P.R.L.
TECHNICAL NOTE

TN-81-~~07~~26

2-D AND 3-D GRAPHICS PACKAGE FOR DRUM PLOTTER

BY

D.R. KULKARNI

July 1981

PHYSICAL RESEARCH LABORATORY
AHMEDABAD.9

2-D AND 3-D GRAPHICS PACKAGE FOR DRUM PLOTTER

by

D.R.Kulkarni

ABSTRACT

A graphics application package consisting of FORTRAN IV subroutines has been developed to draw two and three dimensional pictures on drum plotter attached to the computer system IBM 360/44 under the batch-processing environment of the operating system PS-44. In two dimension it provides subroutines not only for usual point plotting and histogram but also for number of geometrical shapes with a built-in capability of rotating them as desired. In three dimension a method of slicing has been used to draw 3-D graphs as well as approximate 3-D perspective of the solid objects. Finally a true wire-frame perspective could be obtained for the objects defined both in terms of points and lines. Various perspective views can be obtained by changing the view point as well as rotating the object about any arbitrary axis.

Keywords:

1) Batch-computer graphics for drum plotter
2) Two-dimensional plotting
3) Three-dimensional plotting
4) Geometric drawing

TABLE OF CONTENTS

			Page No.
	ABSTRACT		
I	INTRODUCTION	...	1
II	TWO-DIMENSIONAL GRAPHICS	...	5
III	3-D PLOTTING BY METHOD OF SLICING	...	14
IV	GENERATION OF 3-D PERSPECTIVE DISPLAY	...	20
	APPENDIX I	...	27
	APPENDIX II	...	29
	APPENDIX III	...	33
	APPENDIX IV	...	35

INTRODUCTION

The computer-graphics is one of the most fascinating applications of modern computers. The graphical displays generated by computer help considerably in solving many problems in information processing. In fact it would not be an exaggeration to say that the computer graphics add one more dimension to the potentialities of the computer as a problem solving tool. Various types of devices can be attached to the computer to obtain the graphical displays. The choice of these devices would be mainly governed by the requirements of computer users. The display devices such as digital plotters, film recorders etc. are quite slow and are therefore used only as output devices to generate the pictures. Very often these devices are used as off-line independent units. Such a system is quite useful in its own right and may be called as 'batch computer graphics'. However, the appeal and the scope of the term "computer graphics" has been mainly through what is described as 'Interactive Computer Graphics'.

Compared to batch computer graphic system, the interactive computer graphic requires both better hardware support and sophisticated software support. A fast and versatile graphic CRT display unit with a suitable accessory for feed back such as light pen etc., attached to the computer would, in general, form the hardware requirements of the interactive graphic system. It may be noted that the display unit in interactive graphic system serves both as input and output unit. The complex software system is required to support the interactive computer graphics system. Due to fast response of the device, the interactive graphic system makes it possible to have a 'dialogue' between the user and the system. In fact the system can respond to the user as fast as he can respond to it. This on-line conversation is a great asset in applications such as computer - aided design (CAD), computer-aided instruction (CAI), artificial intelligence, information retrieval etc. The design of interactive computer graphics involves all these branches of computer technology that allows us to interact with a computer process by means of pictures.

In this note we report the development of 2-D (two-dimensional) and 3-D(Three dimensional) graphics application package to be used on the system IBM 360/44 with an on-line drum plotter device under the batch-processing environment of the operating system 44 PS. As in any batch computer graphics the drum plotter has been merely used as output device. The package consists of set of subroutines in FORTRAN IV language.

The package displays the picture on the paper in a rectangular frame defined by the user. This frame may be called a screen or a window. Fixing the window centre and varying its size changes the magnification of the display. Further by changing the centre of window it is possible to move the picture in any part of the window.

In 2-D graphic package the points in the object coordinate space may undergo various transformations such as scaling, rotation, translation etc. The transformed points are later subjected to two-dimensional clipping transformation to ensure that the portion of the picture that goes outside the specified window is automatically clipped. Besides the usual 2-D plotting the package consists of subroutines for histogram and various geometrical figures. The package can, thus, be used for geometrical drawing.

In 3-D graphic package the points in the object coordinates space are first transformed by 'viewing transformation' to get the perspective view of the object. The perspective view thus obtained, would undergo the three-dimensional clipping transformation. Subsequently the transformed points would, in general, be subjected to another transformation to eliminate the hidden lines and hidden surfaces of the objects to increase the three-dimensional effect. However, most of the algorithms to eliminate the hidden lines and surfaces are quite time-consuming besides being too narrow in their applicability. We, therefore, do not apply the transformation to remove hidden lines and surfaces. Thus finally we get what can be described as "Wire Frame Drawings" of the solid objects. We obtain these drawings when the object is specified in terms of object-coordinate points as well as when it is specified in terms of lines on the objects. Besides the perspective plotting we give a simple method of plotting the 3-D object by dividing the object in thin slices perpendicular to any co-ordinate axis.

..4

In section 2 we describe the details of the subroutines developed for 2-D pictures. In section 3 we describe the subroutines to plot the 3-D object by method of slicing. In section4 we give the details of perspective plotting of 3-D objects described both in terms of points and lines. The appendix-I gives the transformation matrices to be used in two-dimensional space. The appendix-II gives the transformation matrices used in three-dimensional space. The appendix-III gives the complete list of subroutines developed for this package. The appendix-IV gives the details of the control cards to use these subroutines.

II TWO-DIMENSIONAL GRAPHICS

To display a two-dimensional object on the output device, one primarily requires the points on the object as obtained in the co-ordinate space fixed in it. These points are called the points in the object co-ordinate space. These points usually undergo various transformations of scaling, rotation and translation as described in appendix I. Before these points are displayed on the rectangular frame defined on the device, each one of them is checked to see if it lies outside the frame. If so the point is omitted. In case of objects defined by lines with two points, the decision is not quite straight forward. If both the points lie outside the frame, then of course the line will not be displayed. However, when only one point lies outside the frame and other one inside the frame, it implies that part of the line can be seen in the frame. The operation to determine which portion is seen and which is clipped is basically reduced to find the point of intersection of the line and the sides of the rectangular frame. This process which is known as two-dimensional clipping ensures that portion that goes outside the defined frame has been automatically clipped. The points and the lines which are visible on the frame are subsequently displayed on the output device.

The points in the object coordinate space can be obtained either by direct measurement or they can be generated using mathematical equations. We have developed the package of subroutines to display the two-dimensional pictures when the object points are available. Besides the display of general two-dimensional given points, the subroutines have been developed to obtain various geometrical shapes. We describe the brief description of various subroutines available in the package.

1) SUBROUTINE LINE (X1, Y1, X2, Y2, THETA)

a subroutine to draw a line joining the points (X1,Y1) and (X2, Y2) when rotated through an angle THETA in degrees about the point (X1,Y1). These subroutine takes into account the clipping if any. It calls two subroutines viz. ROTPT and MATMUL.

2) SUBROUTINE EQITRA (X,Y,AL,THETA,S,INDEX)

It draws an equilateral triangle at a point (X,Y) with AL as the length of its sides after rotating it through an angle THETA in degrees about the point (X,Y). When INDEX is non zero, the vertices of the traingle are connected to the central point. The matrix S(3,3) gives the vertices of the triangle in the sequence starting from the vertex on the left hand side in the clockwise direction. It calls the subroutines ROTPT, MATMUL and LINE.

3) SUBROUTINE RCTNGL (X,Y,AL,BL,THETA,S,INDEX)

a subroutine to draw a rectangle with sides AL (vertical) and BL (horizontal) at a point (X,Y) after rotating it through an angle THETA about the point (X,Y). When INDEX is equal to 0 when diagonals of the rectangle are not drawn, 1 when they are drawn, The array S(4,3) gives the vertices of the rectangle in the sequence starting from the lower vertex on the left-hand side in the clockwise direction. It calls the subroutines ROTPT, MATMUL and LINE.

SUBROUTINE PRLGRL (X1,Y1,AV,AH, AVH, THETA, P, INDEX)

C a subroutine to draw a parallelogram with AV as its vertical side and AH as its horizontal side and with an angle AVH in degrees between them after rotating through an angle THETA in degrees about the point (X1,Y1). When INDEX is equal to zero, the diagonals are not drawn and one when they are drawn. The array P(4,3) gives the vertices of the parallelogram in the sequence starting from the lower vertex on the left hand side in the clockwise direction. It calls the subroutines ROTPT, MATMUL and LINE.

Continued to next page

5) SUBROUTINE TRAPZM (X,Y,AV,AHL,AVH,THETA,P,INDEX)

a subroutine to draw a symmetric trapexiumwith lower horizontal side of length AHL and both the vertical sides equal to AV. The angle between AHL and AV at the lower left vertex (X,Y) is AVH in degrees. The trapezium is drawn after rotating it through an angle THETA in degrees about the point (X,Y). INDEX is equal to zero when the diagonals are not drawn and one when they are drawn. The array P(4,3) gives the vertices of the trapezium in the sequence starting from the lower vertex on the left hand side in the clockwise direction. It calls the subroutines ROTPT, MATMUL and KINE.

6) SUBROUTINE CVXPLG (X,Y,NSIDE,AL,THETA,P,INDEX)

a subroutine to draw a symmetric convex polygon of NSIDE sides, each of length AL at a point (X,Y) after rotating it through an angle THETA about the same point. INDEX is equal to zero when the vertices are not connected to the central point and one of they are connected. The array P(15,3) gives the vertices of the polygon in the sequence starting from the lowest left vertex in the clockwise direction. It calls the subroutines ROTPT, MATMUL and LINE.

SUBROUTINE ARC (X1,Y1,X2,Y2,H,W,INDEX)

a subroutine to drawn an arc connecting the points (X1,Y1) and (X2,Y2) with height equal to H at the middle point. INDEX is equal to zero if the mid-point is not connected to the peak of the arc, one if they are connected and two if the arc is divided in four equal parts (angle wise). The array W(5,3) gives the point on the ARC in the sequence starting from the point (X1,Y1) in the clockwise direction. It calls the subroutines ROTPT, MATMUL and LINE.

SUBROUTINE CAP (X1,Y1,X2,Y2,H,NCAP,A,INDEX)

a subroutine to draw an angular cap between the points (X1,Y1) and (X2,Y2) with height equal to H. The cap can be repeated NCAP times inthe same direction. INDEX is equal to zero when the middle point of (X1,Y1) and (X2,Y2) is not joined to the peak of the cap and one when they are connected. The array A(3,3) gives the three vertices in the cap in the sequence starting from the point (X1,Y1) in the clockwise direction. It calls the subroutines ROTPT, MATMUL and LINE.

SUBROUTINE SMCRCL (X1,Y1,X2,Y2,A,INDEX)

a subroutine to draw a semi-circle between the points (X1,Y1) and (X2,Y2). INDEX is equal to zero when bare semi-circle is drawn, one if semi-circle is divided in two parts from the centre and two if it is divided into four equal parts. The array A(4,3) gives the first three points on the semi-circle corresponding to the angle 45°, 90° and 135° and the fourth one is the middle-point of point (X1,Y1) and (X2,Y2). It calls the subroutines ROTPT, MATMUL and LINE.

10) SUBROUTINE PLUS (X,Y,AL,THETA,P)

a subroutine to draw a symmetric plus sign with point (X,Y) as its centre and AL as the length of its four sides after rotating it through an angle THETA in degrees. The array P(4,3) gives the four points of the plus sign in the sequence starting from the right-hand point in the clockwise direction. It calls the subroutines ROTPT, MATMUL and LINE.

11) SUBROUTINE STAR (X,Y,AL,P)

a subroutine to draw a asterisk at the point (X,Y) with AL as the length of its sides. The array P(8,3) gives the eight points of the asterisk sign in the sequence starting from the right-most point in the clockwise direction. It calls the subroutine STAR.

12) SUBROUTINE CIRCLE (X,Y,S,THETA,A,INDEX)

a subroutine to draw a circle with centre as (X,Y) and radius of length S, after rotating it through an angle THETA in degrees. INDEX is equal to zero for a bare circle, one when it is divided in four equal points (anglewise) and two when it is divided into eight equal parts. The array A(8,3) gives the points on the circle when divided in the sequence starting from the right most ppint in the anti-clockwise direction. It calls the subroutines ROTPT, MATMUL, and LINE.

13) SUBROUTINE ELIPSE(X,Y,A,B,THETA,A,INDEX)
a subroutine to draw an ellipse with point (X,Y) as its centre and A and B as its major and minor axis respectively, after rotating it through an angle THETA in degrees. INDEX is equal to zero for a bare ellipse, one when both major and minor axes are drawn and two when it is divided into eight equal parts at the Centre (Anglewise). The array A(8,B) gives the points of the ellipse in the sequence, starting from the right-most point on the ellipse in the anti-clockwise direction. It calls the subroutines ROTPT, MATMUL and LINE.

14) SUBROUTINE GRAPH (N,X,Y,IFSCAL,NFRM,LABEL,IPC)
the subroutine may be used to plot N points. The array X(N) and Y(N) indicate the points on the X and Y axes respectively. IFSCAL is equal to zero if the scale parameters are to be determined by the program, one if they are provided by the user and two if the scale parameters are not to be changed for new plotting. In other words if IFSCAL is equal to 2, one can draw more than one plot on the same paper. The integer parameter NFRM is equal to 1 when both X and Y axes are drawn, 101 when the grid over the X and Y axes is drawn and 201 when no frame is drawn.

The integer parameter LABEL is equal to zero when no description for the X and Y axes is given, one when it given. The variable IPC denotes the symbol to put the point. It can be 11, 13, 15, 17, 19 and 90 when points need not be joined and 12, 14, 16, 18, 20 and 91 when adjacent points are connected by line. It calls the subroutines FRAME2, PRNT and DIGIT.

15) SUBROUTINE HSTGRM (N,X,Y,DX,IFSCAL,LBLXY,LBLHST,LBL) a subroutine to draw histogram for given N points. The arrays X(N) and Y(N) give the points on the X and Y axes, DX gives the width of the column in user's unit for X axis. To decide the parameter IFSCAL see the description for the subroutine GRAPH. To decide the parameter LBLXY, see the description of parameter LABEL in the subroutine GRAPH. The parameter LBLHST is equal to one when each column of the histogram bears the seperate description to be provided by the user. The description is put parallel to Y axis. When LBLHST is equal to zero, no such description is provided by the user. The array LBL (N,10) denotes the working array to store the description for each column of the histogram. It calls the subroutines FRAME2, PRNT and DIGIT.

All the subroutines given above use various system subroutines for plotting which are not described here. We have also not given the details of other subroutines which would not be directly useful to the user. In all the subroutines described above the type of the argument is decided by the default option of the FORTRAN IV language. In order to make use of these routines, the appendix IV may be referred. It must be stated here that the package contains the subroutines only for the most common geometric shapes. However, many more shapes can be added to it. Figure one shows some designs formed using the subroutines in the package.

III 3-D PLOTTING BY METHOD OF SLICING

i) Method

This method of drawing 3-dimensional (3-D) pictures is only an approximation to the perspective 3-D plot. The idea behind this method would be to cut the solid object in thin two-dimensional slices perpendicular to one of the axes and arrange them appropriately along the same axis to obtain nearly the pèrspective view of the object. To illustrate the method clearly we assume that the solid object is defined by a relation

$$Z = f(X, Y)$$

The slices perpendicular to Y axis are obtained for various values of Y with a suitable interval of dy. The two dimensional (X-Z) planes thus obtained, would be placed along the y-direction. We use the 3-D Cartesian Co-ordinate system in which we assume Z to be a vertical axis, X, to be a horizontal axis and Y to be the axis going inside the page. In 2-D representation, however, the Y axis would usually make an angle Θ with the X-axis, which is less than 90 degrees. The first slice would be drawn in the X-Z plane without modification. The subsequent ones would be plotted behind the first only after changing their co-ordinate suitably. These changes would depend on the angle Θ and the increment dy. In general each point (X, Z) is changed as follows:

$$X^1 = X + (n-1)\, dy \, Cos\,(\theta)$$

$$Z^1 = Z + (n-1)\, dy \, Sin\,(\theta)$$

where n denotes the slice number.

A subroutine D3PLOT based on this procedure would be described later. This subroutine plots all the points of the objects, including those which are supposed to be hidden, The appearance of hidden points creates the ambiguity regarding the true 3-D perspective of the object. The simple way to remove the hidden points would be possible in this method. The procedure is, however, quite crude and it also eliminates certain points which are not really hidden from the view. In this procedure the angle need not be specified, instead the increment dx used in the x direction to obtain the points of the X-Z slices is required to be known. Then the co-ordinates of the slices are changed as follows:

$$dz = \sqrt{dy^2 - dx^2}$$

$$X^1 = X + (n-1)dx$$

$$Z^1 = Z + (n-1)dz$$

where n is the slice number

Notice that this method requires increment dy to be greater than dz. Also note that the change in co-ordinate X and the increment dx used in X are made same, thereby the changed co-ordinate would exactly fall on the next value

..16

of X used in the slice. This property has been used to eliminate the hidden points. When plotting the points of the n^{th} slice, each of them is checked against the point of the $(n-1)^{th}$ slice having same value for X coordinate. The point would be plotted only if it is greater than the previous one. A subroutine PLOTD3 may be used to obtain this type of drawings.

ii) Subroutine D3PLOT

As described above the subroutine D3PLOT can be used to plot the 3-D graphs and 3-D picture without removing the hidden points. This subroutine, in fact, plots only one slice at a time. In order to plot an object cut in N slices, the subroutine has be called N times. The subroutine which is written for a single precision is defined as

SUBROUTINE D3PLOT (N,PH,PV,PB,IFSCAL)

Where PH and PV are the arrays of N elements representing the co-ordinates of the slice to be plotted on the horizontal and vertical axes. The scalar variable PB denotes the value on the middle axis for which the above slice has been obtained. IFSCAL is equal to zero if you want to define the new scales. If IFSCAL is equal to one the new plots can be obtained on the same paper using the previous scale. To start with IFSCAL should be assigned zero. Once called the subroutine changes the value of this parameter to one. When this subroutine is called first time, it asks for the following data to be provided by the user.

1st data card (FORMAT (8F10.4))

It reads the variables

HMIN, HMAX, HL, HD, VMIN, VMAX, VL, VD

HMIN and HMAX represents the samllest and the largest values on the horizontal axis, HL denotes the length (in inches) of the horizontal axis and HD indicates the division on the horizontal axis in user's unit.

The variables VMIN, VMAX, VL and VD define the corresponding quantities for the vertical axis.

2nd data card (FORMAT (5F10.5,5I1))

It reads the variables

BMIN, BMAX, BL, BD, THETA, IFBXIO, NFRAME, NMCDH, NMCDV, NMCDB

The variables BMIN, BMAX, BL and BD represent the same quantities for the middle axis as defined in the first data card. THETA is the angle (in degrees) between the horizontal and middle axis. IFBXIO can be either one or not equal to one. If it is one, then the Cartesian Coordinate system is left-handed otherwise it is right-handed one. The parameter NFRAME can also be either one or not equal to one. If it is one, then the Cartesian Coordinate axes would not be drawn, otherwise they would be drawn. The parameter NMCDH, NMCDV, and NMCDB represent the numeric codes for the labels of the horizontal, vertical and middle axes respectively. The label codes are 1, 2 and

3 for the X, Y and Z axis respectively. For 3-D plotting of the solid objects the parameter NFRAME is usually one.

The subroutine D3PLOT calls the subroutines FRAME3 and PLOT3.

iii) Subroutine PLOTD3

This subroutine is used to plot 3-D picture by removing the hidden points. Like subroutine D3PLOT, this subroutine also plots only one slice at a time and has be called repeatedly to complete the picture. It is written in single precision and is defined as

SUBROUTINE PLOTD3 (NHV,PH,PV,NB,KB,DB,IFSCAL,NUM)

where PH and PV are the arrays of NHV elements representing the coordinates of the slice to be plotted on the horizontal and vertical axes. The variable NB denotes the total number of slices to be plotted and KB gives the actual slice number which is being plotted. The variable DB denotes the increment along the middle axis in user's unit. The parameter IFSCAL is same as described for the subroutine D3PLOT. The parameter NUM which should be initialized to zero keeps track of the number of times the subroutine PLOTD3 has been called. The value of NUM gets updated in the subroutine and hence user need not change it.

When called for the first time, this subroutine asks for the following data to be provided by the user in two cards:

st data card (FORMAT(8F10.4))

It reads the variables

HMIN, HMAX, HL, HD, VMIN, VMAX, VL, VD

These variables are the same as described in the 1st card of the subroutine D3PLOT.

d data card (FORMAT(8F10.4))

It reads the variables

BMIN, BMAX, BL, BD

These variables are also described in the 2nd card of the subroutine D3PLOT. While using the program it should be ensured that the increment in DB is greater than the increment in horizontal axis. It calls the subroutines FRAME3 and NPLOT3. While the subroutine D3PLOT can draw Cartesian Coordinate system, the subroutine PLOTD3 would not do so. Consequently the subroutine D3PLOT can also be used in plot 3-D graphs as well.

In order to make use of these routines, user may refer to appendix IV for detail of control cards and instructions. Fig.2 gives the 3-D graph obtained using subroutine D3PLOT.

IV GENERATION OF 3-D PERSPECTIVE DISPLAY

In Section III, we have seen how to obtain the illusion of the 3-D perspective view of the solid objects in a rather naive manner. In this section we generate the true 3-D perspective view known as 'Wire-frame view'. The view is obtained from the perspective projection of the object as viewed from an arbitrary position. Though this view is perfectly perspective, it does not contain enough depth information to interprete the description of the object without ambiguity. For proper interpretation, it is necessary to eliminate the hidden lines and hidden surfaces from wire-frame view. Also there are other depth cues such as intensity variations, stereoscopic view, kinetic depth effect etc. to convey the depth information. However, all these methods require special hardware and consume a good deal of computer time. We have, therefore, confined ouselves to only 'Wire-frame view' of the solid objects.

As stated above, the wire-frame view is the perspective projection of the object on the screen as seen by the viewer. Therefore this view depends on the position of the viewer, his distance from the screen, the size of the screen, its distance from the object and finally on the direction in which the viewer is looking. We give below very brief procedure to generate the wire-frame view as described by Newman and Sproull.

The object is described by the points (X,Y) given in the ject coordinate space which is assumed to be the right-handed rtesian Coordinate system. We now define what is called the ye coordinate system' which has its origin fixed at the view int (position of the viewer in object coordinate space) and its axis pointed in the direction of view. We choose the eye-ordinate system to be a left handed Cartesian Coordinate System that its X_e and Y_e axes will align with the X_s and Y_s axes of e display screen which is placed between the viewer and the ject. The first step to obtain the perspective view is to ansform the point from object space into the eye coordinate system. is transformation called the 'viewing transformation' can be rived by concatenating several rotation and translation operations 3-dimension given in appendix 2. Thus each point (X,Y,Z) in e object coordinate space becomes (X_e, Y_e, Z_e) in the eye coordinate ace. The next step is to project each point $(X_e Y_e Z_e)$ on the reen and find the screen coordinates (X_s, Y_s) of its projected age as measured in the eye coordinate system. If a is the distance tween the screen and the viewer, then using simple geometry we t the following relations

$$X_s = \frac{a.X_e}{Z_e} \quad \text{and} \quad Y_s = \frac{a.Ye}{Z_e} \qquad .....(1)$$

e number X_s and Y_s can be converted to dimensionless fractions dividing by the screen size b and hence we obtain

$$X_s = \frac{a}{b} \; \frac{X_e}{Z_e} \quad \text{and} \quad Y_s = \frac{a}{b} \; \frac{X_e}{Z_e} \qquad ....(2)$$

...22

Further if we try to locate the picture in a specified part of the screen (called viewport) defined by its centre (V_{cx}, V_{cy}) and its extent (V_{sx}, V_{xy}) then one obtains

$$X_s = \frac{a.X_e}{b.Z_e} .V_{sx} + V_{cx}$$

$$Y_s = \frac{a.Y_e}{b.Z_e} .V_{sy} + V_{cy} \qquad ....(3)$$

It may be noted that each point is divided by its Z_e co-ordinate. In fact generating a true perspective image require dividing by depth of each point. Further the expressions involve only the ratio (a/b) and therefore the units of measurement of t parameters a and b are independent of the coordinate system. By playing with these parameters the image on the screen can be magnified or even distored.

Before one obtains the final screen coordinates as given in (3) it would, in fact, be necessary to see that the point (X_e, Y_e, Z_e) lies wihtin the viewing pyramid defining the portion of eye-coordinate space which the viewer can actually see. Otherwise it would not be visible. The conditions that a point be visible are

$$-Z_e \leqslant (a/b)\ X_e \leqslant Z_e$$

and

$$-Z_e \leqslant (a/b)\ Y_e \leqslant Z_e$$

These conditions exclude the points behind the viewpoint. However, the lines can not be processed as easily as points. They must be subjected to three dimensional clipping before being accepted for display. Like two-dimensional clipping, the three-dimensional clipping is performed by finding the points of intersection of the line with the planes of the viewing pyramid. A line may be rejected as invisible if no part of it intersects the pyramid. Otherwise the end points of the visible portion of the line are calculated in the eye-coordinate system.

In this section we describe the subroutines D3PPP and D3PLP. The subroutine D3PPP gives the 3-D view when object is described in terms of points only while the subroutine D3PLP is useful when it is described both in terms of points and lines.

Subroutine D3PPP

This subroutine gives the 3-D wire-frame perspective display of the object described by the points in the object coordinate space. The subroutine is written for single precision and is defined as

SUBROUTINE D3PPP(N,X,Y,Z,IROT,IV,IPC,XE,YE,ZE)

The details of the arguments are as follows:

The argument X, Y and Z are one dimensional arrays of N elements, representing the N coordinates (X,Y,Z) of the object. The parameter IROT is not equal to zero if the object is to be related before obtaining the perspective display. In that case the subroutine would demand the third data card to be described later. Otherwise IROT would be zero. The parameter IV should be zero when this subroutine is called first time. Subsequently it would be made non-zero in the subroutine. However, for a new display requiring new data such as new view point, screen size, angle of rotation etc. it may be again made zero by the user.

The parameter IPC is the parameter required by the system routine of the plotter. It decides both the symbol to display the point and also if the adjoining points are to be connected by the straight line. For various valid numbers for IPC, refer to subroutine GRAPH in section 2. For simple point plotting IPC may be equal to 90. The argument XE, YE and ZE are also the one-dimensional arrays of N element, representing the points of the object obtained after applying the rotational (if required) and viewing transformations. These arrays are kept in the list of arguments for the subroutine only for providing the facility of object-time dimension. User may not have any use of these arrays in the main program.

When this subroutine is called with parameter IV equal to zero, the subroutine demands three data cards.

<u>1st data card (FORMAT (7F10.4))</u>

It reads the variables

XV, YV, ZV, XP, YP, ZP

(XV,YV,ZV) give the coordinate of the viewpoint and

(XP,YP,ZP) give the coordinate of the point on the object at which the viewer is looking.

2nd data card (FORMAT (7F10.4))

It reads the variables

DISTES, SZX, SZY, VCX, VCY, VSX, VSY

The variable DISTES gives the distance between viewpoint and the screen. SZX and SZY give the length of the screen along horizontal and vertical axes respectively. VCX and VCY represent the coordinates of the centre of the **viewport**. VSX and VSY give the extent of the viewport along X and Y axes respectively. By suitably changing the variables VCX, VCY, VSX and VSY the display can be moved anywhere on the screen. By increasing VSX and VSY by equal proportion, one can magnify the display.

3rd data card (FORMAT (7F10.4))

This data card would be required only if the parameter IROT is non-zero. It reads the variables

XL, YL, ZL, AM, AN, ANGLE

This data card defines the axis of rotation given by direction consines (AL,AM,AN) passing through the point(XL,YL,ZL) for rotating the object through angle ANGLE (in degrees) about the axis.

Subroutine D3PLF

It gives the perspective display of the object defined both in terms of its points and lines. The subroutine is written for single precision and is defined by

SUBROUTINE D3PLP (N, X, Y, Z, NL, LSP, LEP, IROT, IV, XE, YE, ZE)

The details of the arguments N, X, Y, Z, IROT, IV, XE, YE and ZE are exactly same as described for the subroutine D3PPP.

The parameter NL gives the number of lines required to define the object. The arrays LSP and LEP having NL elements store the starting and ending points of the line. For example LSP (5) and LEP (5) would give the starting and ending point numbers of the fifth line.

When called this subroutine would demand three data cards if the parameter IV is equal to zero. The third data card would, however, be required only if the parameter IROT is non-zero. The description of these data cards would be exactly similar to those given for subroutine D3PPP.

By way of illustration the figures 3 and 4 have been obtained using the subroutines D3PPP and D3PLP respectively. Appendix III gives the list of subroutines used by D3PPP and D3PLP. Appendix IV gives the list of control cards required to have access to these subroutines.

APPENDIX - I

TWO-DIMENSIONAL MATRIX TRANSFORMATIONS

Two-dimensional transformations can be represented in a uniform way by (3 x 3) matrices. For that a point (X,Y) is appended with a third dummy coordinate of unity to become (X,Y,1). The addition of the third dummy coordinate enables us to represent all the transformations in two-dimensional space in matrix form. We give below the matrix representation of simple transformation of translation, rotation and scaling:

Translation

$$(X^1, Y^1, 1) = (X, Y, 1) \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ T_x & T_y & 1 \end{bmatrix}$$

where T_x and T_y indicate the amount of translation used along x and y axes respectively.

Rotation

$$(X^1, Y^1, 1) = (X, Y, 1) \begin{bmatrix} Cos\theta & -Sin\theta & 0 \\ Sin\theta & Cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

where θ is the angle through which the coordinate space is rotated about the origin in clockwise direction. For anti-clockwise direction the same is $-\theta$.

Scaling

$$(X^1, Y^1, 1) = (X, Y, 1) \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

where S_x, S_y are the scaling factors applied for x and y axes respectively.

When more than one transformation needs to be applied, the resultant final transformation may be obtained just by the product of corresponding matrices in the given sequence. This is called the concatenation of transformations. Thus the complex transformation can be described as concatenations of simple ones. The concatenation is possible only because all simple transformations have been expressed in the matrix form as given above. We illustrate below the use of concatenation for a case in which the point (X,Y) is transformed by rotating the space through the angle θ about the point (R_x, R_y). As rotation transformation can be applied only to rotate the points about the origin, we must first translate points so that (R_x, R_y) becomes the origin. Then apply the rotation through angle and finally translate the point so that the origin is restored. These three operations can be concatenated as follows:

$$(X^1, Y^1, 1) = (X, Y, 1) \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -R_x & -R_y & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ R_x & R_y & 1 \end{bmatrix}$$

(1) (2) (3)

If the values of R_x, R_y and θ, are known, the three matrices may be multiplied to yield one transformation matrix. It should be noted that the third coordinate of unity is only for the sake of mathematical convenience and should, therefore, be ignored while using the points for display purpose.

APPENDIX II

THREE-DIMENSIONAL MATRIX TRANSFORMATION

The matrix formulation of two-dimensional transformations given in Appendix I can be extended easily to three-dimensional space. As in the case of two-dimension, a point (X, Y, Z) in three dimensional space is also appended with the fourth coordinate of unity to express all the transformation in matrix form uniformly. The point (X, Y, Z) then becomes a point (X, Y, Z, 1) in which the dummy coordinate of unity is ignored while displaying the point. We give below the matrix forms of translation, rotation and scaling.

I) Translation

$$(X^1, Y^1, Z^1, 1) = (X, Y, Z, 1) \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ T_x & T_y & T_z & 1 \end{bmatrix}$$

where T_x, T_y and T_z are the components of translation in the X, Y and Z directions respectively.

II) Rotation

In two dimension the rotation has been defined about the given point. In three-dimension the rotation is defined not about the point but about the axis passing through the point. Therefore in 3-dimensional space, three rotational matrices are defined for X, Y and Z axes passing through origin (0, 0, 0).

..30

a) Rotation about X axis through the point (0, 0, 0):-

$$(X^1, Y^1, Z^1, 1) = (X, Y, Z, 1) \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The rotation angle θ is measured clockwise about the origin when looking at the origin from a point on the +X axis. Notice that the transformation matrix affects only the values of the Y and Z coordinates.

b) Rotation about Y axis through the origin (0, 0, 0):-

$$(X^1, Y^1, Z^1, 1) = (X, Y, Z, 1) \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

c) Rotation about Z axis through the origin (0, 0, 0):-

$$(X^1, Y^1, Z^1, 1) = (X, Y, Z, 1) \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The three rotation matrices given above are for right-handed Cartesian Coordinate System. If you are looking along +Z axis with your head along +Y axis, then the right-handed system will have its +X axis at your left and the left-handed system will have it at your right.

III) Scaling

$$(X^1, Y^1, Z^1, 1) = (X, Y, Z, 1) \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where S_x, S_y and S_z are scaling factors for X, Y and Z axes respectively.

Any complex transformation can be considered as the concatenation of many simpler transformations. A transformation for rotation about an arbitrary axis through an arbitrary point can be derived using the above primitive transformations as follows. Suppose (X,Y,Z) is the arbitrary point through which an arbitrary rotation axis with direction cosines (a,b,c) passes. The steps in the rotation through an angle θ about this axis are

1) As all the rotation matrices are defined for the axis passing through origin (0,0,0) we change the origin to the point (X,Y,Z) by translation (matrix T).

2) Again all the rotation matrices are defined only for one of the axis (X,Y or Z) of the system, we rotate the system so that the arbitrary axis with direction cosines (a,b,c) would be aligned along the Z axis. This is done in two steps:

   i) Rotate about the X-axis so that the given axis would lie in (X-Z) plane. The angle α required would be defined by the direction cosines(a,b,c) and is given as

$$\cos\alpha = \frac{c}{\sqrt{b^2+c^2}} \quad \text{and} \quad \sin\alpha = \frac{b}{\sqrt{b^2+c^2}}$$

   and matrix is R_1

ii) Then rotate about the Y axis so that the axis would align exactly with Z axis. The required angle of rotation β would be

$$\cos\beta = \sqrt{b^2+c^2} / \sqrt{a^2+b^2+c^2}$$

and $$\sin\beta = a/\sqrt{a^2+b^2+c^2}$$

This matrix is R_2

3) Now it is possible to apply rotation transformation to rotate the object about the Z_axis through an angle (Matrix $R(\theta)$)

4) Inverse transformation of step 2 (Matrix R_2^{-1} R_1^{-1})

5) Inverse transformation of step 1 (matrix T^{-1})

Thus the complex transformation R_A to rotate the object about the arbitrary axis passing through arbitrary point would be

$$R_A = TR_1R_2R(\theta)R_2^{-1}R_1^{-1}T^{-1}$$

APPENDIX - III

THE LIST OF SUBROUTINES DEVELOPED FOR THE PACKAGE

I) 2-D graphics subroutines:

1)	Sub.	ROT2	9)	Sub.	EQITRA	17)	Sub.	PLUS
2)	Sub.	TRANS2	10)	Sub.	RCTNGL	18)	Sub.	STAR
3)	Sub.	SCALE2	11)	Sub.	PRLGRM	19)	Sub.	CIRCLE
4)	Sub.	ROTPT	12)	Sub.	TRAPZM	20)	Sub.	ELIPSE
5)	Sub.	CLPLN2	13)	Sub.	CVXPLG	21)	Sub.	GRAPH
6)	Fun.	ICD2	14)	Sub.	ARC	22)	Sub.	HSTGRM
7)	Fun.	LAND	15)	Sub.	CAP	23)	Sub.	MATMUL
8)	Sub.	LINE	16)	Sub.	SMCRCL			

II) 3-D plotting by method of slicing:

1)	Sub.	D3PLOT	3)	Sub.	DIGIT	5)	Sub.	PLOTD3
2)	Sub.	FRAME3	4)	Sub.	PLOT3	6)	Sub.	NPLOT3

III) 3-D perspective plotting:

1)	Sub.	D3PPP	8)	Sub.	CLPMAT	15)	Sub.	CLIPLN
2)	Sub.	D3V	9)	Sub.	TRANS3	16)	Fun.	ICODE
3)	Sub.	ROTAAX	10)	Sub.	ROT3	17)	Fun.	LAND
4)	Sub.	MATMUL	11)	Sub.	LEFT			
5)	Sub.	PLOTPT	12)	Sub.	SCALE3			
6)	Sub.	SCREEN	13)	Sub.	D3PLP			
7)	Sub.	VIEW	14)	Sub.	PLOTLN			

IV) Besides these routines the package also uses the following system routines to drive the plotter:-

..34

1) Sub. PLOT

2) Sub. CHAR

3) Sub. SCAL

4) Sub. PRNT

APPENDIX IV

THE DETAILS OF CONTROL CARDS TO USE THE PACKAGE

All the subroutines developed for this package have been stored in the compiled form in the private library called 'OWNLIB'. This library resides on the system disk of the operating system PS44. A FORTRAN user can call these subroutines in his program by using following eleven control cards:

1st	//NAME JOB user's code, time in mts, no. of pages
2nd	//PROG EXEC FORTRAN(MAP)
	FORTRAN DECK
3rd	/*
4th	//SYS005 ACCESS OWNLIB
5th	//EXEC RLNKEDT(SYS005)
6th	/*
7th	//SYS005 ACCESS PLOT, 022=
8th	//SYS003 ACCESS SCRATCH 280=
9th	//EXEC
	DATA CARDS
10th	/*
11th	/& (red card)

Note:- 8th control card is required only when you are trying to plot the alphabetic characters on the plotter. They are required mostly in subroutines GRAPH, HSTGRM and D3PLOT. Further while using the subroutines for two-dimensional drawing, user should specify the size of the rectangular window by defining the variables XMIN, XMAX, YMIN and YMAX in the main program. The same variables are then declared in the COMMON block in the main program as follows:

COMMON/BORDER/XMIN,XMAX,YMIN,YMAX

Then call a plot subroutine as follows:

CALL PLOT(201,XMIN,XMAX,XL,XD,YMIN,YMAX,YL,YD)

to define the scale. To use subroutines GRAPH and HSTGRM some data cards are required. The details of those cards would be available with the author.

REFERENCES

1) *Principles of Interactive Computer Graphics*
by W.M.Newman and R.F.Sproull
McGraw-Hill Book Company (1973)

2) *Interactive Graphics for Computer-aided Design*
by M.D.Prince
Addison-Weslay Publishing Company (1971)

3) *Computer graphics in communications*
by W.A.Fetter
McGraw Hill Book Company (1965)

4) *Scientific & Engineering problem-solving with the Computer*
by W.R.Bennett
Prentice-Hall Inc. (1976)

5) *Computer aided architectural design*
by W.J.Mitchel
Petrocelli/Charter New York (1977)

ACKNOWLEDGEMENT

It is a pleasure to thank Professor S.P.Pandya for his constant interest and encouragement during the course of this work. Author would like to thank Mr.P.S.Shah and Mr.M.S.Patel for modifying the plotter system routine to shift the position of the computer code on the plotting paper. His thanks are also due to Mr.M.V.Bhavsar for providing the subroutines SCAL and PRNT. Finally he would like to thank Miss Meena Shah for her prompt typing of this technical note.

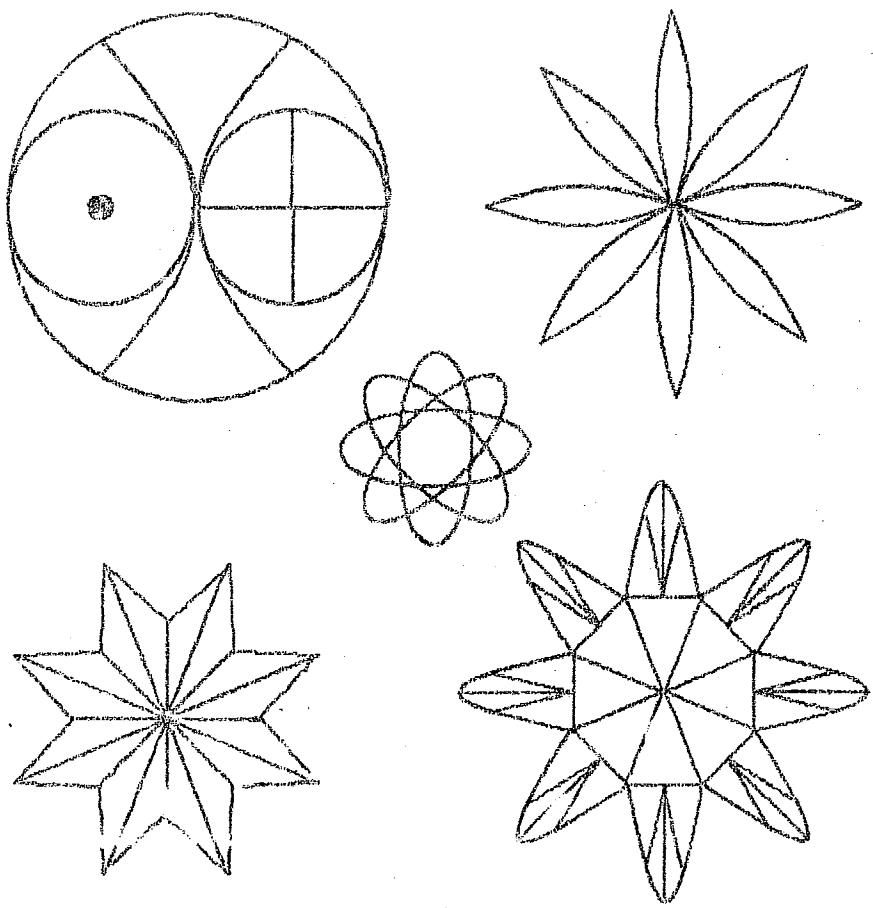

Fig.1   Some 2-D pictures obtained using subroutines for geometric shapes

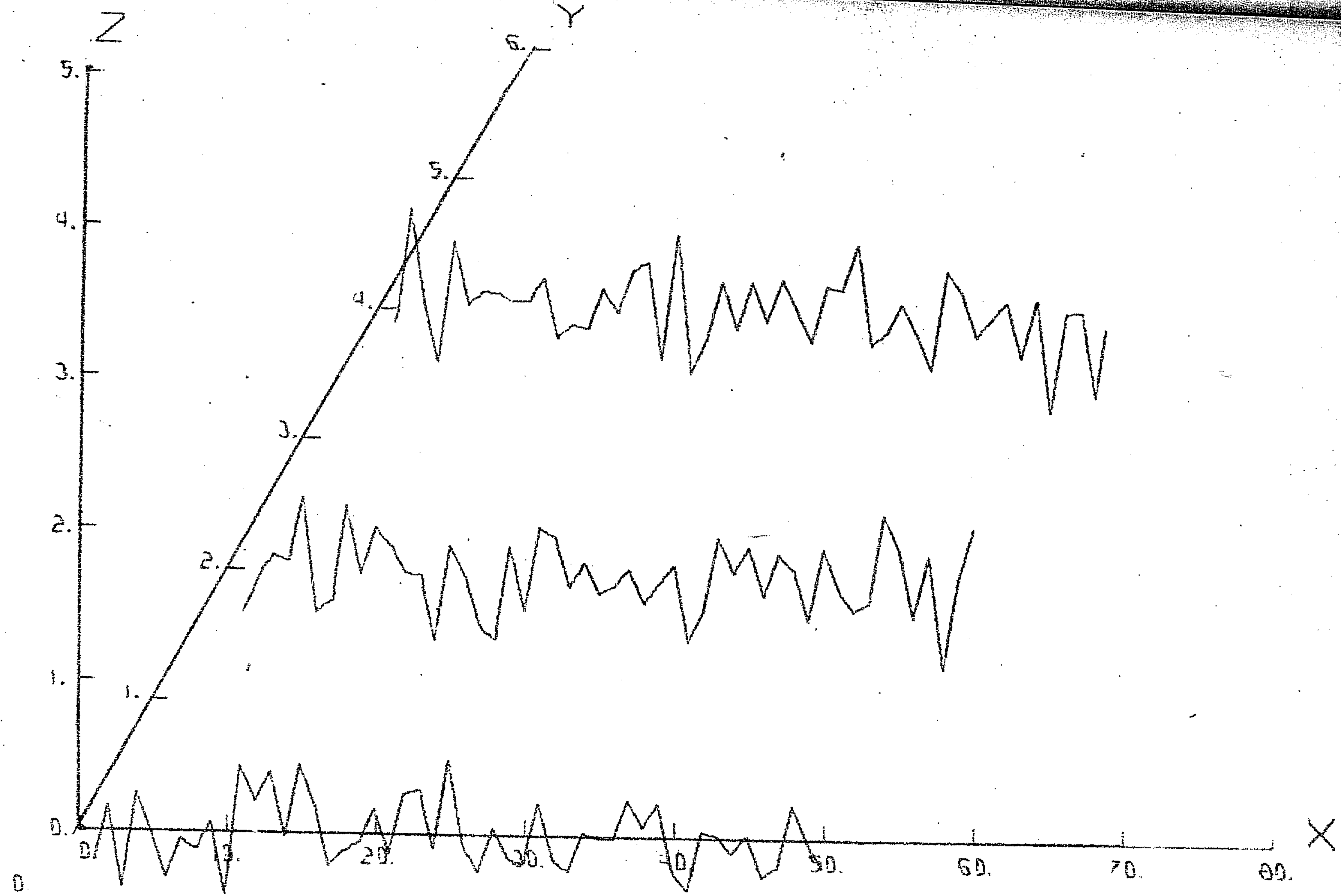

Fig.2 A 3-D graph obtained using the sub.D3PLOT

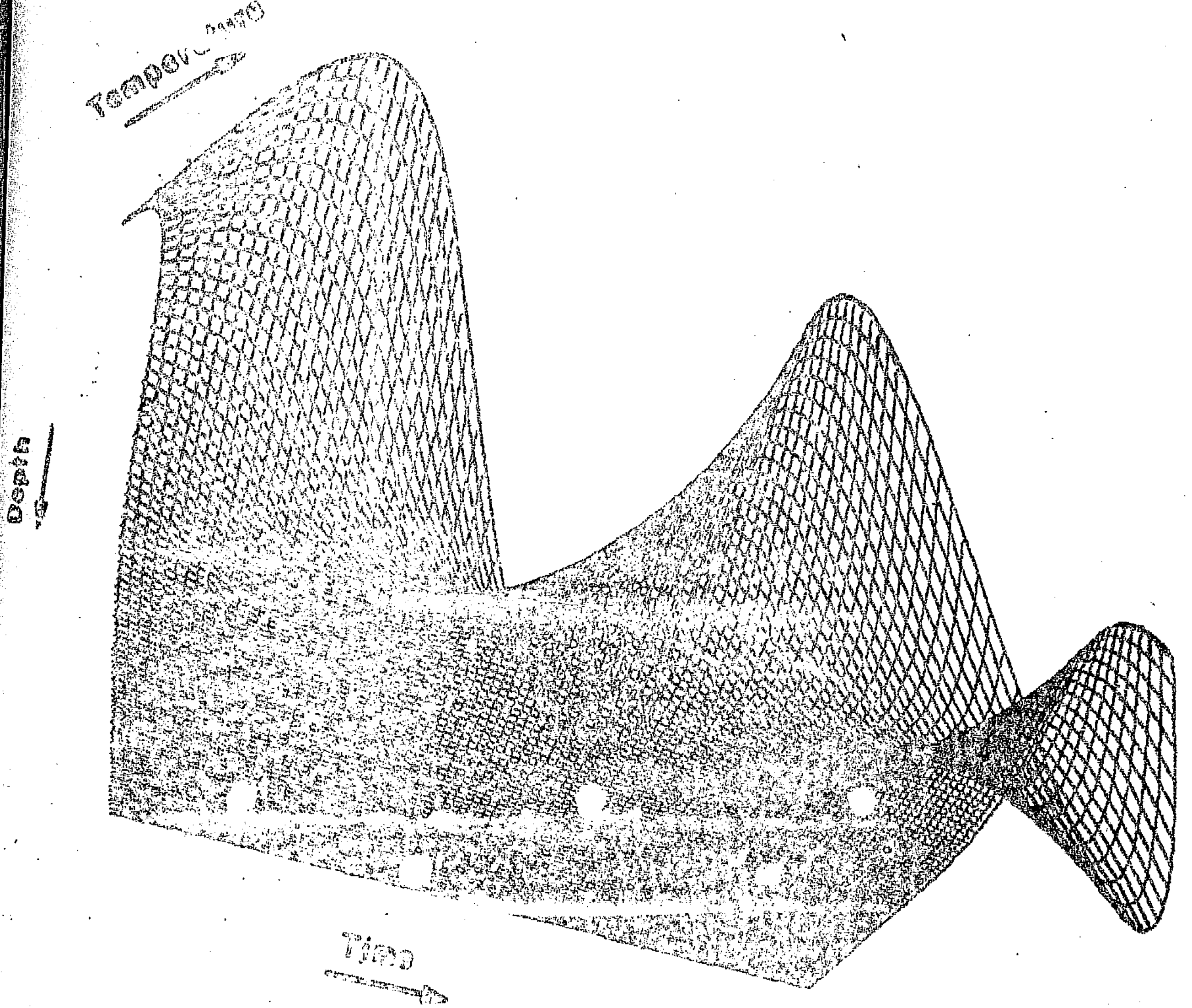

Fig.7 Three-dimensional plot obtained using the Sub.D3FP

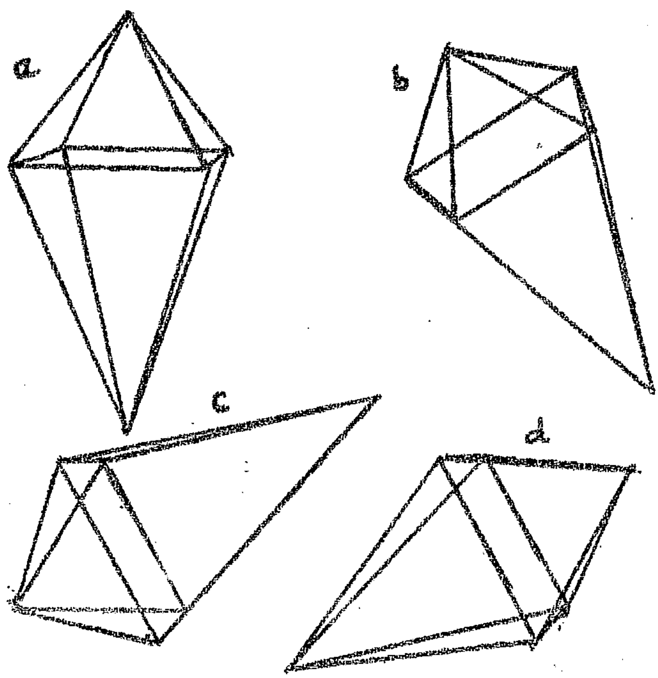

Fig.4 a) A perspective view of a tetrahedron obtained using the sub. D3PLP
b) Another perspective view of the same figure
c) A perspective view a when rotated through -120°
d) A perspective view a when rotated through $+60^{\circ}$