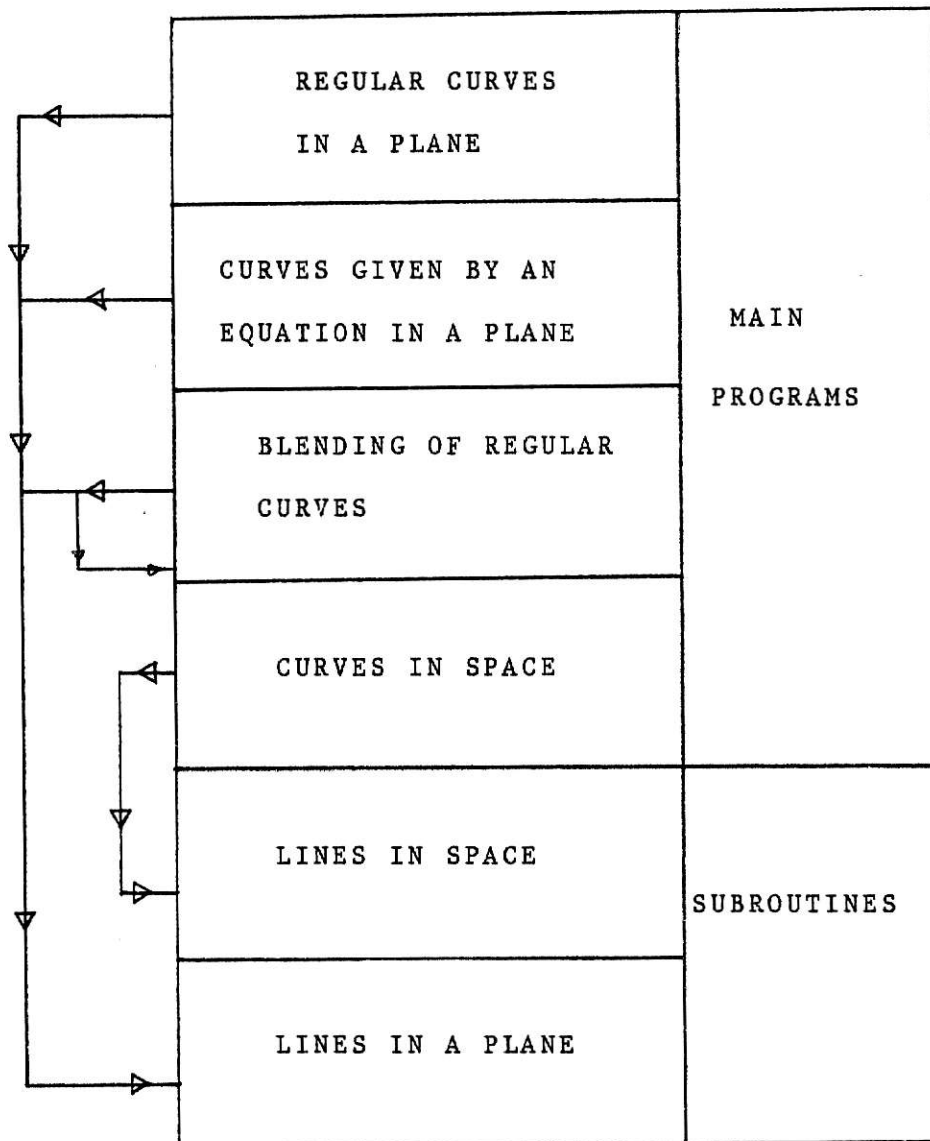


Fig.1.1.1 An outline of the main programs and the subroutines

**THIS BOOK
CONTAINS
NUMEROUS PAGES
THAT WERE
BOUND WITHOUT
PAGE NUMBERS.**

**THIS IS AS
RECEIVED FROM
CUSTOMER.**

CHAPTER 2

LINES IN A PLANE AND IN SPACE

LINES IN A PLANE AND IN SPACE

This Chapter is related to the development of algorithms for the generation of motions along lines in a plane or in space. In order to simplify the algorithmic procedure, the problem of generating the motions along lines in a plane is solved first, and the same principle is applied for generating the motions along lines in space. This Chapter discusses the two more cases of generating motions along lines in a plane with examples of their practical application.

2.1 LINES IN A PLANE

The procedure adopted for the generation of the motion along a line in a plane involves generating a suitable number of incremental steps parallel to the two axes (forming the plane), starting from one end of the line and proceeding to the other end of the line. Since the least command increment for the digital plotter is .005", the algorithms checked on the digital plotter have a least incremental step of .005" along all the three axes, X, Y and Z.

Consider a line to be drawn from the origin (0,0) to a point (5,3) in the X-Y plane. Assume that the incremental step along both X and Y axes is one unit. If starting from the origin, the terminating point of the line is to be reached then there are many ways in which this can be achieved by giving the incremental steps along the X and Y axes. Some of these possibilities are shown in Fig.2.1. Among these several paths denoted by L1, we want to select that path which is the "best fit to the

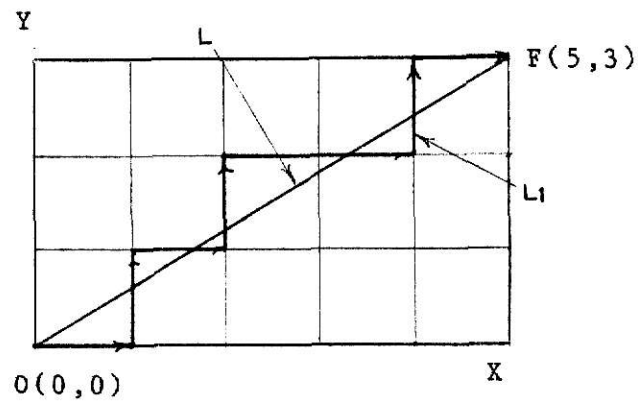
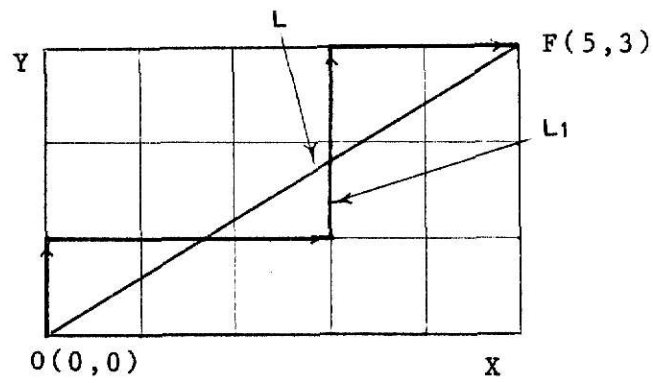
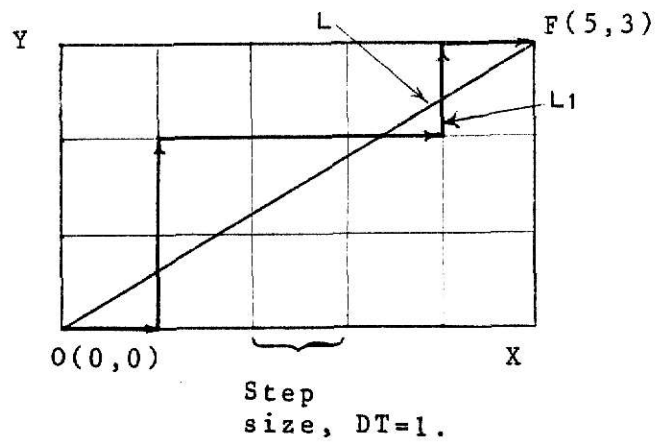


Fig. 2.1 Possibilities of reaching the end point of a line in a plane.

actual line L. For the selection of this best path, certain conditions need to be imposed and in this context the following three important cases have been investigated.

Case 1 : Minimize the maximum absolute deviation of points on both sides of a line. In this case the generated path stays as close to the actual line as possible. For example, to machine a contour that does not mate with another part. An example of Case 1 for the generation of motion along a line is shown in Fig.2.2a.

Case 2 : Minimize the maximum absolute deviation of points on one side of a line. No deviations are permissible on the opposite side of a line. In this case the generated path stays only on one side of a line. An example of this case is shown in Fig.2.2b.

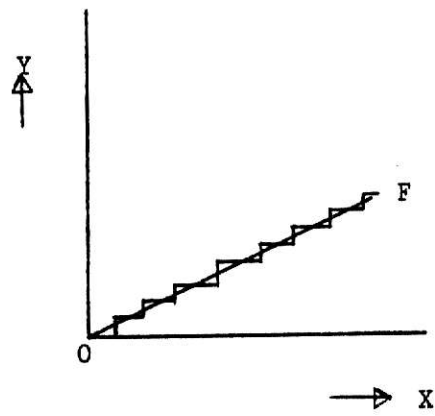
Case 3 : Minimize the maximum absolute deviation of points on one side of a line. No deviations are permissible on the opposite side of a line. In this case the generated path also stays only on one side of a line but opposite to that of Case 2. An example of Case 3 is shown in Fig.2.2c.

An application of Case 2 and Case 3 lies in the machining of a contour of a male or a female part.

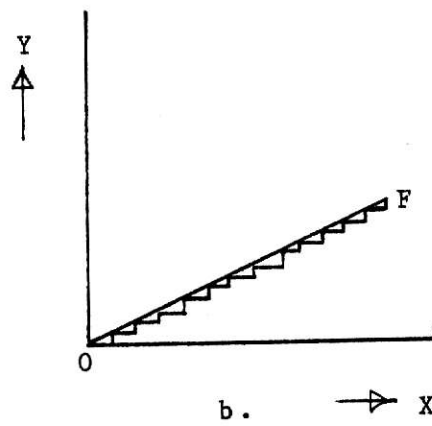
2.1.1 Case 1

For the development of the algorithm for this case, the steps along the two axes are chosen so as to minimize the maximum absolute deviation of points on both sides of a line.

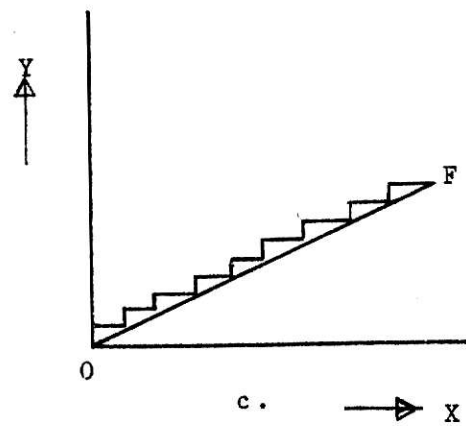
Consider a line OF as shown in Fig.2.3. Let the coordinates of the point F be (X,Y) , and the slope of the line be $C(=Y/X)$.



a.



b.



c.

Fig.2.2 Lines in a plane : a. Case 1, b. Case 2
c. Case 3.

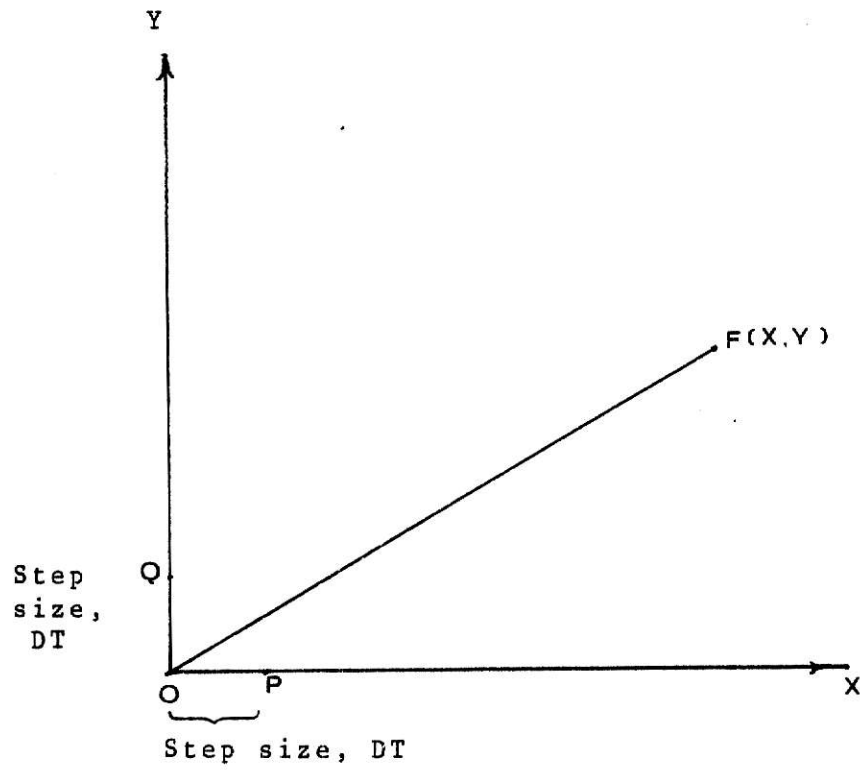


Fig. 2.3 Algorithm for lines in a plane : Case 1

Now, if the motion is to be generated along this line then the incremental steps are taken along the X and Y direction, starting from the point O until the end point of the line is reached. The following steps explain the procedure for generating the motion along the line OF so as to minimize the maximum absolute deviation of the generated points from the line.

Step 1 : Input coordinates (X,Y) of the end point of the line, and the incremental step size DT.

Step 2 : Make X and Y the nearest integer multiples of the step size, DT. This is given by

$$X = (\text{INT}((XN)/M))M/N \quad \text{and}$$

$$Y = (\text{INT}((YN)/M))M/N$$

where $N=1000$ and $M=1000DT$.

Step 3 : Find the number of steps required to reach the end point of the line. This is given by

$$F = (\text{ABS}(X) + \text{ABS}(Y)) / DT$$

Step 4 : Set $A=\text{ABS}(X)$, $B=\text{ABS}(Y)$.

Step 5 : If $A=0$ then go to step 10; otherwise go to step 6.

Step 6 : Calculate the slope of the line, $C=B/A$.

Set $X=0$ and $Y=0$.

Step 7 : Set $X=X+DT$ (See point P in Fig.2.3).

Calculate the absolute value of the deviation of the point P from the line OF. This is given by

$$D1 = \left| \frac{CX - Y}{\sqrt{1 + C^2}} \right| \quad \text{.....2.1}$$

Set $X=X-DT$

Step 8 : Set $Y=Y+DT$ (See point Q in Fig.2.3)

Calculate the absolute deviation of the point Q from the line OF. This is given by

$$D_2 = \left| \frac{CX - Y}{\sqrt{1 + C^2}} \right| \quad \dots\dots 2.2$$

Set $Y=Y-DT$.

Step 9 : If $D_1 < D_2$ then go to step 11; otherwise go to step 10.

Step 10 : Go one step in the Y-direction.

Let $Y=Y+DT$.

Let $F=F-1$.

If $F=0$ then go to step 12; otherwise go to step 5.

Step 11 : Go one step in the X-direction.

Let $X=X+DT$.

Let $F=F-1$.

If $F=0$ then go to step 12; otherwise go to step 5.

Step 12 : Stop.

The computer flow diagram for the above algorithm is shown in Fig.2.4. The computer program in the BASIC language is listed in Appendix 2. Based on this program several plots of lines in a plane are made by using a digital plotter. These plots are depicted in Fig.2.5.

2.1.2 Calculation of the maximum deviation for Case 1

Consider a line OF as shown in Fig.2.6, where the slope of the line is denoted by m . Let C be a point at a distance $P \cdot DT$ vertically below the line i.e. the distance $O'C = P \cdot DT$, where DT is a step size and P is a fraction < 1 . (The algorithm will not generate points that are more than DT below or above

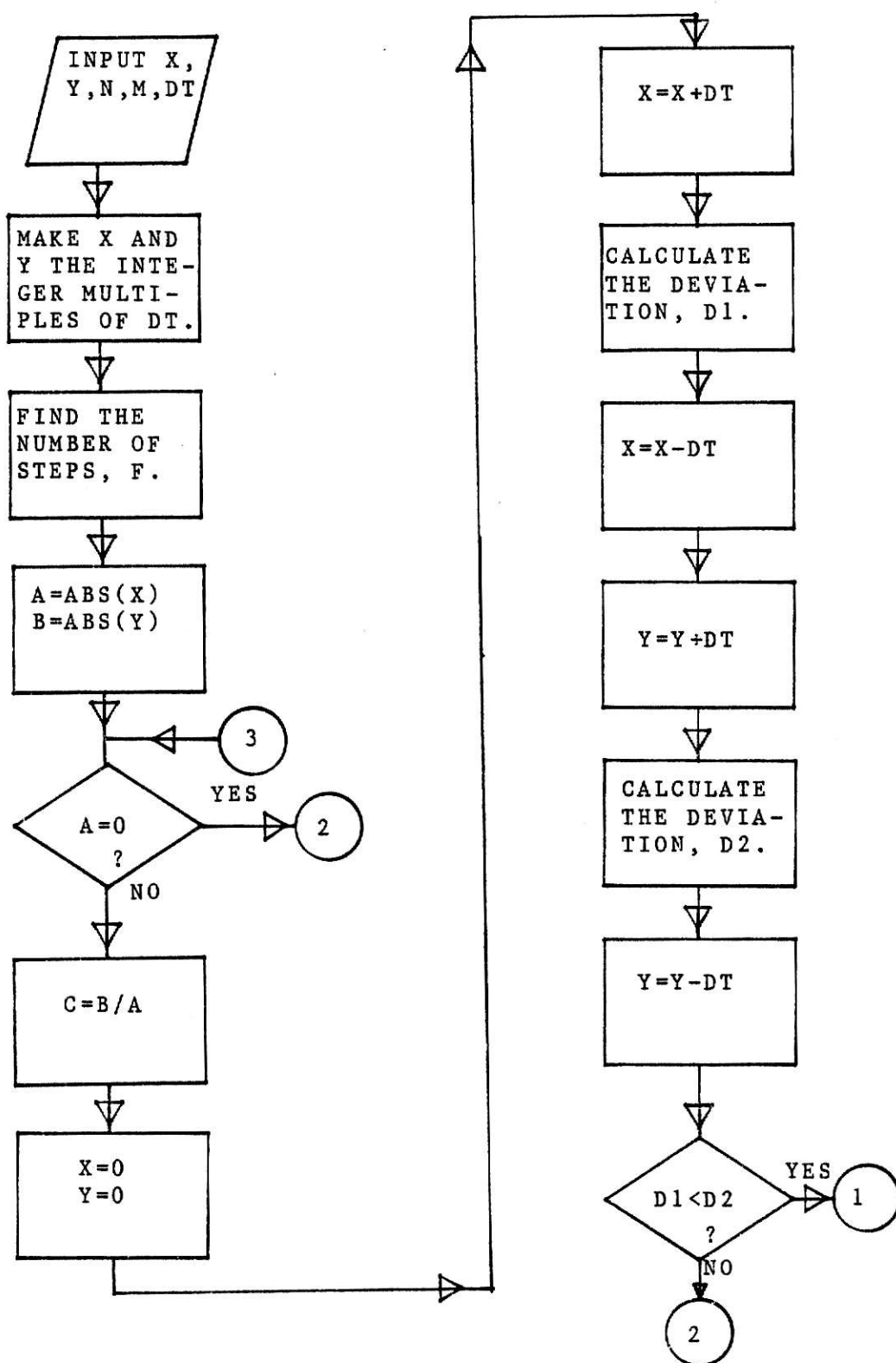


Fig.2.4 Flow diagram for lines in a plane : Case 1.

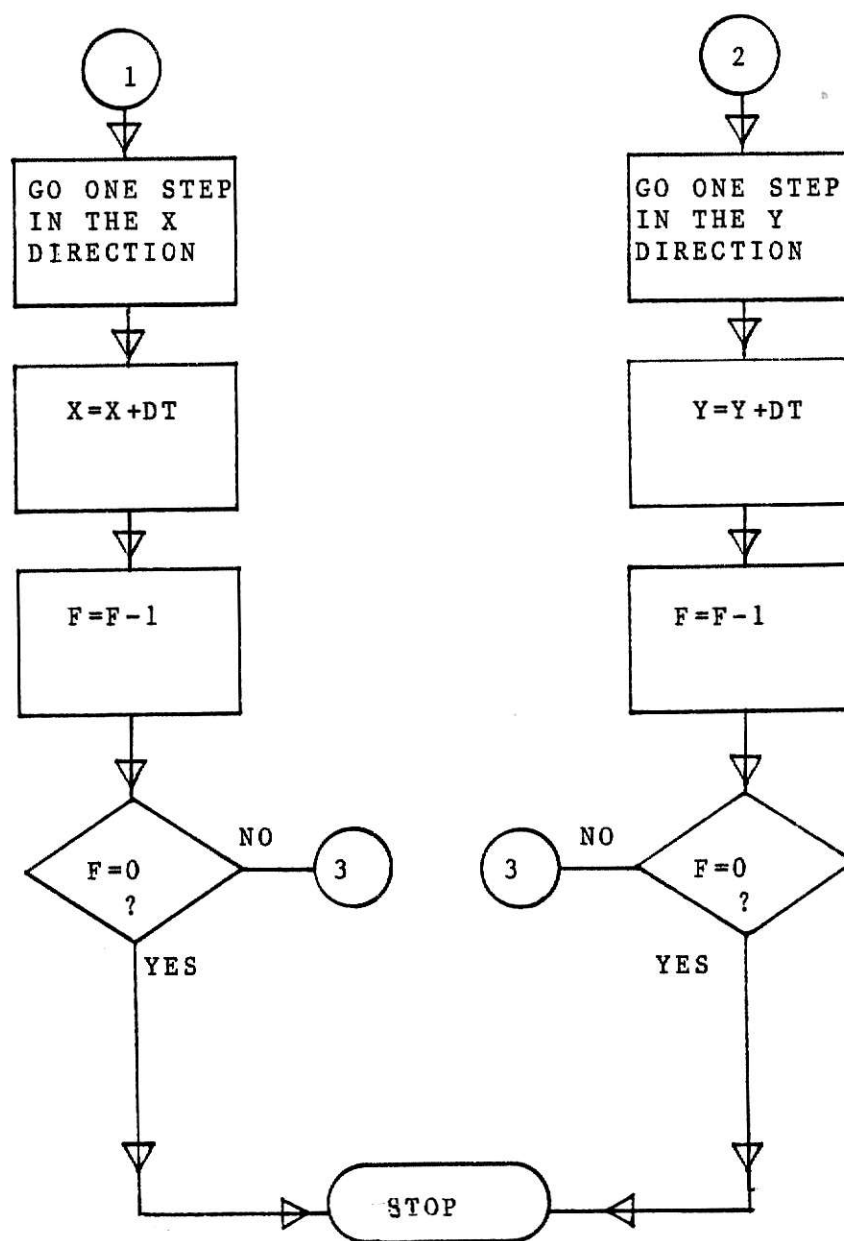


Fig.2.4 (Continued)

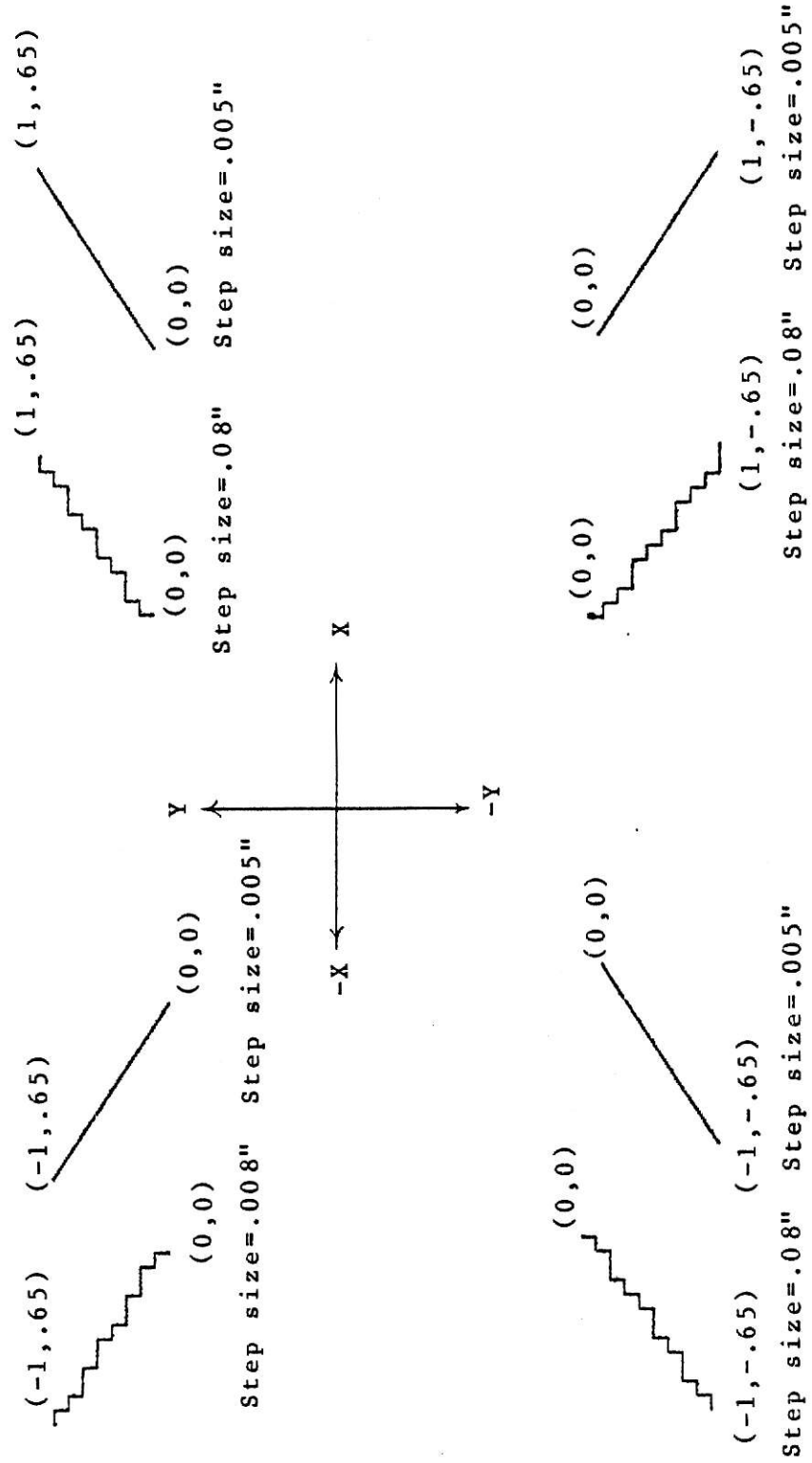


Fig.2.5 Lines in a plane

the line). Let the end point coordinates of the line be given by (F_x, F_y) with respect to the new origin, O' , and a new set of axis $O'X'$ and $O'Y'$.

Now, by giving an incremental step of DT in either the X or Y direction, points D and B are obtained. Their coordinates are given by $(DT, -P.DT)$ and $(0', -P.DT+DT)$, respectively. Let D_1 be the perpendicular distance from D to the line OF and D_2 be the perpendicular distance from B to the line OF . The values of D_1 and D_2 are given by

$$\begin{aligned} D_1 &= \frac{m.DT + P.DT}{\sqrt{1+m^2}} \\ &= \frac{DT(P+m)}{\sqrt{1+m^2}} \end{aligned} \quad \text{.....2.3}$$

and

$$\begin{aligned} D_2 &= \frac{DT - P.DT}{\sqrt{1+m^2}} \\ &= \frac{DT(1-P)}{\sqrt{1+m^2}} \end{aligned} \quad \text{.....2.4}$$

If we always choose the direction of movement by the minimum of D_1 or D_2 , then the maximum deviation occurs when $D_1 = D_2$. Hence, equating D_1 and D_2 , we get

$$\frac{DT(P+m)}{\sqrt{1+m^2}} = \frac{DT(1-P)}{\sqrt{1+m^2}}$$

$$\text{i.e.} \quad 2P = 1-m$$

$$\text{i.e.} \quad P = \frac{(1-m)}{2} \quad \text{.....2.5}$$

Equation 2.5 shows that that when the slope of the line is less than 1 i.e. $m < 1$, the value of P is always positive when the deviation (error) is maximum. This indicates that the point C in Fig.2.5 must be below the line if the slope of the line is

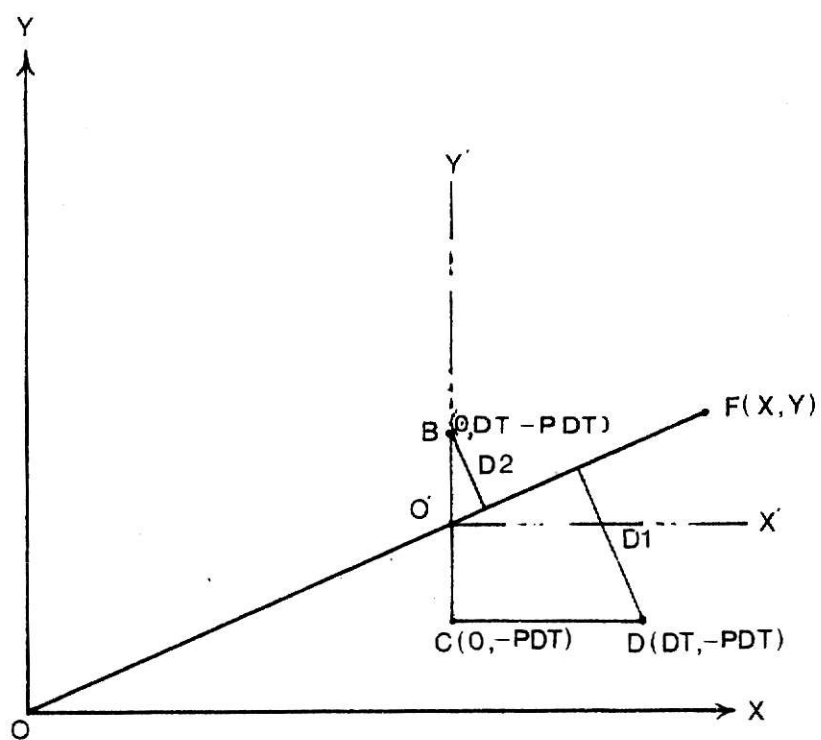


Fig.2.6 Maximum deviation for lines in a plane

less than 1 and the deviation is maximum. However, if m is greater than 1 then the value of P is negative thereby indicating that the point C must lie above the line if the deviation is maximum.

The maximum deviation " D " for different values of $m(<1)$ are shown in Table 2.1. It can be seen from this Table that as the value of m increases, the deviation D increases. The maximum deviation occurs when $m=1$ and $P=0$. Substituting these values of m and P in Equation 2.3, we get

$$\text{maximum deviation } D = \frac{DT}{\sqrt{2}}.$$

If $DT=.005$ ", then

$$\begin{aligned} D &= \frac{.005}{\sqrt{2}} \\ &= 3.53553E-03 \end{aligned}$$

2.1.3 Cases 2 and 3

The stepped paths for Cases 2 and 3 for the generation of motions along a line in a plane are shown in Fig.2.7a and Fig. 2.7b, respectively. In both Cases, the steps are chosen to minimize the maximum absolute deviation of the generated points from the line. Deviations are permitted on one side of the line only. The deviation on the other side is zero.

The procedure developed for Case 2 algorithm is described below with reference to Fig.2.8.

Step 1 : Input coordinates (X,Y) of the end point of the line, and the step size, DT .

Step 2 : Make X and Y the nearest integer multiples of the step size, DT . These are given by

 Table 2.1 Maximum error for lines in a plane

DT=.005"

<u>P</u>	<u>M</u>	<u>D</u>
.45	.1	2.73635E-03
.40	.2	2.94174E-03
.35	.3	3.11294E-03
.30	.4	3.24967E-03
.25	.5	3.3541E-03
.20	.6	3.42997E-03
.15	.7	3.48174E-03
.10	.8	3.51391E-03
.05	.9	3.53065E-03

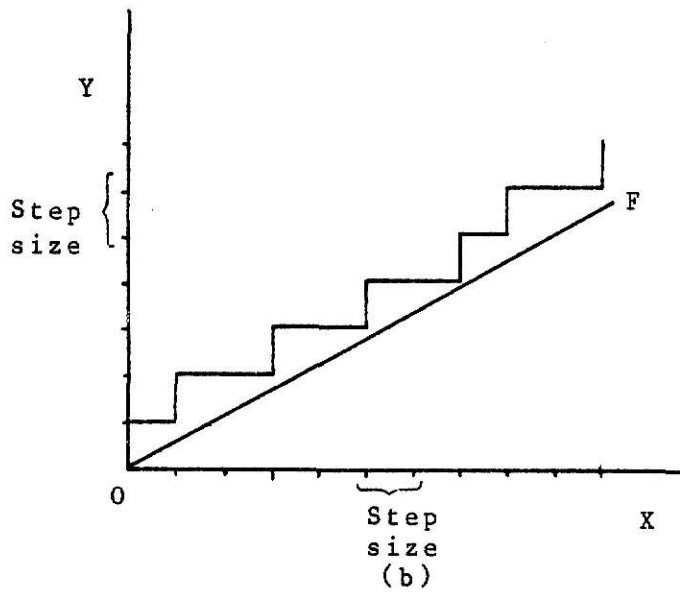
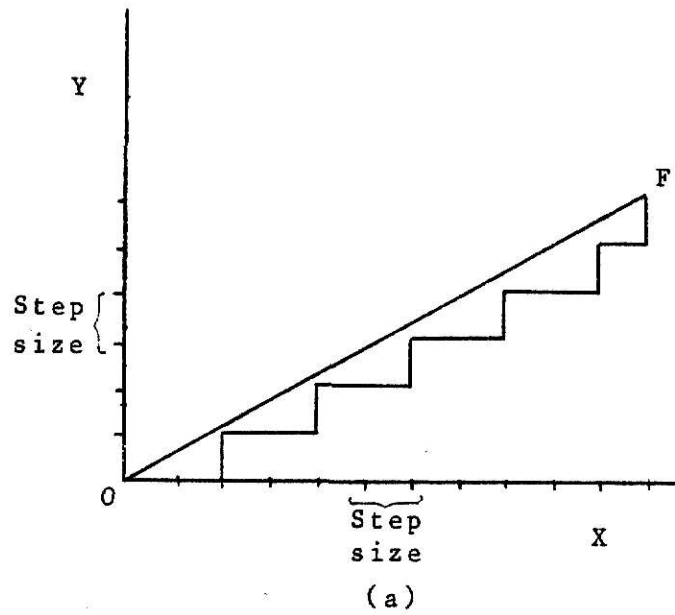


Fig.2.7 . Lines in a plane. (a) Case 2. (b) Case 3.

$$X = (\text{INT}((XN)/M))M/N \quad \text{and}$$

$$Y = (\text{INT}((YN)/M))M/N$$

where $N=1000$ and $M=1000DT$.

Step 3 : Calculate the number of steps required to reach the end point of the line. This is given by

$$F = (\text{ABS}(X) + \text{ABS}(Y)) / DT$$

Step 4 : Initialize $A=0$ and $B=0$.

Step 5 : If $X=0$ then $C=999999$, where C is the slope of the line.
Go to step 8.

If $X \neq 0$ then go to step 6.

Step 6 : Calculate the slope of the line, $C = \text{ABS}(Y) / \text{ABS}(X)$.

Step 7 : Go one step in the X-direction.

Let $F=F-1$

If $F=0$ then go to step 12; otherwise go to step 8.

Step 8 : Let $A=A+DT$

Step 9 : Set $B=B+DT$.

Calculate the slope, $SL=B/A$. (See Fig.2.8).

If $C < SL$ then go to step 11; otherwise go to step 10.

Step 10 : Go one step in the Y-direction.

Let $F=F-1$.

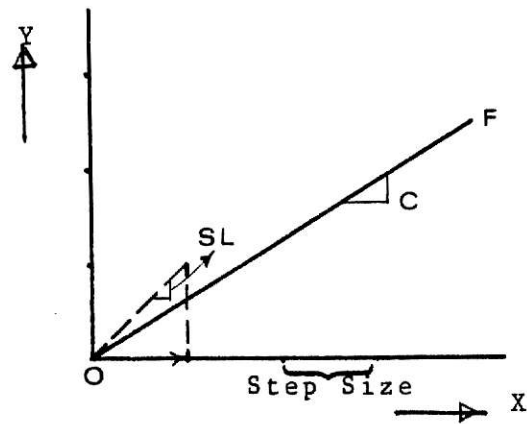
If $F=0$ then go to step 12; otherwise go to step 9.

Step 11 : Set $B=B-DT$.

Go to step 7.

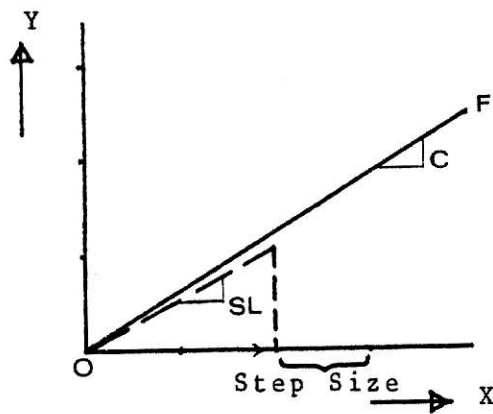
Step 12 : Stop.

The computer flow diagram for the above algorithm is shown in Fig.2.9



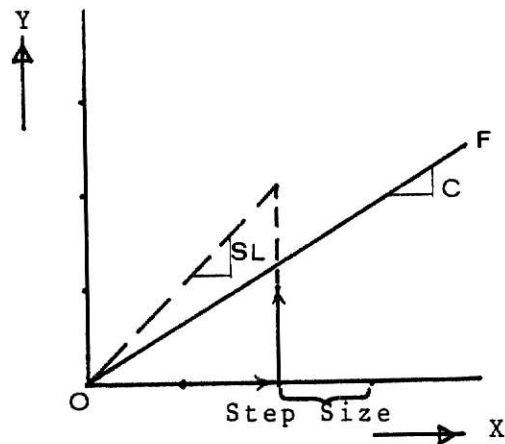
$$SL > C$$

Go one step in the
X-direction.



$$SL < C$$

Go one step in the
Y-direction.



$$SL > C$$

Go one step in the
X-direction

Fig.2.8 Algorithm for lines in a plane : Case 2

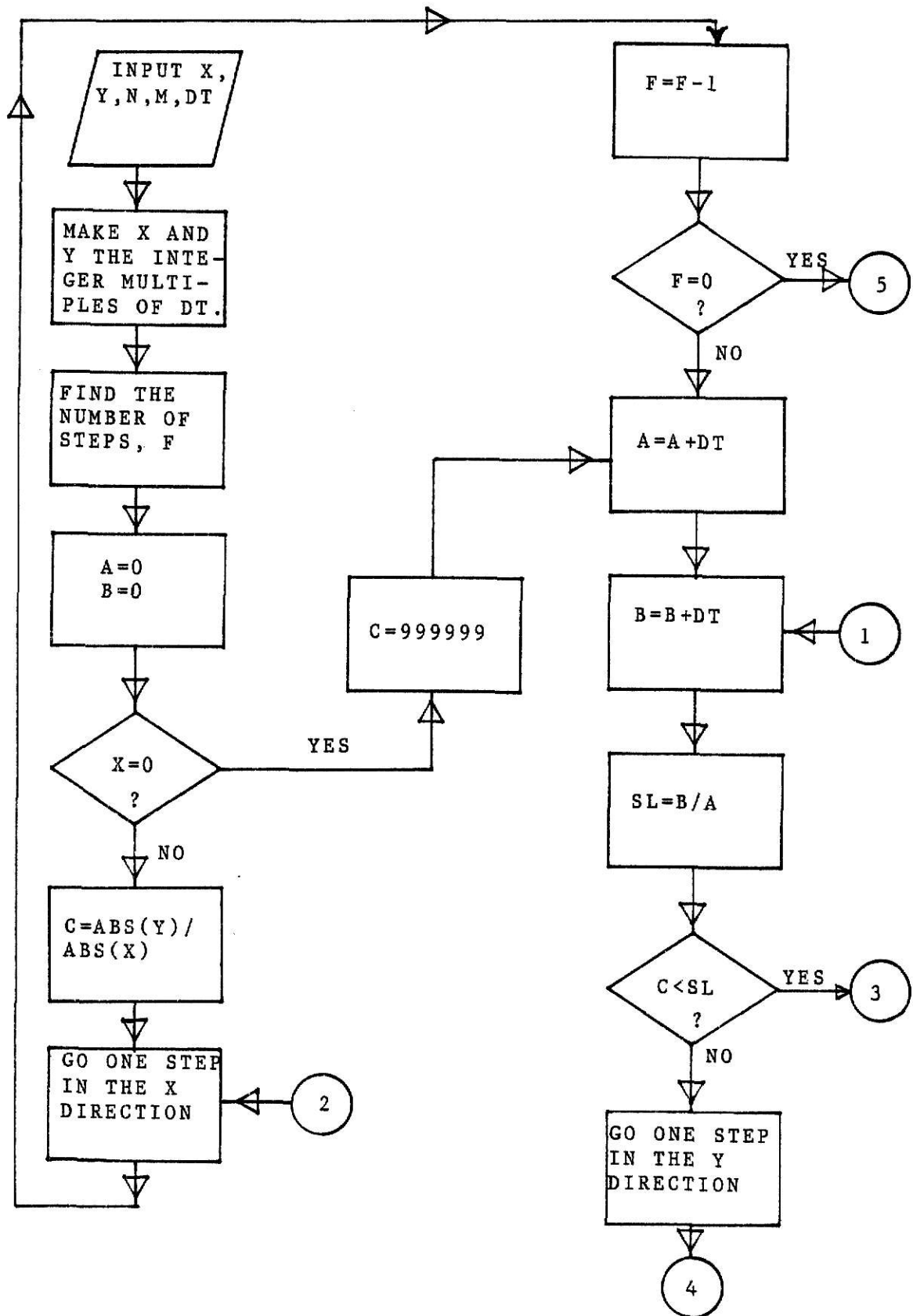


Fig.2.9 Flow diagram for lines in a plane : Case 2.

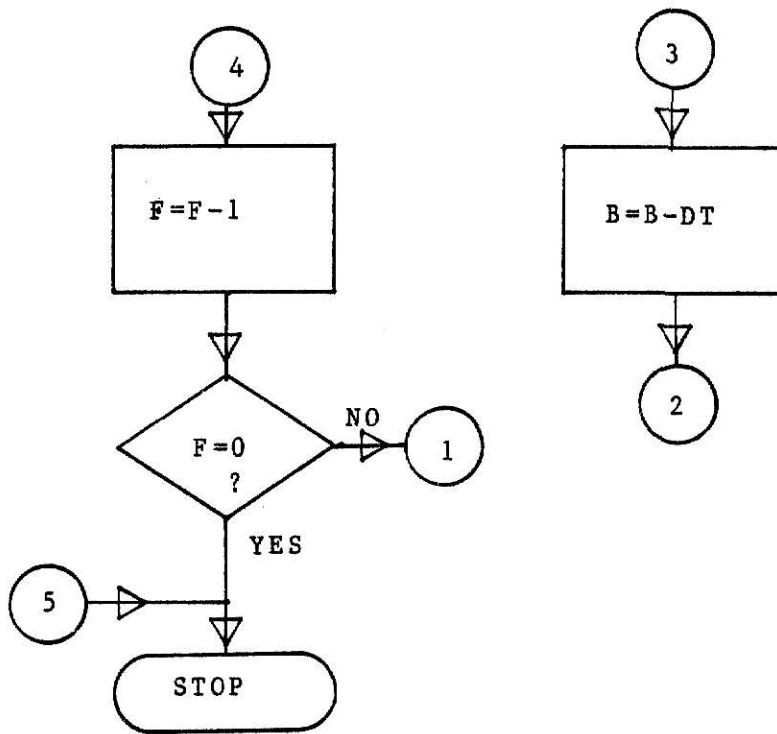


Fig.2.9 (Continued)

The steps required for the development of the algorithm for the generation of the motion along a line for Case 3 are similar to that for Case 2 and are explained below. (See Fig.2.10).

Step 1 : Input coordinates (X,Y) of the end point of the line and the step size, DT.

Step 2 : Make X and Y the nearest integer multiples of the step size, DT. This is given by

$$X = (\text{INT}((XN)/M))M/N \quad \text{and}$$

$$Y = (\text{INT}((YN)/M))M/N$$

where $N=1000$ and $M=1000DT$.

Step 3 : Calculate the number of steps required to reach the end point of the line. This is given by

$$F = (\text{ABS}(X) + \text{ABS}(Y)) / DT$$

Step 4 : Initialize $A=0$ and $B=0$.

Step 5 : If $X=0$ then $C=999999$, where C is the slope of the line. Go to step 8.

If $X \neq 0$ then go to step 6.

Step 6 : Calculate the slope of the line, $C=\text{ABS}(Y)/\text{ABS}(X)$.

Step 7 : Go one step in the Y-direction.

Let $F=F-1$.

If $F=0$ then go to step 12; otherwise go to step 8.

Step 8 : Let $B=B+DT$.

Step 9 : Set $A=A+DT$.

Calculate the slope, $SL=B/A$. (See Fig.2.10).

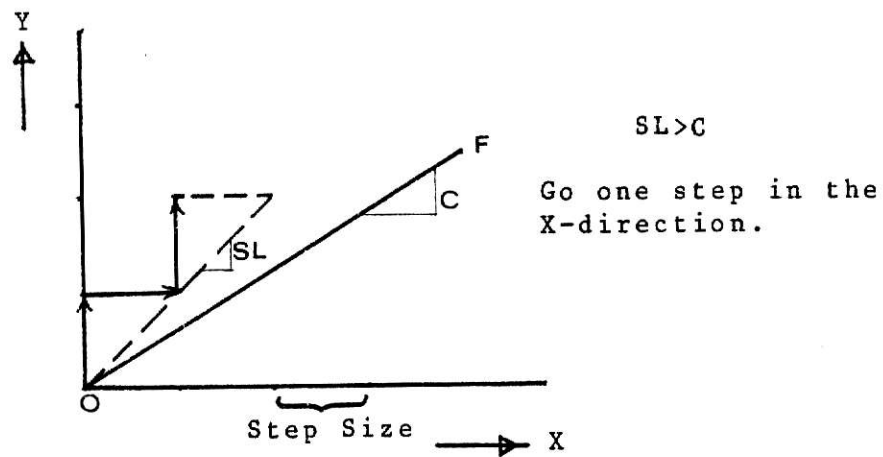
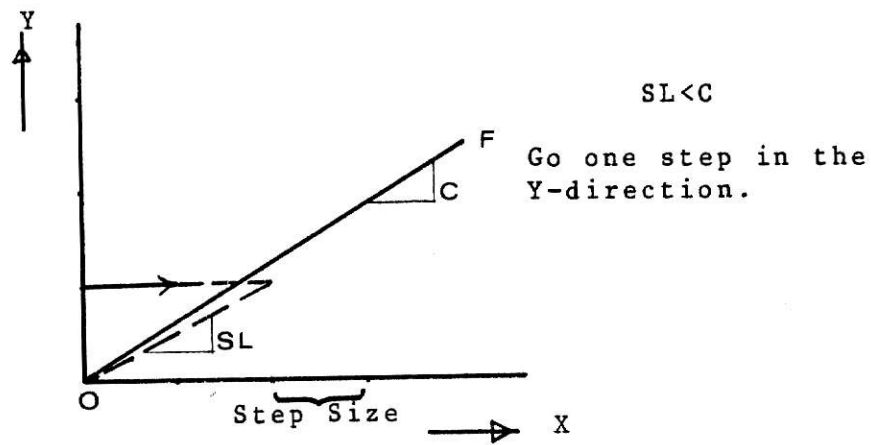
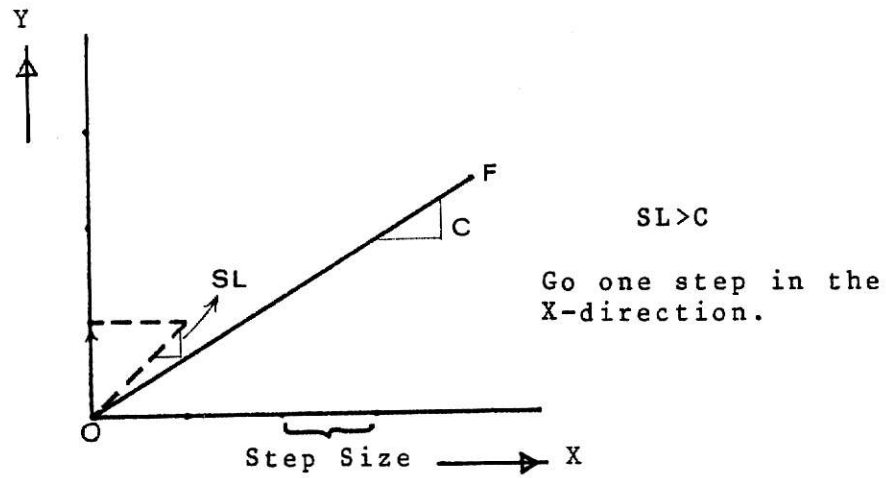


Fig.2.10 Algorithm for lines in a plane : Case 3

If $C > SL$ then go to step 11; otherwise go to step 10.

Step 10 : Go one step in the X-direction.

Let $F = F - 1$.

If $F = 0$ then go to step 12; otherwise go to step 9.

Step 11 : Set $A = A - DT$.

Go to step 7.

Step 12 : Stop.

The computer flow diagram for the above algorithm is shown in Fig.2.11. The listing of the computer program in the BASIC language for cases 2 and 3 is given in Appendix 1. This program uses an input variable "TL" which decides which one of the two cases should be used for generating the motions along lines in a plane. For example, consider a part as shown in Fig.2.12a to be machined along its sides starting from the point 0. If we want to have the motion of a tool as shown in Fig.2.12a then by setting the value of the input variable "TL" in the computer program to any number other than 1, the desired motion of a tool will be generated. If however, the sides are required to be machined with a negative tolerance (See Fig.2.12b) then the value of the input variable should be set equal to 1. If the part is required to be machined in the clockwise direction and a negative tolerance is required to be maintained along its sides (See Fig.2.12c) then the value of TL should be set to any number other than 1. Finally, if the part is required to be machined in the clockwise direction with a positive tolerance along its sides, then this can be achieved by setting the value of the input variable TL equal to 1.

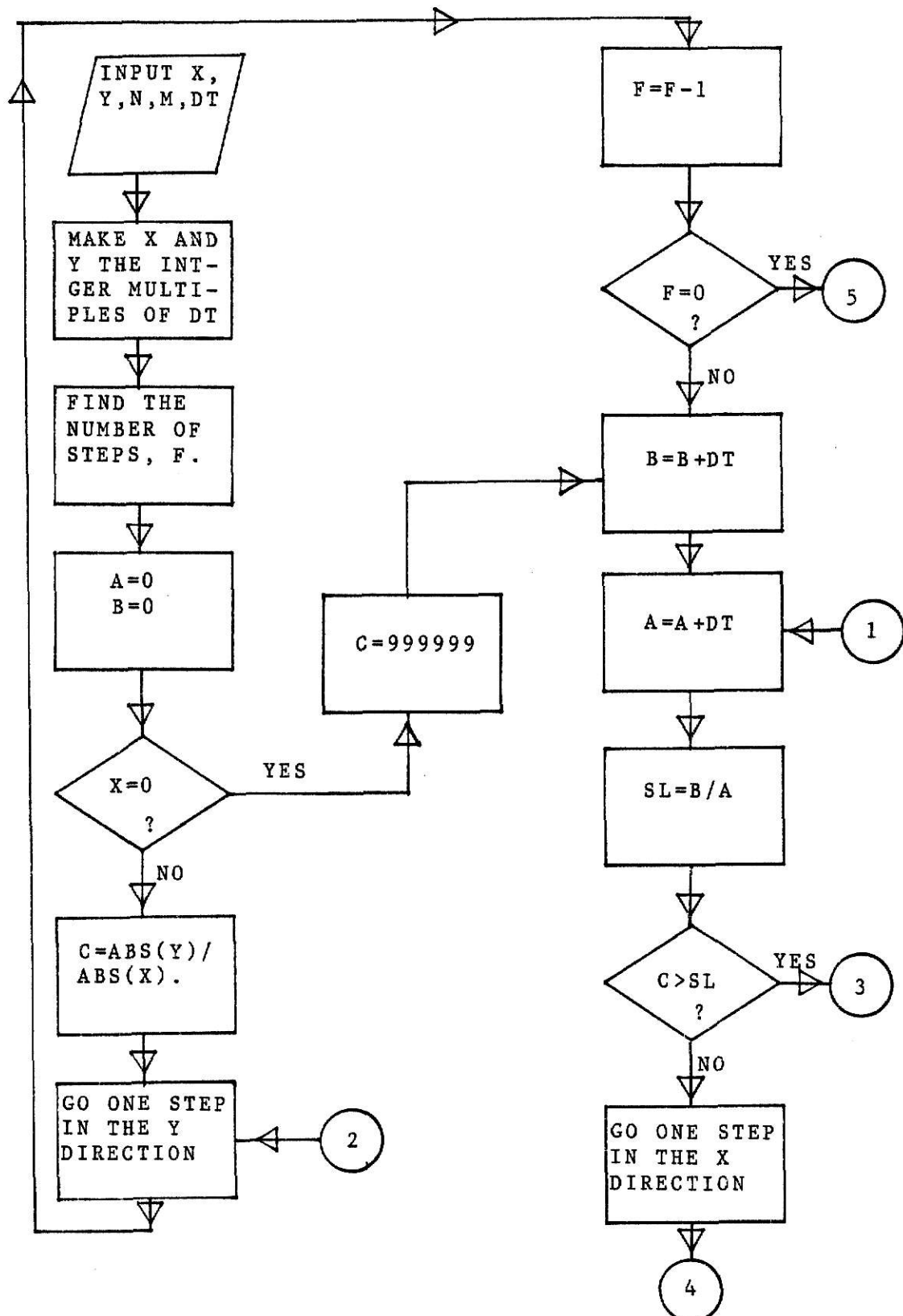


Fig.2.11 Flow diagram for lines in a plane : Case 3

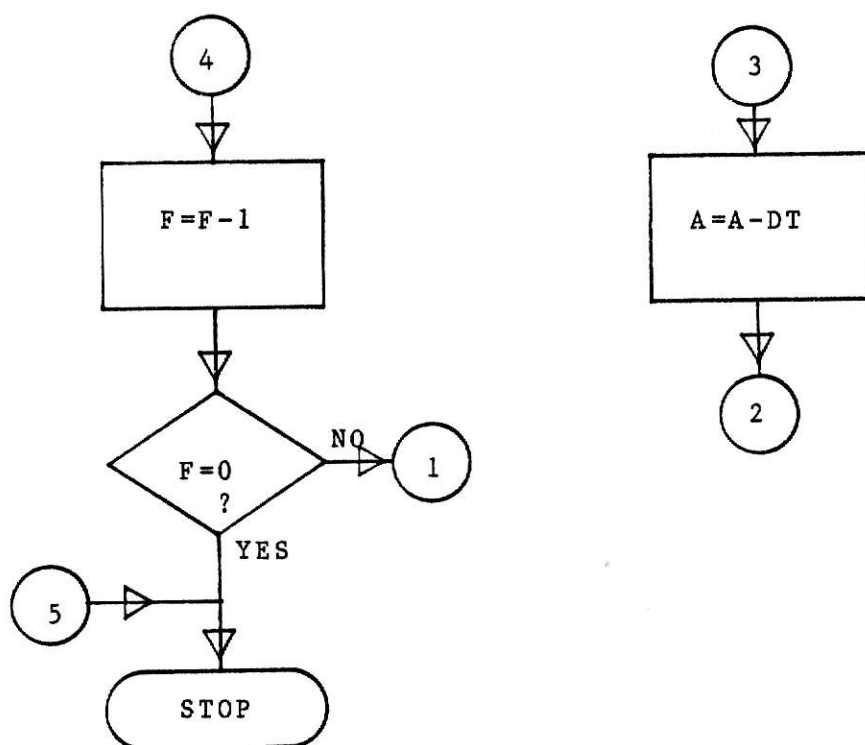


Fig. 2.11 (Continued)

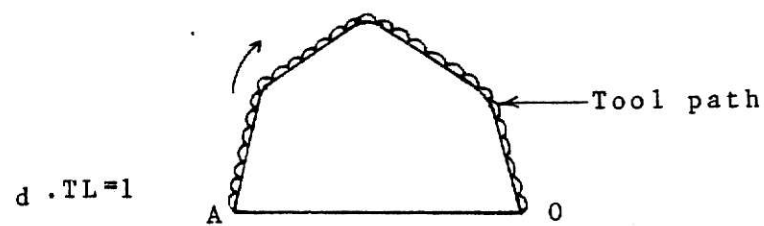
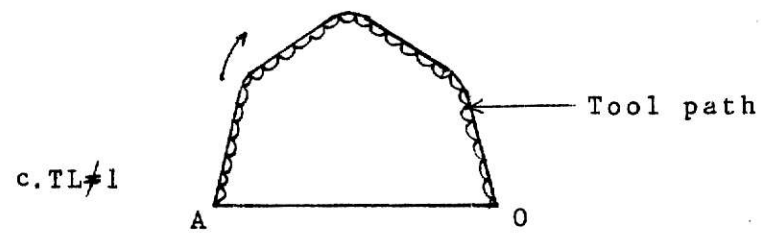
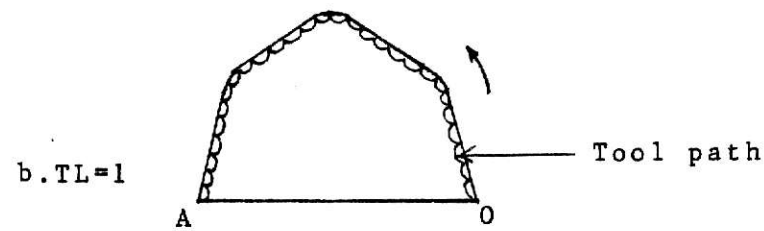
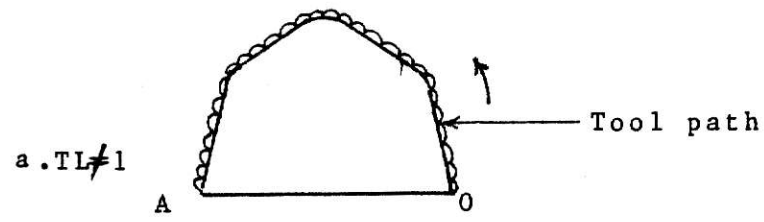


Fig.2.12 Examples of Cases 2 and 3.

Based on the computer program listed in Appendix 1 for Case 2 and 3, several plots of lines are made for $TL=1$ and $TL \neq 1$. These plots are depicted in Fig.2.13 and Fig.2.14.

2.1.4 A practical application of Cases 2 and 3.

Consider two matching parts A and B as shown in Fig.2.15 to be machined on a milling machine. In order that the two parts match perfectly, it is necessary that there should be negative tolerances along the sides QPSR and $Q'P'S'R'$. Hence, for the purpose of matching of these sides, it is preferred that the tool should follow the path as shown in Fig.2.15.

If the side $Q'P'S'R'$ is to be machined (starting from Q') with a negative tolerance then this can be achieved by setting the value of the input variable TL in the computer program to any number other than 1. If the same side is to be machined in the clockwise direction (starting from R') then a negative tolerance on this side can be obtained by setting the value of TL equal to 1. Similarly, the side QPRS can be machined with a negative tolerance.

2.2 LINES IN SPACE

For the development of the algorithm for the generation of the motion along lines in space, the incremental steps are taken along the X, Y and Z directions starting from one end of the line until the other end of the line is reached. The steps along the direction of the three axes are chosen so as to minimize the maximum absolute deviation of the generated

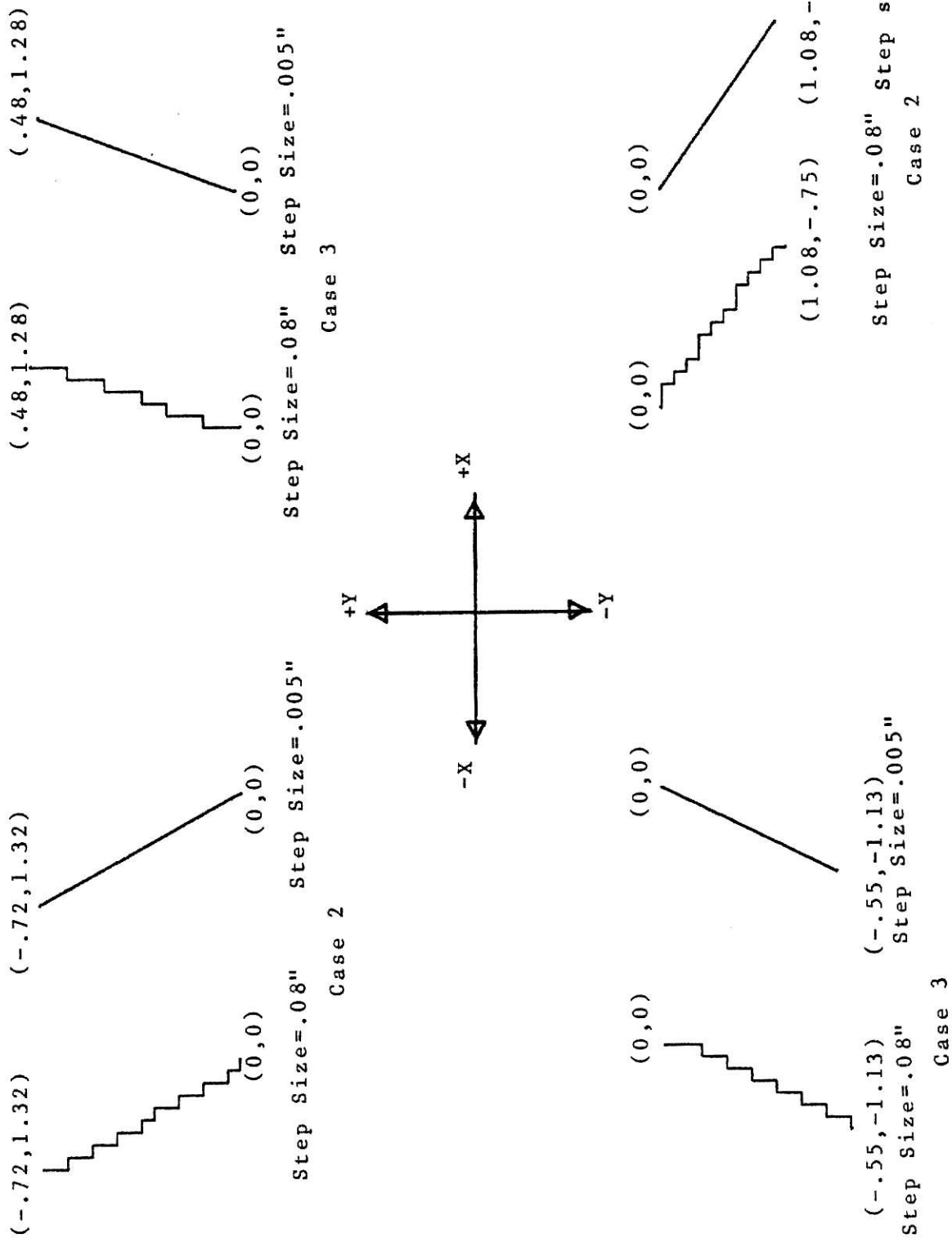


Fig.2.13 Lines in a plane : TL=1

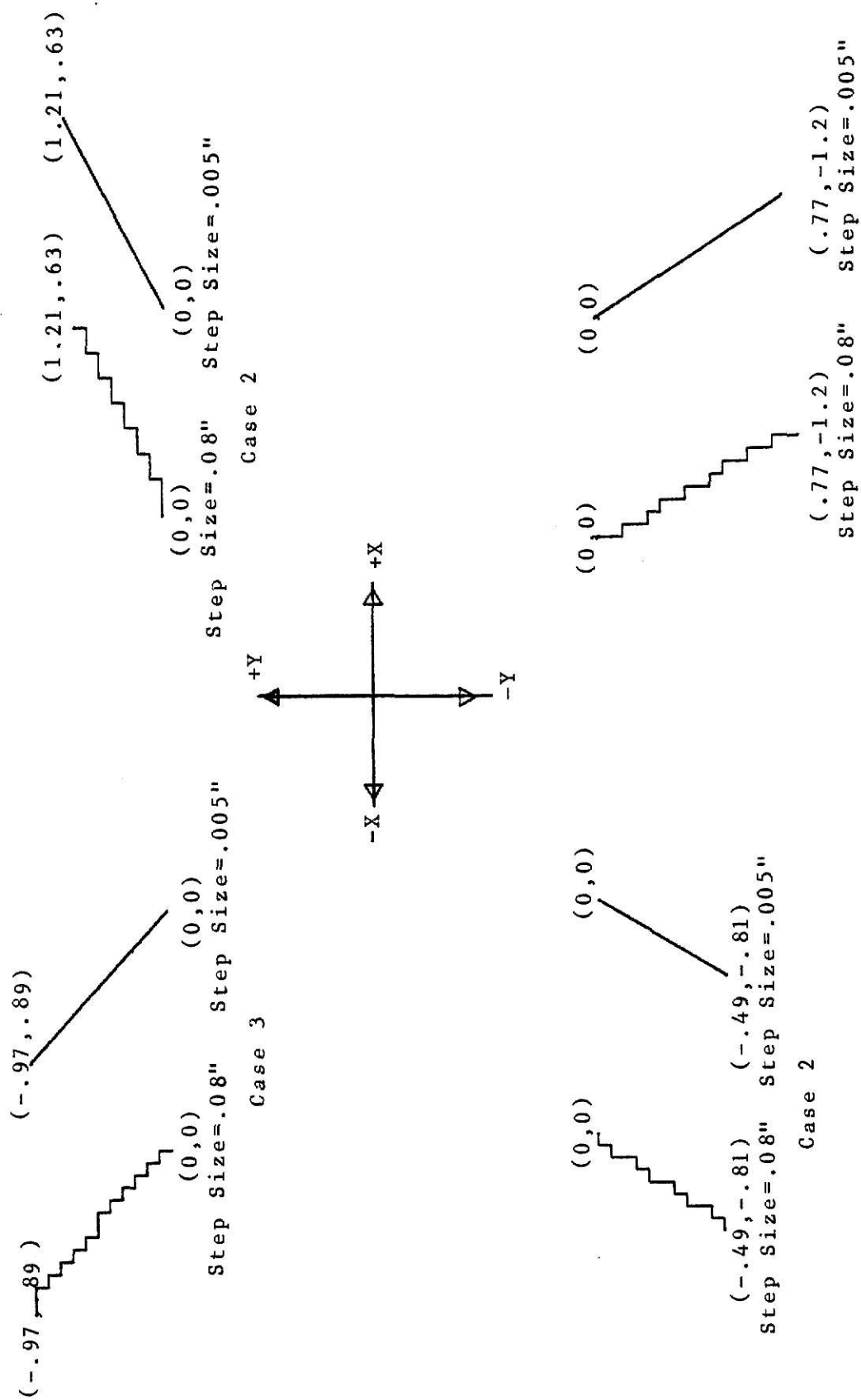


Fig. 2.14 Lines in a plane : TL#1

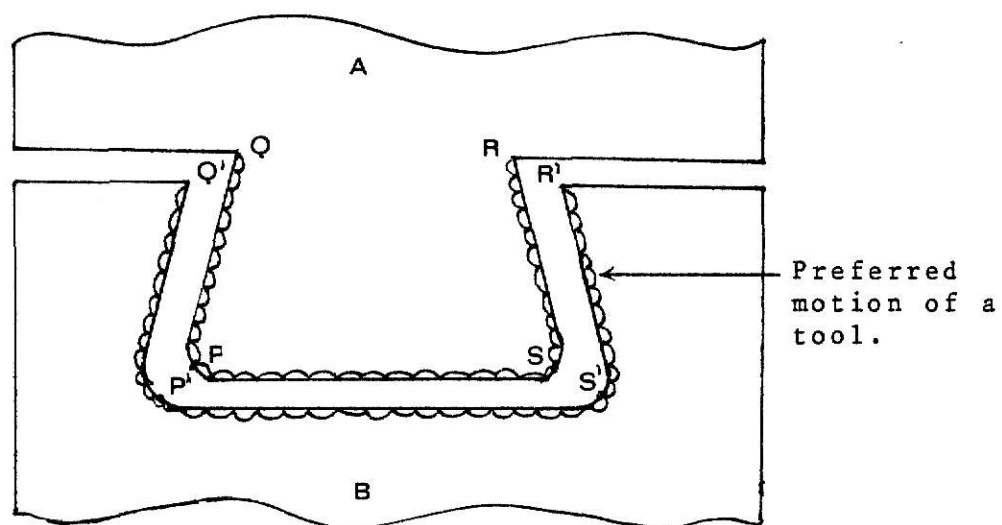


Fig.2.15 Practical application of Cases 2 and 3

points from the line. Since the selection of the direction of motion at each incremental step is based on the comparison of the deviations of the generated points from the line, it is necessary to derive a formula to calculate the deviation of a point in space to a line in space.

2.2.1 To calculate the distance of a point in space to a line in space

Consider a line OA as illustrated in Fig.2.16. The coordinates of the point A are (A_1, A_2, A_3) . Let P be a point in space whose coordinates are (P_1, P_2, P_3) . Draw a line passing through P and perpendicular to OA to meet OA at M. Let the distance PM be denoted by DE.

In order to find DE, we need to find the scalar projection of $V(\vec{OP})$ onto $V(\vec{OA})$. This is given by

$$\frac{V(\vec{OP}) \cdot V(\vec{OA})}{|V(\vec{OA})|} = \frac{(P_1, P_2, P_3) \cdot (A_1, A_2, A_3)}{\sqrt{A_1^2 + A_2^2 + A_3^2}}$$

This represented by $|\vec{OM}|$ in Fig.2.16. The distance DE can be calculated by using the Pythagorean theorem. This is given by

$$DE = \sqrt{|\vec{OP}|^2 - |\vec{OM}|^2}$$

where

$$|\vec{OP}| = \sqrt{P_1^2 + P_2^2 + P_3^2}$$

The above relationship is made use of for the development of the algorithm for the generation of motion along lines in space.

2.2.2 Algorithm for generating the motions on lines in space

The following steps describe in detail the procedure developed for the generation of the motions along lines in space. (Refer to Fig.2.17)

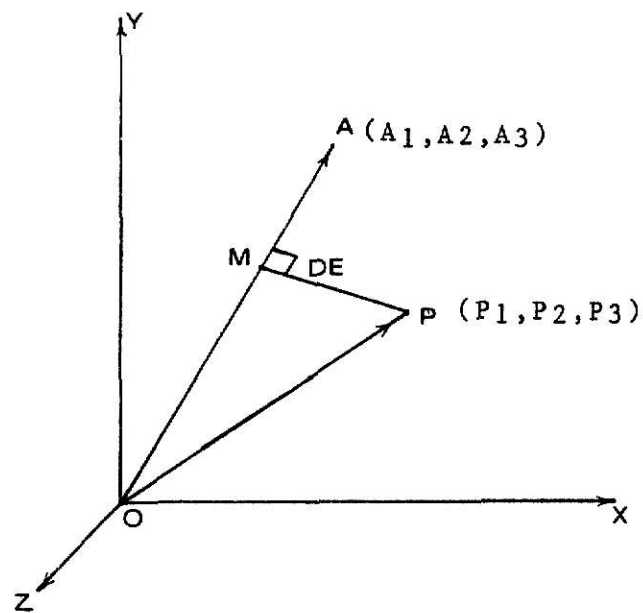


Fig.2.16. Distance of a point in space to a line in space.

Step 1 : Input coordinates (X,Y,Z) of the end point of the line and the step size, DT.

Step 2 : Make X, Y and Z the nearest integer multiples of the step size DT. These are given by

$$X = (\text{INT}((XN)/M))M/N$$

$$Y = (\text{INT}((YN)/M))M/N \quad \text{and}$$

$$Z = (\text{INT}((ZN)/M))M/N$$

where $N=1000$ and $M=1000DT$.

Step 3 : Find the number of steps required to reach the end point of the line. This is given by

$$F = (\text{ABS}(X)+\text{ABS}(Y)+\text{ABS}(Z))/DT$$

Step 4 : Set $A=\text{ABS}(X)$, $B=\text{ABS}(Y)$ and $C=\text{ABS}(Z)$.

Step 5 : Set $X=0$, $Y=0$ and $Z=0$.

Step 6 : If $A=0$ then go to step 9; otherwise go to step 7.

Step 7 : Set $X=X+DT$. (See point L in Fig.2.17)

Calculate the deviation of point L from the line. This is given by

$$DE = \sqrt{X^2+Y^2+Z^2 - \frac{(AX+BY+CZ)^2}{A^2+B^2+C^2}}$$

Set $D1=DE$

Step 8 : Set $X=X-DT$.

Go to step 10.

Step 9 : Set $D1=1000$.

Step 10 : If $B=0$ then go to step 13; otherwise go to step 11.

Step 11 : Set $Y=Y+DT$. (See point M in Fig.2.17).

Calculate the deviation of the point M from the line. This is

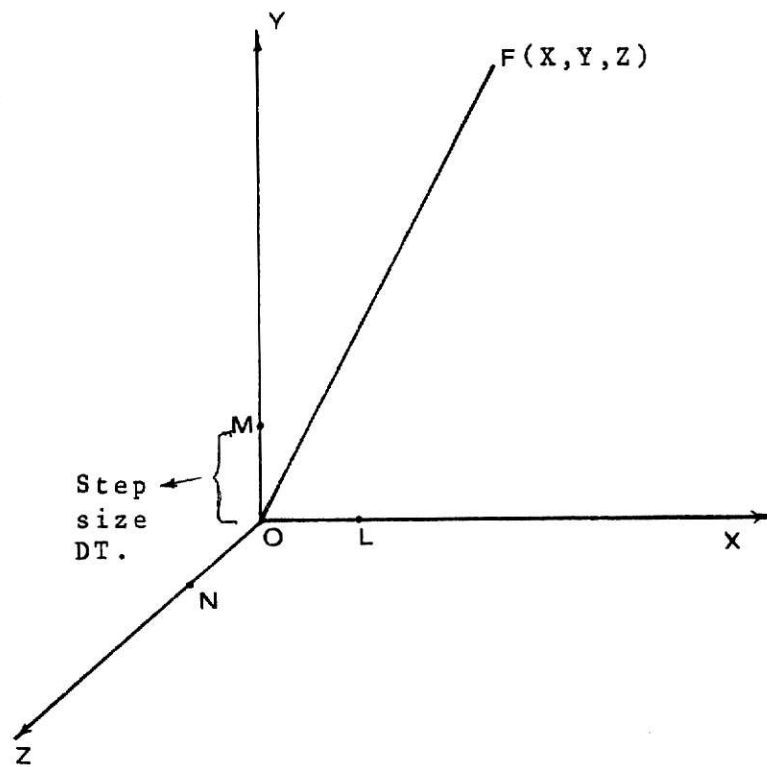


Fig.2.17. Algorithm for lines in space.

given by

$$DE = \sqrt{X^2 + Y^2 + Z^2 - \frac{(AX + BY + CZ)^2}{A^2 + B^2 + C^2}}$$

Set $D2 = DE$.

Step 12 : Set $Y = Y - DT$.

Go to step 14.

Step 13 : Set $D2 = 1000$.

Step 14 : If $C = 0$ then go to step 17; otherwise go to step 15.

Step 15 : Set $Z = Z + DT$. (See point N in Fig.2.17).

Calculate the deviation of the point N from the line. This is given by

$$DE = \sqrt{X^2 + Y^2 + Z^2 - \frac{(AX + BY + CZ)^2}{A^2 + B^2 + C^2}}$$

Set $D3 = DE$.

Step 16 : Set $Z = Z - DT$.

Go to step 18.

Step 17 : Set $D3 = 1000$.

Step 18 : If $D1$ is less than $D2$ and $D3$ then go to step 19.

If $D2$ is less than $D1$ and $D3$ then go to step 20.

If $D3$ is less than $D1$ and $D2$ then go to step 21

Step 19 : Go one step in the X-direction.

Let $X = X + DT$ and $F = F - 1$.

If $F = 0$ then go to step 22; otherwise go to step 6.

Step 20 : Go one step in the Y-direction.

Let $Y = Y + DT$ and $F = F - 1$.

If $F = 0$ then go to step 22; otherwise go to step 6.

Step 21 : Go one step in the Z-direction.

Let $Z = Z + DT$ and $F = F - 1$.

If $F=0$ then go to step 22; otherwise go to step 6.

Step 22 : Stop.

The computer flow diagram for the above algorithm is shown in Fig.2.18. This algorithm can also be used for the generation of the motion along a line in a plane. The listing of the computer program for the generation of lines in a plane and space is shown in Appendix 1. In order to demonstrate the generation of lines in space, lines are plotted in the X-Y plane by using a digital plotter, where the Z axis is treated as a line passing through the origin and at an angle of 45° to the +X axis. These plots of lines are shown in Fig.2.19.

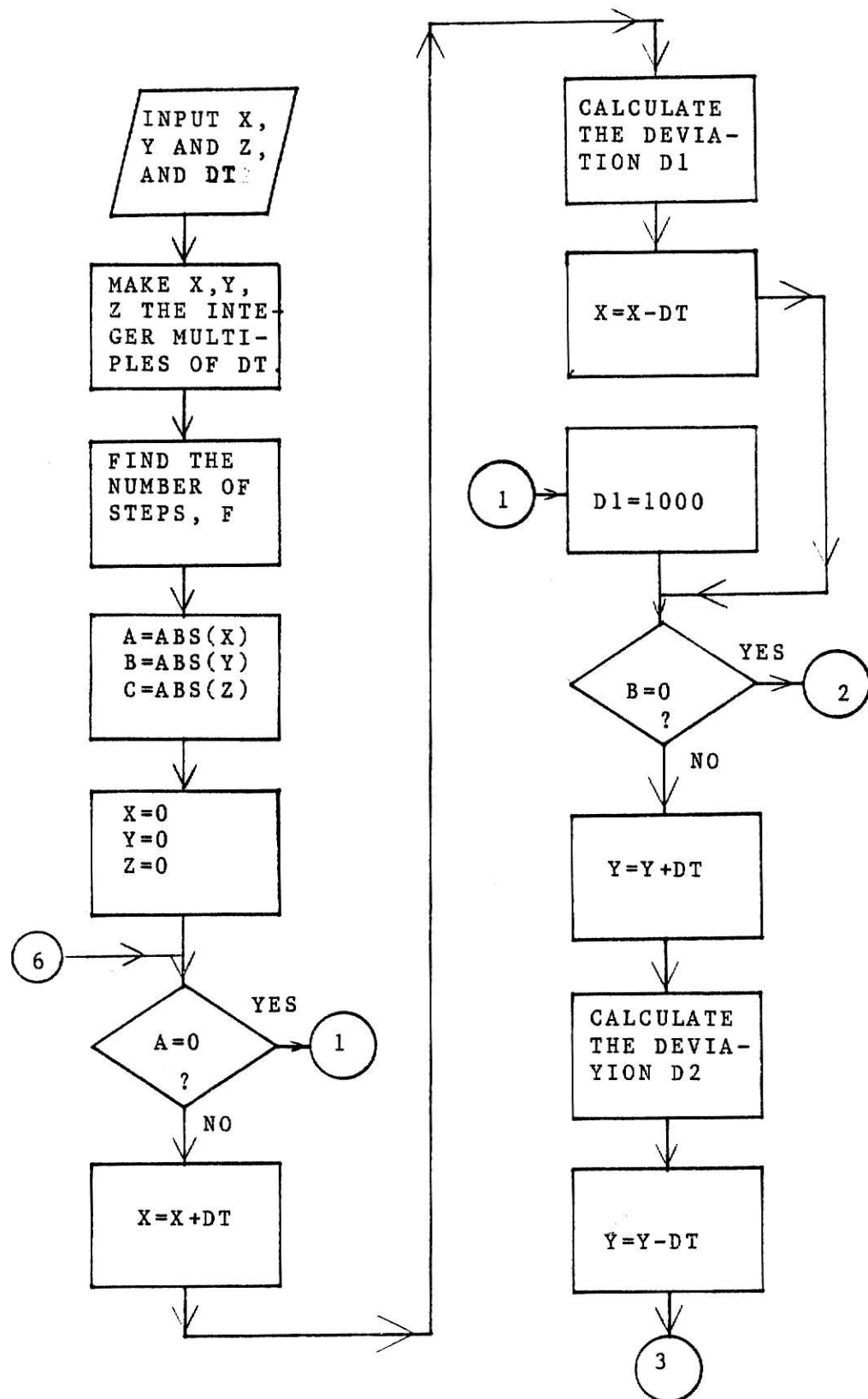


Fig.2.18 Flow diagram for lines in space

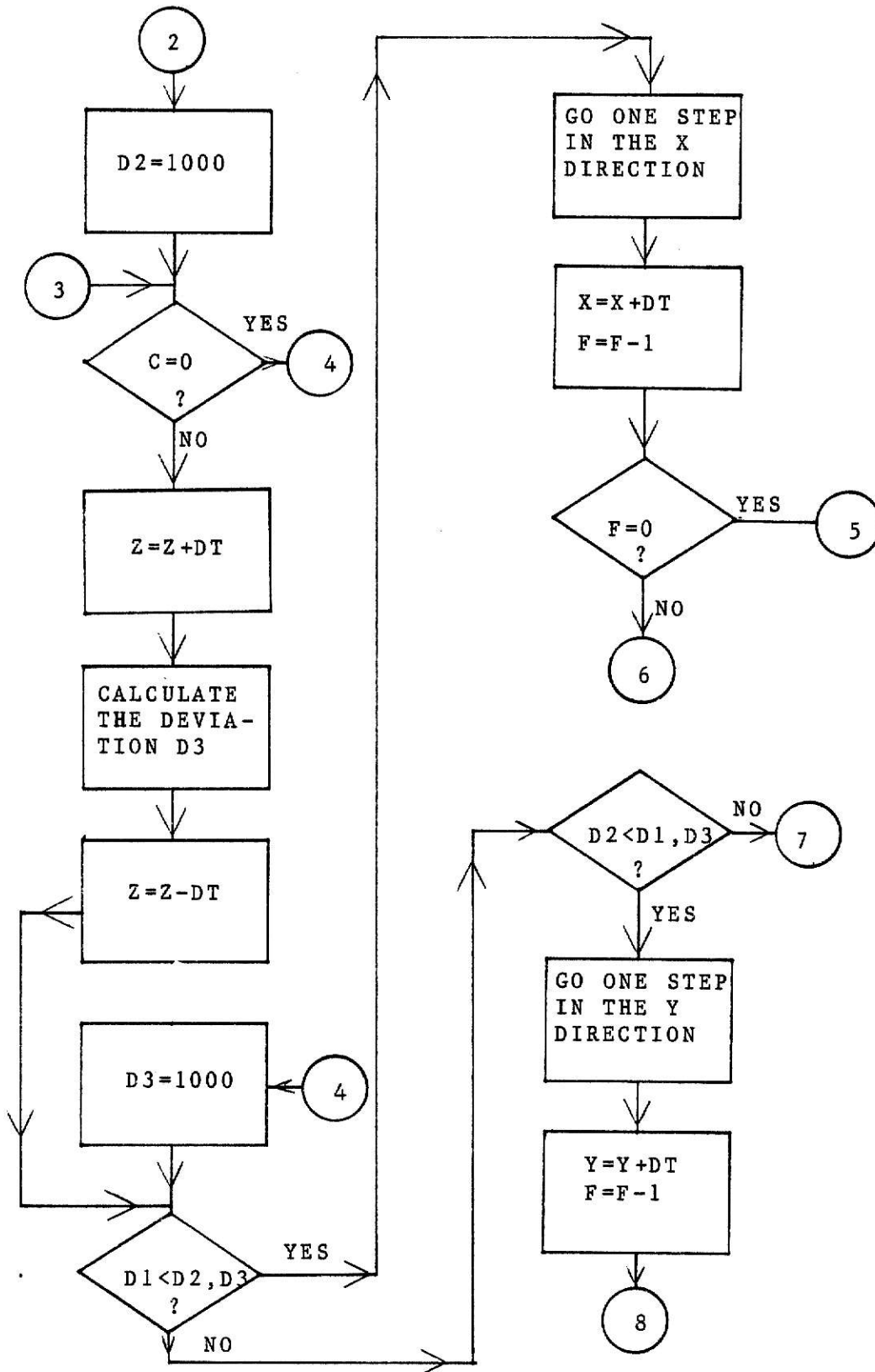


Fig.2.18 (Continued)

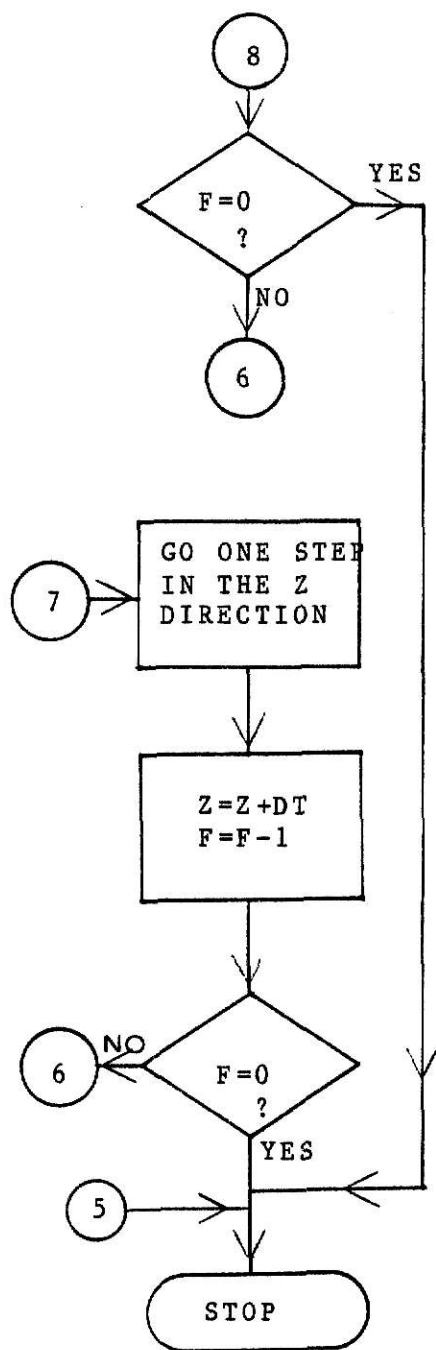


Fig.2.18 (Continued)

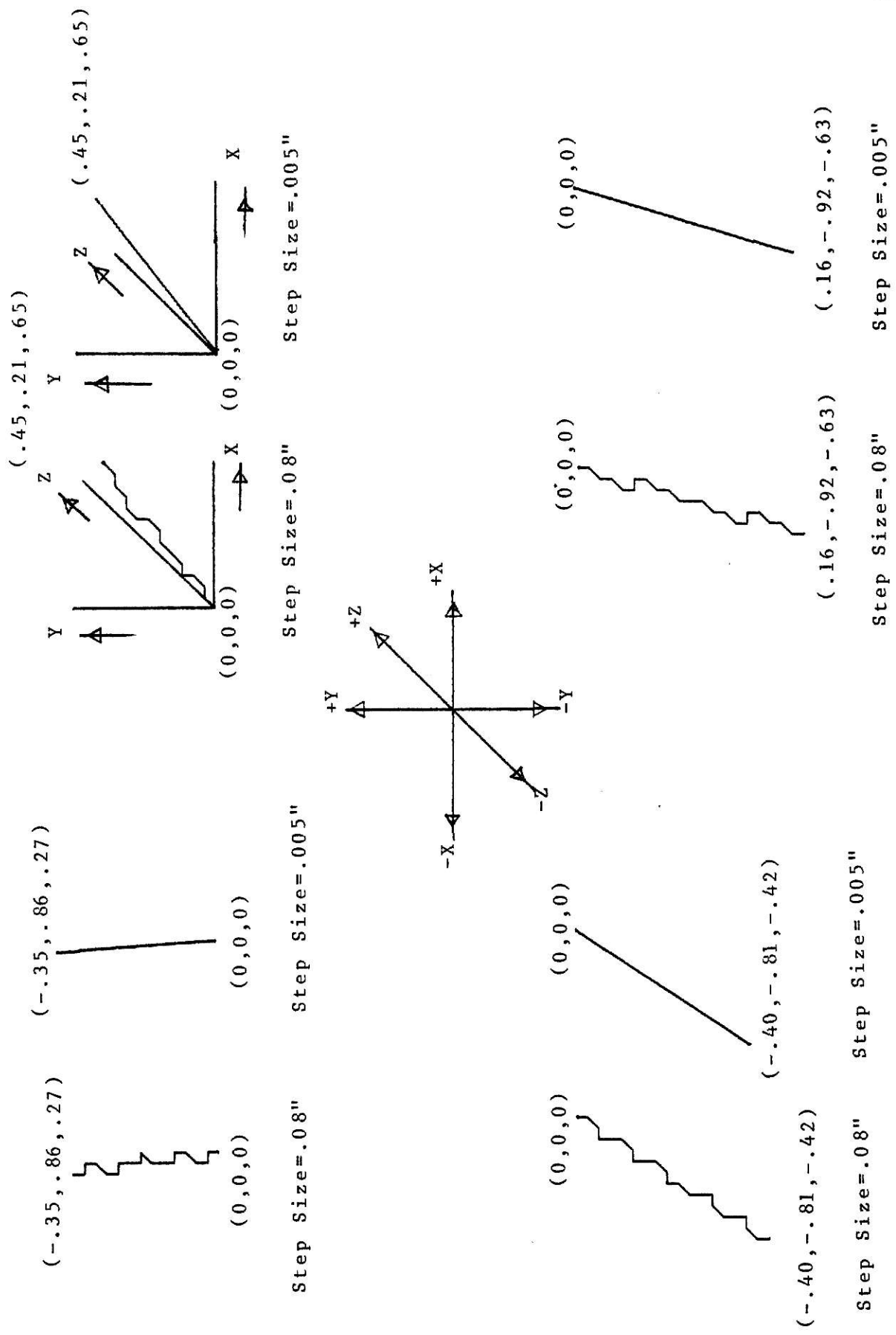


Fig.2.19 Lines in Space

CHAPTER 3
REGULAR CURVES IN A PLANE

REGULAR CURVES IN A PLANE

This chapter describes the algorithms for the generation of motions along some regular curves. The regular curves considered are circles, parabolas, ellipses and hyperbolas. The motions along these curves are generated by making straight line approximations. The programs discussed in this chapter use the programs in Chapter 2. The main programs for the generation of the regular curves call the subroutine programs presented in Chapter 2. These subroutine programs can be for any one of the three cases for the generation of motions along a line in a plane.

3.1 CIRCULAR ARCS

Consider a circular arc of length $r\theta$ illustrated in Fig. 3.1(a). If straight line approximations are to be made for the generation of motion along this arc, there are three possible ways in which this could be done :

- i) The linear segments are chordal to the arc (Chordal method). Refer to Fig.3.1(b).
- ii) The linear segments are tangent to the arc (Tangential method). Refer to Fig.3.1(c).
- iii) The linear segments are calculated such that they intersect the arc leaving equal tolerance inside and outside the arc (Secantial method). Refer to Fig.3.1(d).

Now, if the tolerance value of t is to be maintained on the arc PP_n , the intermediate points on this arc could be defined as a function of angle θ , where θ is the incremental angle as

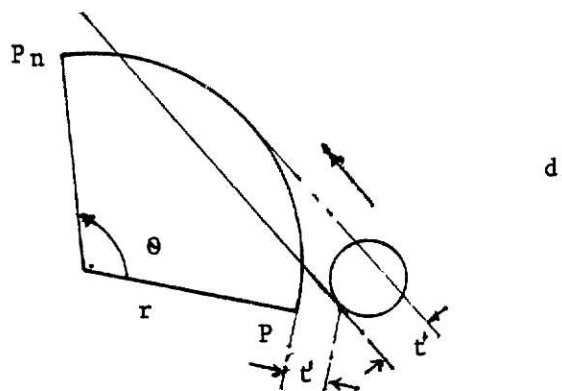
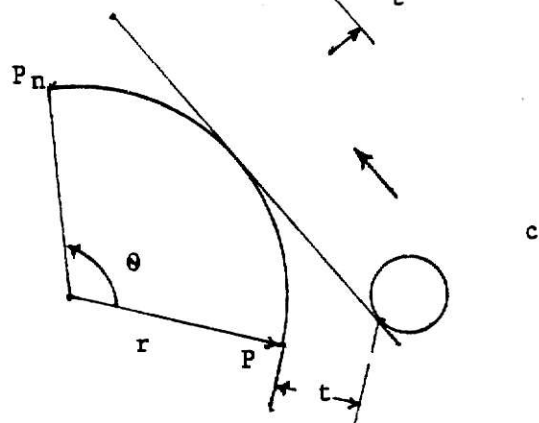
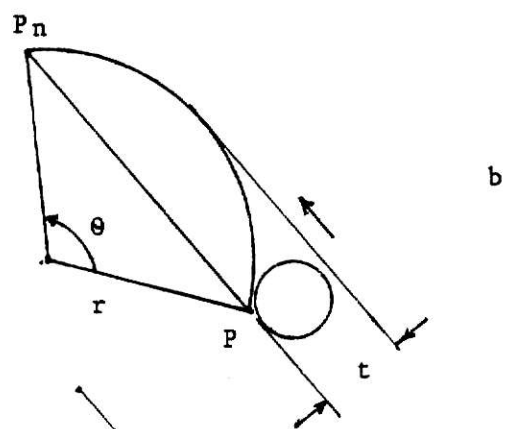
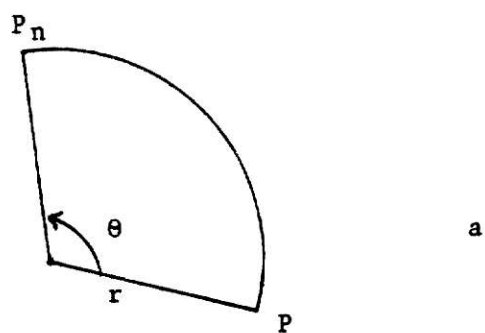


Fig.3.1 a. Circular arc of length $r\theta$. b. Chordal method.
c. Tangential method. d. Secantial method.

shown in Fig.3.2. For the fixed tolerance "t", the the value of the angle θ varies depending upon whether the linear approximation made is chordal, tangential or secantial. The expressions for the angle θ in terms of the radius r and tolerance t for all the three conditions are given below :

$$\text{i) } \theta \text{ (Chordal)} = 2\cos^{-1}(1-t/r) \quad \dots 3.1$$

$$\text{ii) } \theta \text{ (Tangential)} = 2\cos^{-1}(r/(t+r)) \quad \dots 3.2$$

$$\text{iii) } \theta \text{ (Secantial)} = 2\cos^{-1}((1+t/r)/(1-t/r)) \quad \dots 3.3$$

For the representation of the circle, polar coordinates are used which are given by

$$X=r\cos\theta ;$$

$$Y=r\sin\theta .$$

Now, consider the circular arc shown in Fig.3.2 to be drawn with a tolerance of t . Then depending on the secantial, tangential or chordal methods, angle θ could be selected from the equations 3.1, 3.2 or 3.3 to give the required accuracy.

If the motion is to be generated in the counter clockwise direction, then the first point on the curve will be P_1 . The point P_1 is represented by its coordinates X_1 and Y_1 which are given by

$$X_1=r\cos\theta ;$$

$$Y_1=r\sin\theta .$$

The next point on the curve is P_2 which is represented by its coordinates X_2, Y_2 which are given by

$$X_2=r\cos(\theta+\theta) ;$$

$$Y_2=r\sin(\theta+\theta) .$$

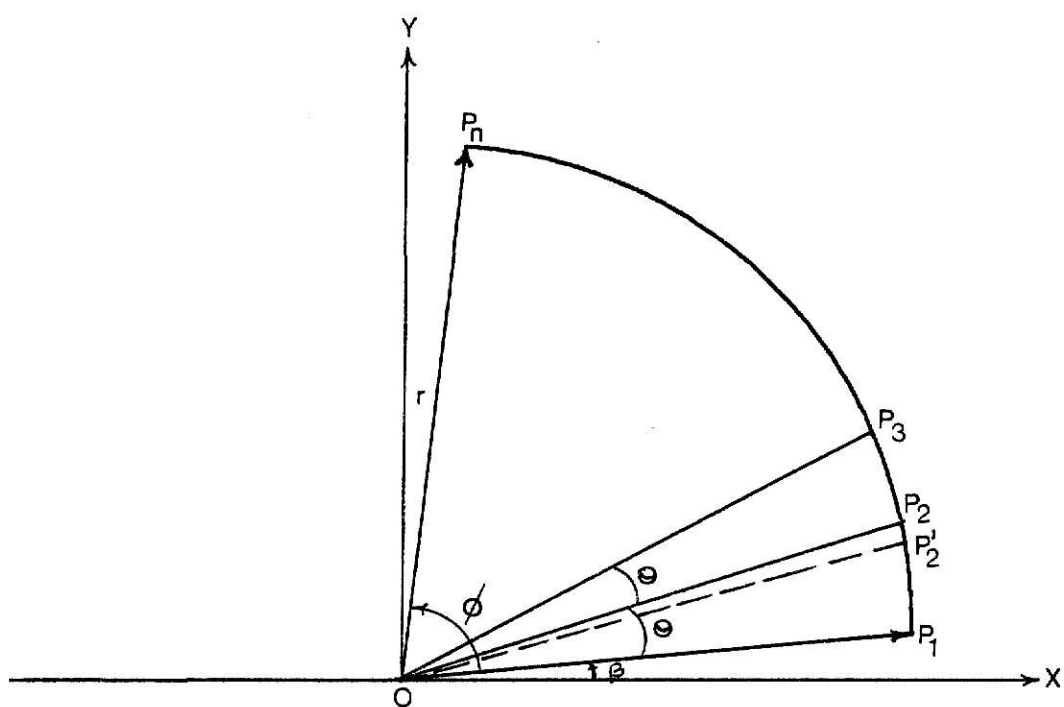


Fig.3.2 Generation of motion along a circular arc

Since the coordinates of P_1 and P_2 are known, a line can be drawn joining those two points. The coordinates of P_2 with respect to P_1 as origin are given by

$$X=X_2-X_1=r\cos(\beta+\theta)-r\cos\beta ;$$

$$Y=Y_2-Y_1=r\sin(\beta+\theta)-r\sin\beta$$

Hence for the generation of the first line segment the above values of X and Y will be fed into the line subroutine. It should be noted that in the line subroutine X and Y are made nearest integer multiples of the step size $DT(.005")$. Let these new values be XI and YI . Thus, it is obvious that the end point coordinates of the first generated line segment will not necessarily be X_2 and Y_2 . In order to compensate for this error, the coordinates of the point P_2 are adjusted and are given by

$$X_2' = X_2 - (X - XI) ;$$

$$Y_2' = Y_2 - (Y - YI) .$$

This is represented by point p_2' in Fig.3.2.

If the coordinates of the third point P_3 are (X_3, Y_3) then the values of input variables X and Y that will be fed into the line subroutine for the generation of the second line segment are given by

$$X=X_3-X_2' ;$$

$$Y=Y_3-Y_2' .$$

The process is continued until the arc of length $r\phi$ is obtained.

The computer flow diagram for the generation of a circular arc is shown in Fig.3.3. The computer program in BASIC language is listed in Appendix 2. The input variables are given below:

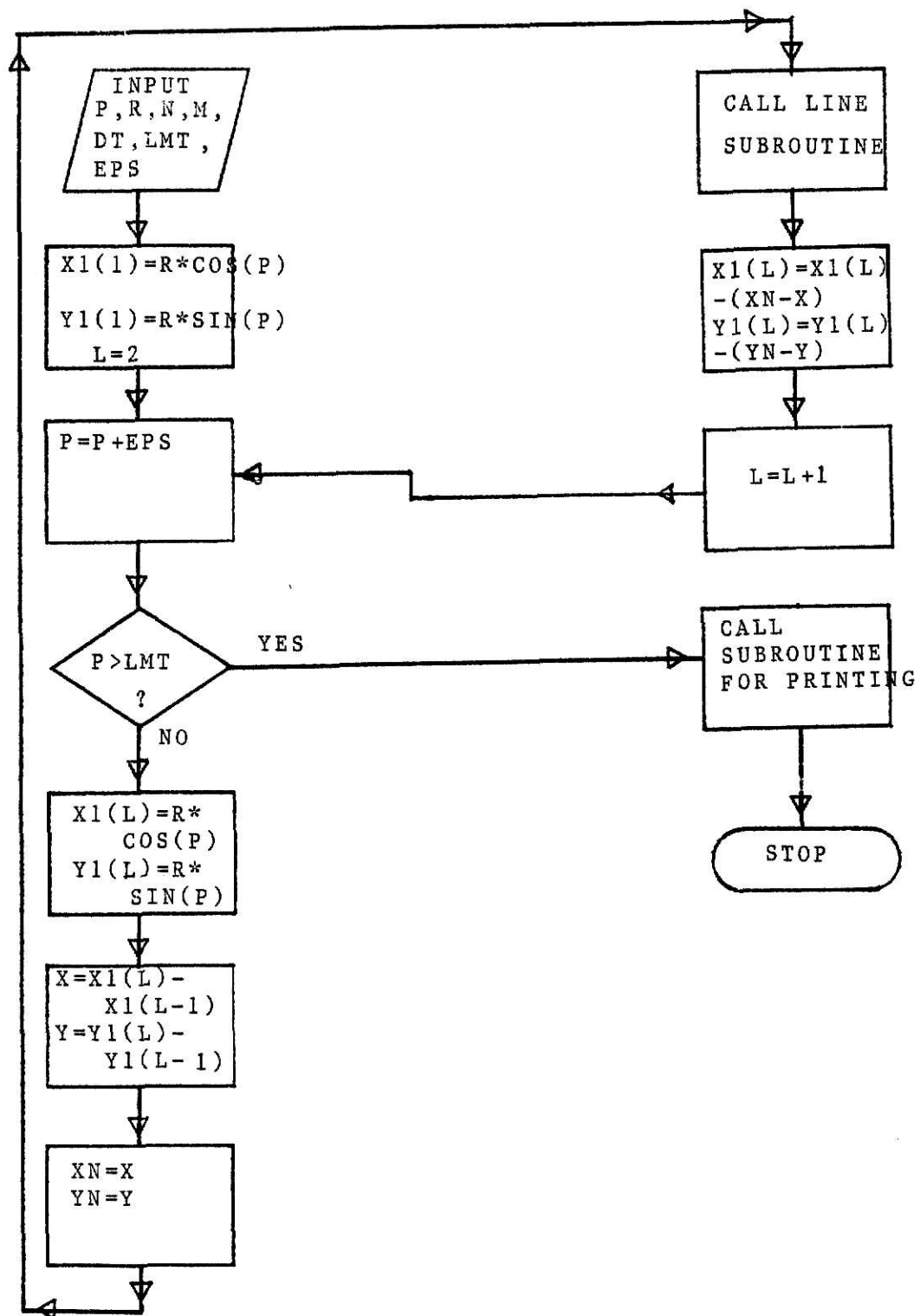


Fig.3.3 Flow diagram : An arc of a circle

P = Angle in radians made by the position vector passing through the origin and the starting point of the arc and the +X axis.

R = Radius of the circle.

EPS = Incremental angle in radians.

LMT = Angle in radians made by the position vector passing through the origin and the terminating point of the arc and the +X axis.

DT = Step size (.005") of the plotter.

N = 1000 ... This value is used in the line subroutine.

M = 5 ... This value is used in the line subroutine.

The plots of circular arcs are shown in Fig.3.4 and 3.5. The plots made in Fig. 3.4 make use of the line subroutine for the case 1 discussed in Chapter 2 and those which are shown in Fig.3.5 make use of the line subroutines for cases 2 and 3 also discussed in Chapter 2.

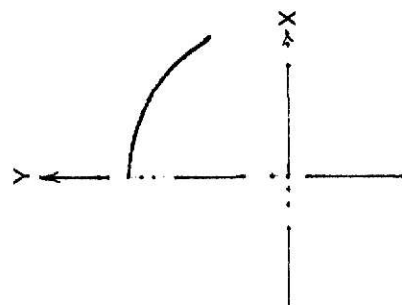
3.2 PARABOLIC ARCS

In the generation of a parabolic arc, the parametric representation of a parabola is used. This given by

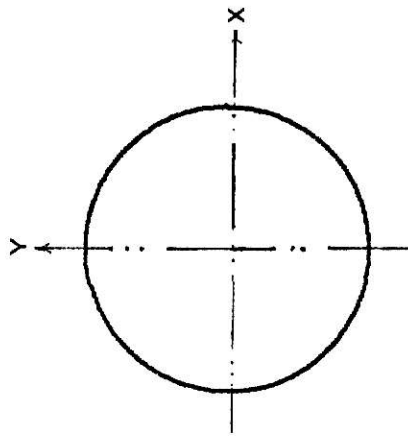
$$X = at^2$$

$$Y = 2at$$

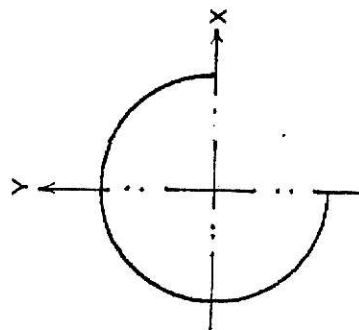
where $0 \leq t \leq \infty$ sweeps out the entire parabola. The parabola, unlike the circle, is not a closed curve. Thus the amount of parabolic arc to be generated must be limited by choosing a maximum value of t , say $t_{\max}$. This can be done by limiting the range of either the X-coordinates or the Y-coordinates.



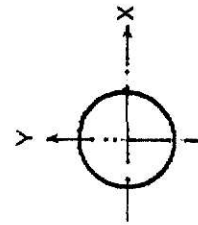
Radius=0.85"
 Starting Angle=30°
 Ending Angle=90°
 Incremental Angle=.04 radian



Radius=0.75"
 Incremental Angle=.04 radian



Radius=0.60"
 Starting Angle=0°
 Ending Angle=270°
 Incremental Angle=.04



Radius=0.25"
 Incremental Angle=.04radian

Fig.3.4 Arcs of Circles: Case 1.

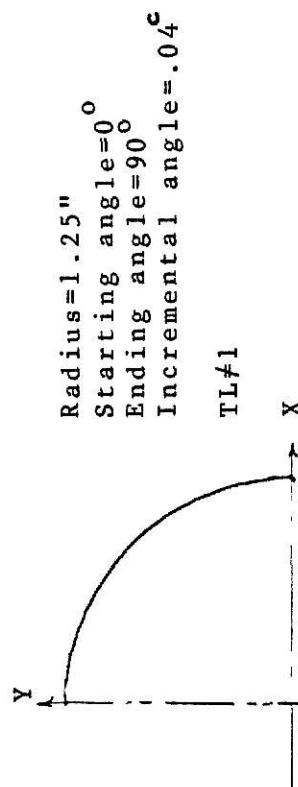
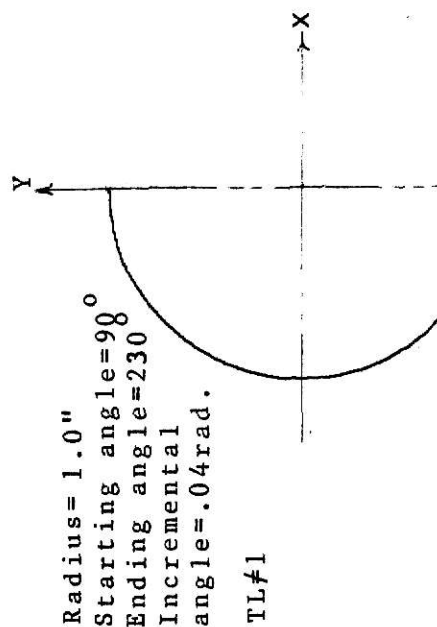
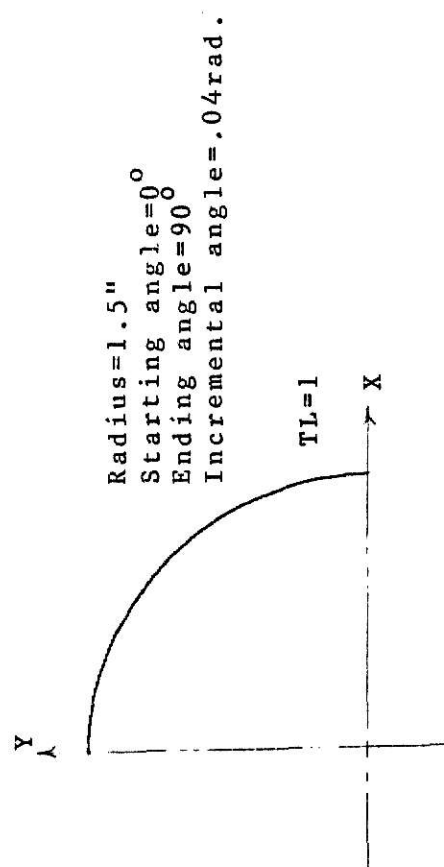
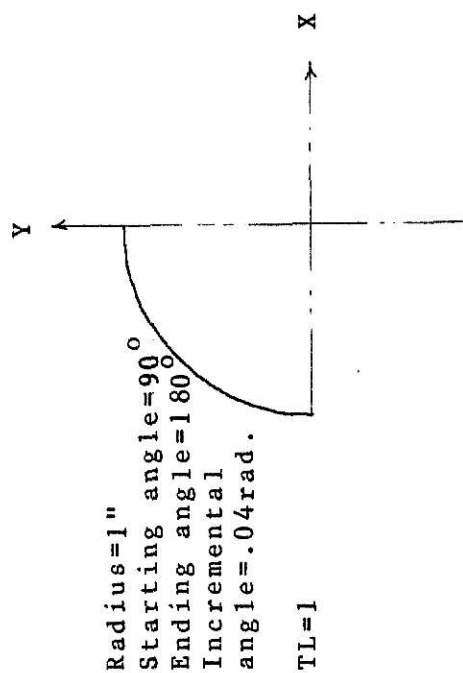


Fig.3.5 Arcs of circles : Cases 2 and 3

If the range of the X-coordinate is limited then

$$t_{\max} = \sqrt{X_{\max}/a} \quad \dots 3.4$$

If the range of the Y-coordinate is limited then

$$t_{\max} = Y_{\max}/2a \quad \dots 3.5$$

Once $t_{\max}$ is established, an algorithm can be developed to calculate N1 representative points for the parabola in the first quadrant. Once the parabola is obtained in the first quadrant, the rest of it could be made to appear in the fourth quadrant by its reflection about the X-axis.

For $t_{n+1} = t_n + dt$; equations 3.4 and 3.5 become

$$X_{n+1} = at_n^2 + 2at_ndt + a(dt)^2$$

$$Y_{n+1} = 2at_n + 2adt$$

which can be written as

$$X_{n+1} = X_n + Y_ndt + a(dt)^2$$

$$Y_{n+1} = Y_n + 2adt$$

The above relationship is used for the development of the algorithm. The computer flow diagram for the algorithm is shown in Fig. 3.6. This flow diagram is for the parabola in the first quadrant. In the algorithm a fixed number of points (N1) is specified and a constant increment in t is used. Parabolas with displaced centers are obtained by suitable rotation.

The listing of the generalized computer program in the BASIC language is shown in Appendix 2. It can be used to generate parabolic paths in all quadrants. The input variables used in this program are given below:

DT = Step size (.005") of the plotter.

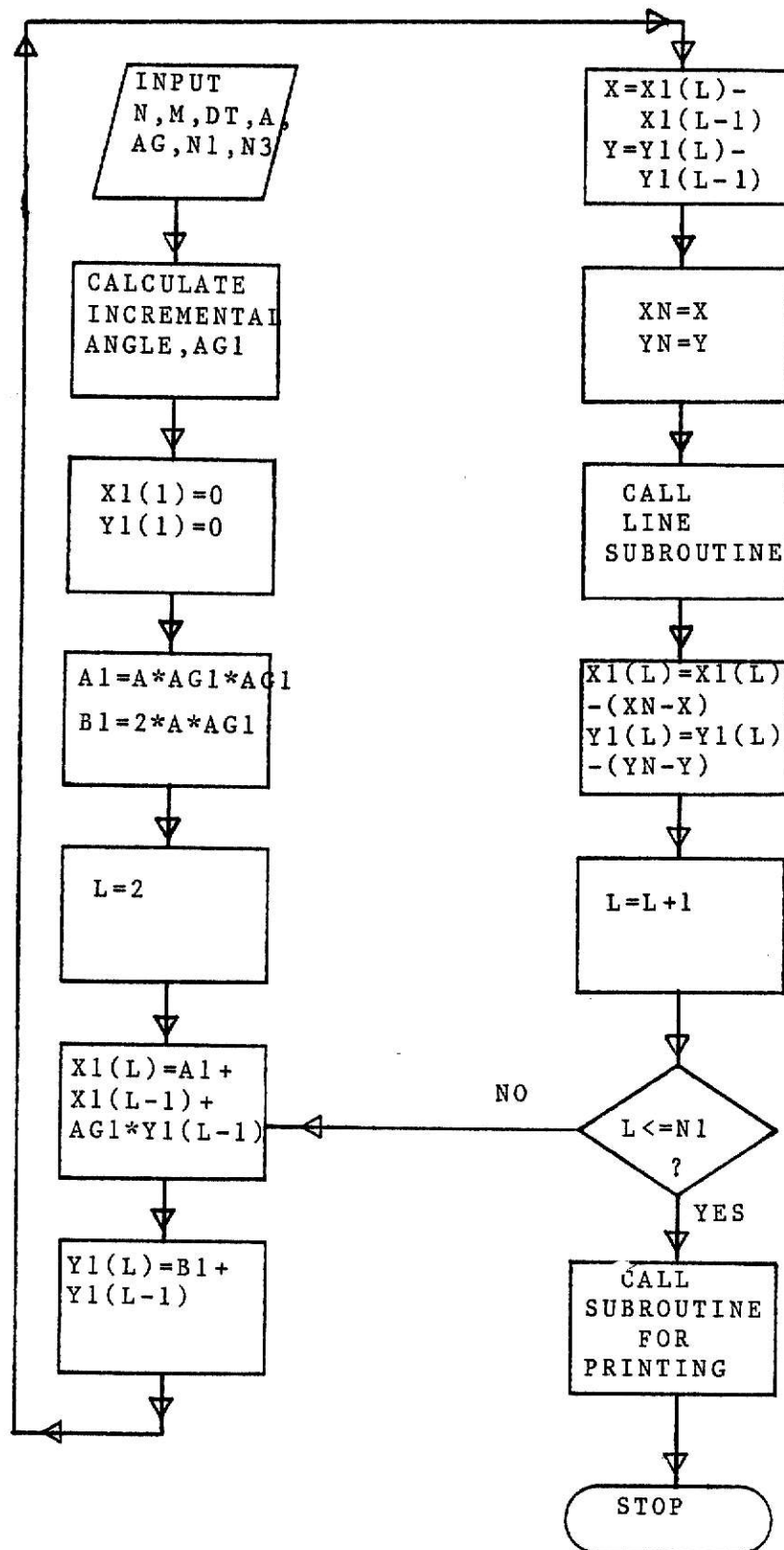


Fig.3.6 Flow diagram for a parabola

A = Distance from the focus to the vertex of the parabola.

AG = Maximum angle in degrees.

N1 = Number of points on the parabola.

N = 1000

M = 5

The values of N and M are used in the line subroutine. The incremental angle, AG1 in radians, is calculated from :

$$AG1 = (AG/(N1-1)) \cdot (180/\pi)$$

As in the case of circle, the generated path is comprised of small straight line segments. When the line subroutine is called, the incremental coordinates of the end of the line segment are made nearest integer multiples of the step size DT(.005") as explained in Chapter 2. The error resulting from this is taken care of in the main program for the generation of the parabolic path in the same way as explained in the circular arc section(3.1). The same procedure will be adopted for all the curves that will be discussed subsequently in order to take care of this type of error. The parabolic arcs drawn in all quadrants are depicted in Fig.3.7. The line subroutine for the case 1 discussed in Chapter 2 is called in order to make these plots. The line subroutines for the cases 2 and 3 can also be used to generate motion along parabolic arc.

3.3 ELLIPTIC ARCS

The parametric approach is used for the generation of ellipses. This is given by

$$X = a \cos \theta$$

$$Y = b \sin \theta$$

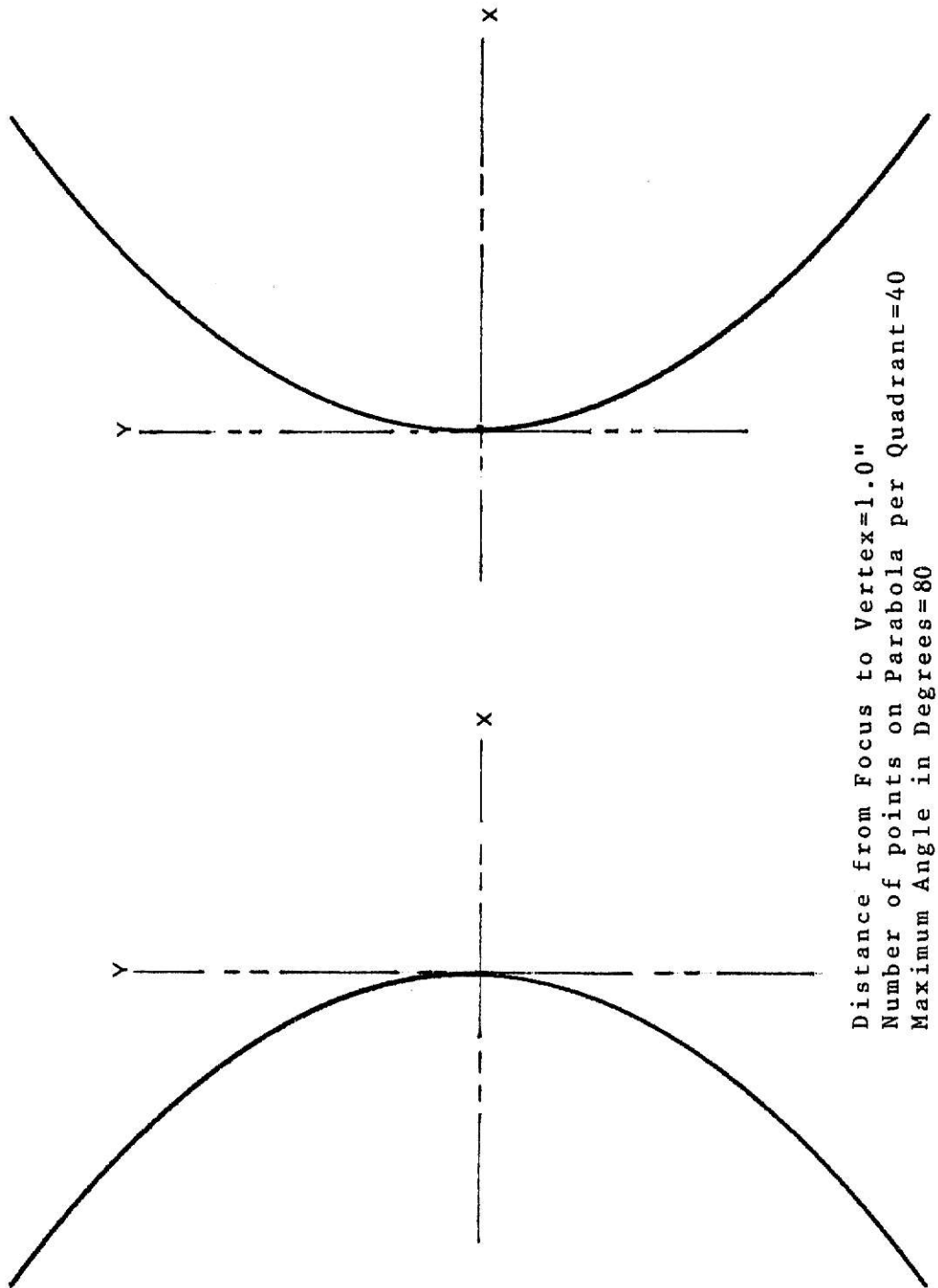


Fig.3.7 Parabolas

where θ is the parameter and a and b are semi-major axis and semi-minor axis respectively. By varying θ between 0 and 2π the entire ellipse can be generated. A specified number of points can be used to represent the ellipse by taking fixed increments in the parameter θ . If n is the number of points, then the incremental value of θ is given by $2\pi/(n-1)$

The computer flow diagram shown in Fig.3.8 is a more generalized one where the ellipse can be assumed to be inclined at an angle i to the horizontal as shown in Fig.3.9

Hence

$$X' = X \cos i - Y \sin i$$

$$Y' = X \sin i + Y \cos i$$

$$\text{or } X' = a \cos \theta \cos i - b \sin \theta \sin i$$

$$Y' = a \cos \theta \sin i + b \sin \theta \cos i$$

If the ellipse has a displaced center, given by (X_c, Y_c) , Introduction of these coordinates yields

$$X'_{n+1} = X_c + X'_n \cos i - Y'_n \sin i$$

$$Y'_{n+1} = Y_c + X'_n \sin i + Y'_n \cos i$$

The computer program in the BASIC language is listed in Appendix 2. The input variables in this program are the number of points ($N1$), the length of the semi-major axis ($A1$), the length of the semi-minor axis ($B1$) and the inclination angle (L). This program assumes the center of the ellipse to be at the origin. The input values are $N=1000$, $M=5$ and $DT=.005$, which are used in the line subroutine.

By using this program, plots were made on the digital plotter and are shown in Fig.3.10

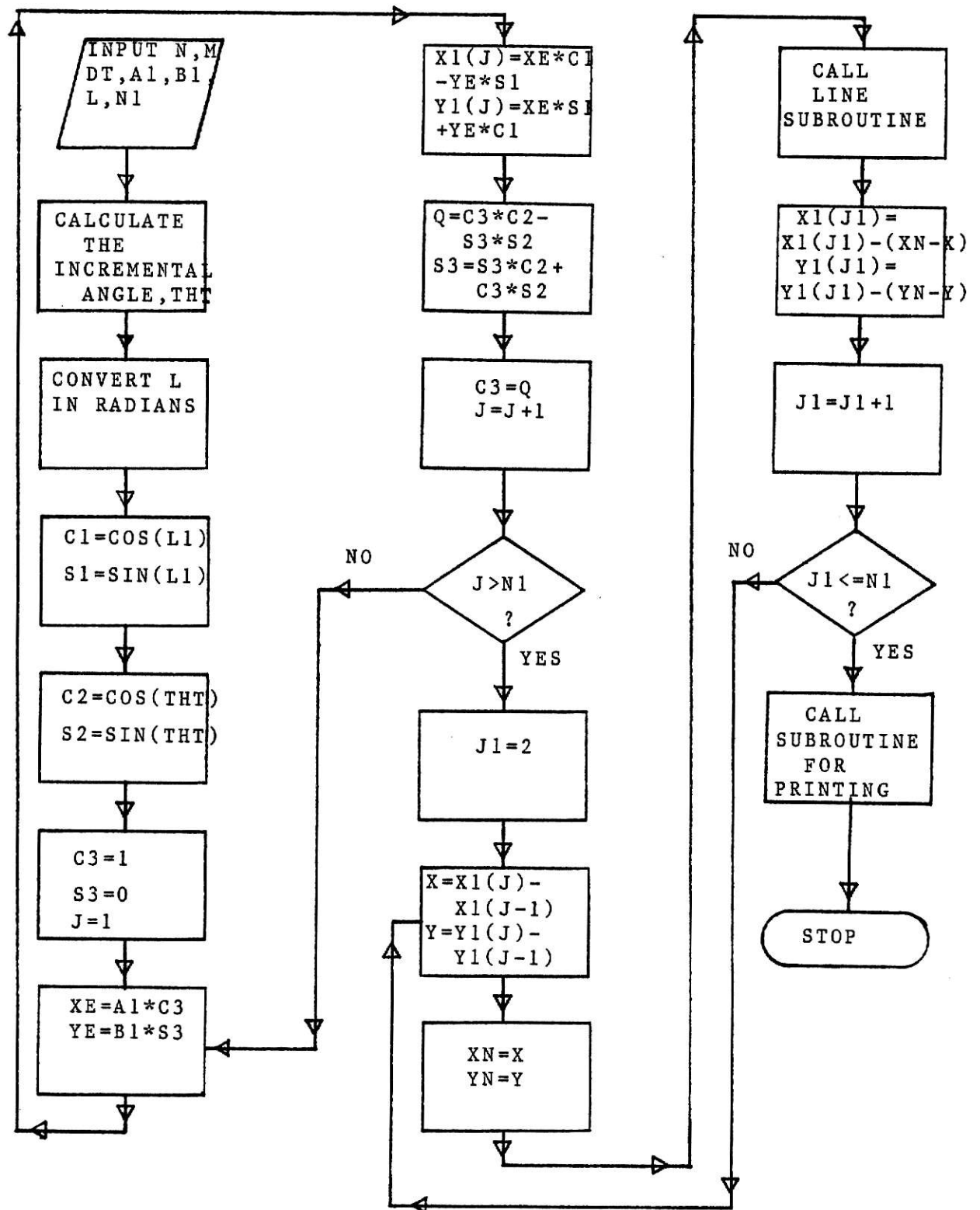


Fig.3.8 Flow diagram for an ellipse

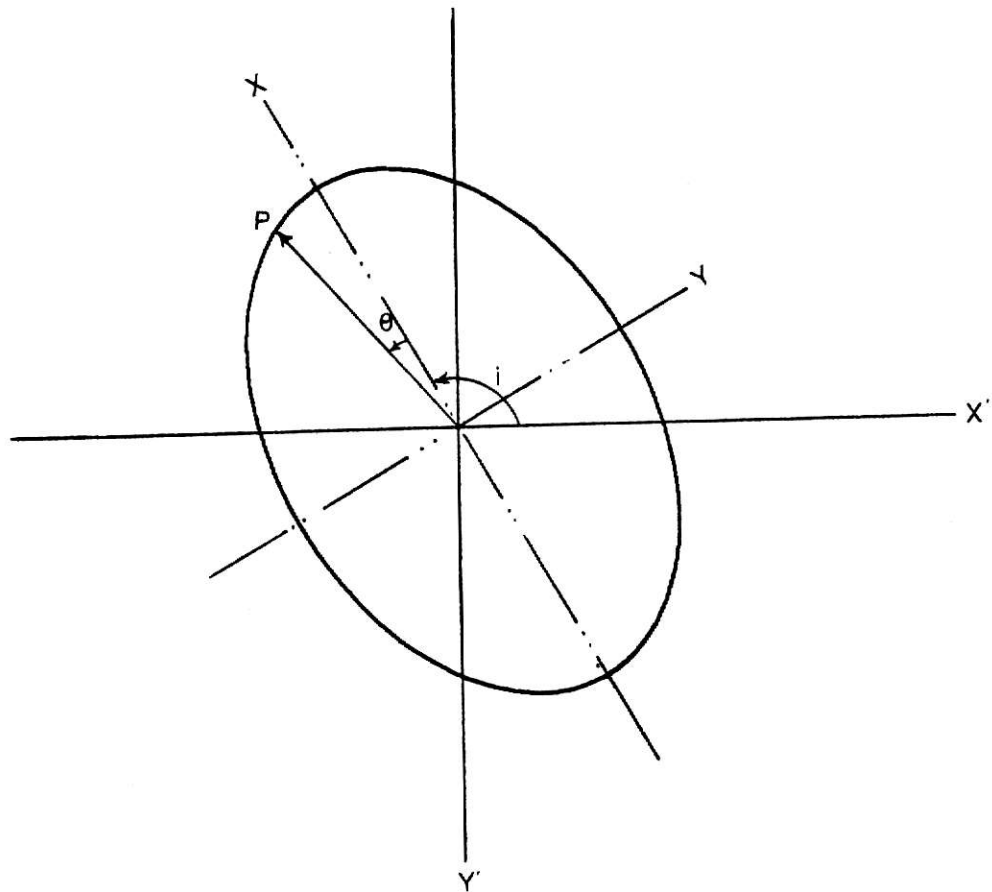


Fig.3.9 Ellipse inclined at an angle i to the horizontal

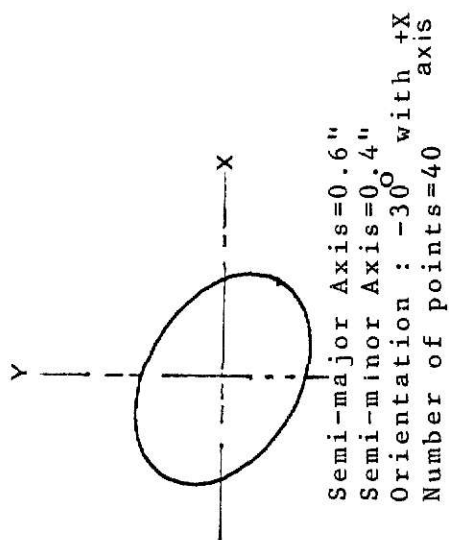
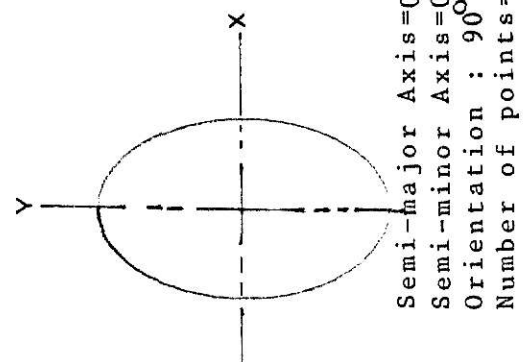
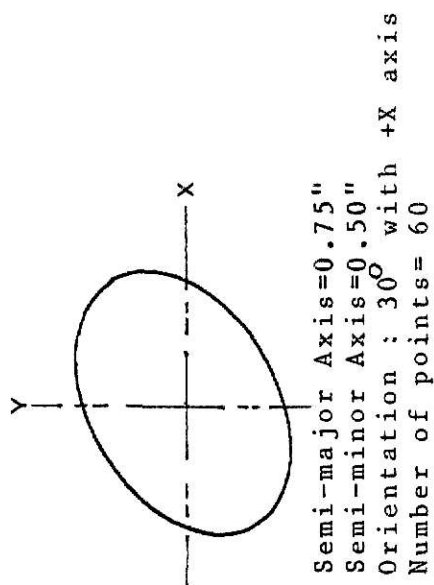
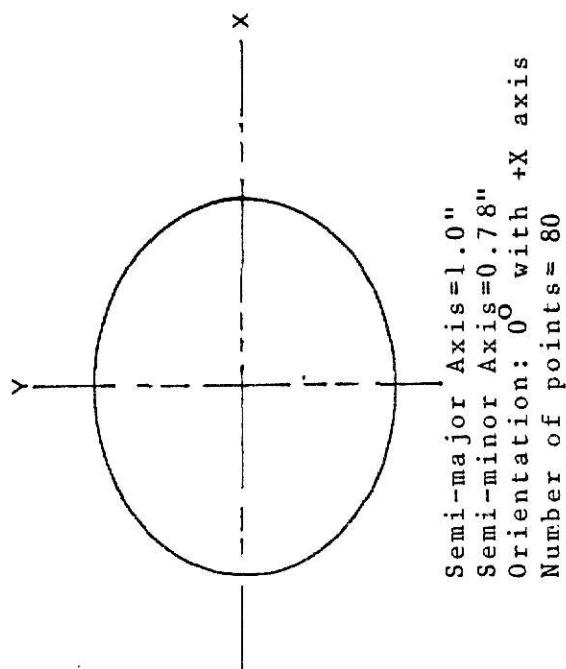


Fig.3.10 Ellipses

3.4 HYPERBOLIC ARCS

A parametric representation of a hyperbola is given by

$$X = \pm a \operatorname{SEC} \theta$$

$$Y = \pm b \operatorname{TAN} \theta ,$$

where $0 \leq \theta < \pi/2$ yields the desired hyperbola. For the generation of the hyperbolic path double-angle formulas are used to obtain an efficient algorithm.

If $X_n = \pm a \operatorname{SEC} \theta$ then

$X_{n+1} = \pm a \operatorname{SEC}(\theta + d\theta)$, where $d\theta$ is the incremental angle.

$$X_{n+1} = \pm \frac{a}{\cos(\theta + d\theta)}$$

$$X_{n+1} = \pm \frac{a}{\cos \theta \cos d\theta - \sin \theta \sin d\theta}$$

Multiplying denominator and numerator by $b/\cos \theta$, gives

$$X_{n+1} = \pm \frac{ab/\cos \theta}{b/\cos \theta [\cos \theta \cos d\theta - \sin \theta \sin d\theta]}$$

$$X_{n+1} = \pm \frac{ab \operatorname{SEC} \theta}{b \cos d\theta - b \operatorname{TAN} \theta \sin d\theta}$$

$$X_{n+1} = \pm \frac{b X_n}{b \cos d\theta - b \operatorname{TAN} \theta \sin d\theta}$$

If $Y_n = \pm b \operatorname{TAN} \theta$ then

$$Y_{n+1} = \pm b \operatorname{TAN}(\theta + d\theta)$$

$$Y_{n+1} = \pm \frac{b \operatorname{TAN} \theta + b \operatorname{TAN} d\theta}{1 - \operatorname{TAN} \theta \operatorname{TAN} d\theta}$$

$$Y_{n+1} = \pm \frac{b(Y_n + b \operatorname{TAN} \theta)}{b - Y_n \operatorname{TAN} d\theta}$$

The computer flow diagram shown in Fig.3.11 uses these relationships. The listing of the computer program is given in Appendix 2. The input variables are the distance from the center of a hyperbola to its vertex ($A3$), the slope of asymptotes, $\pm b/a$

and the number of points on a hyperbola(N_1). Some hyperbolic arc plots made by using this algorithm are depicted in Fig.3.12.

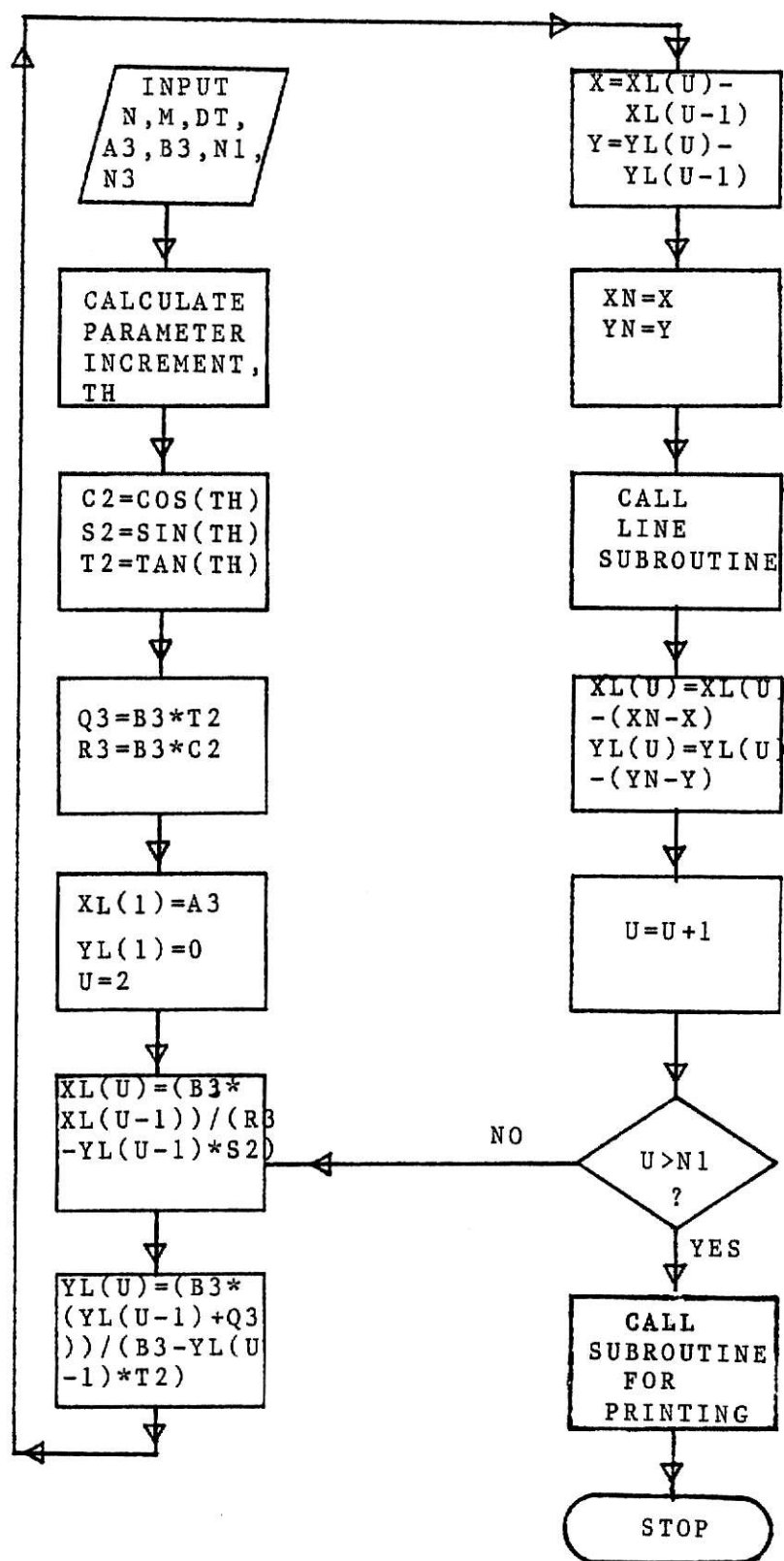


Fig.3.11 Flow diagram for a hyperbola

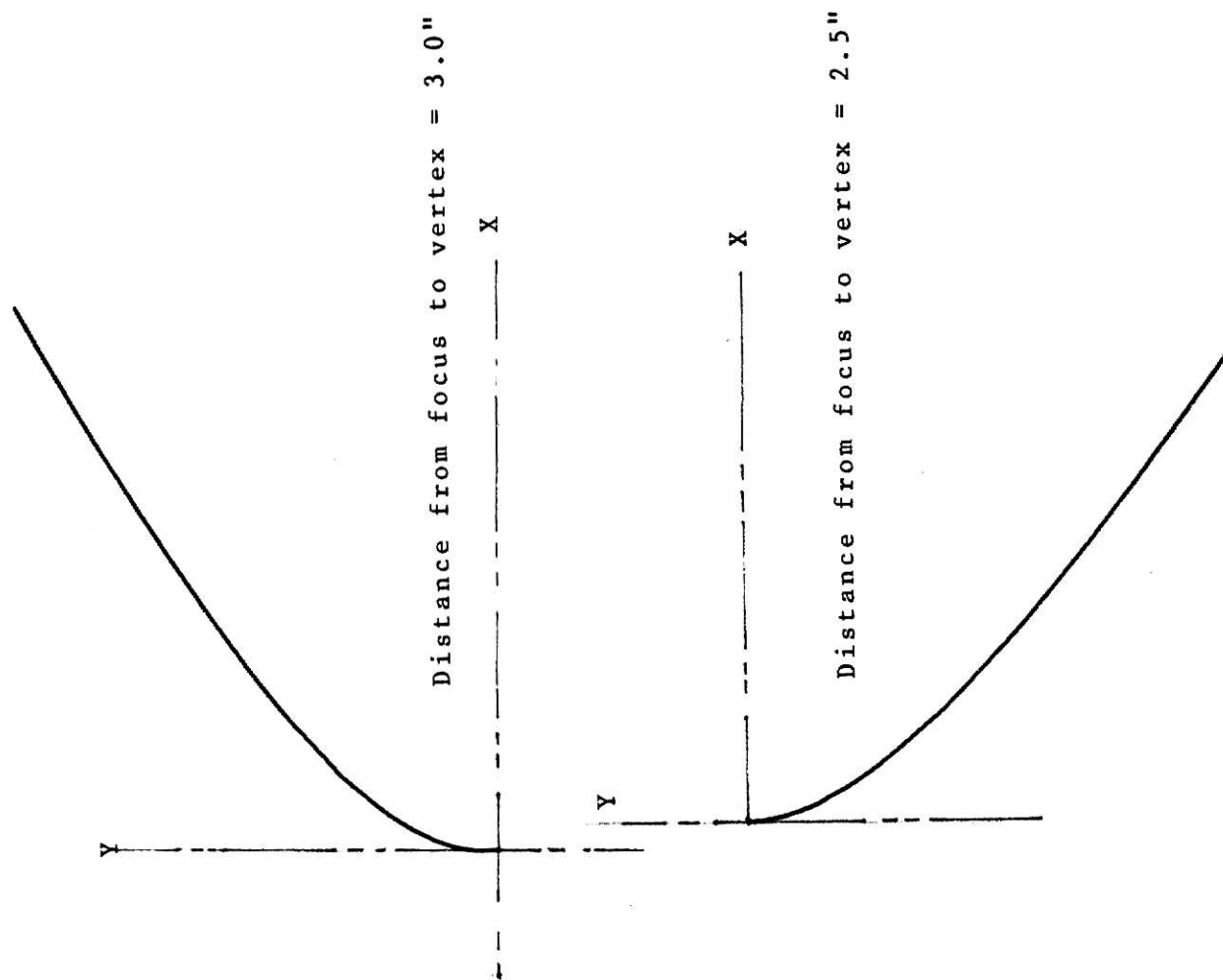


Fig.3.12 Hyperbolic arcs

CHAPTER 4

BLENDING OF REGULAR CURVES IN A PLANE

BLENDING OF REGULAR CURVES IN A PLANE

In Chapter 3, the generation of motion along regular curves was discussed in detail. However, it is often desirable to wrap a continuous curve around a construction, and a single math model may be inappropriate to satisfy design constraints. The possible approach to this problem is by joining two elliptic, circular or parabolic arcs. This chapter describes these possibilities in detail.

4.1 ELLIPTIC ARC BLENDING [10]

Before considering the problem of elliptic arc blending, we will first discuss the conditions under which the design of a single elliptic arc is possible.

4.1.1 Design of an elliptic arc

Consider Fig.4.1, where it is desired to create an elliptic arc between two points P_1 and P_2 with a horizontal slope at P_1 ($m_1=0$) and an input slope, m_2 , at P_2^* . We will seek an elliptic arc in standard position whose equation is given by

$$\frac{(X-X_c)^2}{a^2} + \frac{(Y-Y_c)^2}{b^2} = 1 \quad \text{.....4.1}$$

where (X_c, Y_c) is the center of the ellipse and a and b are semi-major axis and semi-minor axis respectively. The horizontal slope at P_1 implies that the center is somewhere on the Y axis, that is, $X_c=0$. The parameters Y_c , a and b can be derived by setting up three equations in three unknowns. These equations are obtained by inputting the X and Y coordinates of P_1 and P_2 into the Equation 4.1, and the desired slope at P_2 into the equation of the derivative. However, it should be noted that the solution does not necessarily exist for all input values of $*Y_2 < Y_1$

slope m_2 . Appendix 3 shows the derivation of the slope m_2 which must exceed (cannot be equal to) twice the slope (in magnitude) of the line between P_1 and P_2 . This is given by

$$m_2 > 2 \left| \frac{Y_1 - Y_2}{X_1 - X_2} \right| \quad \dots\dots 4.2$$

If, we do not wish our elliptic arc to have two values (which occurs when an arc is longer than a quadrant), we must not allow our input slope to exceed the vertical position; that is, it must not become positive for this case. When $m_2 = \infty$, we have

$$Y_2 = \frac{-b^2 X_2}{a^2 (Y_2 - Y_c)} = \infty$$

i.e. since "a" cannot be zero, $Y_c = Y_2$.

Substituting $Y_2 = Y_c$ in the equation of an ellipse given by

$$\frac{X_2^2}{a^2} + \frac{(Y_2 - Y_c)^2}{b^2} = 1, \text{ we get}$$

$$a = X_2$$

At point P_1 , Equation 4.1 reduces to

$$(Y_1 - Y_c)^2 = b^2 \quad \dots\dots 4.3$$

At point P_2 , Equation 4.1 gives

$$\frac{X_2^2}{a^2} + \frac{(Y_2 - Y_c)^2}{b^2} = 1$$

$$\text{i.e.} \quad \frac{X_2^2}{a^2} = \frac{b^2 - (Y_2 - Y_c)^2}{b^2} \quad \dots\dots 4.4$$

Substitute b^2 from Equation 4.3 into Equation 4.4 to get

$$\frac{X_2^2}{a^2} = \frac{(Y_1 - Y_c)^2 - (Y_2 - Y_c)^2}{b^2} \quad \dots\dots 4.5$$

As shown above when $m_2 = \infty$, $Y_2 = Y_c$ and $X_2 = a$

Hence, from Equation 4.5, we have

$$b = (Y_1 - Y_c)$$

Hence, if the elliptic arc is to be single valued between points P_1 and P_2 then the value of the input slope m_2 , should lie between the slope of the lines given by L_1 and L_2 (See Fig.4.1).

Similar to the example of Fig.4.1, a design problem may specify an elliptic arc between P_1 and P_2 with a vertical slope at P_2 as depicted in Fig.4.2. In this case the input slope at P_1 would have to be between zero and $1/2$ the slope (in magnitude) of the line between P_1 and P_2 . That is,

$$1/2 \cdot \left| \frac{Y_1 - Y_2}{X_1 - X_2} \right| > m_1 \geq 0 \quad \text{.....4.6}$$

Hence, given three input conditions, $P_1(X_1, Y_1)$, $P_2(X_2, Y_2)$ and the slope at P_2 , m_2 (the slope at P_1 is made 0 by suitable translation and/or rotation), it can be checked to determine if it is feasible to have an elliptic arc between P_1 and P_2 . If so, the values of the parameters a , b and Y_c can be determined as shown in Appendix 3. These are given as

$$Y_c = \frac{Y_2^2 - Y_1^2 - m_2 X_2 Y_2}{2Y_2 - 2Y_1 - m_2 X_2} \quad \text{.....4.7}$$

$$a^2 = \frac{X_2(Y_1 - Y_c)^2}{m_2(Y_c - Y_2)} \quad \text{.....4.8}$$

$$b^2 = \frac{a^2 m_2 (Y_c - Y_2)}{X_2} \quad \text{.....4.9}$$

4.1.2 Elliptic blending

Consider Fig.4.3, where it is desired to have elliptic arc/arcs to pass through the three points given by P_1 , P_2 and P_3 with continuity of slope at P_2 .

This example differs from the characteristics of those just discussed in that we have points P_1 , P_2 and P_3 through

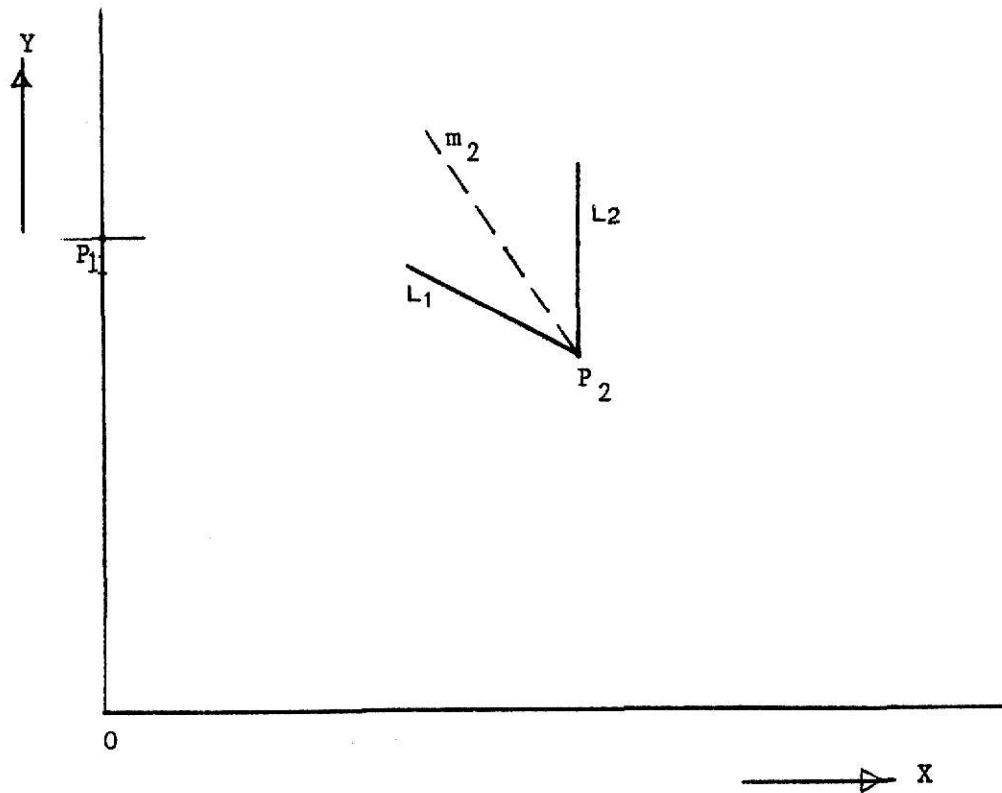


Fig.4.1 Input conditions for elliptic arc design.

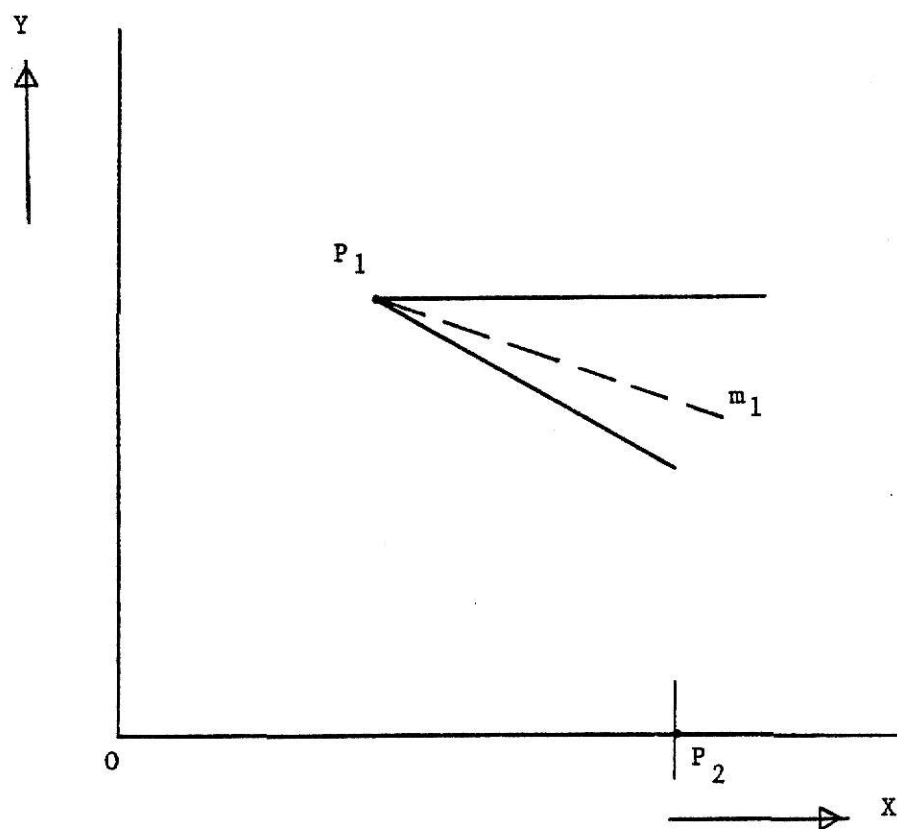


Fig.4.2 Alternative to Fig.4.1 for elliptic arc design.

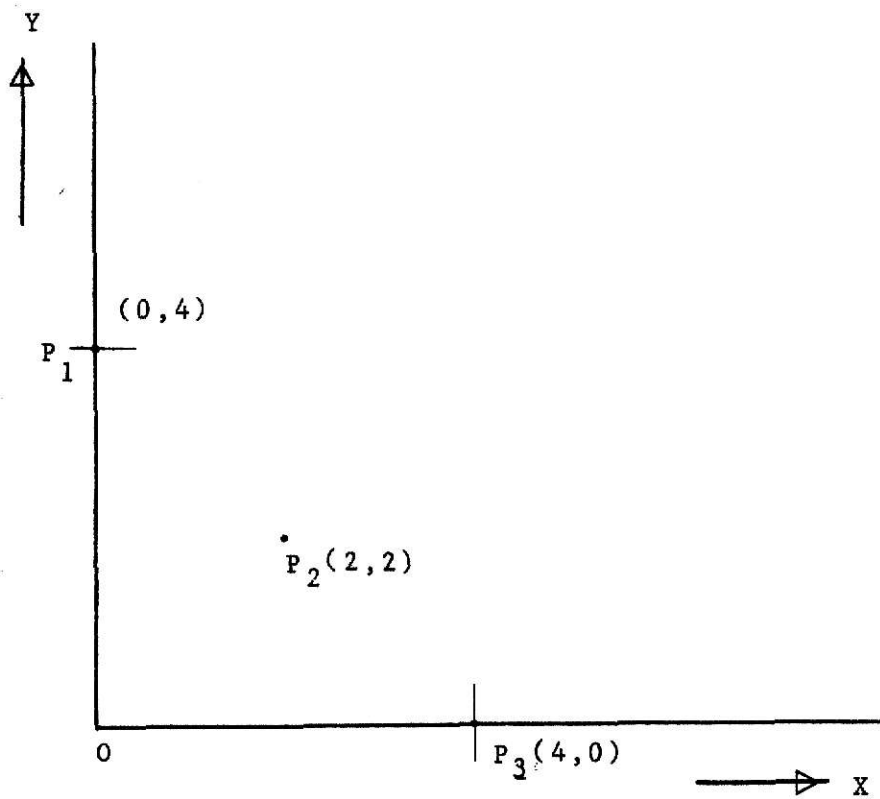


Fig.4.3 Example of input constraints for elliptic blending.

which we have an option of blending two arcs or, if it meets all criteria, a single elliptic arc. An attempt will be made to consider two elliptic arcs in this section. First of all, it is necessary to determine whether a solution exists. By setting the slope constraints given by Equations 4.2 and 4.6 equal, we get,

$$2 \frac{(Y1-Y2)}{(X1-X2)} = 1/2 \cdot \frac{(Y2-Y3)}{(X2-X3)} \quad \dots\dots 4.10$$

By suitable translation, it is always possible to have P_1 on the vertical axis and P_3 on the horizontal axis. In this case $X_1=0$ and $Y_3=0$.

Substituting these values in Equation 4.7, we get

$$2 \frac{(Y2-Y1)}{X2} = 1/2 \cdot \frac{Y2}{(X2-X3)} \quad \dots\dots 4.11$$

Solving for X_2 , we get the limiting value for X_2 , denoted by $X_2(L)$. The value of $X_2(L)$ is determined by

$$X_2(L) = \frac{4X_3(Y2-Y1)}{3Y2-4Y1} \quad \dots\dots 4.12$$

Consider an example where $(X_1, Y_1)=(0, 4)$, $(X_2, Y_2)=(3, 3)$ and $(X_3, Y_3)=(4, 0)$. If we substitute all data in Equation 4.12 except X_2 , we obtain

$X_2(L) = 16/7$, which is smaller than $X_2=3$, which clearly indicates that it is possible to have two elliptic arcs pass through P_1 , P_2 and P_3 with continuity of slope at P_2 . Using $X_2=3$ and other input data on each side of Equation 4.11, we see that the slopes must be less than $-2/3$ and greater than $-3/2$, respectively. So if m_c is the common slope

we are at liberty to input m_c in the range

$$-2/3 > m_c > -3/2$$

This shown in Fig.4.4. However, if the value of Y_1 is changed from 4 to 5 and that of X_3 is changed from 4 to 5 then by using $X_2=3$ and other input data on each side of Equation 4.11, we see that the slopes must be less than $-4/3$ and greater than $-3/4$, respectively, which is not a feasible condition. Hence two elliptic arcs cannot pass through the three points given by (0,5), (3,3) and (5,0), with continuity of slope at (3,3). Once it is confirmed that two elliptic arcs can pass through the three points P_1 , P_2 and P_3 with continuity of slope at P_2 , the equation of an ellipse in standard position through the points P_1 and P_2 is obtained by calculating the values of the parameters Y_c , a and b from Equations 4.7, 4.8 and 4.9. The motion along this elliptic arc is generated by using the algorithm described in Chapter 3. Similarly the motion along the elliptic arc through P_2 and P_3 can be generated.

4.2 BLENDING OF TWO CIRCULAR ARCS [10]

The required matching of two circular arcs at a point is more restrictive than matching of two elliptic arcs. This may be seen by noting that there exists a range of ellipses from P_1 to P_2 , as in Fig.4.1. A corresponding range of input slopes at P_2 are possible within certain constraints. This, of course, create options to choose among a range of ellipses. However, only one circular arc with its center on the Y axis can be drawn from P_1 to P_2 . Similarly one circular arc with its center on the X axis can be drawn from P_2 to P_3 .

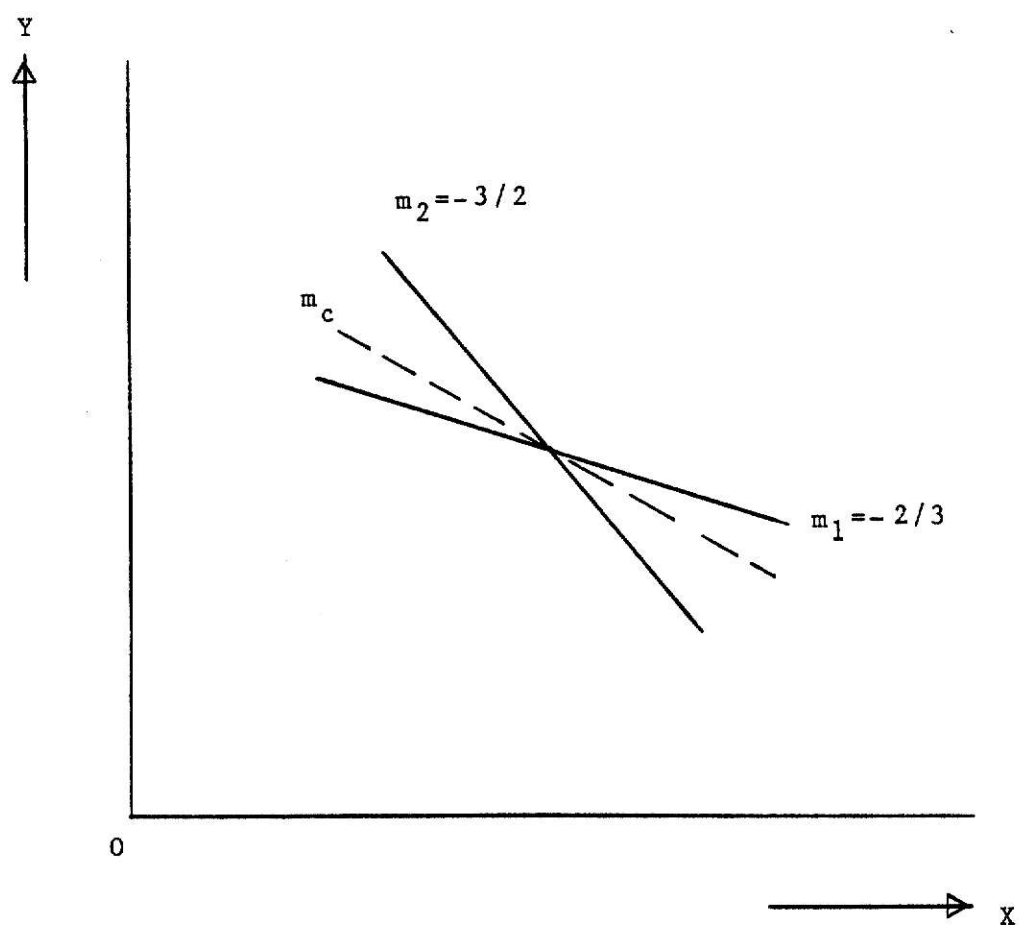


Fig.4.4 Acceptable range of the input slope, m_c .

In view of this restriction, we cannot have two circular arcs tangent at a particular value of P_2^* . It is possible to specify one of the coordinates and write a computer program to search for the other. Thus, for the P_1 and P_2 of the examples, we may set the Y coordinate of P_2 equal to 3 and seek the X coordinate that will produce the desired tangency at P_2 .

Since circles are in the family of ellipses, the limiting value of X_2 , denoted by $X_2(L)$, as explained for the elliptic arcs, also holds good for the circular arcs. This is given by Equation 4.12 as

$$X_2(L) = \frac{4X_3(Y_2 - Y_1)}{3Y_2 - 4Y_1}$$

The procedure for deriving the value of X_2 is described below with the aid of Fig.4.5.

Select a trial value of X_2 and by using the coordinates of P_1 and P_2 , derive the equation of the perpendicular bisector of the P_1P_2 line. Let this line intersect the Y axis at a point "h". Similarly, derive the equation of the perpendicular bisector of the line P_2P_3 and let it intersect the X-axis at a point "k". Using Y-intercept at "h" and the coordinates of P_2 , we can determine the slope m_1 , of the line L_1 , which connects "h" to P_2 . Since "h" is at the center of the first arc (because the slope is horizontal at P_1), L_1 is perpendicular to that arc at P_2 . In the similar way, the slope of the line L_2 (given by m_2) can be derived. Now for our trial point X_2 to be a solution, m_1 must be equal to m_2 . The formulas for m_1 and m_2 in terms of the input coordinate data are given by

* Single valued functions have been considered here.

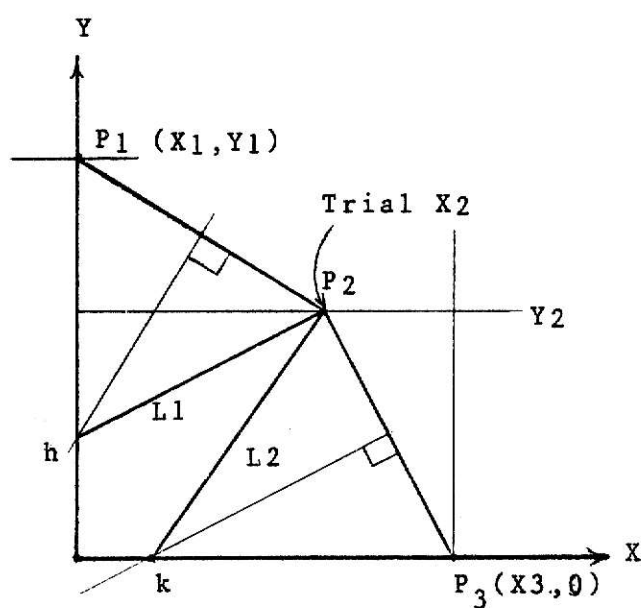


Fig.4.5 Input data used in searching for solution for two blended circular arcs.

$$m_1 = \frac{Y_2}{X_2} + \frac{X_2^2 - Y_1^2 + Y_2^2}{2X_2(Y_1 - Y_2)}$$

and

$$m_2 = \frac{2Y_2(X_2 - X_3)}{(X_2 - X_3)^2 - Y_2^2}$$

By setting $m_1 = m_2$, the equation is solved for the value of X_2 by a trial and error method. For successive values of X_2 from $X_2(L)$ to X_3 , $m_1 - m_2$ must start off at $X_2(L)$ with a negative sign (the reason for this is explained below). If the sign does not change with successive trials, no solution exists. If a sign change does occur between two trial values then a solution exists between these two values. The computer program in the BASIC language for finding the solution of X_2 is listed in Appendix 3. The input variables for this program are ;

- i) Coordinates of point P1.
- ii) Coordinates of point P3.
- iii) Ordinate of point P2.
- iv) Number of points N_1 , dividing $(X_3 - X_2(L))$ into equal intervals.

The prescribed trial and error process may be defined as follows :

- i) Determine $X_2(L)$. Potential solutions exist if and only if $X_2(L) < X_3$.
- ii) Divide $X_3 - X_2(L)$ into equal intervals, say N_1 .
- iii) At each end point, compute $m_1 - m_2$ and determine its sign.
- iv) When the sign changes, that interval is subdivided

* Refer to Appendix 3 for the derivation.

into subintervals(N_1).

v) The trial procedure is repeated until a change in sign for $m_1 - m_2$ is noted.

The value of X_2 is obtained to a desired accuracy by controlling the length of the subinterval.

For data in the first quadrant of a coordinate system, unless $(m_1 - m_2)$ is negative when using $X_2(L)$ as the first trial solution, there can be no solution since m_1 increases with increasing X_2 , and m_2 decreases with increasing X_2 . Hence, $(m_1 - m_2)$ is always increasing. So, $(m_1 - m_2)$ must start as negative if it is going to get to zero and produce a solution. Another necessary condition to produce a solution is that m_1 be positive when the trial X_2 is equal to X_3 . This would ensure that $(m_1 - m_2)$ is positive because m_2 is zero when X_2 is set equal to X_3 . A solution can exist if and only if $(m_1 - m_2)$ is negative when X_2 is set equal to $X_2(L)$, and m_1 is positive when X_2 is set equal to X_3 .

For $X_1=0$, $Y_1=4$, $Y_2=2$, $X_3=4$ and $Y_3=0$, the value for X_2 is found by using the computer program listed in Appendix 3. The value of X_2 is 3.464.

4.3 PARABOLIC BLENDING [3]

The technique for parabolic blending presented here was first suggested by A.W. Overhauser. The interpolation scheme considers four consecutive points simultaneously. A smooth curve between the two interior points is generated by blending two overlapping parabolic segments. The first parabolic segment

is defined by the first three points, and the second parabolic segment is defined by the last three points.

Consider four consecutive points in a plane or space specified by the position vectors P_1 , P_2 , P_3 and P_4 . Two overlapping parabolas $P(r)$ and $Q(s)$ between these points are shown in Fig.4.6. Each parabola passes through three points and each is defined relative to its local coordinate system. The parabola $P(r)$ passes through P_1 , P_2 and P_3 and is governed by the following equation, relative to the ur-coordinate system.

$$u = P(r) = \alpha r(d-r), \quad \dots\dots 4.13$$

where, r is measured along the chord length P_1P_3 and u is measured perpendicular to r in the plane defined by P_1 , P_2 and P_3 . The chord length between P_1 and P_3 is d . A parabola can be completely specified by two end points P_1 , P_3 and a third point P_2 on the curve. The value of the constant α is chosen such that the parabola $P(r)$ passes through P_2 .

In a similar manner, the parabola $Q(s)$ is defined so as to pass through the points P_2 , P_3 and P_4 . The equation of this parabola is given by

$$v = Q(s) = \beta s(e-s), \quad \dots\dots 4.14$$

where, s is measured along the chord P_2P_4 , v is perpendicular to s in the plane defined by P_2 , P_3 and P_4 , and β is chosen such that the parabola passes through P_3 . The chord length between p_2 and P_4 is e .

The parameter t is now chosen as the distance along the chord length between P_2 and P_3 . A curve $C(t)$, which is a blend

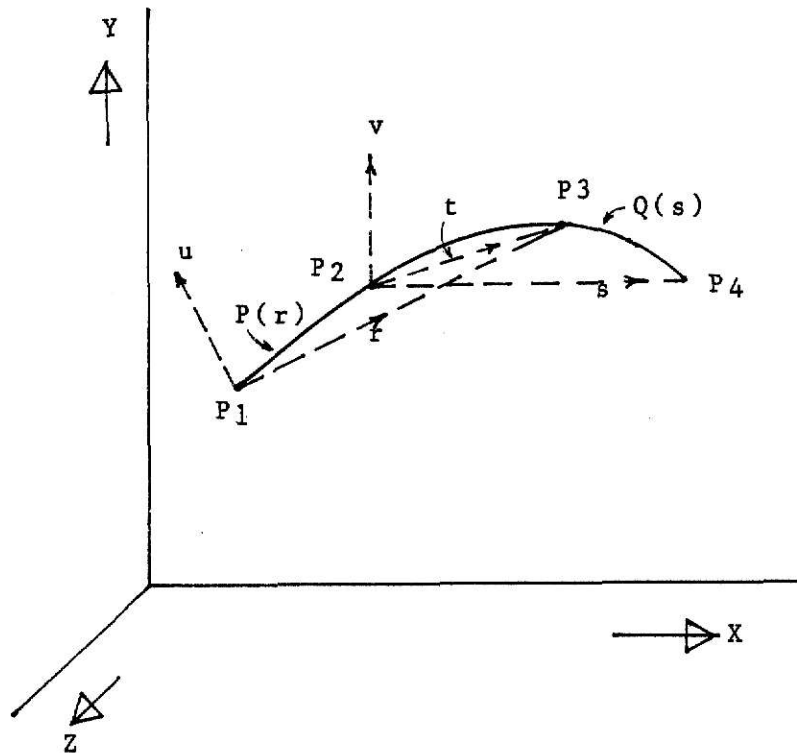


Fig.4.6 Parabolic blending

of the two overlapping parabolas, is constructed between P_2 and P_3 by using an interpolation scheme. The blending curve $C(t)$ is defined by

$$C(t) = [1-(t/t_0)]P(r) + [t/t_0]Q(s) \quad \text{.....4.15}$$

where, t_0 is the distance between P_2 and P_3 . The coefficients of $P(r)$ and $Q(s)$ act as blending functions, varying linearly between 1.0 and 0, and 0 and 1.0 respectively.

The position vectors P_1 , P_2 , P_3 and P_4 are specified in terms of the cartesian XYZ-coordinate system, whereas the blending parabolas $P(r)$ and $Q(s)$ are specified in terms of a local coordinate system. To derive the parametric parabolic equation in terms of the XYZ coordinate system, consider the geometry as shown in Fig.4.7. In Fig.4.7a, P_2J is perpendicular to the chord between P_1 and P_3 . Thus, the ur-plane can be defined by the vector dot product

$$(P_2-J).(P_3-P_1) = 0 \quad \text{.....4.16}$$

If J is located in the ur-plane at $r=xd$ then in the XYZ-coordinate system

$$J = P_1 + x(P_3-P_1) \quad \text{.....4.17}$$

and Equation 4.16 may be written as

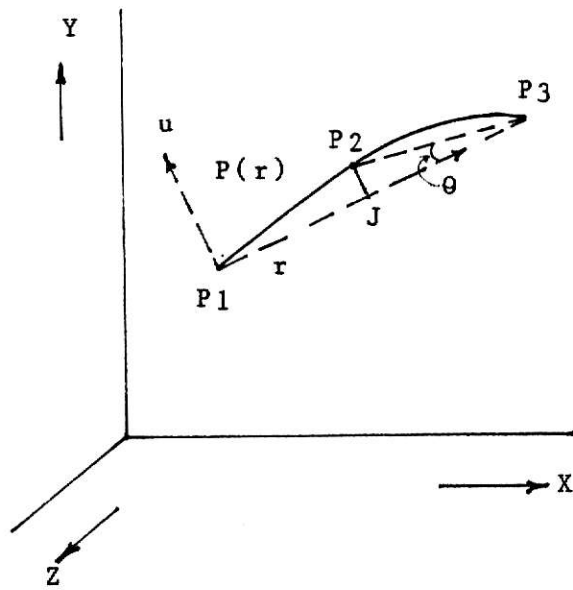
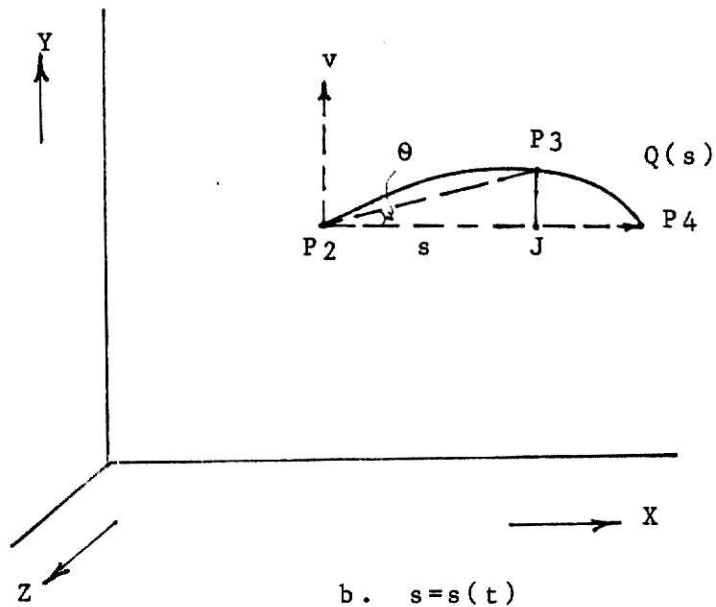
$$\{P_2-[P_1+x(P_3-P_1)]\}.(P_3-P_1) = 0 \quad \text{.....4.18}$$

Solving for x in the XYZ-coordinate system yields

$$x = \frac{(P_2-P_1).(P_3-P_1)}{(P_3-P_1)^2} \quad \text{.....4.19}$$

$$= \frac{(P_2-P_1).(P_3-P_1)}{d^2} \quad \text{.....4.20}$$

With this information the vector equation for a point P on the parabola $P(r)$, relative to the XYZ-coordinate system, is given

a. $r=r(t)$ b. $s=s(t)$ Fig.4.7 Geometric relationships : $r(t)$ and $s(t)$

by

$$P(r) = P_1 + \frac{r}{d}(P_3 - P_1) + \alpha r(d-r)(P_2 - J) \quad \dots\dots 4.20$$

Using Equations 4.17 for J, we get

$$P(r) = P_1 + \frac{r}{d}(P_3 - P_1) + \alpha r(d-r)[(P_2 - P_1) - x(P_3 - P_1)] \quad \dots\dots 4.21$$

It is required to determine α and the parametric equation for $r(t)$. Since in the ur-coordinate system $P(xd)=P_2$, it follows that the vector equation P_2-J is

$$P_2 - J = \alpha xd(d-xd)(P_2 - J) \quad \dots\dots 4.22$$

or

$$\alpha = \frac{1}{d^2 x(1-x)}$$

The required relationship for $r=r(t)$ can be obtained from the geometry shown in Fig.4.7. It follows that

$$r = xd + t \cos \theta \quad \dots\dots 4.23$$

where

$$\cos \theta = (P_3 - P_2) \cdot \left(\frac{P_3 - P_1}{t_0 d} \right) \quad \dots\dots 4.24$$

For the parabola $Q(s)$, similar equations can be derived using Fig.4.7b. For the parabola $Q(s)$, the relationship $s=s(t)$ is

$$s = t \cos \theta = t \left[(P_3 - P_2) \cdot \left(\frac{P_4 - P_2}{t_0 e} \right) \right]$$

and

$$Q(s) = P_2 + \frac{s}{e} (P_4 - P_2) + \beta s(e-s)[(P_3 - P_2) - x(P_4 - P_2)] \quad \dots\dots 4.25$$

Once the points are specified, the procedure is to calculate x by using Equation 4.19 and α using Equation 4.22. For a given value of t , r is given by Equations 4.23 and 4.24. Finally, points on the curve, $P(r)$, are calculated using

Equation 4.20. This procedure is then repeated for the $Q(s)$ parabola.

To continue generating a curve through additional points, a blending curve $C_i(t_i)$ is formed between each adjacent pair of points. This creates a continuous curve which is also continuous in first derivative at the internal data points. These first derivatives can be easily determined if Equation 4.15 is written as

$$C(t) = P(t) + \left(\frac{t}{t_0}\right)[Q(t) - P(t)].$$

Then

$$\frac{dC}{dt} = \frac{dP}{dt} + \left(\frac{t}{t_0}\right)\left(\frac{dQ}{dt} - \frac{dP}{dt}\right) + \left(\frac{1}{t_0}\right)(Q - P).$$

At point P_2 on the blending curve, $t=0$ and $P=Q$. Thus

$$\left(\frac{dC}{dt}\right)_{P_2} = \left(\frac{dP}{dt}\right)_{P_2}.$$

That is, the slope of the blending curve equals the slope of the parabola $P(r)$ at P_2 . Likewise, at P_3 on the blending curve, $t=t_0$ and $P=Q$. Thus

$$\left(\frac{dC}{dt}\right)_{P_3} = \left(\frac{dQ}{dt}\right)_{P_3}.$$

Parabolic blending can be used only for internal segments of a curve. The two end segments must each be a single parabola, defined through the first and the last three data points respectively.

When an artist, stylist, or designer sketches, he or she uses short, overlapping strokes to produce the contour desired. This is like the technique that can be used with parabolic blending. Once the sketch is defined in a computer, then the

vectors defining the points can be quickly displayed in a variety of ways. In some applications it may be desirable to sketch a shape by using parabolic blending and then use the resulting junction points as data for other techniques.

An algorithm which will implement the parabolic blending technique described above is given in Appendix 3. As an example, the parabolic blending technique was applied to the data points given by $P_1(0,0)$, $P_2(1,1)$, $P_3(3,2)$, $P_4(4,3)$. The resulting curve between the two points P_2 and P_3 was determined by using the computer program in the BASIC language shown in Appendix 3 (for parabolic blending), and the curve was plotted by using the digital plotter. This plot is depicted in Fig.4.8.

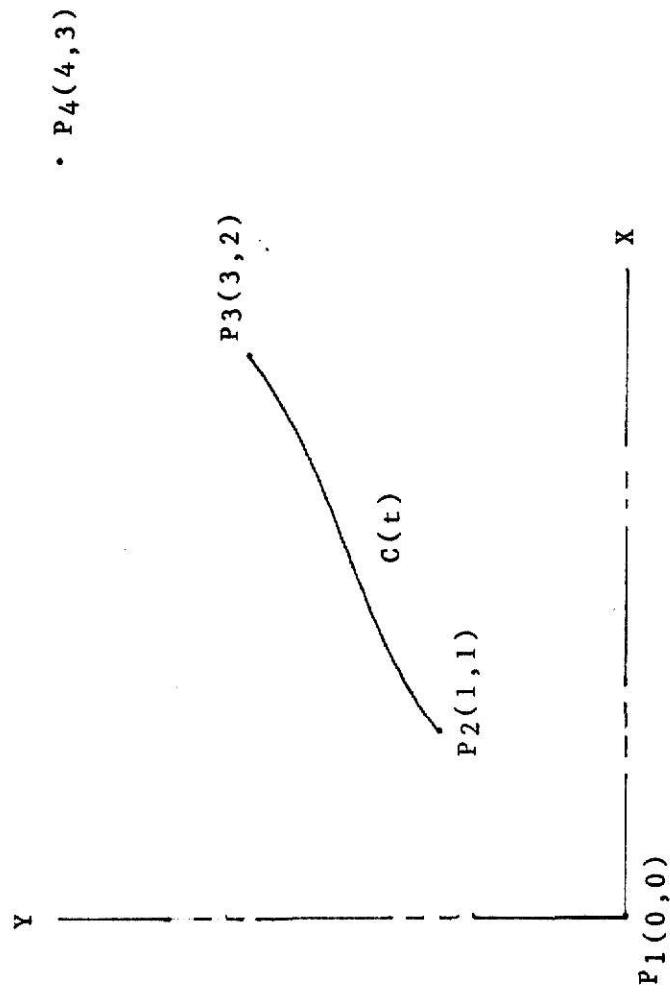


fig.4.8 Parabolic blending for example in the text.

CHAPTER 5

MOTION IN A PLANE GIVEN BY AN EQUATION

MOTION IN A PLANE GIVEN BY AN EQUATION

This chapter discusses the generation of curves that are more general than those discussed in Chapter 3. The procedure has been carried out by categorizing the curves given by equations into the following three groups :

i) Curves given by an equation, where Y is a function of X and for a given value of X , there exists only one value of Y .

This can be expressed as

$$Y = \sum_{i=0}^n a_i X^i, \text{ where } n > 0 \text{ and } a_i \text{ is a constant}$$

ii) Curves given by an equation, where X is a function of Y and for a given value of Y , there exists only one value of X .

This can be expressed as

$$X = \sum_{i=0}^n b_i Y^i, \text{ where } n > 0 \text{ and } b_i \text{ is a constant}$$

iii) Curves given by an equation of the form as given below:

$$F(X, Y) = 0$$

5.1 CURVES GIVEN BY AN EQUATION : $Y=F(X)$

Consider a motion to be generated along the curve given by an equation $Y=F(X)$ within the range of X values given by X_M and X_X . The following steps are carried out for the generation of the curve :

Step 1 : Input X_M into the given equation and calculate the corresponding value of Y . Let this value be Y_1 .

Step 2 : Increase X_M by a small amount DX . This incremental size depends on the accuracy desired.

If $(X_M+DX) < X_X$ then go to step 3; otherwise go to step 7.

Step 3 : Find the corresponding value of Y . Let this value be Y_2 .

Step 4 : Feed the values of $X=(X_M+DX-X_M)$ and $Y=Y_2-Y_1$ into the line

subroutine and obtain the motion along the first line segment.

Step 5 : Since in the line subroutine X and Y are made nearest integer multiples of the step size, the end point coordinates of the first line segment will not necessarily be (X_M+DX, Y_2) . Let the new adjusted coordinates be (X_2', Y_2') .

Reassign $X_M=X_2'$ and $Y_1=Y_2'$.

Step 6 : Go to Step 2.

Step 7 : Stop.

The detailed computer flow diagram is illustrated in Fig.5.1. Based on this flow diagram , the computer program is written in the BASIC language and is listed in Appendix 4.

The input variables for this program are given below :

X_M = Lower range of X value.

XX = Higher range of X value.

DX = Least increment along X.

The constants $N=1000$, $M=5$ and $DT=.005$ are used in the line subroutine. Several curves drawn by using this program are shown in Fig.5.2 and Fig.5.3.

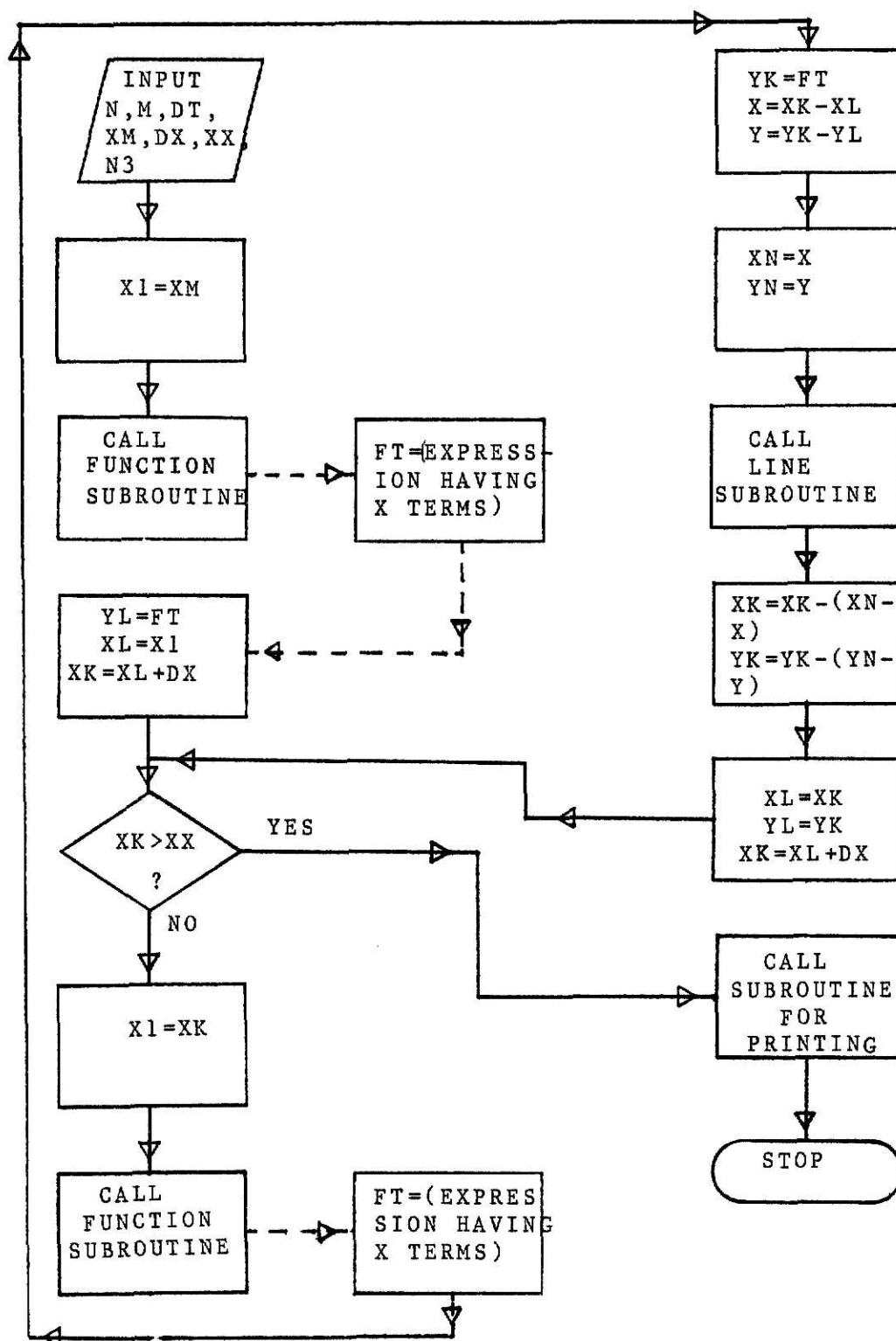
5.2 CURVES GIVEN BY AN EQUATION : $X=F(Y)$

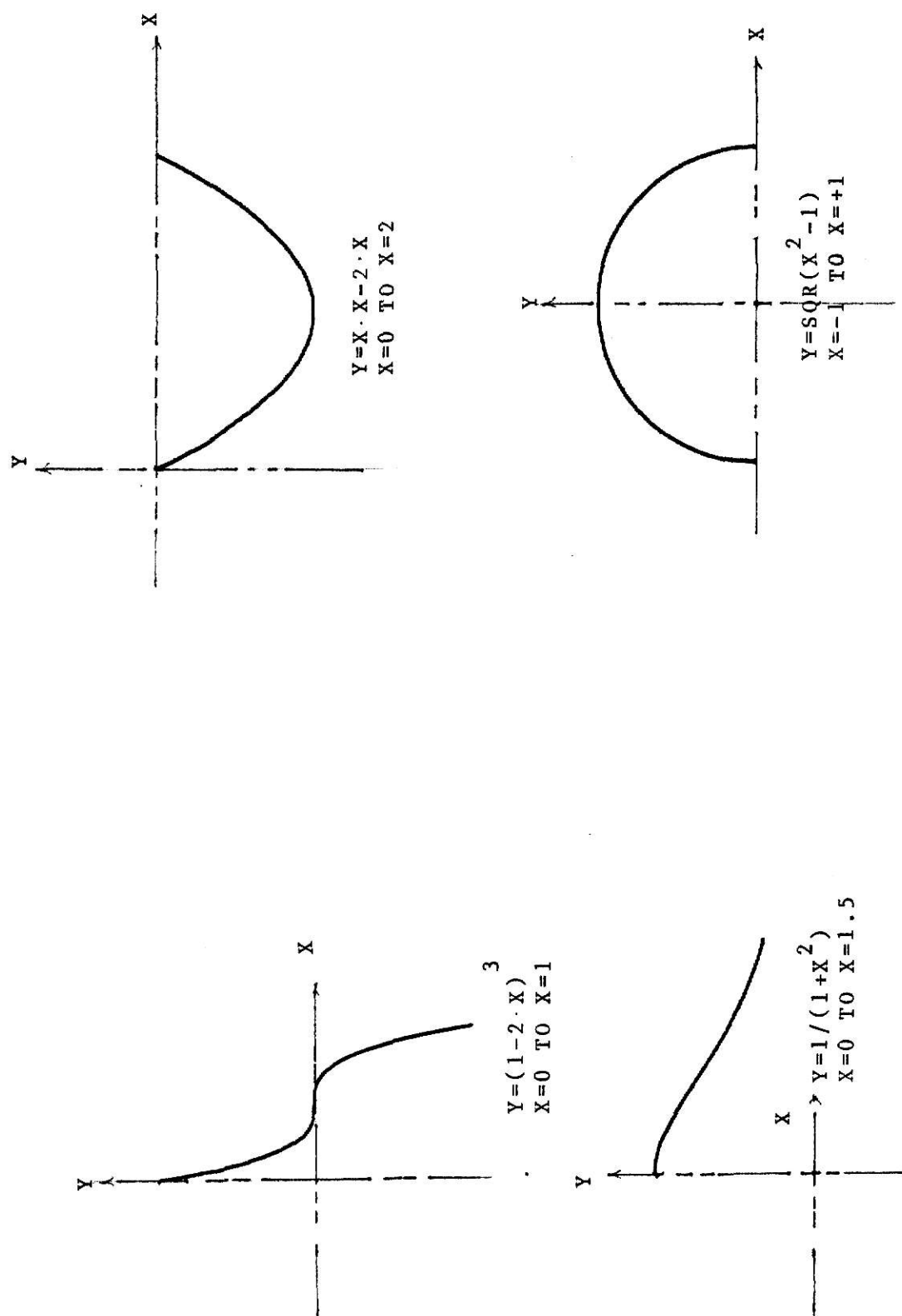
The algorithm for the curves given by an equation $X=F(Y)$ is similar to the one explained for the curves given by an equation $Y=F(X)$. The detailed computer flow diagram is shown in Fig.5.4. The input variables are given below :

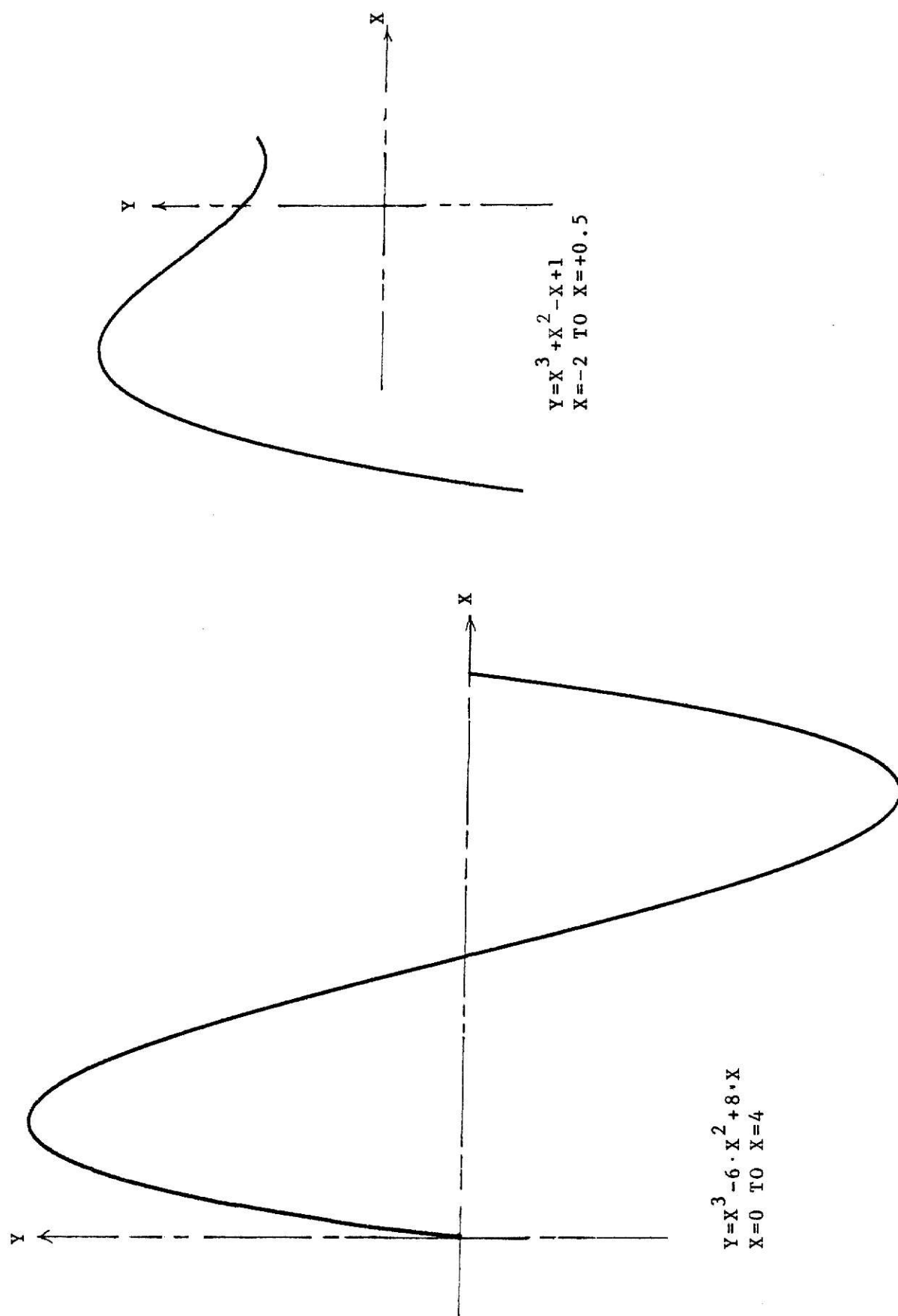
Y_M = Lower range of Y value.

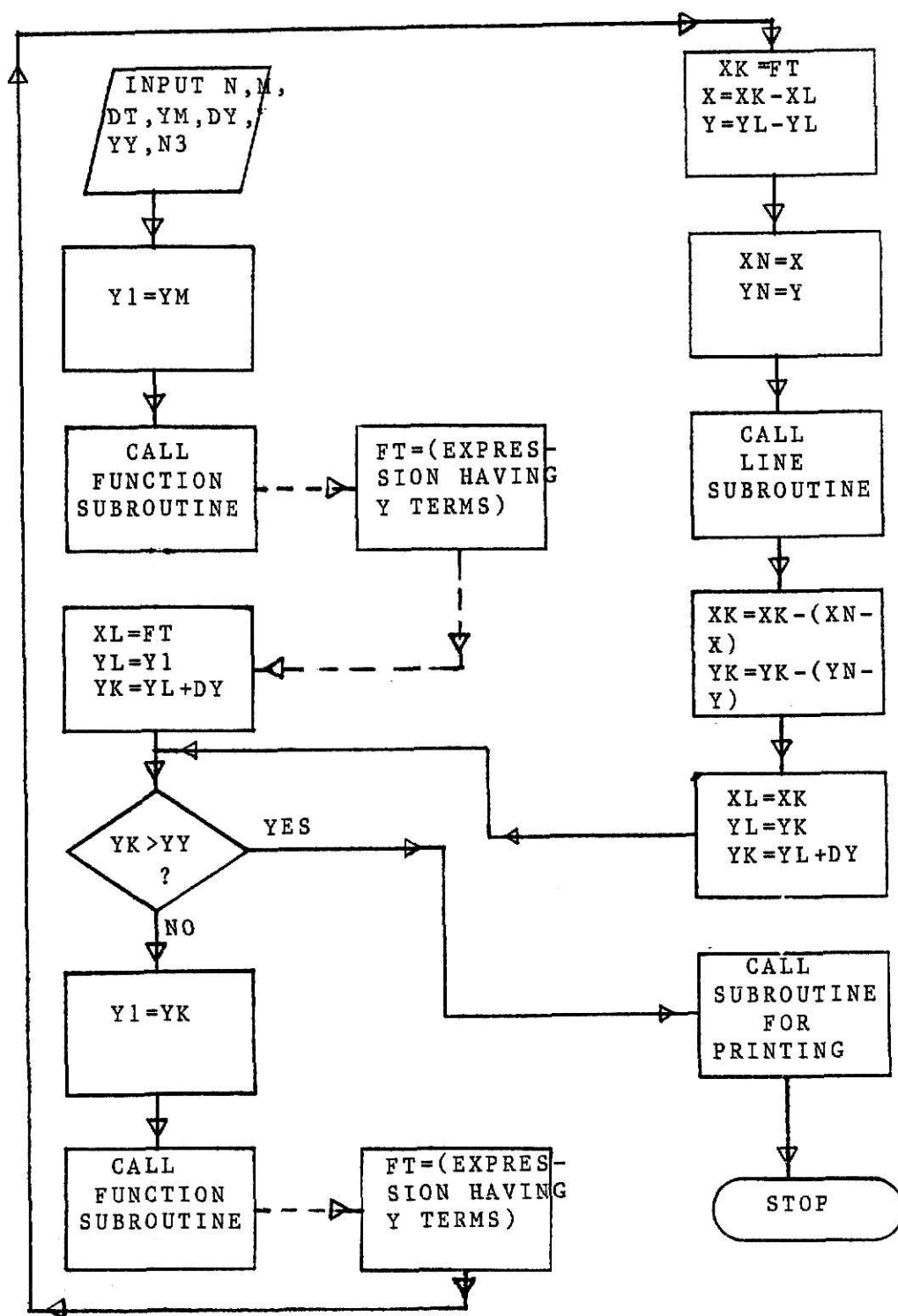
YY = Higher range of Y value

DY = Least increment along Y.

Fig.5.1 Flow Diagram : Equation of Curve $Y=F(X)$

Fig.5.2 Curves in a Plane : $Y=F(X)$

Fig.5.3 Curves in a Plane : $Y=F(X)$

Fig.5.4 Flow Diagram : Equation of Curve $X=F(Y)$

The constants, $N=1000$, $M=5$ and $DT=.005$, are used in the line subroutine. The listing of the computer program in the BASIC language is shown in Appendix 4.

5.3 CURVES GIVEN BY AN EQUATION : $F(X,Y)=0$

Consider a motion to be generated along the curve given by an equation $F(X,Y)=0$ within a range of X values given by $A1$ and $B1$. The following steps explain in detail the procedure for generating the path :

Step 1 : Input the value of $X(=A1)$ into the equation $F(X,Y)=0$. This will result in obtaining a one variable equation given by $F(Y)=0$.

Step 2 : Solve the equation, $F(Y)=0$ for Y . In order to do this, several numerical methods can be used. However, the one being considered here is "The Newton Method". The following steps are used for finding the roots of the function $F(Y)$.

- i) Select a trial point Y_1 .
- ii) Find the value of $F(Y)$ at $Y=Y_1$.
- iii) Obtain the first order derivative of $F(Y)$, given by $F'(Y)$.
- iv) Find the value of $F'(Y)$ at $Y=Y_1$.
- v) Find the new trial point given by

$$Y_2 = Y_1 - F(Y_1)/F'(Y_1)$$
- vi) Find $F(Y_2)$.
- vii) If the absolute value of $F(Y_2)$ is less than ϵ (the value of ϵ is chosen depending on the accuracy desired) then stop. Store the value of Y_2 .

If the absolute value of $F(Y2)$ is greater than ϵ then assign $Y1=Y2$. Go to step ii.

Step 3 : $X=A1+DX$, where DX is the least increment which can be selected depending on the accuracy desired.

If $X>B1$ then go to Step 5; otherwise go to Step 4.

Step 4 : Go to Step 2.

Step 5 : Stop.

As explained for the other curves, the coordinates of all the points on the curve are fed to the line subroutine to generate the path for the equation desired.

The computer flow diagram for this algorithm is shown in Fig. 5.5. The listing of the computer program in the BASIC language is illustrated in Appendix 4. The input variables for this program are given below :

$A1$ = Lower range of X value.

$B1$ = Higher range of X value.

NM = Approximate value of Y .

DX = Least increment along X .

The values of $N(=1000)$, $M(=5)$ and $DT(=.005)$ are used in the line subroutine. Some curves plotted for different equations are shown in Fig.5.6 and Fig.5.7.

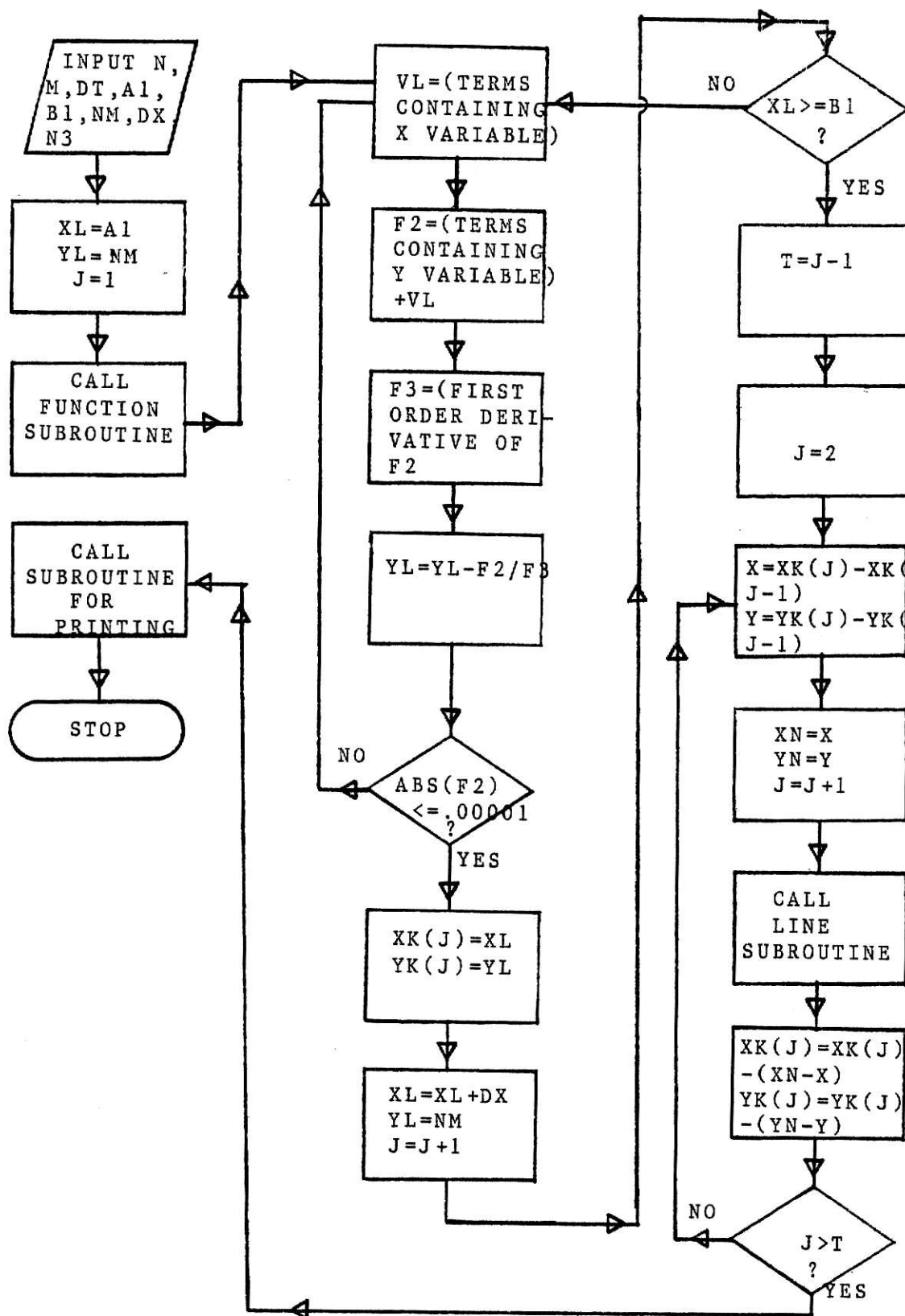


Fig.5.5 Flow Diagram for Generalized Curve in a Plane

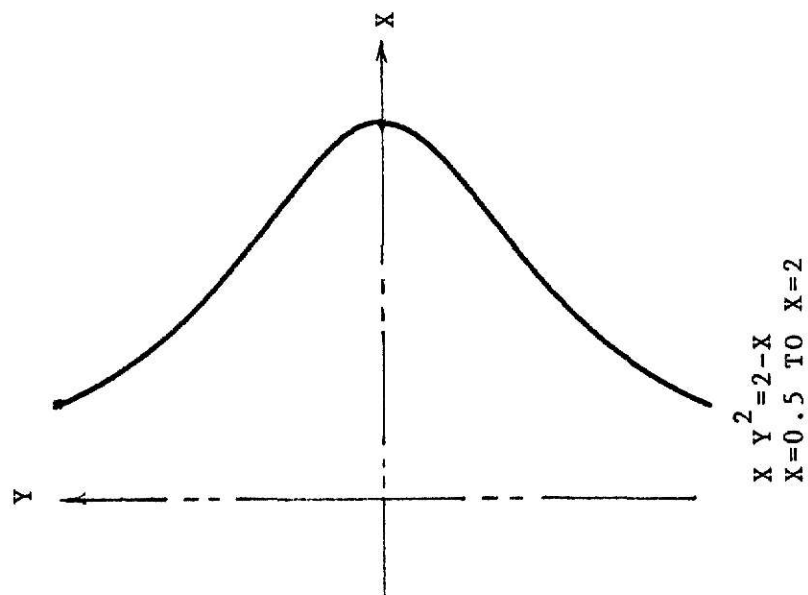
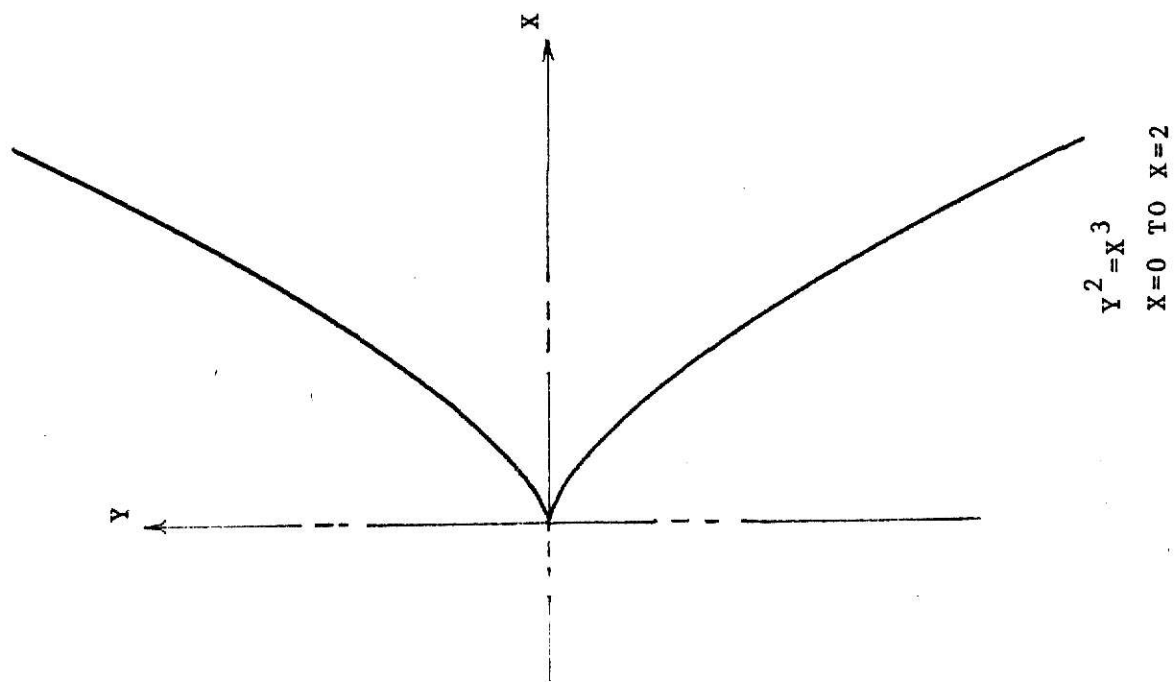
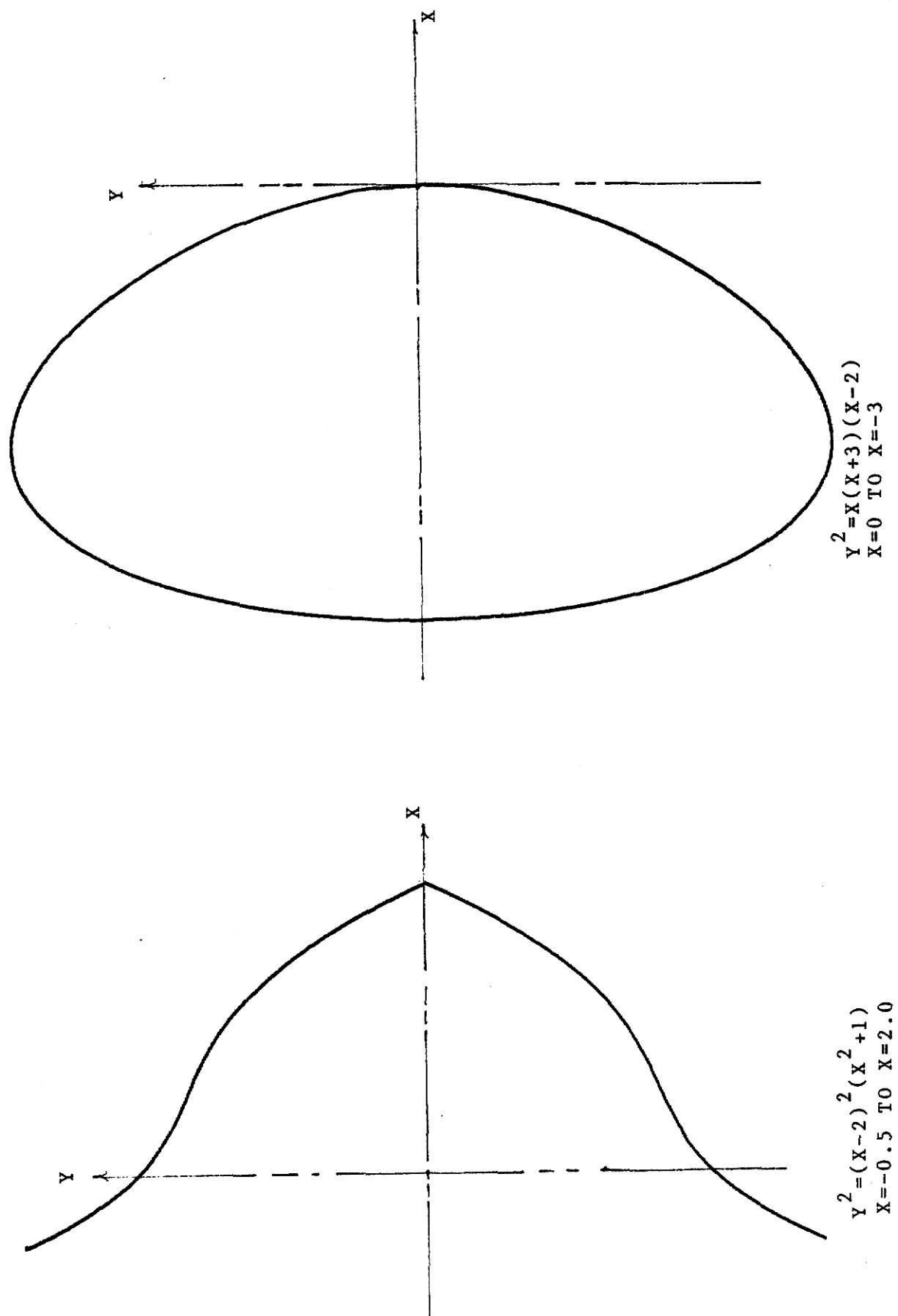


Fig.5.6 Curves in a Plane : Any Equation: $F(X,Y)=0$

Fig.5.7 Curves in a Plane : Any Equation : $F(X,Y)=0$

CHAPTER 6
CURVES IN SPACE

CURVES IN SPACE

Most objects encountered in the real world are three-dimensional in nature. In order to facilitate machining of these objects, it is essential to generate motions of a tool along curves in space. Space curves can be displayed on a two-dimensional plane by treating the Z-axis as a line passing through the origin, at an angle of 45° with +X-axis. The present chapter extends the previous discussions for curve descriptions to a three dimensional space curve.

The study has been carried out by classifying space curves into the following two groups :

i) Curves given by the equation of the intersection of two surfaces.

ii) Bezier curves.

6.1 CURVES GIVRN BY THE INTERSECTION OF TWO SURFACES

Consider two surfaces defining a space curve given by

$$F(X,Y,Z) = 0 \quad \text{.....6.1}$$

$$G(X,Y,Z) = 0 \quad \text{.....6.2}$$

Now, if a motion is to be generated along this space curve between the two values of Z given by AI and AF ($AF > AI$), it is required to obtain the coordinates of the desired number of points along the curve so that straight line approximations can be made to generate the required curve. The procedure for determining the coordinates of points along the curve is explained below :

The value of Z(=AI) is substituted in Equations 6.1 and 6.2 which results in obtaining two equations with two variables,

given by

$$F_1(X,Y) = 0 \quad \text{.....6.3}$$

$$G_1(X,Y) = 0 \quad \text{.....6.4}$$

These two equations can be solved for X and Y by using the "Newton Raphson method". For the solution of these equations by the "Newton Raphson method", it is necessary to have initial approximate values for X and Y. Let these values be X=NM and Y=NN. The values of $F_1(X,Y)$ and $G_1(X,Y)$ are calculated at X=NM and Y=NN. If one of these values is greater than ϵ (ϵ is taken as .0001 for the algorithm explained below), then the new values of X and Y are calculated(explained in a moment). At these new values of X and Y, the values of $F_1(X,Y)$ and $G_1(X,Y)$ are calculated. The procedure is continued until the values of both, $F_1(X,Y)$ and $G_1(X,Y)$ are less than ϵ . The corresponding values of X and Y, when $F_1(X,Y)$ and $G_1(X,Y)$ are less than ϵ , are stored. These stored values of X and Y are used as the initial approximate values for the next point on the curve given for $Z=AI+DZ$, where DZ is a small increment along the Z-direction and whose value depends on the accuracy desired. The same method is used to calculate the values of X and Y for $Z=AI+DZ$. This is continued until the value of Z is equal to AF. Once the coordinates of the desired number of points are obtained, the motion along the curve,given by the intersection of two surfaces, can be generated by making straight line approximations. The computer program listed in Appendix 1, for the lines in space, is treated as a subroutine to achieve this. The following steps explain the algorithm in detail :

Step 1 : Set $Z=AI$ (Lower range of Z value)

Set $X=NM$ (Approximate value of X)

Set $Y=NN$ (Approximate value of Y)

Step 2 : Substitute the value of Z in the the Equations 6.1 and 6.2 to obtain two equations in two unknowns which are given by

$$F_1(X,Y) = 0$$

$$G_1(X,Y) = 0$$

Step 3 : Substitute $X=NM$ and $Y=NN$ in $F_1(X,Y)$ and $G_1(X,Y)$.

Step 4 : If $ABS(F_1(X,Y)) > .0001$ then go to step 6; otherwise go to step 5.

Step 5 : If $ABS(G_1(X,Y)) > .0001$ then go to step 6; otherwise go to step 9.

Step 6 : Input $\frac{d}{dY} F_1(X,Y)$, $\frac{d}{dX} F_1(X,Y)$, $\frac{d}{dY} G_1(X,Y)$ and $\frac{d}{dX} G_1(X,Y)$, and calculate the values of each of these derivatives at $X=NM$ and $Y=NN$.

Step 7 : Calculate the new values of X and Y as given below :

$$\begin{bmatrix} X(\text{new}) \\ Y(\text{new}) \end{bmatrix} = \begin{bmatrix} X \\ Y \end{bmatrix} - \begin{bmatrix} \frac{d}{dX} F_1 & \frac{d}{dY} F_1 \\ \frac{d}{dX} G_1 & \frac{d}{dY} G_1 \end{bmatrix}^{-1} \begin{bmatrix} NM \\ NN \end{bmatrix}$$

Set $X=X(\text{new})$ and $Y=Y(\text{new})$.

Step 8 : Go to step 2.

Step 9 : Store the values of X and Y corresponding to the value of Z .

Step 10 : Let $Z=Z+DZ$. DZ is a small incremental value along the Z -direction and is selected depending on the accuracy desired.

Step 11 : If $Z > AF$ then go to step 12; otherwise go to step 2.

Step 12 : Stop.

Once the coordinates of the desired number of points along the curve are obtained, the line subroutine for lines in space

is called to generate the motion along the curve. The detailed computer flow diagram is shown in Fig.6.1. The listing of the computer program in the BASIC language is given in Appendix 5. Based on this computer program several three dimensional curves have been plotted and are shown in Fig.6.2.

6.2 BEZIER CURVES [3]

The Bezier curves are named after P.E. Bezier of Renault who is credited with conceiving the procedure by which any span or set of points may be used to develop a curve with certain special properties. The curve generated will be a polynomial of degree $n-1$, which may be generated by varying a parameter t between 0 and 1.

A Bezier curve is associated with the "vertices" of a polygon which uniquely define the curve shape. Only the first and the last vertices of the polygon actually lie on the curve; however, the other vertices define the derivative, order, and shape of the curve. Thus in general the curve is defined by an open polygon as shown in Fig.6.3. Since the curve shape will tend to follow the polygon shape, changing the vertices of this polygon gives the user a much greater intuitive feeling for input/output relationships. If the order of the curve is to be increased then this could be achieved by specifying another interior vertex. Any change in the vertices of a portion of the curve will effect the curve within that portion of the curve. The rest of the portion of the curve will be unaffected.

6.2.1 Mathematical basis of the Bezier curve

The basis function for the generation of Bezier curve is

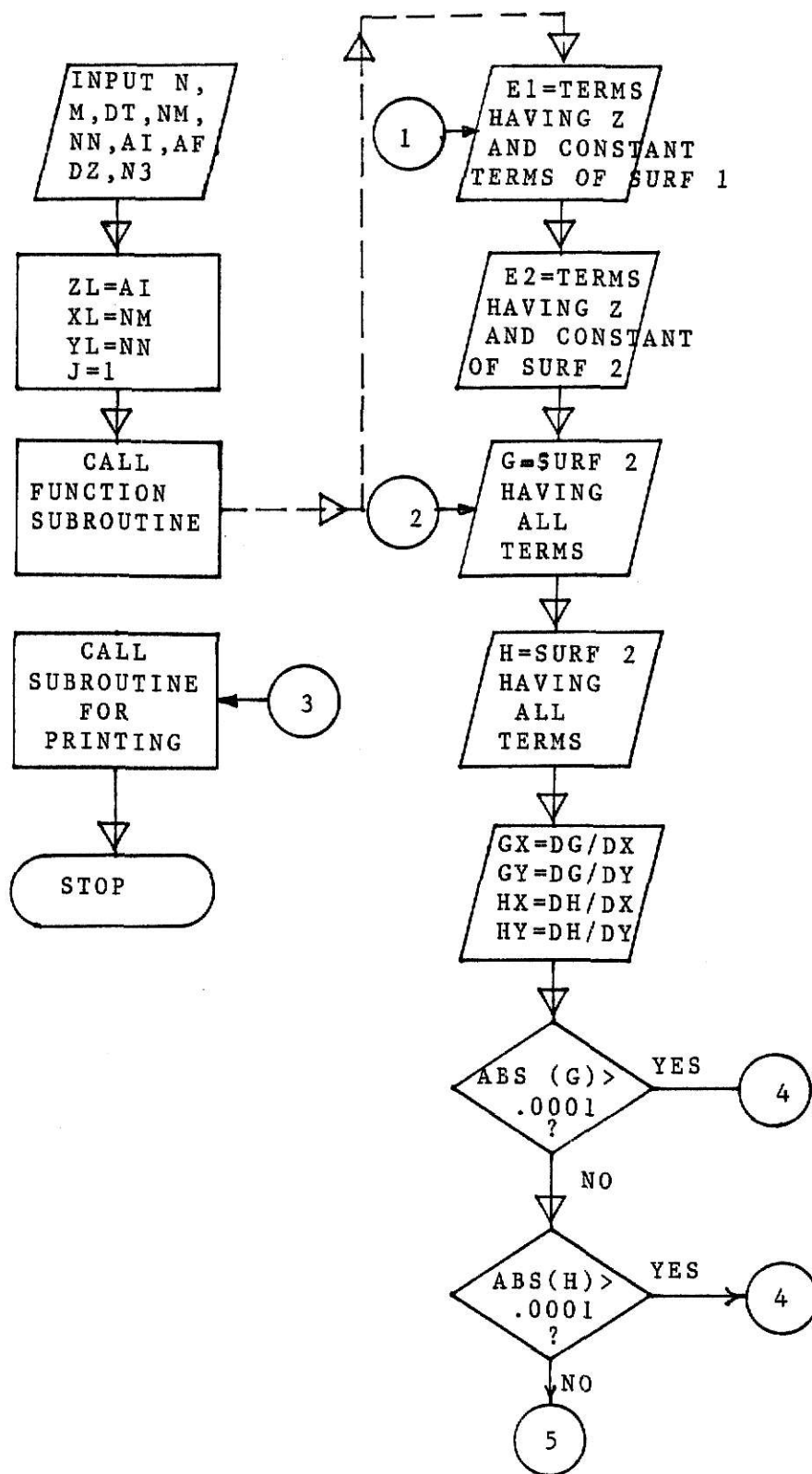


Fig. 6.1 Flow diagram for curves in space

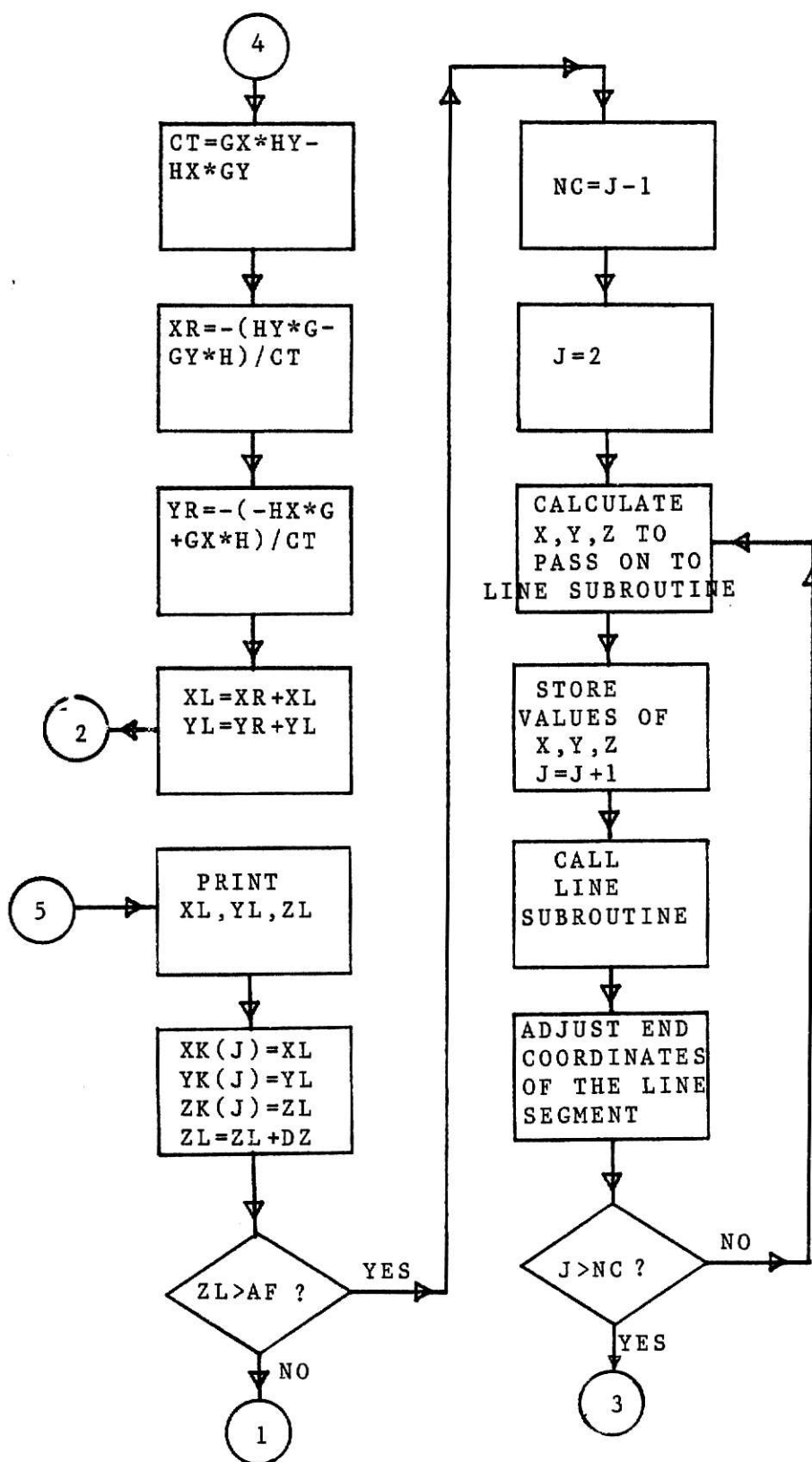


Fig.6.1 (Continued)

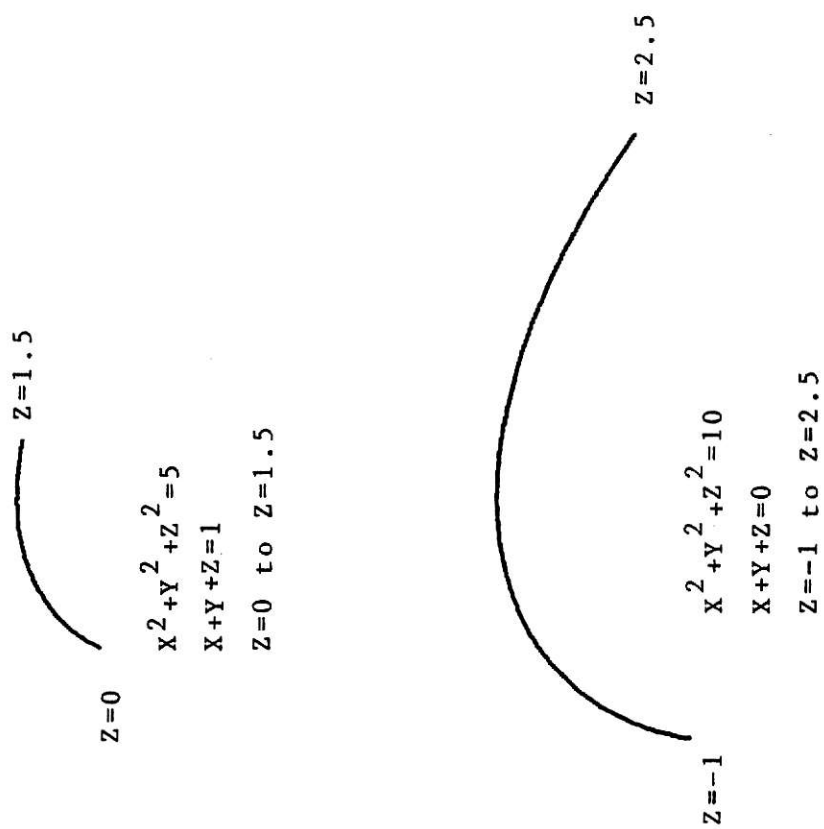


Fig.6.2 Curves in space projected on a plane

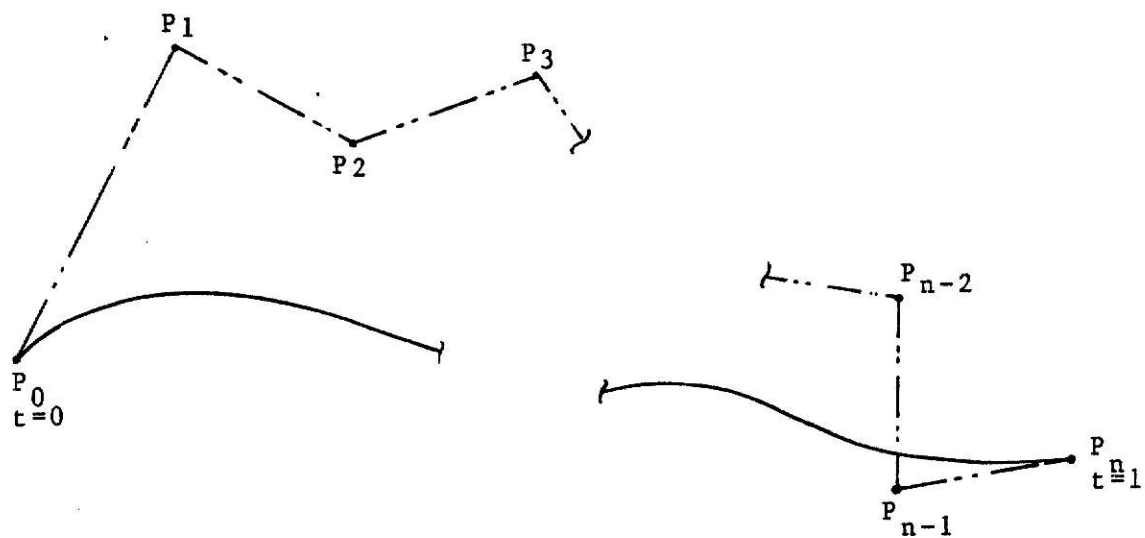


Fig.6.3 Nomenclature for Bezier curves.

given by

$$F_{n,i}(t) = \binom{n}{i} t^i (1-t)^{n-i} \quad \dots\dots 6.5$$

where

$$\binom{n}{i} = \frac{n!}{i!(n-i)!} \quad \dots\dots 6.6$$

where n is the degree of the polynomial and i is the particular vertex in the ordered set from 0 to n . In general an n^{th} order polynomial is specified by $(n+1)$ vertices. The curve points are given by

$$P(t) = \sum_{i=0}^n P_i F_{n,i}(t), \quad 0 < t < 1 \quad \dots\dots 6.7$$

where P_i contains the vector component of the various vertices.

At the starting point of a curve segment,

$$\lim_{t \rightarrow 0} F_{n,0}(t) = \frac{n!(1)(1-0)^{n-0}}{n!} = 1.$$

From Equation 6.7, we get

$$P(0) = P_0.$$

At the end point of a curve segment,

$$\lim_{t \rightarrow 1} F_{n,n}(t) = \frac{n!(1)^n(t)^{n-n}}{n!(1)} = 1.$$

Hence from Equation 6.7, we get

$$P(1) = P_n.$$

This indicates that the vertices P_0 and P_n lie on the actual curve segment at the starting point and the end point respectively.

Another important characteristic of the basis function is that maximum values occur at $t=i/n$. This value is given by

$$F_{n,i}\left(\frac{i}{n}\right) = \binom{n}{i} \frac{i^i (n-i)^{n-i}}{n^n}$$

For a cubic curve this simplifies to

$$F_{3,1}(1/3) = 4/9$$

$$F_{3,2}(2/3) = 4/9$$

The procedure for generating the points along the Bezier curve is explained below with reference to Fig.6.4.

For this case $n=3$. Assume equal increments in the parameter t , say $t=0, 1/4, 1/2, 1$. Then using Equations 6.5 and 6.6 yields

$$\begin{aligned} F_{3,0}(t) &= \frac{3!}{0! 3!} t^0 (1-t)^3 \\ &= (1-t)^3 \\ F_{3,1}(t) &= \frac{3!}{1! 2!} t(1-t)^2 \\ &= 3t(1-t)^2 \\ F_{3,2}(t) &= \frac{3!}{2! 1!} t^2(1-t) \\ &= 3t^2(1-t) \\ F_{3,3}(t) &= \frac{3!}{3! 0!} t^3(1-t)^0 \\ &= t^3 \end{aligned}$$

The results are tabulated in Table 6.1. From Equation 6.7, we have

$$P(1/4) = \frac{27}{64} P_0 + \frac{27}{64} P_1 + \frac{9}{64} P_2 + \frac{1}{64} P_3$$

Thus, to create a cubic curve segment, it is only necessary to specify the four polygon vertices and then calculate points along the curve for $0 < t < 1$ using Equations 6.5 and 6.6. A similar procedure is adopted for generating the points for the curve of any order. It is not necessary to explicitly consider parametric derivatives. A user can quickly learn to predict the shape of a curve which will be generated by a certain polygon shape.

6.2.2 Advantages of Bezier curves

i) Bezier curves require no input derivatives, just data

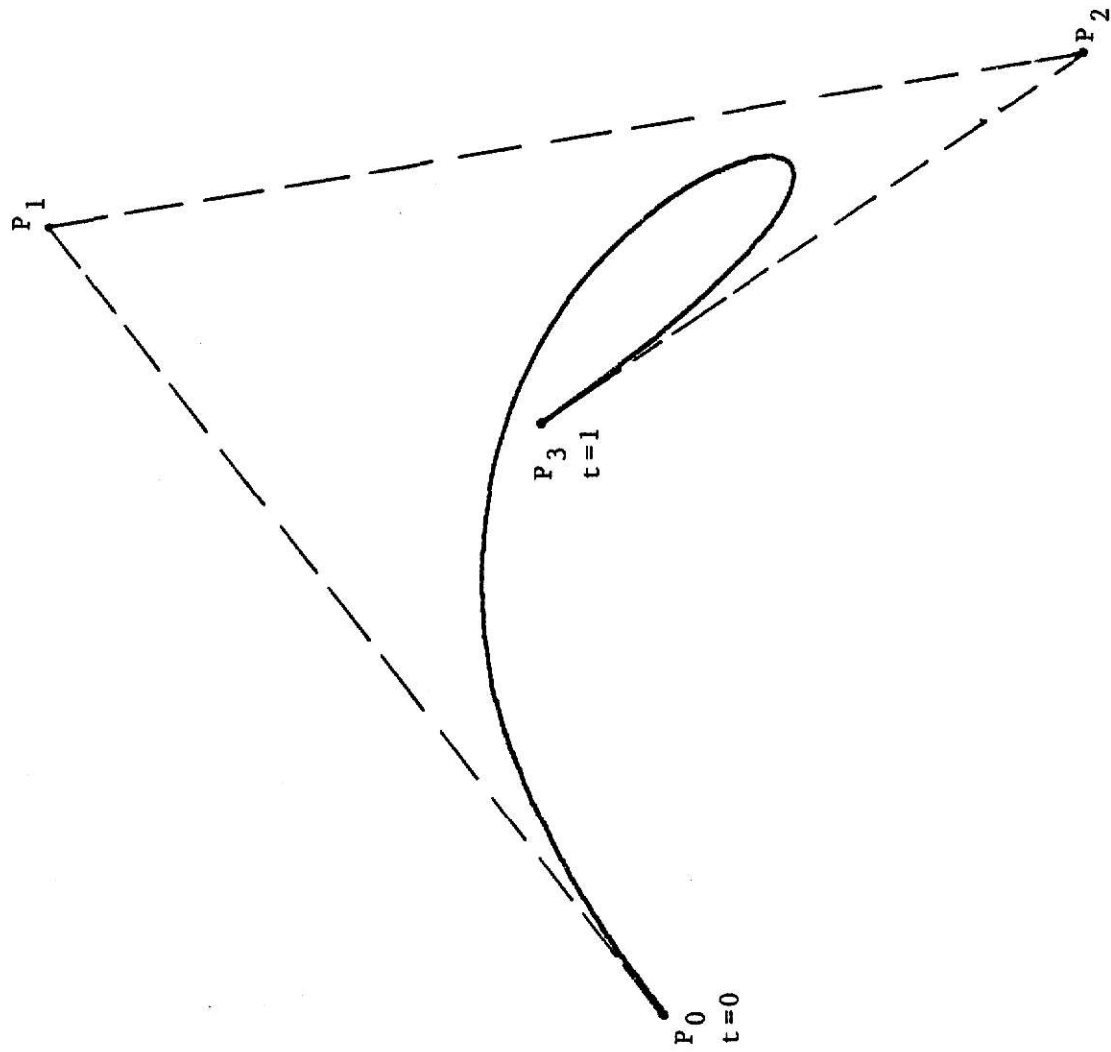


Fig.6.4 Bezier polygon for cubics

Table 6.1. The results of an example for a cubic curve

The values of $F_{3,i}$					
$i \backslash t$	0	1/4	1/2	3/4	1
0	1	27/64	1/8	1/64	0
1	0	27/64	3/8	9/64	0
2	0	9/64	3/8	27/64	0
3	0	1/64	1/8	27/64	1

points.

ii) Since the degree of a Bezier polygon depends strictly on the number of points in a designated span, polynomials of different orders may be automatically developed from span to span.

iii) Bezier curves are well suited to interactive graphics, since the data points are guides or controls to the shape. That is, they influence the curve shape, although the curve itself passes through only the first and last points of a designated span of points. The change of location of data points done easily at a console, changes the influence on the resulting curve.

iv) Since the Bezier technique develops a single equation for each variable, computer storage, I/O, and computation should be somewhat decreased.

The computer flow diagram for the generation of a Bezier curve for two and three dimensional cases is shown in Fig.6.5. The listing of the computer program in the BASIC language is given in Appendix 5. Based on this program several two dimensional curves have been plotted and are shown in Fig.6.6.

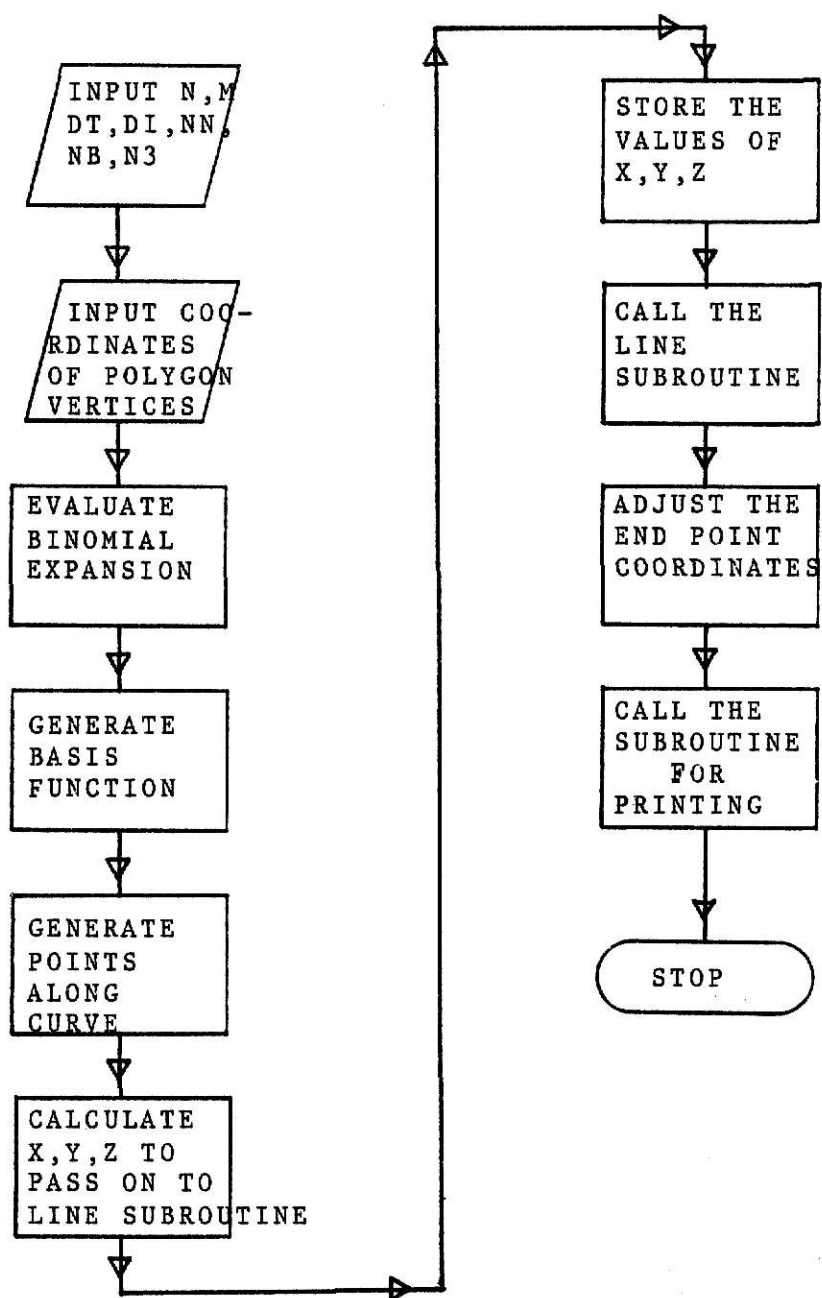


Fig.6.5 Flow Diagram for Bezier curves.

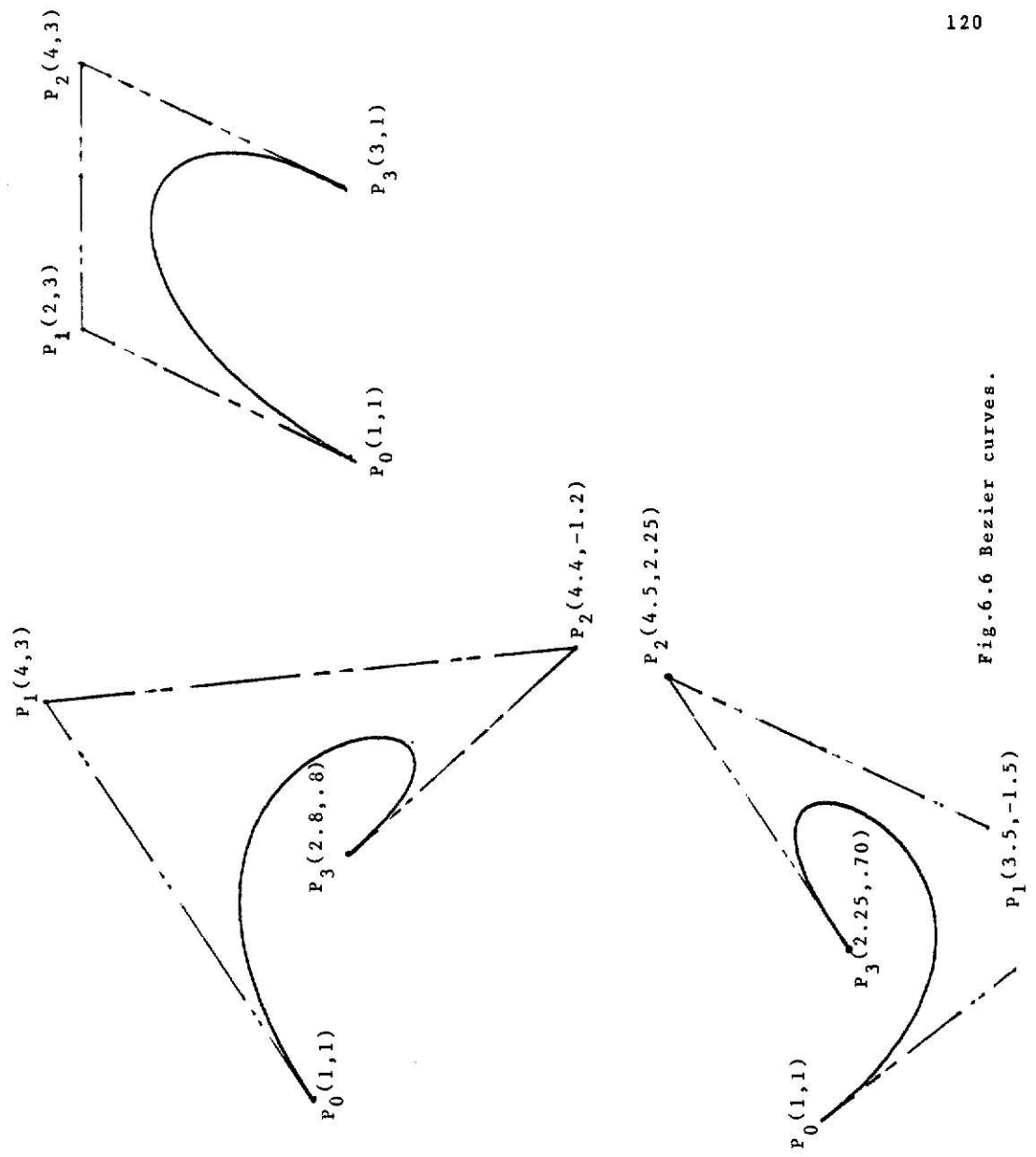


Fig.6.6 Bezier curves.

APPENDIX 1

LINE SUBROUTINE FOR LINES IN A PLANE

```

980 PRINT"INPUT IS X,Y,N,M,DT,N3"
982 REM N=1000,M=1000DT,DT=STEP SIZE,N3=DT/.005
985 INPUT X,Y,N,M,DT,N3
988 REM MAKE X AND Y THE NEAREST INTEGER MULTIPLES OF DT
990 X=(INT((X*N)/M))*M/N
992 Y=(INT((Y*N)/M))*M/N
993 DIM V(2000),A$(8)
994 A$="pqrstuvw":REM INPUT CHARACTER STRING
995 REM CALCULATE THE NUMBER OF STEPS
996 F=(ABS(X)+ABS(Y))/DT
1000 IF X>0 THEN 1006
1002 R=7
1004 GOTO 1008
1006 R=3
1008 IF Y>0 THEN 1014
1010 P=5
1012 GOTO 1018
1014 P=1
1018 A=ABS(X):B=ABS(Y)
1020 X=0:Y=0:Z=0
1022 IF A=0 THEN 1065
1025 C=B/A
1026 REM INCREASE X BY DT
1030 X=X+DT
1032 REM CALCULATE D1 FOR COMPARING THE DEVIATIONS
1035 D1=ABS(C*X-Y)
1040 X=X-DT
1042 REM INCREASE Y BY DT
1045 Y=Y+DT
1046 REM CALCULATE D2 FOR COMPARING THE DEVIATIONS
1047 D2=ABS(C*X-Y)
1050 Y=Y-DT
1053 REM FIND THE SMALLEST OF D1 AND D2
1055 IF D1>D2 THEN 1065
1057 V(K)=R
1059 F=F-1:X=X+DT
1060 IF F=0 THEN 1110
1062 K=K+1
1063 GOTO 1022
1065 V(K)=P
1070 F=F-1:Y=Y+DT
1072 IF F=0 THEN 1110
1074 K=K+1
1076 GOTO 1022
1110 GOSUB 1295
1115 STOP
1295 REM THIS IS A SUBROUTINE FOR PRINTING
1297 PRINT CHR$(17)

```

```
1300     PRINT "z           "  
1305 FOR J=1 TO K  
1307 FOR J1=1 TO N3  
1310 PRINT MID$(A$,V(J),1)  
1314 NEXT J1  
1318 NEXT J  
1325 RETURN
```

LINES IN A PLANE : CASE 2 AND CASE 3

```

1480 REM THIS PROGRAM IS FOR THE PATH ABOVE OR BELOW THE LINE
1482 REM DT=STEP SIZE
1483 REM N3=DT/.005
1484 REM IF TL=1 THEN THE PATH IS ABOVE THE LINE IN FIRST QUADRA
NT
1486 REM THE ROTATION OF THIS LINE IN THE ANTI-CLOCKWISE
1487 REM DIRECTION GIVES THE PATH IN THE OTHER QUADRANTS.
1488 REM IF TL>1 THEN THE PATH IS OPPOSITE TO THE ABOVE LINE.
1489 REM N,M ARE FOR MAKING X,Y THE INTEGER MULTIPLES OF DT
1490 PRINT "INPUT IS X,Y,N,M,DT,TL,N3"
1495 INPUT X,Y,N,M,DT,TL,N3
1496 IF X<>0 THEN 1500
1497 C=9999999:L=0:REM GIVE A LARGE VALUE FOR SLOPE C
1498 A=9999999:GOTO 1565
1499 REM FOR A LARGE VALUE OF "A" C IS GREATER THAN SLOPE SL AT
X=0
1500 IF TL=1 THEN 1540
1510 IF X>0 THEN 1525
1515 IF Y>0 THEN 1530
1518 REM CS=2 MEANS CASE 2 AS EXPLAINED IN THE TEXT
1520 CS=2 : GOTO 1565
1525 IF Y<0 THEN 1530
1527 CS=2: GOTO 1565
1528 REM CS=3 MEANS CASE 3 AS EXPLAINED IN THE TEXT
1530 CS=3: GOTO 1565
1540 IF X>0 THEN 1555
1545 IF Y>0 THEN 1560
1550 CS=3: GOTO 1565
1555 IF Y<0 THEN 1560
1557 CS=3:GOTO 1565
1560 CS=2
1565 PRINT CHR$(19
)
1594 DIM A$(8),V(2000)
1595 X=(INT((X*N)/M))*M/N
1597 Y=(INT((Y*N)/M))*M/N
1600 F=(ABS(X)+ABS(Y))/DT
1601 PRINT F
1602 A$="pqrstuvw"
1603 IF X=0 THEN 1605
1604 C=ABS(Y)/ABS(X):A=0:B=0:L=1
1605 IF X>0 THEN 1620
1610 P=7
1615 GOTO 1625
1620 P=3

```

```

X
1625 IF Y>0 THEN 1640
1630 Q=5
1635 GOTO 1641
1640 Q=1
1641 IF Y=0 THEN 1645
1642 IF X=0 THEN 1650
1643 IF CS=3 THEN 1680
1644 REM          C A S E          2
1645 V(L)=P:A=A+DT
1646 F=INT(F)-1
1648 PRINT F
1649 IF F=0 THEN 1665
1650 L=L+1
1652 B=B+DT
1654 REM SL=SLOPE
1655 SL=B/A
1658 B=B-DT
1660 IF SL>C THEN 1625
1661 V(L)=Q:B=B+DT:F=INT(F)-1
1663 PRINT F
1664 IF F>0 THEN 1650
1665 N2=L
1666 PRINT N2
1668 REM CALL THE SUBROUTINE FOR PRINTING
1670 GOSUB 1295
1675 STOP
1678 REM          C A S E          3
1680 V(L)=Q:B=B+DT
1684 F=INT(F)-1
1688 PRINT F
1692 IF F=0 THEN 1720
1695 L=L+1
1698 A=A+DT
1700 SL=B/A
1703 A=A-DT
1705 IF SL<C THEN 1625
1706 V(L)=P:A=A+DT:F=INT(F)-1
1707 PRINT F
1709 IF F>0 THEN 1695
1720 N2=L
1721 PRINT N2
1725 GOSUB 1295
1730 STOP

```

LINE SUBROUTINE FOR LINES IN SPACE

```

1985      REM LINE SUBROUTINE FOR LINES IN A PLANE AND SPACE
1990 PRINT "INPUT IS X,Y,Z,N,M,DT,N3
1991 REM N=1000,M=1000DT,DT=STEP SIZE,N3=DT/.005
1992 INPUT X,Y,Z,N,M,DT,N3
1993 REM MAKE X,Y AND Z THE NEAREST INTEGER MULTIPLES OF DT
1994 X=(INT((X*N)/M))*M/N
1995 Y=(INT((Y*N)/M))*M/N
1997 Z=(INT((Z*N)/M))*M/N
1998 REM CALCULATE THE NUMBER OF STEPS, F.
1999 F=(ABS(X)+ABS(Y)+ABS(Z))/DT
2000 K=1
2002 DIM V(2000),A$(8)
2008 A$="pqrstuvwxyz"
2010 IF X>0 THEN 2015
2012 R=7
2013 GOTO 2016
2015 R=3
2016 IF Y>0 THEN 2021
2018 P=5
2019 GOTO 2023
2021 P=1
2023 IF Z>0 THEN 2028
2025 Q=6
2026 GOTO 2030
2028 Q=2
2030 A=ABS(X):B=ABS(Y):C=ABS(Z)
2032 X=0:Y=0:Z=0
2035 IF A=0 THEN 2055
2036 REM INCREASE X BY DT
2038 X=X+DT
2039 REM CALL SUBROUTINE TO CALCULATE THE DEVIATION
2040 GOSUB 2250
2045 D1=DE
2050 X=X-DT
2052 GOTO 2058
2055 D1=1000
2058 IF B=0 THEN 2076
2059 REM INCREASE Y BY DT
2060 Y=Y+DT
2062 REM CALL SUBROUTINE TO CALCULATE THE DEVIATION
2063 GOSUB 2250
2065 D2=DE
2066 PRINT D2
2072 Y=Y-DT
2074 GOTO 2078
2076 D2=1000
2078 IF C=0 THEN 2095
2079 REM INCREASE Z BY DT
2080 Z=Z+DT
2082 REM CALL SUBROUTINE TO CALCULATE THE DEVIATION
2083 GOSUB 2255

```

```

2086   D3=DE
2090  Z=Z-DT
2092  GOTO 2098
2095  D3=1000
2096  REM FIND THE LEAST DEVIATION
2098  IF D1<D2 THEN 2120
2100  IF D2<D3 THEN 2135
2105  V(K)=Q
2106  PRINT V(K)
2107  Z=Z+DT:F=F-1
2109  IF F=0 THEN 2200
2110  K=K+1
2112  GOTO 2035
2120  IF D1<D3 THEN 2125
2122  GOTO 2105
2125  V(K)=R
2126  PRINT V(K)
2127  X=X+DT:F=F-1
2130  IF F=0 THEN 2200
2132  K=K+1
2133  GOTO 2035
2135  V(K)=P
2136  PRINT V(K)
2137  Y=Y+DT:F=F-1
2140  IF F=0 THEN 2200
2145  K=K+1
2147  GOTO 2035
2200  GOSUB 2375
2205  STOP
2245  REM SUBROUTINE TO CALCULATE THE DEVIATION
2250  OM=(A*X+B*Y+C*Z)^2/(A^2+B^2+C^2)
2255  OP=(X^2+Y^2+Z^2)
2260  DE=OP-OM
2265  RETURN
2375  REM THIS IS A SUBROUTINE FOR PRINTING
2378  PRINT CHR$(17)
2382  PRINT "z"
2385  FOR J=1 TO K
2388  FOR J1=1 TO N3
2393  PRINT MID$(A$,V(J),1)
2396  NEXT J1
2398  NEXT J
2400  RETURN
READY

```

APPENDIX 2

CIRCLE

```
35 REM      THIS PROGRAM IS FOR DRAWING AN ARC OF A CIRCLE
36 REM P=STARTING ANGLE OF AN ARC IN RADIANS MADE WITH +X-AXIS
37 REM R=RADIUS OF A CIRCLE
38 REM DT=STEP SIZE
39 REM LMT=FINAL ANGLE OF AN ARC IN RADIANS MADE WITH +X-AXIS
40 REM EPS=INCREMENTAL ANGLE IN RADIANS
41 REM N,M ARE USED TO MAKE X,Y THE INTEGER MULTIPLES OF DT
42 REM THE VALUES OF N AND M ARE USED IN THE LINE SUBROUTINE
45 PRINT "INPUT IS P,R,N,M,DT,LMT,EPS"
47 DIM A$(8),V(4000),X1(1000),Y1(1000)
50 INPUT P,R,N,M,DT,LMT,EPS
52 REM PARAMETRIC REPRESENTATION OF A CIRCLE
55 X1(1)=R*COS(P):Y1(1)=R*SIN(P)
60 L=2
62 K=1
65 P=P+EPS
66 PRINT P
67 IF P>LMT THEN 120
70 X1(L)=R*COS(P):Y1(L)=R*SIN(P)
75 X=X1(L)-X1(L-1):Y=Y1(L)-Y1(L-1)
77 PRINT X,Y
78 REM STORE THE VALUES OF X AND Y
80 XN=X:YN=Y
82 REM CALL THE LINE SUBROUTINE
85 GOSUB 990
86 REM ADJUST THE VALUES OF X1(L) AND Y1(L)
90 X1(L)=X1(L)-(XN-X)
91 Y1(L)=Y1(L)-(YN-Y)
95 L=L+1
100 GOTO 65
110 REM CALL THE SUBROUTINE FOR PRINTING
120 GOSUB 1295
130 STOP
```

PARABOLA

```

160 REM THIS PROGRAM DRAWS PARABOLA IN A PLANE
162 REM A=DISTANCE FROM FOCUS TO VERTEX OF PARABOLA
164 REM AG=MAXIMUM ANGLE IN DEGREES
166 REM QU=QUADRANT IN WHICH PARABOLA IS REQUIRED
168 REM N=NUMBER OF POINTS ON PARABOLA
170 REM X( )=ARRAY CONTAINING X-COORDINATES OF POINTS ON PARABOL
A
172 REM Y( )=ARRAY CONTAINING Y-COORDINATES OF POINTS ON PARABOL
A
175 PRINT "INPUT IS N,M,DT,,A,AG,N1,QU,N3"
180 INPUT N,M,DT,A,AG,N1,QU,N3
181 DIM A$(8),V(3000),X1(1000),Y1(1000)
182 IF QU=1 THEN 189
183 IF QU=2 THEN 188
184 IF QU=3 THEN 187
185 REM INPUT STRING CHARACTERS DEPENDING UPON THE QUADRANT
186 A$="tsrqpuvw" :GOTO 190
187 A$="tuvspqrw" :GOTO 190
188 A$="pwvsturq" :GOTO 190
189 A$="pqrstuvw"
190 REM STRING CHARACTERS WILL NOT BE USED IN LINE SUBROUTINE
191 AG1=AG/((N1-1)*57.2957795)
192 REM CALCULATE THE INCREMENTAL ANGLE
193 K=1
195 X1(1)=0:Y1(1)=0
198 REM PARAMETRIC REPRESENTATION OF PARABOLA
200 A1=A*AG1*AG1
205 B1=2*A*AG1
210 L=2
215 X1(L)=A1+X1(L-1)+AG1*Y1(L-1)
220 Y1(L)=B1+Y1(L-1)
225 X=X1(L)-X1(L-1):Y=Y1(L)-Y1(L-1)
226 PRINT X,Y
227 XN=X:YN=Y:REM STORE THE VALUES OF X AND Y
230 REM CALL THE LINE SUBROUTINE
231 GOSUB 990
233 REM ADJUST THE END POINTS OF THE LINE SEGMENT
235 X1(L)=X1(L)-(XN-X):Y1(L)=Y1(L)-(YN-Y)
240 L=L+1
242 IF L<=N1 THEN 215
244 REM CALL THE SUBROUTINE FOR PRINTING
245 GOSUB 1295
250 STOP

```

ELLIPSE

```

60      REM THIS PROGRAM DRAWS AN ELLIPSE IN AN PLANE
65 REM A1=LENGTH OF SEMI-MAJOR AXIS
70 REM B1=LENGTH OF SEMI-MINOR AXIS
75 REM L=INCLINATION OF MAJOR AXIS IN DEGREES
80 REM N1=NUMBER OF POINTS ON ELLIPSE
85 REM X( )=ARRAY CONTAINING X-COORDINATES OF POINTS ON ELLIPSE
90 REM Y( )=ARRAY CONTAINING Y-COORDINATES OF POINTS ON ELLIPSE
95 PRINT "INPUT IS N,M,DT,A1,B1,L,N1"
100 INPUT N,M,DT,A1,B1,L,N1
101 REM THE VALUES OF N AND M ARE USED IN THE LINE SUBROUTINE
102 DIM A$(8),V(4000),X1(1000),Y1(1000)
103 K=1
104 REM CALCULATE THE INCREMENTAL ANGLE
105 THT=2*3.14156/(N1-1)
110 L1=L/57.2957795
115 C1=COS(L1)
120 S1=SIN(L1)
125 C2=COS(THT)
130 S2=SIN(THT)
135 C3=1
140 S3=0
145 FOR J=1 TO N1
150 XE=A1*C3
155 YE=B1*S3
160 X1(J)=XE*C1-YE*S1
165 Y1(J)=XE*S1+YE*C1
170 Q=C3*C2-S3*S2
175 S3=S3*C2+C3*S2
180 C3=Q
185 NEXT J
190 J1=2
195 X=X1(J1)-X1(J1-1):Y=Y1(J1)-Y1(J1-1)
198 REM STORE THE VALUES OF X AND Y
200 XN=X:YN=Y
203 REM CALL THE LINE SUBROUTINE
205 GOSUB 990
207 REM ADJUST THE VALUES OF X1(J1),Y1(J1)
210 X1(J1)=X1(J1)-(XN-X):Y1(J1)=Y1(J1)-(YN-Y)
212 PRINT J1
215 J1=J1+1
217 IF J1<=N1 THEN 195
218 REM CALL THE SUBROUTINE FOR PRINTING
220 GOSUB 1295
225 STOP

```

HYPERBOLA

```

10 REM THIS PROGRAM DRAWS HYPERBOLA IN A PLANE
15 REM A3=DISTANCE FROM CENTER OF HYPERBOLA TO VERTEX
16 REM B3=ITS VALUE DETERMINES SLOPE OF ASYMTOTES=+-B3/A3
18 REM N1=NUMBER OF POINTS ON HYPERBOLA
20 REM DT=STEP SIZE
22 REM N3=DT/.005
23 REM N,M ARE USED TO MAKE X,Y INTEGER MULTIPLES OF DT
25 PRINT "INPUT IS N,M,DT,A3,B3,N1,N3"
30 INPUT N,M,DT,A3,B3,N1,N3
31 K=1
32 DIM A$(8),V(2000),XL(200),YL(200)
33 REM CALCULATE PARAMETER INCREMENT
35 TH=3.1415592654/(2*(N1-1))
38 REM CALCULATE COS,SIN,TAN OF PARAMETER INCREMENT
45 C2=COS(TH):S2=SIN(TH):T2=TAN(TH)
46 REM INITIALIZE
50 Q3=B3*T2:R3=B3*C2
55 XL(1)=A3:YL(1)=0
60 FOR U=2 TO N1
65 XL(U)=(B3*XL(U-1))/(R3-YL(U-1)*S2)
70 YL(U)=(B3*(YL(U-1)+Q3))/(B3-YL(U-1)*T2)
75 X=XL(U)-XL(U-1):Y=YL(U)-YL(U-1)
77 PRINT X,Y
80 XN=X:YN=Y
85 GOSUB 990
90 XL(U)=XL(U)-(XN-X):YL(U)=YL(U)-(YN-Y)
95 NEXT U
100 N2=K
105 GOSUB 1295
110 STOP

```

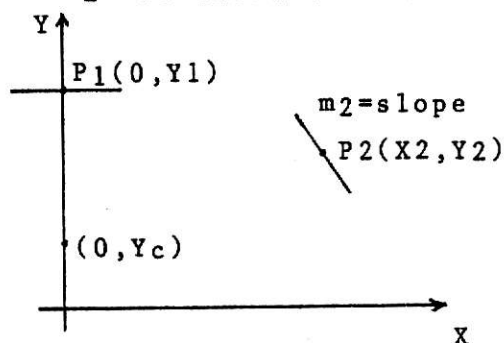
APPENDIX 3

DERIVATION OF INPUT SLOPE LIMIT FOR
AN ELLIPSE IN STANDARD POSITION WITH
CENTER ON Y-AXIS

The equation of an ellipse in standard position with its center on the Y-axis is given by

$$\frac{x^2}{a^2} + \frac{(Y-Y_c)^2}{b^2} = 1 \quad \text{.....A1}$$

We need to find the equation of an ellipse that produces an arc having an horizontal slope at P₁ and a slope m₂ to some location at P₂* as shown below:



If m₂ were ∞ then we would have an elliptic quadrant from P₁ to P₂. In this case (as shown in the text) a=X₂, Y_c=Y₂, and b=(Y₁-Y_c). If m₂ were positive then we would have more than a quadrant. In view of this we will assume $0 \geq m_2 \geq -\infty$ and proceed to find the lower limit of m₂. From Equation A1, the derivative is :

$$Y' = - \frac{b^2 X}{a^2 (Y-Y_c)} \quad \text{.....A2}$$

The ellipse passes through P₁(0, Y₁). Hence from Equation A1, we have

$$(Y_1 - Y_c)^2 = b^2 \quad \text{.....A3}$$

* Y₂ < Y₁

The ellipse also passes through $P_2(X_2, Y_2)$. Hence from Equation A1, we have

$$\frac{X_2^2}{a^2} + \frac{(Y_2 - Y_c)^2}{b^2} = 1 \quad \text{.....A4}$$

From Equation A2, we have

$$m_2 = \frac{-b^2 X_2}{a^2 (Y_2 - Y_c)} \quad \text{.....A5}$$

From Equation A4, we have

$$\frac{X_2^2}{a^2} = \frac{b^2 - (Y_2 - Y_c)^2}{b^2} \quad \text{.....A6}$$

By substituting b^2 in terms of Y from Equation A3 in Equation A6, we have,

$$\frac{X_2^2}{a^2} = \frac{(Y_1 - Y_c)^2 - (Y_2 - Y_c)^2}{b^2}$$

i.e.

$$X_2^2 = \frac{a^2}{b^2} (Y_1 - Y_2)(Y_1 + Y_2 - 2Y_c)$$

or

$$\frac{b^2}{a^2} = \frac{(Y_1 - Y_2)(Y_1 + Y_2 - 2Y_c)}{X_2^2} \quad \text{.....A7}$$

By substituting the value of b^2/a^2 given by Equation A7 into Equation A5, and after simplification, we get

$$m_2 = - \frac{(Y_1 - Y_2)(Y_1 + Y_2 - 2Y_c)}{X_2^2} \frac{X_2}{(Y_2 - Y_c)}$$

or

$$\frac{(Y_2 - Y_1)}{X_2} \frac{(Y_2 + Y_1 - 2Y_c)}{(Y_2 - Y_c)} \quad \text{.....A8}$$

The limiting value of m_2 can be found by taking the limit of m_2 as Y_c approaches negative infinity. This is given by

$$\lim_{Y_c \rightarrow -\infty} m_2 = \frac{2(Y_2 - Y_1)}{X_2}$$

DERIVATION OF THE EQUATION OF AN ELLIPSE
IN STANDARD POSITION WITH CENTER ON THE Y-AXIS

The equation of an ellipse in standard position is given by

$$\frac{x^2}{a^2} + \frac{(y-y_c)^2}{b^2} = 1 \text{ (Center on Y-axis) } \dots\dots A1$$

The equation of the derivative is given by

$$y' = -\frac{b^2 x}{a^2 (y-y_c)} \dots\dots A2$$

By substituting the coordinates of the two points in Equation A1, we get

$$\frac{(y_1-y_c)^2}{b^2} = 1 \dots\dots A9$$

and

$$\frac{x_2^2}{a^2} + \frac{(y_2-y_c)^2}{b^2} = 1 \dots\dots A10$$

Solving Equation A2 for b^2 , and denoting y' by m , we get

$$b^2 = \frac{a^2 m (y_c - y_2)}{x_2} \dots\dots A11$$

Now, substituting the value of b^2 given by Equation A11 into Equations A9 and A10, we get

$$\frac{x_2 (y_1 - y_c)^2}{a^2 m (y_c - y_2)} = 1 \dots\dots A12$$

and

$$\frac{x_2^2}{a^2} + \frac{x_2 (y_2 - y_c)^2}{a^2 m (y_c - y_2)} = 1 \dots\dots A13$$

Since the left sides of Equations A12 and A13 equal unity, these two expressions can be equated. By doing so, we get

$$\frac{X_2(Y_1 - Y_c)^2}{a^2 m(Y_c - Y_2)} = \frac{X_2^2}{a^2} + \frac{X_2(Y_2 - Y_c)^2}{a^2 m(Y_c - Y_2)}$$

Dropping out a^2 , which is one of the unknowns, and solving for Y_c , we get

$$Y_c = \frac{Y_2^2 - Y_1^2 - mX_2Y_2}{2Y_2 - 2Y_1 - mX_2} \quad \dots\dots A14$$

From Equation A12 we get

$$a^2 = \frac{X_2(Y_1 - Y_c)^2}{m(Y_c - Y_2)} \quad \dots\dots A15$$

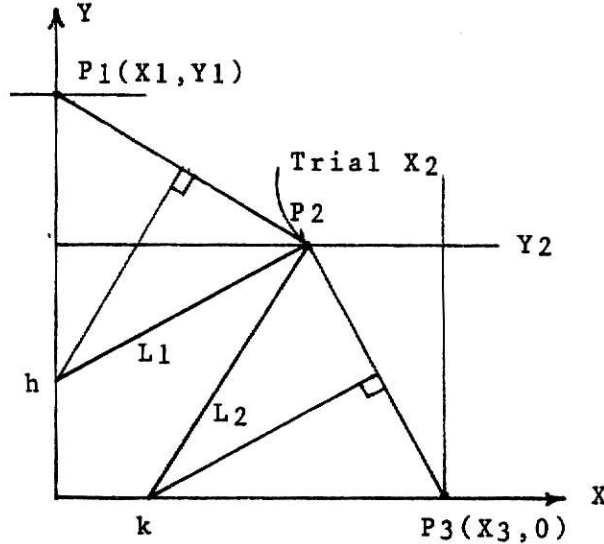
Equation A11 gives an expression for b^2 , namely;

$$b^2 = \frac{a^2 m(Y_c - Y_2)}{X_2}$$

Thus given the points $(0, Y_1)$ and (X_2, Y_2) , we can find an ellipse that goes through both points if the required slope at (X_2, Y_2) is less than $2(Y_2 - Y_1)/X_2$.

DERIVATION OF SLOPES m_1 AND m_2
FOR
CIRCULAR ARC BLENDING

As discussed in the text, consider the geometry as depicted below :



The slope of the line P_1P_2 is $(Y_2 - Y_1)/(X_2 - X_1)$. The slope of the perpendicular bisector of the line P_1P_2 is $-(X_2 - X_1)/(Y_2 - Y_1)$. The perpendicular bisector of the line P_1P_2 passes through the point whose coordinates are $((X_1 + X_2)/2, (Y_1 + Y_2)/2)$, and has a slope $-(X_2 - X_1)/(Y_2 - Y_1)$. Hence the intercept made by this line on the Y-axis is given by

$$\begin{aligned} \text{Intercept(Y-axis)} &= (Y_1 + Y_2)/2 + \frac{(X_2 - X_1)^2}{2(Y_2 - Y_1)} \\ &= \frac{(Y_2 - Y_1)^2 + (X_2 - X_1)^2}{2(Y_2 - Y_1)} \end{aligned}$$

The slope of the line L_1 denoted by m_1 can now be determined and is given by

$$m_1 = \frac{Y_2 - \frac{(Y_2 - Y_1)^2 + (X_2 - X_1)^2}{2(Y_2 - Y_1)}}{X_2}$$

Simplifying and putting $X_1 = 0$, we get

$$m_1 = \frac{Y_2}{X_2} + \frac{X_2^2 - Y_1^2 + Y_2^2}{2X_2(Y_1 - Y_2)}$$

The slope of the line line P2P3 is $(Y_2)/(X_2 - X_3)$. The slope of the perpendicular bisector of the line P2P3 is $-(X_2 - X_3)/Y_2$. The perpendicular bisector of the line P2P3 passes through $((X_2 + X_3)/2, Y_2/2)$, and has a slope $-(X_2 - X_3)/Y_2$. Hence the intercept made by this line on the Y-axis is given by

$$\begin{aligned} \text{Intercept(Y-axis)} &= Y_2/2 + \frac{(X_2 - X_3)}{2Y_2} \\ &= \frac{Y_2^2 + (X_2 - X_3)^2}{2Y_2} \end{aligned}$$

The intercept on the X-axis is given by

$$\begin{aligned} \text{Intercept(X-axis)} &= \frac{\text{Intercept(Y-axis)}}{(X_2 - X_3)/Y_2} \\ &= \frac{Y_2^2 + (X_2 - X_3)^2}{2(X_2 - X_3)} \end{aligned}$$

The slope of the line L₂ denoted by m₂ can now be determined and is given by

$$m_2 = \frac{-Y_2}{\frac{Y_2^2 + (X_2 - X_3)^2}{2(X_2 - X_3)} - X_2}$$

Simplifying, we get

$$m_2 = \frac{2Y_2(X_2 - X_3)}{(X_2 - X_3)^2 - Y_2^2}$$

CIRCULAR ARC BLENDING

```
4 REM      THIS PROGRAM IS FOR BLENDING OF TWO CIRCULAR ARCS
5 PRINT "INPUT IS X1,Y1,Y2,X3,Y3,N1"
6 REM COORDINATES OF P1 ARE (X1,Y1)
7 REM COORDINATES OF P2 ARE (X2,Y2)
8 REM COORDINATES OF P3 ARE (X3,Y3)
9 REM N1=NUMBER OF PARTS INTO WHICH THE INTERVAL IS DIVIDED
10 INPUT X1,Y1,Y2,X3,Y3,N1
15  $XT = 4 * X3 * (Y2 - Y1) / (3 * Y2 - 4 * Y1)$ 
17 PRINT XT
18 IF  $XT \geq X3$  THEN 65
20  $DX = (X3 - XT) / N1$ 
21 PRINT DX
22 IF  $DX < .0001$  THEN 70
24  $M1 = Y2 / XT + (XT^2 - Y1^2 + Y2^2) / (2 * XT * (Y1 - Y2))$ 
25  $M2 = 2 * Y2 * (XT - X3) / ((XT - X3)^2 - Y2^2)$ 
26 IF  $(M1 - M2) > 0$  THEN 50
27  $XT = XT + DX$ 
28 GOTO 24
50 PRINT XT,(XT-DX)
55  $XT = XT - DX : DX = DX / N1$ 
60 GOTO 21
65 PRINT "SOLUTION DOES NOT EXIST"
70 STOP
READY
```

PARABOLIC BLENDING

```

1 REM THIS PROGRAM IS FOR PARABOLIC BLENDING
2 REM N=1000,M=1000DT,DT=STEP SIZE
3 REM N1=NUMBER OF POINTS ON BLENDED CURVE
4 REM R(1, )=ARRAY CONTAINING X-COORDINATES OF FOUR POINTS
5 REM R(2, )=ARRAY CONTAINING Y-COORDINATES OF FOUR POINTS
6 REM R(3, )=ARRAY CONTAINING Z-COORDINATES OF FOUR POINTS
12 PRINT "INPUT IS N,M,DT,N1"
14 INPUT N,M,DT,N1
15 DIM A$(8),V(2000)
16 DIM R1(3,4),T(3,4),S(3,3),XL(3,40),XQ(3,40),XP(3,40),T1(3)
17 DIM S1(3),M1(3)
20 R1(1,1)=0:R1(1,2)=1:R1(1,3)=3:R1(1,4)=4
21 R1(2,1)=0:R1(2,2)=1:R1(2,3)=2:R1(2,4)=3
22 R1(3,1)=0:R1(3,2)=0:R1(3,3)=0:R1(3,4)=0
32 NP=(R1(1,3)-R1(1,2))^2+(R1(2,3)-R1(2,2))^2
33 NQ=(R1(3,3)-R1(3,2))^2
34 PRINT NP,NQ
35 T0=SQR(NP+NQ)
36 FOR J1=1 TO 3 :REM SET UP FOR FIRST PARABOLA
37 FOR I1=1 TO 3 :REM SET UP FOR SECOND PARABOLA
39 S(I1,J1)=R1(I1,J1)
41 T(I1,J1)=R1(I1,J1+1)
42 NEXT I1
43 NEXT J1
44 H2=1
45 GOSUB 80 :REM GENERATE PARABOLA THROUGH FIRST THREE POINTS
46 H2=2
47 GOSUB 80 :REM GENERATE PARABOLA THROUGH LAST THREE POINTS
50 KS=0
53 FOR E1=0 TO 1 STEP(1/(N1-1))
54 PRINT E1
55 KS=KS+1 :REM GENERATE BLENDED CURVE
57 FOR KA=1 TO 3
59 XL(KA,KS)=(1-E1)*XQ(KA,KS)+E1*XP(KA,KS)
60 PRINT XL(KA,KS)
61 NEXT KA
65 NEXT E1
66 K=1
67 FOR J1=2 TO KS
68 X=XL(1,J1)-XL(1,J1-1):Y=XL(2,J1)-XL(2,J1-1)
69 Z=XL(3,J1)-XL(3,J1-1)
71 XN=X:YN=Y:ZN=Z
72 REM CALL THE LINE SUBROUTINE
73 GOSUB 1994
74 XL(1,J1)=XL(1,J1)-(XN-X):XL(2,J1)=XL(2,J1)-(YN-Y)
75 XL(3,J1)=XL(3,J1)-(ZN-Z)
77 NEXT J1
78 GOSUB 1295 :REM CALL THE SUBROUTINE FOR PRINTING
79 STOP
80 IF H2=1 THEN 100 :REM H2=1 INDICATES FIRST PARABOL

```

```

81 REM H2=2    INDICATES SECOND PARABOLA
85 FOR I1=1 TO 3
88 T1(I1)=T(I1,2)-T(I1,1)
90 S1(I1)=T(I1,3)-T(I1,1)
94 M1(I1)=T(I1,3)-T(I1,2)
95 PRINT S1(I1)
96 NEXT I1
97 GOTO 116
100 FOR I1=1 TO 3
105 T1(I1)=S(I1,2)-S(I1,1)
110 S1(I1)=S(I1,3)-S(I1,1)
112 M1(I1)=S(I1,3)-S(I1,2)
113 PRINT S1(I1)
114 NEXT I1
116 U2=0
118 FOR I2=1 TO 3
120 U=S1(I2)*S1(I2)
122 U=U+U2
123 U2=U
124 PRINT U
125 NEXT I2
127 DS=SQR(U)
129 V2=0
131 FOR I2=1 TO 3
133 V=T1(I2)*S1(I2)
135 V=V+V2
137 V2=V
139 NEXT I2
141 XA=V/U
143 AC=1/(U*XA*(1-XA))
145 IF H2=2 THEN 158
147 W2=0
149 FOR I2=1 TO 3
151 W1=M1(I2)*S1(I2)
152 W1=W1+W2
154 W2=W1
155 NEXT I2
156 TC=W1/(T0*DS) :REM CALCULATE COS(THETA)
157 GOTO 162
158 TC=V/(T0*DS) :REM CALCULATE COS(THETA)
162 KS=0
163 REM CALCULATE POINTS ON PARABOLA
165 FOR E2=0 TO 1 STEP(1/(N1-1))
167 KS=KS+1
169 TM=T0*E2
171 RM=TM*TC
172 REM TEST FOR FIRST OR SECOND PARABOLA
173 IF H2 =2 THEN 188
175 RM=RM+XA*DS
177 FOR J=1 TO 3
181 XQ(J,KS)=S(J,1)+(RM/DS)*S1(J)+AC*RM*(DS-RM)*(T1(J)-XA*S1(J)

```

X

183 NEXT J

185 NEXT E2

186 RETURN

188 FOR J=1 TO 3

192 $XP(J,KS) = T(J,1) + (RM/DS) * S1(J) + AC * RM * (DS - RM) * (T1(J) - XA * S1(J))$

194 NEXT J

196 NEXT E2

198 RETURN

APPENDIX 4

EQUATION OF A CURVE : $Y=F(X)$

```

X
60 REM THIS PROGRAM IS FOR AN EQUATION OF THE FORM Y=F(X)
61 REM DT=STEP SIZE
62 REM N,M ARE USED TO MAKE X,Y INTEGER MULTIPLES OF DT
63 REM XM=LOWER RANGE OF X VALUE
64 REM DX=LEAST INCREMENT
65 REM XX=HIGHER RANGE OF X VALUE
67 PRINT "N,M,DT,XM,DX,XX,N3"
70 INPUT N,M,DT,XM,DX,XX,N3
72 DIM A$(8),V(3000)
75 X1=XM
77 K=1
78 REM CALL THE FUNCTION SUBROUTINE
80 GOSUB 200
85 YL=FT
90 XL=X1
92 PRINT XL
95 XK=XL+DX
100 IF XK>XX THEN 150
105 X1=XK
107 PRINT XK
108 REM CALL THE FUNCTION SUBROUTINE
110 GOSUB 200
112 YK=FT
115 X=XK-XL:Y=YK-YL
116 REM STORE THE VALUES OF X AND Y
120 XN=X:YN=Y
122 REM CALL THE LINE SUBROUTINE
125 GOSUB 990
128 REM ADJUST THE VALUES OF XK,YK
130 XK=XK-(XN-X):YK=YK-(YN-Y)
135 XL=XK:YL=YK
140 GOTO 95
145 CALL THE SUBROUTINE FOR PRINTING
150 GOSUB 1295
160 STOP
195 REM THIS IS A FUNCTION SUBROUTINE
200 FT =X1*X1*X1+X1*X1+2*X1+2
205 RETURN

```

EQUATION OF A CURVE : $X=F(Y)$

```

300      REM THIS PROGRAM IS FOR AN EQUATION OF THE FORM  $X=F(Y)$ 
301 REM DT=STEP SIZE
302 REM N,M ARE USED TO MAKE X,Y INTEGER MULTIPLE INTEGERS OF DT
303 REM YM=LOWER RANGE OF Y VALUE
304 REM DY=LEAST INCREMENT
305 REM YY=HIGHER RANGE OF Y VALUE
310 PRINT "INPUT IS N,M,DT,YM,DY,YY,N3"
315 INPUT N,M,DT,YM,DY,YY,N3
317 DIM A$(8),V(3000)
320 Y1=YM
322 K=1
323 REM CALL THE FUNCTION SUBROUTINE
325 GOSUB 400
327 XL=FT
329 YL=Y1
332 PRINT YL
335 YK=YL+DY
338 IF YK>YY THEN 370
340 Y1=YK
342 PRINT YK
343 REM CALL THE FUNCTION SUBROUTINE
345 GOSUB 400
347 XK=FT
350 X=XK-XL:Y=YK-YL
351 REM STORE THE VALUES OF X AND Y
352 XN=X:YN=Y
353 REM CALL THE LINE SUBROUTINE
354 GOSUB 990
355 REM ADJUST THE VALUES OF XK AND YK
356 XK=XK-(XN-X):YK=YK-(YN-Y)
358 XL=XK:YL=YK
360 GOTO 335
368 REM CALL THE SUBROUTINE FOR PRINTING
370 GOSUB 1295
380 STOP
398 REM THIS IS A FUNCTION SUBROUTINE
400 FT=Y1*Y1:REM FUNCTION  $X=Y^2$  IS CONSIDERED
405 RETURN

```

EQUATION OF ANY CURVE IN A PLANE

```

50 REM THIS PROGRAM IS FOR ANY EQUATION OF A CURVE IN A PLANE
51 REM DT=STEP SIZE
52 REM N,M ARE USED TO MAKE X,Y INTEGER MULTIPLES OF DT
53 REM A1=LOWER/HIGHER RANGE OF X VALUE
54 REM B1=LOWER/HIGHER RANGE OF X VALUE
55 REM NM=INITIAL APPROXIMATE VALUE OF Y
56 REM DX=LEAST INCREMENT VALUE OF X
57 REM N3=DT/.005
60 PRINT "INPUT IS N,M,DT,A1,B1,NM,DX,N3"
62 INPUT N,M,DT,A1,B1,NM,DX,N3
63 DIM A$(8),V(3000),XK(1000),YK(1000)
65 XL=A1
70 YL=NM
72 J=1:K=1
75 GOSUB 110
78 REM CALL SUBROUTINE FOR PRINTING
80 GOSUB 1295
90 STOP
100 REM "NEWTON METHOD IS USED FOR THIS ALGORITHM
102 REM VL IS AN EXPRESSION CONTAINING ONLY X TERMS
104 REM F2 IS AN EXPRESSION CONTAINING ONLY Y ANY XY TERMS
106 REM F3 IS THE FIRST ORDER DERIVATIVE OF F2 WITH RESPECT TO Y
110 VL=-(XL^3)
112 F2=YL*YL+VL
113 F3=2*YL
120 YL=YL-F2/F3
125 IF ABS(F2)<=.00001 THEN 135
130 GOTO 110
135 XK(J)=XL:YK(J)=YL
137 PRINT XL,YL
138 IF A1>B1 THEN 150
140 XL=XL+DX:YL=NM:J=J+1
145 IF XL>=B1 THEN 170
147 GOTO 110
150 XL=XL-DX:YL=NM:J=J+1
155 IF XL<=B1 THEN 170
160 GOTO 110
170 T=J-1
175 FOR J=2 TO T
177 REM CALCULATE X AND Y TO PASS ON TO THE LINE SUBROUTINE
178 X=XK(J)-XK(J-1):Y=YK(J)-YK(J-1)
179 REM STORE THE VALUES OF X AND Y
180 XN=X:YN=Y

```

```
182 REM CALL THE LINE SUBROUTINE
183 GOSUB 990
184 REM ADJUST THE END COORDINATES OF THE LINE SEGMENT
185  $XK(J) = XK(J) - (XN - X)$  :  $YK(J) = YK(J) - (YN - Y)$ 
190 NEXT J
210 RETURN
```

APPENDIX 5

CURVES IN SPACE

```

48      REM THIS PROGRAM IS FOR GENERATING SPACE CURVE
50 REM DT=STEP SIZE
51 REM N,M ARE USED TO MAKE X,Y INTEGER MULTIPLES OF DT
52 REM NM=APPROXIMATE VALUE OF X
53 REM NN=APPROXIMATE VALUE OF Y
54 REM AI=LOWER RANGE OF Z VALUE
55 REM AF=HIGHER RANGE OF Z VALUE
56 REM DZ=LEAST INCREMENT ALONG Z
62 REM "NEWTON RAPHSON METHOD" IS USED
65 PRINT "INPUT IS N,M,DT,NM,NN,AI,AF,DZ,N3"
70 INPUT  N,M,DT,NM,NN,AI,AF,DZ,N3
72 DIM V(3000),A$(8),XK(700),YK(700),ZK(700)
75 ZL=AI:J=1
80 XL=NM:YL=NN
85 K=1
88 REM CALL FUNCTION SUBROUTINE
90 GOSUB 100
91 REM CALL SUBROUTINE FOR PRINTING
92 GOSUB 2375
95 STOP
96 REM E1=EXPRESSION CONTAINING Z AND CONSTANT TERMS OF SURFACE
1  REM E2=EXPRESSION CONTAINING Z AND CONSTANT TERMS OF SURFACE
2
98 REM G=SURFACE 1:REM H=SURFACE 2
99 REM GX=DG/DX,GY=DG/DY:REM HX=DH/DX,HY=DH/DY
100 E1=ZL*ZL-5
105 E2=ZL-1
110 G=XL*XL+YL*YL+E1
115 H=XL+YL+E2
120 GX=2*XL:GY=2*YL
125 HX=1:HY=1
130 IF ABS(G)>.00001 THEN 150
135 IF ABS(H)>.00001 THEN 150
140 GOTO 177
145 REM COMPUTATION FOR FINDING INVERSE OF A 2*2 MATRIX
150 CT=GX*HY-HX*GY
155 XR=-(HY*G-GY*H)/CT
160 YR=-((-HX*G)+GX*H)/CT
165 XL=XR+XL
170 YL=YR+YL
175 GOTO 110
177 PRINT XL,YL,ZL
180 XK(J)=XL:YK(J)=YL:ZK(J)=ZL
200 IF AI>AF THEN 220

```

```
205     ZL=ZL+DZ:J=J+1
210 IF ZL>AF THEN 240
215 GOTO 100
220 ZL=ZL-DZ:J=J+1
225 IF ZL<AF THEN 240
230 GOTO 100
240 NC=J-1
241 PRINT NC
245 FOR J=2 TO NC
247 REM CALCULATE X AND Y TO PASS ON TO THE LINE SUBROUTINE
248 X=XK(J)-XK(J-1):Y=YK(J)-YK(J-1):Z=ZK(J)-ZK(J-1)
249 REM STORE THE VALUES OF X AND Y
250 XN=X:YN=Y:ZN=Z
252 REM CALL THE LINE SUBROUTINE (SPACE LINES)
255 GOSUB 1994
260 XK(J)=XK(J)-(XN-X):YK(J)=YK(J)-(YN-Y):ZK(J)=ZK(J)-(ZN-Z)
265 NEXT J
270 RETURN
```

BEZIER CURVE FOR 3-D OR 2-D

```

20 REM BEZIER CURVE ALGORITHM FOR 2-D OR 3-D CURVE
25 REM DT=STEP SIZE
26 REM N,M ARE USED TO MAKE X,Y,Z INTEGER MULTIPLES OF DT
27 REM DI=2 FOR 2-D,DI=3 FOR 3-D
28 REM NN=NUMBER OF VERTICES IN BEZIER POLYGON
29 REM NB=NUMBER OF POINTS ALONG BEZIER CURVE
30 REM X2( ,1)=ARRAY CONTAINING X COMPONENT OF POLYGON VERTICES
31 REM Y2( ,1)=ARRAY CONTAINING Y COMPONENT OF POLYGON VERTICES
32 REM Z2( ,1)=ARRAY CONTAINING Z COMPONENT OF POLYGON VERTICES
33 REM X1(1, )=X-COMPONENT OF POINTS ALONG BEZIER CURVE
34 REM Y1(1, )=Y-COMPONENT OF POINTS ALONG BEZIER CURVE
35 REM Z1(1, )=Z-COMPONENT OF POINTS ALONG BEZIER CURVE
36 REM N3=DT/.005
38 PRINT "INPUT IS N,M,DT,DI,NN,NB,N3"
40 INPUT N,M,DT,DI,NN,NB,N3
45 DIM A$(8),V(2000),X1(1,100),Y1(2,100),Z1(3,100)
46 DIM E(1,50),X2(50,1),Y2(50,1),Z2(50,1)
47 DIM FT(20)
62 K=1
63 A$="pqrstuvwxyz"
65 REM INPUT CO-ORDINATES OF POLYGON VERTICES
70 X2(1,1)=1:X2(2,1)=2:X2(3,1)=4:X2(4,1)=3
75 Y2(1,1)=1:Y2(2,1)=3:Y2(3,1)=3:Y2(4,1)=1
105 NM=NN-1
107 REM BINOMIAL EXPANSION
110 P=0:I=0
115 IF P=0 THEN 145
120 S=1
125 S=S*P
130 P=P-1
135 IF P=0 THEN 155
140 GOTO 125
145 FT(I)=1
150 GOTO 160
155 FT(I)=S
160 PRINT FT(I)
162 I=I+1:P=I
163 PRINT P
164 IF P<=NN THEN 120
165 KT=1
168 REM GENERATE BASIS FUNCTION
170 FOR CT=0 TO 1 STEP 1/(NB-1)
175 FOR L=0 TO NM
180 E(1,L+1)=(FT(NM)/(FT(L)*FT(NM-L)))*CT^L*(1-CT)^(NM-L)
182 NEXT L
184 REM GENERATE POINTS ALONG A 2-D OR 3-D BEZIER CURVE
186 F2=0
187 FOR I=1 TO NN

```



```

188  F1=E(1,I)*X2(I,1)
189  F1=F1+F2
190  F2=F1
191  NEXT I
192  G2=0
193  FOR I=1 TO NN
194  G1=E(1,I)*Y2(I,1)
195  G1=G1+G2
196  G2=G1
197  NEXT I
200  X1(1,KT)=F2
205  Y1(2,KT)=G2
210  IF DI=2 THEN 225
211  H2=0
212  FOR I=1 TO NN
213  H1=E(1,I)*Z2(I,1)
214  H1=H1+H2
215  H2=H1
216  NEXT I
220  Z1(3,KT)=H2
225  KT=KT+1
226  PRINT F2,G2,H2
230  NEXT CT
235  NS=KT-1
238  REM CALCULATE X,Y Z TO PASS ON TO THE LINE SUBROUTINE
240  FOR KT=2 TO NS
245  X=X1(1,KT)-X1(1,KT-1)
247  Y=Y1(2,KT)-Y1(2,KT-1)
250  IF DI=2 THEN 254
252  Z=Z1(3,KT)-Z1(3,KT-1)
253  REM STORE THE VALUES OF X,Y,Z
254  XN=X:YN=Y:ZN=Z
255  REM CALL THE LINE SUBROUTINE (LINES IN SPACE OR IN A PLANE)
256  GOSUB 1994
258  REM ADJUST THE END POINT COORDINATES OF THE LINE SEGMENT
260  X1(1,KT)=X1(1,KT)-(XN-X):Y1(2,KT)=Y1(2,KT)-(YN-Y)
262  IF DI=2 THEN 270
265  Z1(3,KT)=Z1(3,KT)-(ZN-Z)
270  NEXT KT
280  REM CALL THE SUBROUTINE FOR PRINTING
290  GOSUB 2375
295  STOP

```

References

1. R. S. Pressman, and J. E. Williams, Numerical Control and Computer Aided Manufacturing, John Wiley & Sons, N.Y., 1977.
2. P. Bézier, Numerical Control Mathematics and Applications, John Wiley & Sons, N.Y., 1970.
3. D. F. Rogers, and J. A. Adams, Mathematical Elements for Computer Graphics, McGraw-Hill, N.Y., 1976.
4. N. O. Olesten, Numerical Control, Wiley-Interscience, 1970.
5. E. D. Smith, M. Salkover and H. K. Justice, Calculus, John Wiley and Sons, N.Y., 1946.
6. F. Scheid, Theory and Problems in Numerical Analysis, McGraw Hill, N.Y., 1968.
7. W. K. Giloi, Interactive Computer Graphics, Prentice-Hall, Inc, N.J., 1978.
8. I. C. Braid, Designing with Volumes, Computer Aided Design Group, University of Cambridge, 1973.
9. D. McPherson, Advances in Computer Aided Manufacture, North Holland Publishing Co, 1977.
- 10 S. H. Chasen, Geometric Principles and Procedures for Computer Graphic Applications, Prentice-Hall, N.J., 1978.
11. Sorcerer Technical Manual, The Sorcerer of the Exidy Inc, California, 1979.
12. FANUC Tape Drill-Model C, Operator Manual, 1977.
13. L. Leithhold, The Calculus with Analytic Geometry, Harper & Row, N.Y., 1976.

GENERATING THREE DIMENSIONAL CUTTER PATHS
FOR
AN XY OR XZ CONTOUR MILLING MACHINE

by

ASHOK N. KABADI

B.E., University of Bombay, 1976

AN ABSTRACT OF A MASTER'S THESIS

submitted in partial fulfillment of the
requirements for the degree

MASTER OF SCIENCE

Department of Industrial Engineering
KANSAS STATE UNIVERSITY
Manhattan, Kansas

1981

ABSTRACT

The objective of this research is to generate three dimensional cutter paths for a two dimensional numerically controlled milling machine. The study has been carried out by generating motions along lines in a plane and space. Motions along a line are obtained by developing algorithms to generate incremental steps along each of the three axes viz, X, Y and Z, starting from one end of a line until the end point of that line is reached. In order to generate motions along curves in a plane and space, straight line approximations are made. The algorithms that are developed, are checked with a plot on a digital plotter which is interfaced with the Exidy Sorcerer micro-computer. To demonstrate the plots of lines and curves in space, the Z-axis is treated as a line passing through the origin at an angle of 45° with the +X-axis. The plots of lines and curves in a plane and space reveal that the numerically controlled milling machine will be able to generate three dimensional cutter paths when its table and turret are under the control of the Exidy Sorcerer micro-computer.