DEVELOPMENT OF AN INTERACTIVE COMPUTER GRAPHICS SYSTEM WITH APPLICATION TO DATA FITTING

BY

BAKIR ABDUL RASOUL HASSAN AL-HASHEMY

B.Sc.(Leeds), M.Sc.(Wales)

A Doctoral Thesis submitted in partial fulfilment
of the requirements for the award of
Doctor of Philosophy of the Loughborough University of Technology
May, 1978.

Supervisor: MR. G.N.C. GRANT
Department of Computer Studies

DECLARATION

I declare that the following thesis is a record of research work carried out by me, and that the thesis is of my own composition. I also certify that neither the thesis nor the original work contained herein has been submitted to this or any other institute for a degree.

To My Family

ACKNOWLEDGMENTS

I would like to express my gratitude to my supervisor, Mr. G.N.C. Grant, to whom I am greatly indebted for his advice and counsel throughout the period of my research and for the thoughtful guidance he has supplied during the preparation of this thesis.

I would also like to express my special thanks to Professor D.J. Evans who has provided continual encouragement and advice.

My thanks go to all the other members of the staff of this department.

It is a pleasure to acknowledge Miss J.M. Briers for her patience and diligence in typing this thesis.

Finally, my thanks to the Ministry of Higher Education of Iraq for providing me with financial support.

CONTENTS

CHAPTER 1

INTRODUCTION

The first important manifestation of computer graphics was at M.I.T. in 1963, when Ivan Sutherland demonstrated his SKETCHPAD system on the TX2 computer at Lincoln Laboratory. In this demonstration a cathode-ray tube was used in such a way that it generated geometric figures and by using a light-pen, the figures on the screen could be drawn and manipulated. Although he was preceded by some earlier graphics hardware development (e.g. WHIRLWIND 1, 1950), it was Sutherland who made the breakthrough in man-machine communication, to the extent of interacting with a computer by means other than bits, numbers or the omnipresent punched card. Following this work Project MAC was initiated at M.I.T. in early 1964, involving the use of time-shared terminals and later the display consoles. During the same period, General Motors were developing their system DAC-1 (Design Augmented by Computer), which involved the use of CRT displays: these subsequently became the prototype of the IBM 2250 display.

The concepts of the computer-driven display, light-pen interaction with the CRT trace, and time-sharing are adequate to make up the technology of on-line computer graphics for drawing or drafting. In the famous M.I.T. SKETCHPAD, a drawing was constructed of entities or 'instances' which could be manipulated and reproduced on the screen. This demonstrated that the computer had an 'understanding' of the picture, could make calculations based on it, and display the result. This concept is the basis for the extension of computer graphics into several application areas such as computer-aided design, artificial intelligence and information retrieval. For example, interactive graphics displays can be used as a powerful tool in the computer-aided design of products and systems. In many cases such displays provide an ideal interface between man and machine; the designer can exercise insight and judgement, while the computer undertakes involved calculations and analyses at high speed.

In general terms, computer graphics means images generated by computer, while interactive computer graphics means that the human must be involved with the computer through these images. The scope of computer graphics usage can vary in the level of sophistication involved, and may be classified as follows:-

(i) Historically, conventional batch-processing computer system first produced graphic output only via line printers, incremental graph plotters, video displays, etc. These are relatively expensive resources and graphic output is usually off-lined into a faster peripheral (typically magnetic tape) and consequently driving the plotter as a background job.

(ii) The introduction of time-sharing systems has made possible the idea of interactive computing. Thus, conversational input typically at a teletype terminal can be made to initiate computer output, and in particular, graphics output on a plotter or CRT. This is effectively 'semi-interactive graphics', since the process is interactive but no graphical input is involved.

(iii) The appearance of storage tube displays allowing a graphical input capability (e.g. by means of cross-hair cursor), has made it possible to build systems which enable the user to incorporate graphical interaction. However, this type of system still lacks the selective erasure capability which allows dynamic interaction. Such systems could be designated as 'static interactive graphics'. A wide range of scientific applications which do not require dynamic interaction can utilize such systems, if a well-designed graphic interface and suitable interactive techniques are used.

(iv) The use of a refreshable display and some input device (e.g. light-pen), allows change of the picture in dynamic fashion in real time. It is often desirable to remove part of the picture or change its position, thus requiring a highly interactive graphics capability; this implies an intelligent terminal with its own display processor and input capability. With an increase in size and sophistication of the display processor, the graphics sub-system can become independent of the host computer, including its communication front-end. Thus it is possible for some graphics systems to be used as stand-alone configurations (for relatively small applications).

(v) With more general interactive graphics systems where further computing resources are needed, the display system would be linked to a large machine which would provide the extra facilities to run more extensive graphics problems. This may require the computer to understand the picture which the user has drawn on the screen. That is, the computer must have a useful knowledge of the topology or the connectivity of the picture as well as the coordinates of significant points and elements. The programming and storage philosophy required for this level of sophistication is necessarily more involved.

Computer graphics, computer-based pictorial information processing, has appealed to computer designers and users since the early days. Both hardware and software developments have continued to improve graphical facilities, but the progress has been relatively slow. A number of quite sophisticated systems have been designed and built, but at a cost that makes them uneconomical for many users, and simply not available to most. As yet less attention has been given to the development of simple and relatively cheap systems, though the feasibility of such systems have been explored in recent years. At the same time, the need for graphics as a medium of information interchange has grown tremendously. With this increasing demand for computer graphics, terminal costs have reduced, allowing a wider use of terminals for this purpose. Consequently the need for improved software techniques for graphics has become apparent.

Therefore, interactive computer graphics has changed considerably over the past few years from an expensive, dedicated activity restricted to a privileged few users to a relatively low-cost, time-shared activity available to many users in most scientific/technical environments.

The aim of this work is to provide a graphical facility which is relatively low-cost both in regard to capital cost and run time overheads, and making efficient use of the available computing resources. The system must be generally available, general-purpose, portable, powerful and easily augmented by application programs in selected areas. The display terminal available in the early stages of development was the storage tube (Tektronix 4010). This display requires less software and processor power for its support than does a refreshable display. Admittedly, such a display limits the interaction and animation capabilities of the system but for the average application programmer such restrictions are not serious. The basic graphic software package provided is made accessible to Fortran programmers while circumventing some of the limitations of the language for graphics applications.

This work was also aimed at the possible exploitation of some scientific application areas, in particular the use of man-machine interaction in data-fitting problems as a starting point for many more applications. Data-fitting is an area in which a batch processing environment is especially frustrating, because intermediate results and graphs of results provide additional insight and are needed in order to proceed intelligently.

The work reported in this thesis is organized into two parts. Part I presents a review study of the existing graphics facilities in terms of hardware and software (Chapter 2), interactive input techniques (Chapter 3) and the organization of graphics output processes and application data structures (Chapter 4). Finally, in Part I, a full account is presented concerning the development and implementation of the basic graphics software package LIGHT. Part II contains a detailed discussion of the implementation of several application programs which employ the basic graphics software developed in Part I. The applications cover the following problem areas:

(1) Interpolatory Data Fitting - IDF

(2) Interactive Contour Tracing - ICT

(3) Triangular Mesh Generation - TMG

Finally, full program listings of the basic software and the application modules are given in the Appendices accompanying this thesis.

PART I

GRAPHICS SYSTEM

CHAPTER 2

A REVIEW OF FACILITIES FOR COMPUTER GRAPHICS

1. INTRODUCTION

A man-computer graphics system consists of the man (liveware), the hardware and the programming (software). In general, the hardware consists of a computer, a display driven by the computer, and input devices which permit the user to give instructions and data to the computer based on his evaluation of the picture (information) displayed on the screen. The programming consists of the basic system programs (e.g. UNIX system) and applications programs which permit the solution of specific problems. There is also the device interface software which enables the user's program to communicate with the display (e.g. LIGHT package, see Chapter 5).

It is immediately apparent that the hardware-programming system must be considered in its entirety, since all portions interact to provide the over all capability.

2. GRAPHICS HARDWARE

2.1 Hardcopy Output Devices

In an interactive terminal configuration some means of hardcopy production is usually useful and often essential. Here, we briefly mention the different types available:-

(i) Incremental Plotter

This is the most common graphical output hardcopy device. There are now two main types of incremental plotter: drum and flatbed. Paper on a drum plotter is supplied in a roll which is mounted at the rear of the device. The paper then passes over the drum and is either allowed to hang freely or, more usually, is wound onto a take-up spool. A pen attachment is mounted above the drum and can move across the width of the paper. The more advanced drum plotters have multi-pen arrangements which are useful for plotting in several colours.

Flatbed plotters are available in the form of a horizontal table, or may be mounted at a fixed angle to the horizontal. The pen is suspended in a gantry arrangement which moves over the surface of the paper.

All digital plotting is accomplished by drawing straight lines in certain fixed directions. In the case of the drum plotter for example, movement of the drum alone can cause a line to be drawn in the direction of rotation of the drum, while movement of the pen alone can cause a line to be drawn at right angles to the direction of rotation of the drum.

In order to reduce the demand on the processor time of the central computer caused by excessive data throughput, off-line plotting systems have been developed. A controller processes a simplified form of the plotter input (which has been stored on magnetic tape or disc), into the incremental steps required by the plotter. The main drawback of the incremental plotters is their low plotting speeds.

(ii) Electrostatic Plotter

This is an alternative to incremental plotters. Here the output is not in the form of line vectors, but in the form of small dots. A dot may be 'drawn' only on a grid point and the distance between grid points is of the same order as the length of an incremental plotter step. The general arrangement of this plotter consists of dielectric coated plotting paper which passes over a writing head containing minute conducting styli which are selectively able to deposit electrostatic charges on the paper. The positions of the charge deposits become visible on passing through a liquid toner suspension. Speeds of up to 1200 lines per minute are obtainable - comparable to the speed of a lineprinter for a small or medium range computer configuration. It is possible with the addition of an interface to a storage tube, to produce hardcopy reproduction of the screen image on the plotter.

(iii) Microfilm Recorder

This is another alternative to the incremental plotter but is commercially expensive. However, its speed makes it an economic proposition if vast quantities of output need to be produced.

The most powerful microfilm recorder currently available is the FR80 made by Information International Incorporated. Basically the FR80 consists of an III15 computer which accepts data stored on magnetic tape and displays the result on a high precision CRT. Various cameras may be used to record the information displayed in hardcopy form, on 35mm film, 16mm film or microfiche form.

2.2 Graphical Displays

Basically there are four types of graphical displays:-

(i) Direct view storage tube

(ii) Indirect view storage tube

(iii) Beam driven refresh display

(iv) Raster display

All the above types of displays rely on CRT-related technology for electron beam generation and control; details of this may be found in [1]. An electron beam, having struck a phosphor coated screen, will cause the phosophor to glow for a short period, the intensity level falling as time increases. The length of time that the glow remains visible is termed the 'persistence' of the phosphor. In order to maintain the image on the screen for longer periods it must be redrawn and it is in the redrawing techniques that the differences between the above displays are to be found.

Storage tubes are so named because a representation of the screen image is stored in the form of an electric charge pattern which is continuously copied to the screen. The storage device is actually part of the display terminal. Refresh and raster display on the other hand need some form of external storage device (display file/buffer) from which to obtain the data to redraw the image.

Picture drawing on all types of display is accomplished by referencing a notional grid, precisely as is done on a plotter. Each grid point is addressable, though some may in effect lie outside the screen dimensions and are therefore never visible. The more visible points there are, the more accurate is the screen image, provided of course that the resolution obtainable by the focusing mechanism of the display is sufficient to enable adjacent points to be distinguishable by eye. Each type of display is now described more fully.

(i) Direct view storage tube (e.g. Tektronix 4010)

The writing beam is not focused directly into the screen, but onto a grid of fine wire situated immediately behind the screen and coated with dielectric, on which a pattern of charge is deposited and retained. This pattern is then effectively copied from the storage tube to the screen by a continuous flood of slow moving electrons. Selective erasure i.e. erasure of part of the image without erasure of the whole, is virtually impossible using the storage tube technique. Complete erasure of the screen is accomplished by applying a positive pulse of about half a second duration to the storage grid. This has the side effect of producing a visible flash across the entire screen.

(ii) Indirect view storage tube (e.g. Princeton 801)

This is a variation on the previous storage technique and allows selective erasure. The image change is stored on a circular target which is about an inch in diameter and composed of a wafer of silicon and silicon oxide. The target is continuously scanned in a raster fashion by an electron beam to give the screen image. To remove part of the image, the target must be retraced in a special mode along those vectors not required.

(iii) Beam driven refresh display (e.g. DEC GT42)

The alternative method of maintaining an image on the screen is to redraw the picture before it has faded. The persistence of the phosphor governs the rate at which the screen must be refreshed in order to avoid flicker. For most phosphors used in CRT's designed for interactive graphics, flicker will be avoided if the picture is refreshed 30 or more times per second. Maintaining this rate has two consequences:-

(a) high speed circuitry must be used to convert the digital signal received from the computer to the analogue signals required by the CRT.

(b) the data which describes the picture to be displayed must be readily accessible.

The beam driven refresh display is a vector device which obtains its data for drawing the picture from a 'display file'. The display file is composed of 'display instructions', so named because their form bears a close resemblance to machine instructions. The set of instructions which may be used is called the 'display instruction set', the size of the set depending on the hardware features of the display. For example, the DEC GT42 instruction set basically comprises

- set graphic mode
- Jump
- NO-OP
- Load status Register A
- Load status Register B

together with six data word formats that accompany the instructions:-

- character data format
- short vector mode
- long vector mode
- point data mode
- graph plot X(Y) mode
- relative point mode

For further details see reference [2].

The instructions in the display file are executed autonomously by the 'display processor unit' (DPU), and one such execution of all instructions in the display file is termed a 'refresh cycle'. The final instruction in the display file should be one of the two types:-

(a) an instruction to halt the DPU

(b) an instruction to direct the DPU to the first instruction of the display file.

By adopting method (a), the DPU may be synchronised to the internal clock of the host computer; a flag to restart the DPU is set when a timing signal (or clock interrupt) is received. Thus at the end of the refresh cycle there will be a pause before the next cycle is started, the refresh cycle plus the pause being termed a 'frame'. The number of frames per second is known as the 'refresh rate' and by the above synchronisation technique the refresh rate may be maintained at a constant value provided that the relevant clock interrupt is not received before the DPU has finished its refresh cycle. If this happens then the refresh rate must be lowered with the possibility of picture flicker resulting. By adopting method (b), the refresh rate is not constant but varies according to the number of instructions in the display file. This procedure is not normally adopted as problems are caused if the display file manipulation is attempted while the DPU is executing instructions.

(iv) Raster Display

The sequential scan refresh display, or raster display, can use a standard domestic television monitor for its display screen. The screen elements, corresponding to grid points, are sequentially scanned, a double scan of alternate lines being adopted to reduce flicker. Data is stored for each element in the order required by the raster scan.

Finally, Table 2.1 presents a comparison of the various graphics display types discussed above, and Table 2.2 shows the hardware feature of the Tekronix 4010, DEC GT42 and Vector General. Table 2.3 presents an approximate cost of these hardware devices, circa 1976.

FEATURE	DISPLAY TYPE			
	(i) Direct view storage tube	(ii) Indirect view storage tube	(iii) Beam-driven refresh display	(iv) Raster scan refresh display
Picture drawing technique	Wire grid storage copied to screen	Raster scan of silicon target	Continuous retrace of display file	Raster scan from storage device or memory
Picture generation speed	Dependent on line speed	Dependent on line speed	Fast	Fast
Picture quality	Good	Poor	Good	Poor (at present)
Minimum line speed	110 Baud	110 Baud	1 M Baud	30 M Baud
Local memory requirements	None	None	10 K bytes	100 K bytes
Selective erasure	No	Yes	Yes	Yes
Time for complete erasures	500 ms	400 ms	20 ms	30 ms
Capacity limitation	Resolution	Resolution	Beam driven speed	Resolution
Intensity level	One only	Several	Several	Several

TABLE 2.1: Comparison of Display Types

Feature		Storage Tube	Refresh Display	
		Tektronix 4010	GT42	VG-3D3I
Lines:	solid	Yes	Yes	Yes
	long dash	No	Yes	No
	short dash	No	Yes	Yes
	Dotted	No	No	Yes
	chained	No	Yes	Yes
Blinking:		No	Yes	Yes
Characters:	normal set	64	127	192
	rotation	No	No	Yes
	italics	No	Yes	by rotation
Intensity levels:		1	8	32
Intensity modulations:		No	No	Yes
Transformation:	rotation	No	No	3D
	translation	No	No	3D
	scaling	No	No	3D
Windowing:		No	No	optional
Arc circle generator:		No	No	optional

TABLE 2.2: Hardware Features

DRUM PLOTTERS:	
CIL Midas (34 cm)	£2050
CIL Economist-1 (92 cm)	£3750
FLATBED PLOTTERS:	
Ferranti	from £27000
ELECTROSTATIC PLOTTERS:	
Sintrom 800 (8.5 inch) plotter	£4300
Sintrom 800A (8.5 inch) plotter/printer	£4600
MICROFILM PLOTTERS:	
Ferranti	from £40,000
III FR80	\$210,000
DIRECT-VIEW STORAGE TUBES:	
Tektronix 4010	from £3300
Tektronix 4010-1(hard copy unit compatible)	from £3700
Tektronix 4051	from £5300
INDIRECT-VIEW STORAGE TUBES:	
Princeton 801	from £8000
REFRESH DISPLAYS:	
GT42	£12000
IMLAC ODS-4/DINO system	£15410
Vector General Series 3	from £27400
Vector General 3400	from £45500
CDC 777 Cybergrahpic terminal	£67100

TABLE 2.3: Approximate Costs of Hardware

2.3 Input Devices

For a display to be termed an interactive terminal it must have some means of permitting its user to input data. Without this facility the display is really an output only device. The various input devices currently available are:

- keyboard
- function buttons (switches)
- control dials
- joystick, tracker ball and mouse
- digitiser cursor
- tablet and stylus
- lightpen
- storage tube graphical cursor

More detail on these devices may be found in [1]. However, a detailed study of the associated programming techniques is presented in the next chapter.

2.4 Various Graphical Terminal Configurations

Numerous designs of graphics display terminal configurations are possible. This section presents some examples of these configurations based upon the tasks assigned to each part of the system. These tasks may be divided into the following categories as in [4]:

(a) Display file maintenance - (DFM) for refresh display only.

(b) Interactive demand servicing (IDS).

(c) Extensive numerical calculation (EC).

Examples of terminal configurations:-

(i) storage tube terminal

The nature of the storage tube makes it particularly suitable for use as a remote terminal with no local processor. With this configuration (Fig.2.1), there is no provision for any computation to be done at the terminal site. A hardcopy unit at the terminal site would enable plots of the screen to be obtained in about ten seconds; this is therefore very useful additional equipment for such a terminal.

(ii) storage tube terminal with a small processor

This configuration (Fig.2.2), is the obvious enhancement to (i) providing some processing power at the terminal site. With appropriate software, the processor is able to cope with a limited amount of interactive demand servicing. A terminal providing just this system in one package is the Tektronix 4051. In addition to the standard Tektronix storage tube, this has a microprocessing unit for programming in Basic, 8K-32K bytes of memory and a cartridge tape unit. The Fortran version is the Tektronix 4081.

An interface is available which allows some direct storage tubes (Tektronix 4010) to behave as a refresh display. The intensity of the writing beam is reduced to prevent charge storage and is focused onto the screen; this is called 'write through only'. However, the performance of operation in this mode can not compete with the more expensive refresh displays.

(iii) low cost refresh display having its own processor and core memory

Because of the need to refresh the screen image many times a second, some core memory is required at the terminal site (Fig.2.3.) to hold the display file (or equivalent). In addition, the display

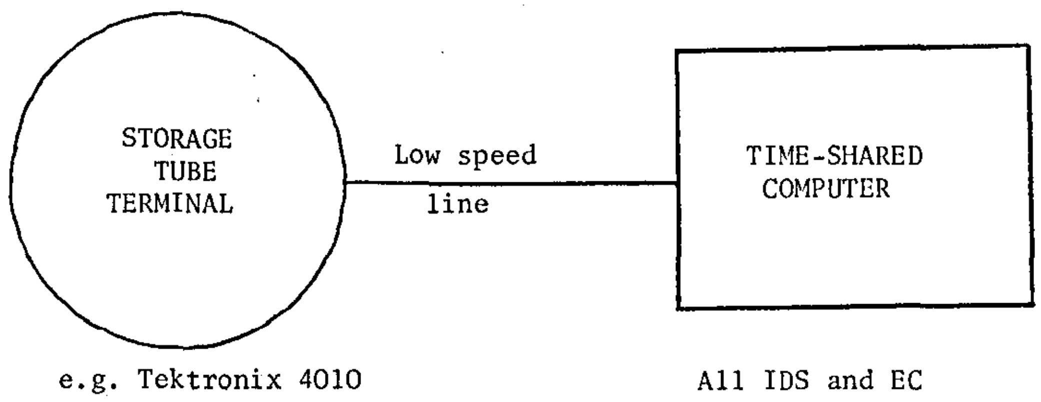


FIGURE 2.1: Storage Tube Configuration

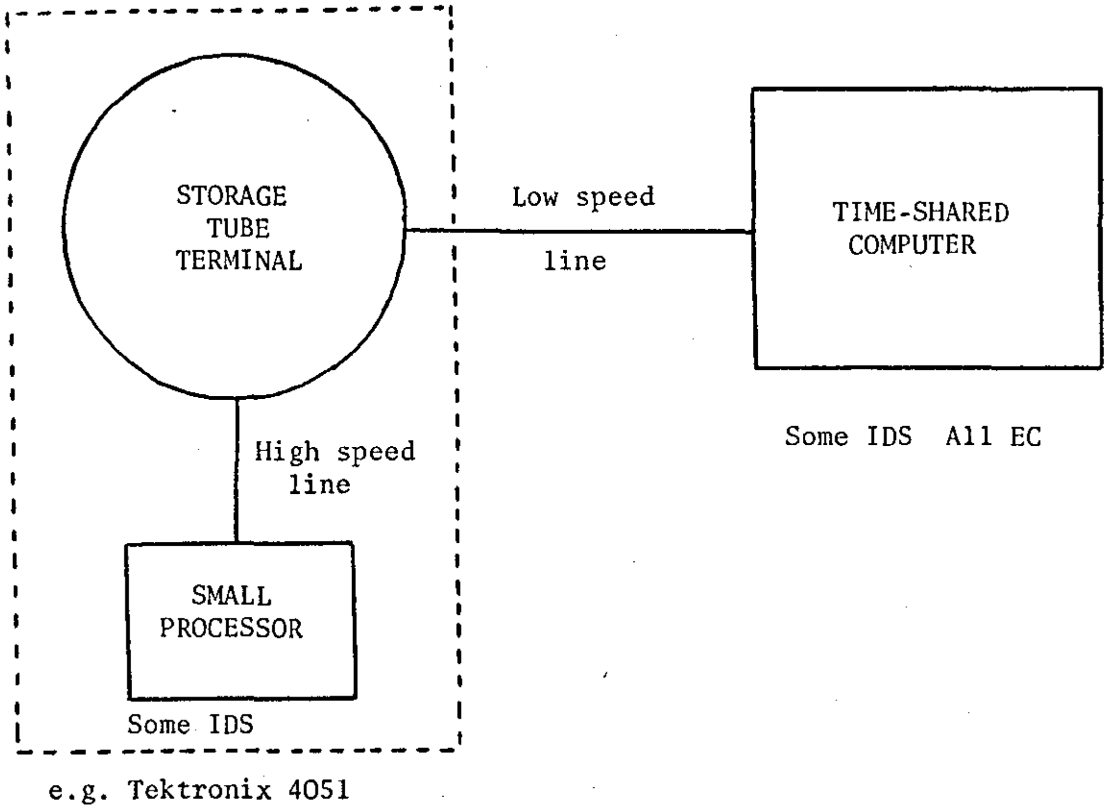


FIGURE 2.2: Storage Tube and Small Processor Configuration

file must be updated when necessary and this is best done by providing a local processor, which can also be used for interactive demand servicing. This type of terminal is supplied as one package e.g. the DEC-GT42.

(iv) high cost refresh display with its own processor and core storage directly interfaced to a mini-computer

This, (Fig.2.4), provides a multi-user environment (e.g. Vector General 3400- PDP 11/45). The display's core and processor hold and maintain the display file. The minicomputer can be dedicated to interactive demand servicing. The extensive calculations which the minicomputer could not reasonably cope with would be sent down the link to the large computer for execution in batch mode.

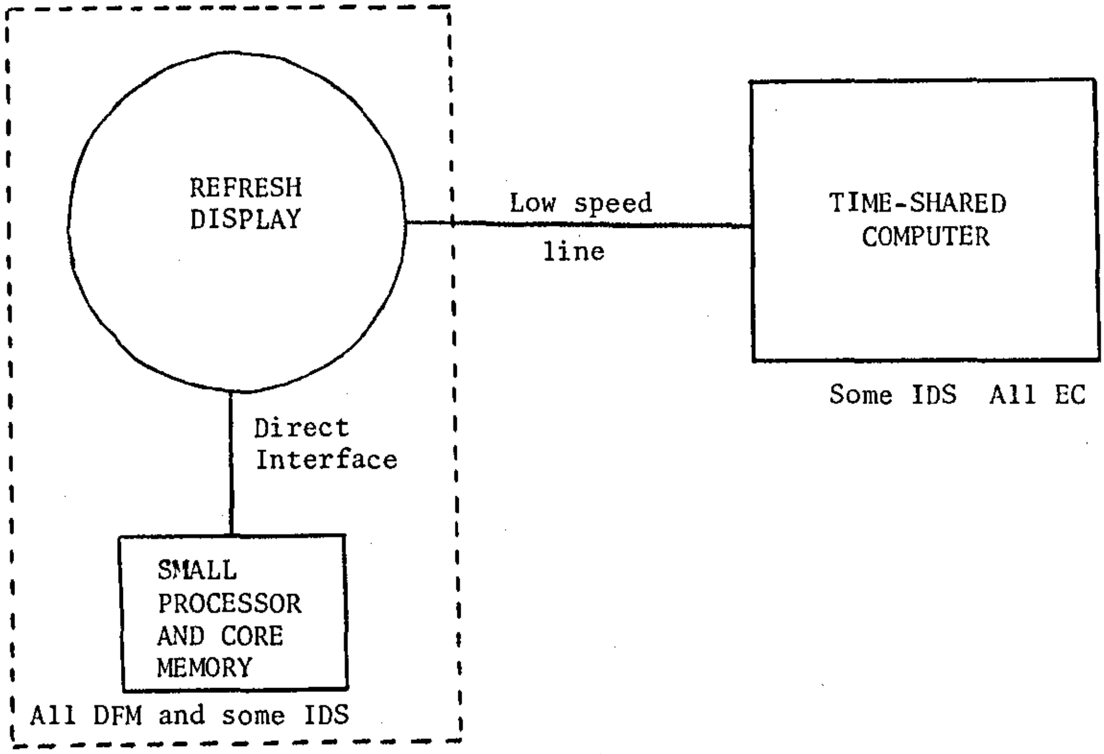
3. GRAPHICS SOFTWARE

3.1 Structure of Graphics Software

In general there are three main and distinct 'layers' of software [5] in any interactive graphical computer program:-

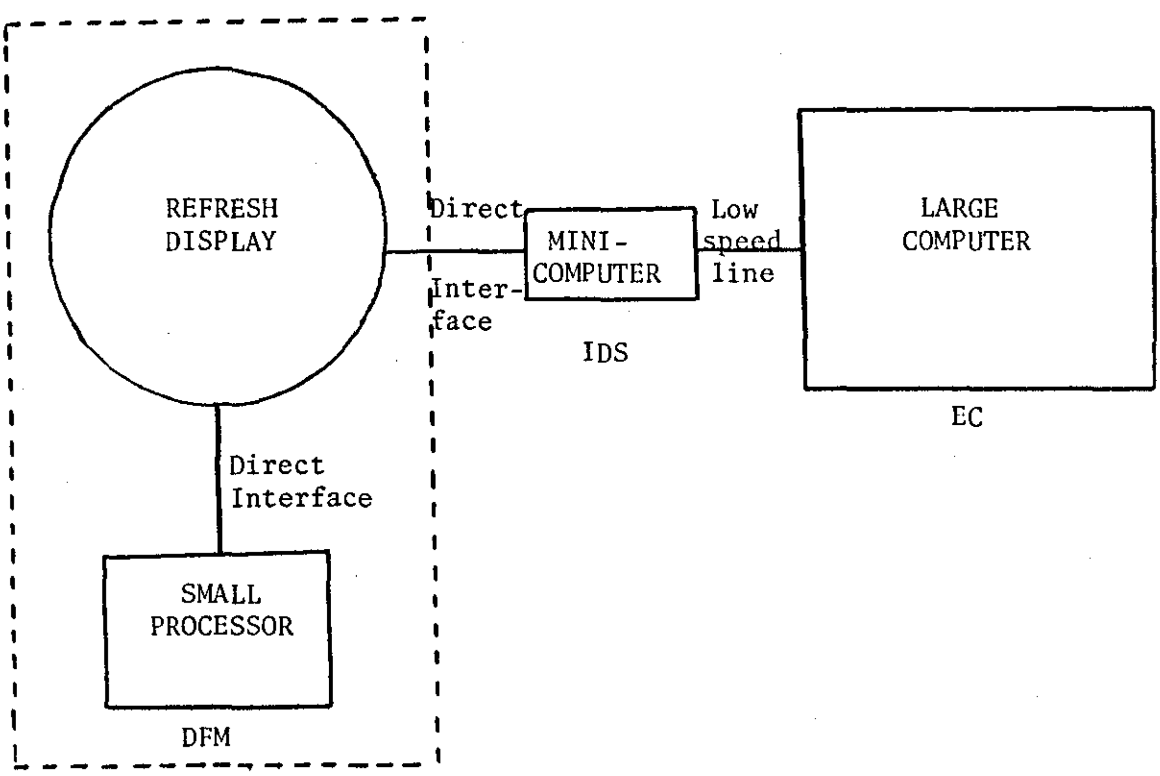
(i) the lowest level, that 'closest' to the hardware is called 'basic' software and consists largely, for a given application, of a basic graphics package with its associated device drivers and may include some simple data-structuring and file handling routines. Parts of this basic software will of necessity, be written in machine dependent code; transfer of an application system from one host computer to another or from one graphical device to another will thus affect this layer of software more than any other.

(ii) The middle layer is known as 'general purpose' as this is normally written with a whole range of application areas in mind.



e.g. DEC-GT42

FIGURE 2.3: Low Cost Refresh System Configuration



e.g. Vector General

FIGURE 2.4: High Cost Refresh System Configuration

Such general purpose routines are assembled from basic software building blocks. Examples, in this layer are routines for messages, menu display and removal, and routines for utility purposes such as clipping.

(iii) The top layer is the actual application program. This software is built on both of the bottom two layers. In other words, an application program will call basic routines directly or (usually) via the general purpose routines. To operate a given application program on a different computer, or a different graphical terminal or under a different operating system, few changes should be needed at this level in a well-designed, well-layered system.

A simplified schematic diagram of Fig.2.5 shows a typical software package and the user program. Here we are considering the simple configuration of a mainframe computer driving a graphical display terminal. At the highest level there is the user program which calls routines in the display software package. Those routines of the package which are user-callable are known as 'front-end' routines, and these in turn call other routines which are not user callable and are therefore termed 'back-end' routines. At the lowest level there is the program known as the 'device driver' whose function is:-

(a) translate simple instructions from backend into low level instructions (i.e. machine code) which will drive the graphical terminal.

(b) handle communication between the computer and display terminal by sending instructions and data and (in the case of an interactive terminal) by receiving and interpreting data.

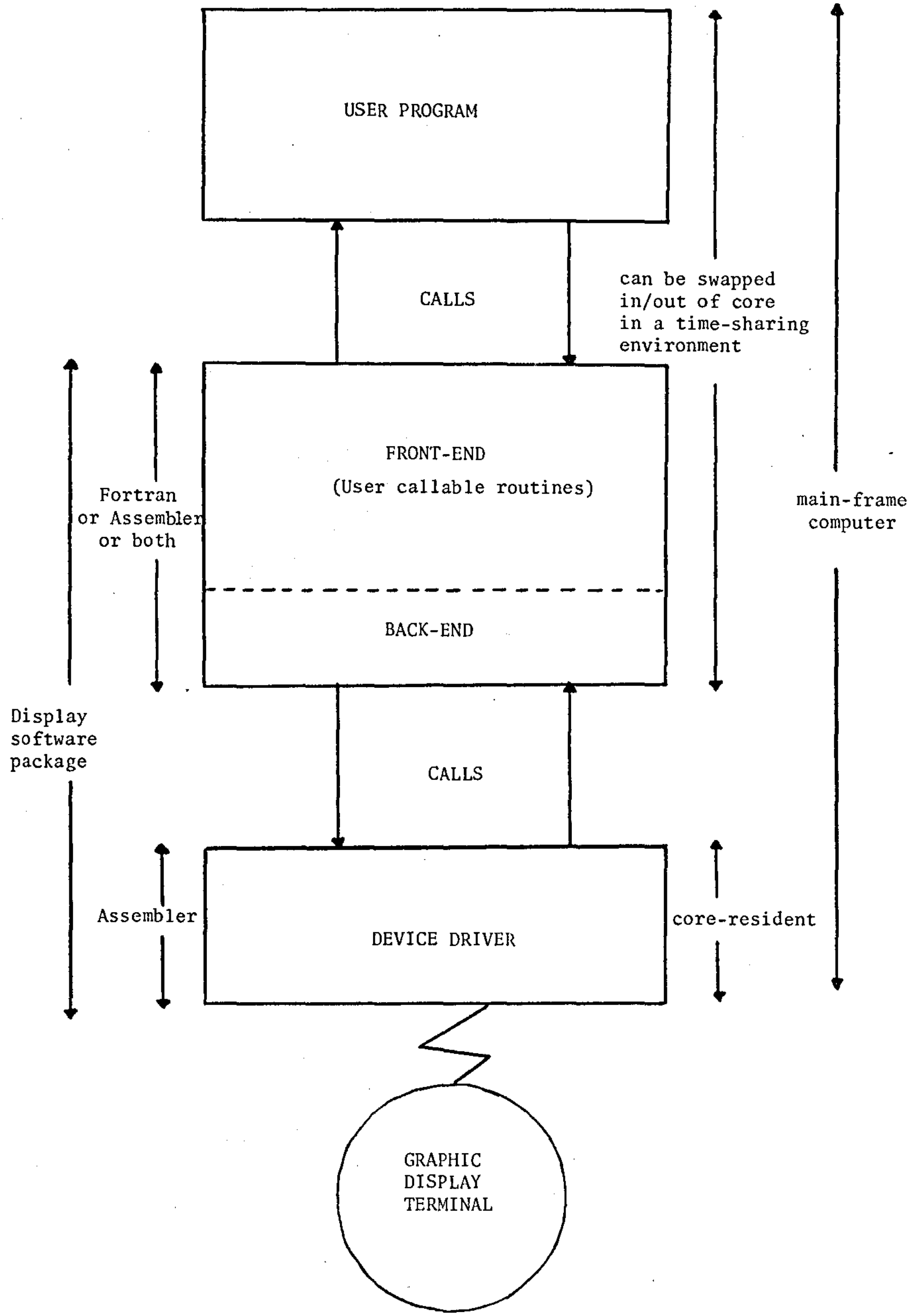


FIGURE 2.5: Simplified Schematic Diagram of a Typical Graphics Software

The device driver software communicates directly with the input/ output hardware of the computer and is of necessity written in assembly language.

3.2 Features of a Typical Graphics Software Package

3.2.1 Initialising the graphics package

The graphics software package must provide facilities for:

(i) device nomination

(ii) resetting to default values.

(i) Device nomination

Before any drawing can be made, the device has to be prepared for drawing or character generation. If the package was designed to operate under a 'closed' system, i.e. a dedicated computer and one screen, the device nomination may not be an explicit part of the package; switching on the system will prepare the device for calls from the graphics package. Nomination may involve clearing the device's display file (and hence clearing the screen for a refresh display) at the start of a run, but this is usually under the control of a further subroutine.

(ii) Resetting

All graphics packages will have various defaults to simplify initialisation, the number of these depending on the range of facilities offered by the package. For plotting packages, these are default axes, grid marks, curve resolution, maximum size of allowable plot etc. For display-oriented packages there will also be default settings for picture sensitivity, character size, brightness, window size etc. In some existing packages such as GINO-F [3] and GPGS [6] all default values are set in initialisation, while other packages require, in addition to the setting of default values, various preparatory calls before drawing can commence. For example:

(a) Plotting package:

DISSPLA [7] requires a title, page border, definition of physical origin and subplot area, plus axes types and units.

(b) Other packages:

PICTURE SYSTEM [8] requires the refresh rate and the time interval between updating the display file to be specified as parameters to the initialising routine.

3.2.2 Picture generation

Picture generation routines form a large part of any graphics package. These routines are used to drive indirectly the drawing mechanism of the device through either assembly coded routines or structured display file. These routines form part of the 'front-end' of the graphics package. Calls to routines are interpreted by the graphics package into a series of calls to device driver routines which the application programmer is not allowed to (nor would wish to) access. In the case of a refresh device, the device driver may be a separately running program interpreting the structure of the current display file and regularly refreshing the screen with the current picture.

We can divide picture generation calls into three classes:

(i) Basic drawing routines

These are routines which translate directly into device driver calls, the most common of which is straight line drawing. Graphical devices normally allow such drawing action as MOVE,LINE, DOT and character with the addition of different line types, intensities, pen sensitivity etc.

(ii) Multiple drawing routines

Multiple drawing routines are those routines which are a kind of 'shorthand' for common objects, and replace a whole series of calls to the more basic picture generation routines. These are middle level routines provided by the package to simplify the work of the programmer at the expense of the size of the package.

(iii) Complex drawing routines

These require much more preparation than multiple drawing routines either by the package or by the programmer. An example is CUPID [9] which has one complex drawing routine which plots a graph or histogram, and the axes with annotation. Every parameter (except the data file name) has a default value but can be altered by calls prior to PLOT.

Graphics packages are designed with different users in mind and for this reason provide fewer complex features and greater control over basic line drawing. For example GINO-F and PICTURE SYSTEM provide many more basic drawing routines than complex routines. On the other hand DISSPLA and PLOT-10 [10] are designed for plotting of graphical or other data under various systems of axes and does not demand the programmer to write code to draw the axes, tick marks, etc. Instead they provide subroutine calls to simplify the programming when the drawing follows a fixed format.

3.2.3 Picture administration

This involves the segmentation of pictures, the setting of picture parameters and picture manipulation.

(i) Picture Segmentation

This is the dividing up of a picture into segments and this is only useful when the graphics package stores the picture represented by lines and characters in some form of file. The segments may be required for identification or to be used for building up complex pictures in a similar manner to the usage of subroutines in Fortran. Segments are used to obviate the necessity to regenerate the complete picture when only one part is changed. The normal usage of this form of segmentation is with the refresh device in which a lightpen is used to identify some part of the whole picture. But it can also be used with a storage-tube device where identification is made with the cursor (requiring a search of the display file to locate the picture from the coordinates). GINO-F offers the facility of storing a picture segment either on discs or in local arrays to be recalled when required.

(ii) Picture Parameters

These are used by graphics packages to give sections of code hardware options such as picture intensity, lightpen sensitivity etc. When picture segmentation is provided it is usual for each segment to have a header containing the segment name (or number) together with various picture parameters.

Packages that provide for altering these parameters contain a varying number of routines. For example, GINO-F has a routine for each type of parameter while DISSPLA provides a different routine for each parameter setting.

(iii) Picture Manipulation

This is concerned with the display file and thus applies to display file based packages e.g. Picture Book [11]. Manipulation usually involves deleting and copying of pictures, changing the position of a picture, etc.

3.2.4 Transformation

Transformation facilities provided by various packages could be divided into three categories:

(i) Linear Transformation

(ii) Windowing Transformation

(iii) Perspective Projection.

(i) Linear Transformation

The four primitive transformations that are applied to an object in two or three-dimension are translation, scaling, rotation and shear. These transformations are applied to the coordinates of all points and end-points of lines forming the object. The basis of these transformations is the 4x4 transformation matrix using homogeneous coordinates. A typical package would usually provide routines to update the current transformation matrix, using these primitive transformations and producing more complex transformations.

(ii) Windowing Transformation

This is concerned with the mapping of the picture (object) coordinates into the physical screen coordinates. One could assume if all picture drawing would remain within the limit of the screen, a large amount of coordinate checking could be removed from many graphics packages. But when a linear transformation is applied, for example, on an existing picture on the screen, it is very likely that the transformed picture will have some of its lines transformed off the screen. Hence, there is a need to clip the lines to a rectangular area or a 'window' and then project the clipped picture on to the display 'viewport'. Thus, a windowing transformation has two phases, firstly to determine if a line exceeds a window boundary, and secondly to calculate the cut-off point. Routines such as clipping and viewing are usually provided to facilitate these operations. Both would define how much of the picture should be visible and where the visible portion should be placed on the screen. Finally there is often the necessity for zooming into a detailed part of a picture or placing several viewports on one screen.

(iii) Perspective Projection

The combination of a perspective transformation with a projection is often called a perspective projection. It represents a transformation from three space to two space. Some packages, for example GINO-F and PICTURE SYSTEM, provide the programmer with transformation routines such as these, for greater control of viewing of three-dimensional objects projected onto the plane of the display screen.

Chapter 3

INTERACTIVE INPUT AND RELATED PROGRAMMING METHODOLOGY

1. INTRODUCTION

The organization of interactive input is probably the most important factor in the design of an interactive graphics program. Well planned man-machine communication is vital if the program is to be successfully used.

In this chapter we develop and apply the philosophies underlying graphical input techniques [12]. Initially, an attempt is made to formulate conceptually the mechanics of man-machine interaction and the different processes involved in the communication process. Then, the practicality of these concepts is exploited in the field of software techniques for interactive input, so that a better understanding of the nature of man-machine interaction is obtained and subsequently used in the development and design of interactive computer graphics systems. These techniques would provide facilities in the basic graphics packages for application programmers or could be used directly in the application programs. This area has received considerable attention in various contexts [13],[14].

Finally, a number of common input devices are examined separately, and the different programming techniques that may be used with such interactive input devices are highlighted. Most of these techniques have been incorporated in the design of the interpolatory data fitting packages in Chapter 6.

2. THE MAN-MACHINE DIALOGUE

The combination of man and machine can leave design decisions to man and calculation to the machine (computer). The main distinction between 'design decisions' and 'calculation' is that the latter are readily programmed and the former are not, being a function of experience etc. The sharing of the work seems useful in many areas of design and is efficiently achieved when the man-machine interface is well-defined.

Ross [15] pointed out that the fundamental mechanics of communication involve the transfer of 'atomic' components such as characters or numbers. Therefore to convey an idea or a meaning from one body to another requires (1) a process of analysis through semantic, syntactic and lexical phases to define the stream of atomic components, (2) the transfer of these components through a suitable medium, and (3) a process of synthesis to reassemble the idea or meaning through lexical, syntactic and semantic phases. These phases retain their identity even though the rate of interaction may vary. In a telephone conversation where the atomic components are the sound syllables, the sentences are short and the rate of interaction is relatively high. Telegrams typically contain a single whole message or request, a letter typically consists of a series of requests, or pieces of information. Batch input to a computer corresponds to a letter. A Teletype input command compares to a telegram and 'interactive graphics' compares to telephonic or face to face dialogue. Therefore, conceptually we may recognise that there are three distinct levels at which interaction could take place in an interactive graphics system:

(i) Lexical Level:- corresponds to interactive facilities in the basic graphics package.

(ii) Syntactic Level:- corresponds to command languages in general.

(iii) Semantic Level:- corresponds typically to application programs.

Each level of processing of computer input and output corresponds to processes of synthesis and analysis respectively. Corresponding processes of analysis and synthesis occur in the brain of the operator. Fig.3.1 illustrates these ideas, and Fig.3.2 relates these to the different levels of program code used in a man-machine dialogue.

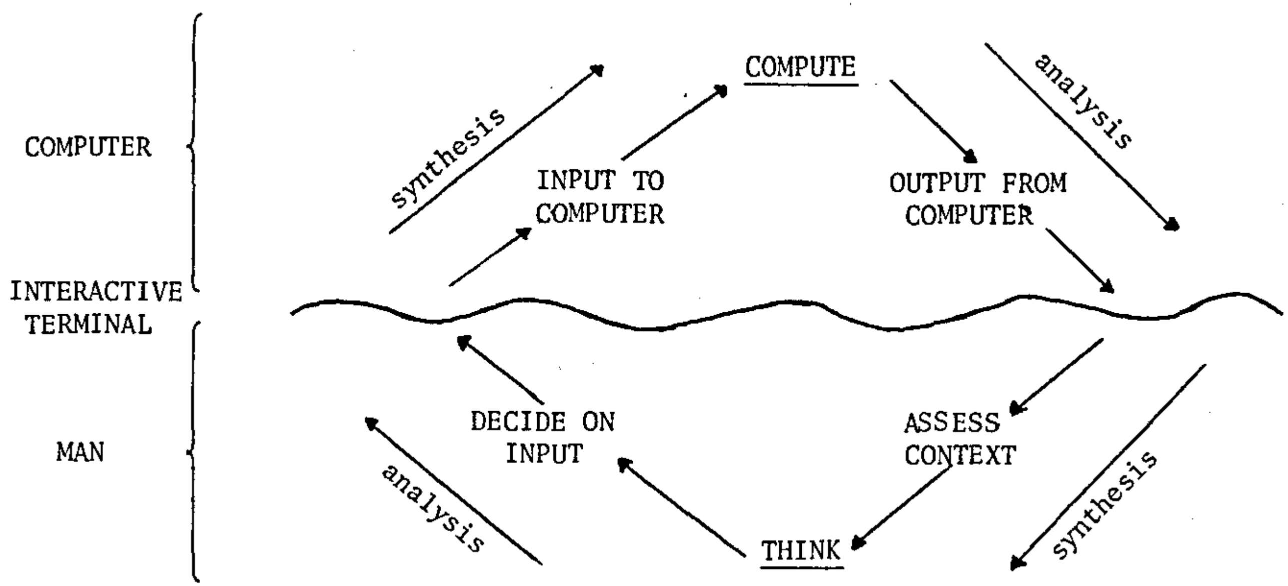


FIGURE 3.1: Man-Computer Communication Processes

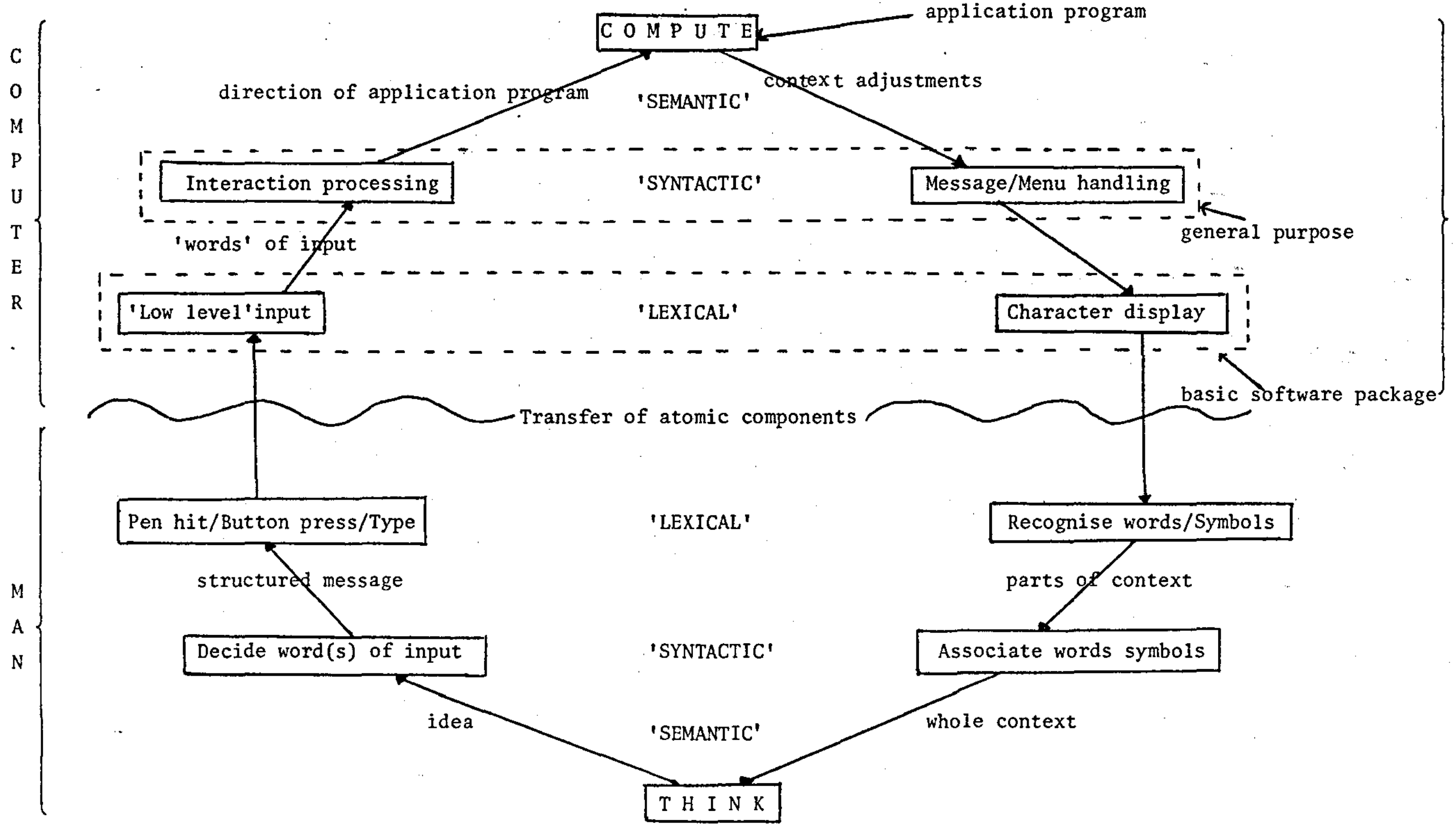


FIGURE 3.2: Man-Computer Communication Processes and Program Code

3. INTERACTIVE INPUT SOFTWARE

Interactive graphics adds new dimensions to the more general conceptual framework of man-machine communication. A variety of schemes for interactive graphics are available today. The following discussion describes the basic principles and requirements of interactive input.

3.1 Layers of Input Software

Figure 3.3 shows how input software can be divided conceptually into three distinct layers corresponding to the three communication levels mentioned previously.

Each annulus represents a piece of code and each circle represents an interface. The user (operator) is considered to be at the centre acting on the 'physical input device'. Input involves action by the user, terminal and the interactive program.

(i) User - a physical operation, such as the press of a 'key' on a keyboard, or placing of a lightpen over a picture element.

(ii) Terminal - the 'prompting' of the operator prior to his physical action; the delivery of all the data of input operations to the program, e.g. a character, a picture part plus x,y coordinate; and 'echoing' the input data subsequently.

(iii) Interactive program - the organization of prompting, echoing and the processing of the input received.

3.2 Interactive Input Facilities

Consider a program which needs some data value at a point from the user before continuing execution. For example:

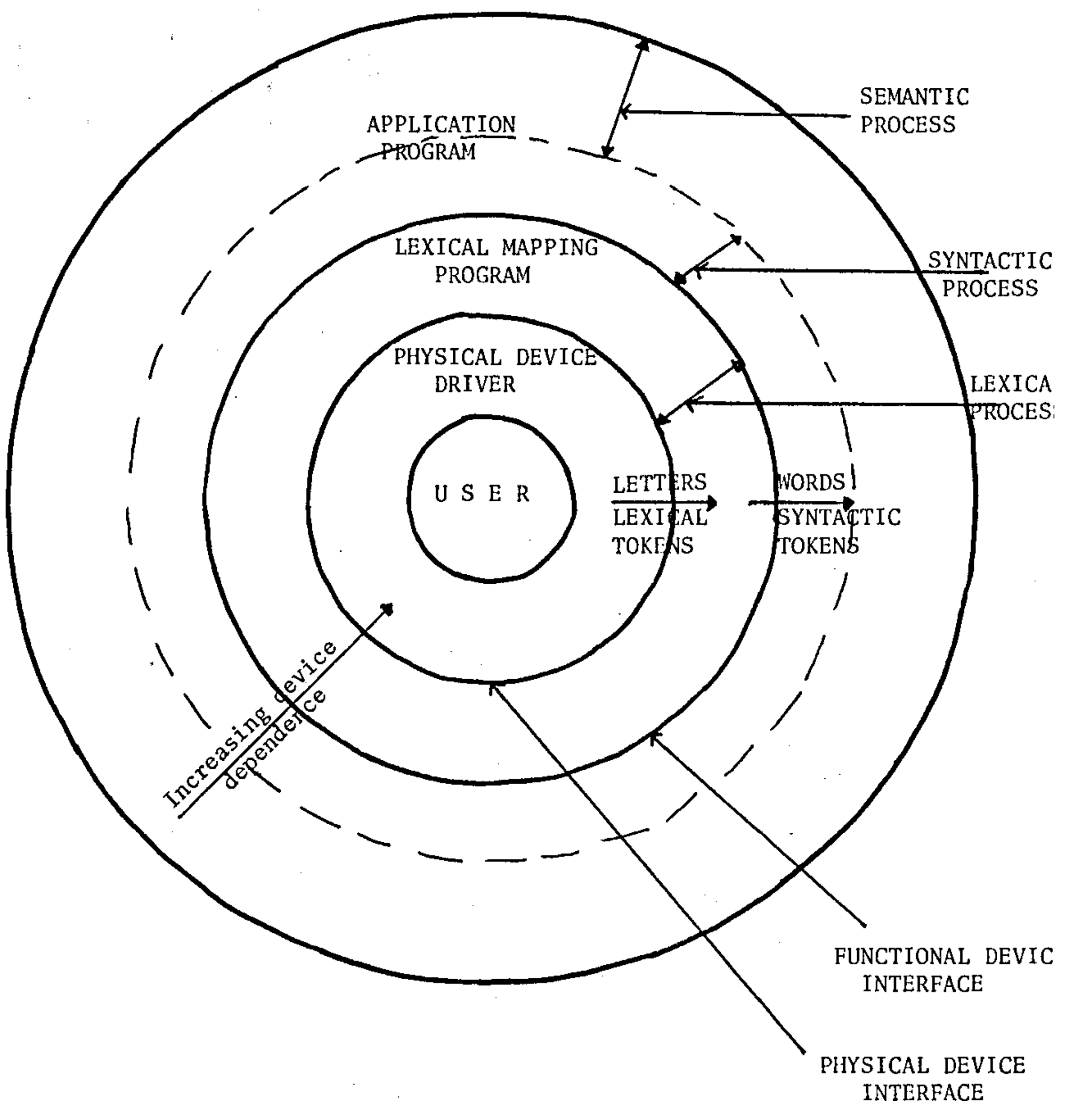


FIGURE 3.3: Interfaces and Layers of Input Software

1. characters
2. picture segment number
3. x,y coordinate pairs

Input such as these are considered to be delivered by 'functional input devices' which are conceptually more convenient for programmers. The actual input data (such as a picture segment or coordinate pair) are placed in an agreed format in a global or common area (e.g. COMMON in Fortran). Ideally functional input devices are characterised by having:

(i) an identifying number such as 1,2,3,.....etc.

(ii) a value or set of values generated by user action, with an associated location or locations in the 'common' block where the value(s) would be placed.

(iii) an associated physical device or devices such as keyboard key, lightpen, or cross-hair cursor, from which the input actually arrives; including a 'trigger' which actually causes the dispatch of the input data.

(iv) a prompt to the user to announce the program's readiness for functional input.

(v) an echo to the user to announce the recognition of his input.

Therefore, from these characteristics, one may conclude that interactive input programming facilities should include the following actions:-

(a) attach physical devices to functional devices and specify triggers;

(b) activate the required functional input devices;

(c) broadcast the various prompts associated with (b);

(d) send an echo to the user to confirm his input;

(e) request and wait for an event;

(f) read data from input devices.

These facilities have been variously combined in basic graphics packages into a set of high level procedures. Some facilities are unavaiable in some packages, particular activities having been selected and fixed into the low level code of the device driver or even incorporated into the hardware of the physical input devices themselves (e.g. Tektronix 4010 cross-hairs cursor).

(a) attachment of physical devices to functional devices and specification of 'triggers':

Attachment needs to be largely by default. For example:-

Type	Functional Devices	Physical Devices	Trigger
1	Single character	Key on keyboard	Key itself
2	Picture segment number	Lightpen	Sensing light
3	x,y pair	Cross-hair cursor	Pressing a key

The implementation of attachments requires 'mapping', which is usually performed within device drivers, e.g. within a keyboard driver, keys are mapped to characters. Mapping however may occur in a higher level code, e.g. keyboard driver characters could be mapped into integers, reals and valid character strings, which have been defined within the driver or in data which is fed to it via an application program. This mapping can be seen to be a process of lexical analysis. Physical devices produce 'atomic components' of input, or 'lexical tokens' (see Fig.3.3), or single characters or single picture segment numbers. The mapping programs convert this stream of lexical tokens into syntactic tokens or words. Each operable physical device must be considered as an indivisible entity. This notion is important if we are to design high performance man-machine operations into an application, and manage the human factors properly. This is by no means always possible with currently manufacturerd hardware and available basic graphics packages. For example, a light pen is not necessarily just one physical device as many assume; it is usually two - a light pen sensor plus some kind of switch - each supplying an atomic component of input and each being separately attachable or specifiable as triggers. Mapping programs may be local to an installation or an application area. They may involve trivial one-to-one mappings such as keyboard keys to characters; or one-to-many such as keyboard keys to sets of characters, x,y pairs; or many-to-one as shown below:-

	Physical Devices	Functional Devices
1:1	Keyboard keys	Characters
1:many	Keyboard keys	Characters x,y pair Function no.
many:1	Keyboard keys Lightpen Tablet scribe	Characters

Specification of triggers is not usually provided for by basic graphics packages. It is the kind of thing that has usually been seen as a characteristic of the physical device and often brought to the functional level without option. For example, the cursor input routine in LIGHT (see Chapter 5) is a direct mapping of thumbwheels to an x,y pair with a particular trigger pre-specified.

(b) Activation of functional devices:

Activation of a functional input device will, through the physical attachments, activate a set of physical devices. These physical devices should then be highlighted in some way for the user.

(c) and (d) Broadcast of prompts and echoes:

Bleeps and Blinks from an interactive terminal are an essential part of effective communication. As skill develops, direct response to prompts become less observable, indeed anticipation becomes more and more noticeable with experienced users. However, the right bleeps and blinks will then become an integral part of the continued success of the dialogue. Referring back to (b) above the activation of the physical device may be highlighted by prompts such as:

- a pair of cross-hairs on the display screen
- a bell on a keyboard

The operation of a device may be confirmed by echoes such as a small marker on the display screen .... etc.

Successful operation of physical devices is recognisable by the terminal user on a separate level from the successful acceptance of the input by the software at the functional level. The feel of the terminal device is an important aspect of user acceptability apart from the confidence that the program is operating correctly.

(e) Request and wait for event:

It should be noted that an event as far as the user is concerned is physical, e.g. the pressing of a key, or the turn of the thumbwheel (on the Tektronix 4010). An event as far as the program is concerned, is all that happens between passing control over to the user and control being returned to the application program on an acceptable trigger.

(f) Read data from input devices:

Input data is placed in a common area which acts as a functional input device buffer.

3.3 Application Program and Interactive Input

The semantic process of interpreting the meaning of communication from the user is performed in the code of the application program. The syntactic processes of synthesising a sentence from words and analysing a sentence into words are often coded as an integral part of an application program. This usually takes the form of a 'command language'. Thus, the user of an interactive program must be provided with a set of commands by which to control the execution path of an application program. This approach induces modular program design and simplifies program overlay control. These commands must fulfil two requirements: they should control which processes are activated, and they should contain the data associated with these processes. Thus, it is generally advisable to define with precision the range of commands that the program will accept, and to define the form or syntax of each command. By doing so, we have defined a command language, the language 'spoken' by the user as he operates the interactive program. As he addresses the computer in this way, the computer addresses him, by means of displays and printed messages; thus a dialogue is maintained. Fig. 3.4 shows diagramatically the conceptual relationship between the command language and the application program modules. These commands may be presented on the display in the form of menus (tables), so that at any time the user can see what commands are available to him. He can then point at a specific command by the use of a device such as a lightpen or cross-hair cursor, and this is conveyed to the application program for interpretation. A menu may comprise one part which corresponds to the main modules of the application program, and a second part providing command options such as 'zoom'.

An important facility that is sometimes overlooked in the design of an interactive system, is a backup or recovery procedure. User mistakes as well as program errors can cause extraordinary damage to data files. Because repairs can be costly in user and computer time, it is often wise to provide a fail-safe option which enables the user to backup to a point prior to the accident.

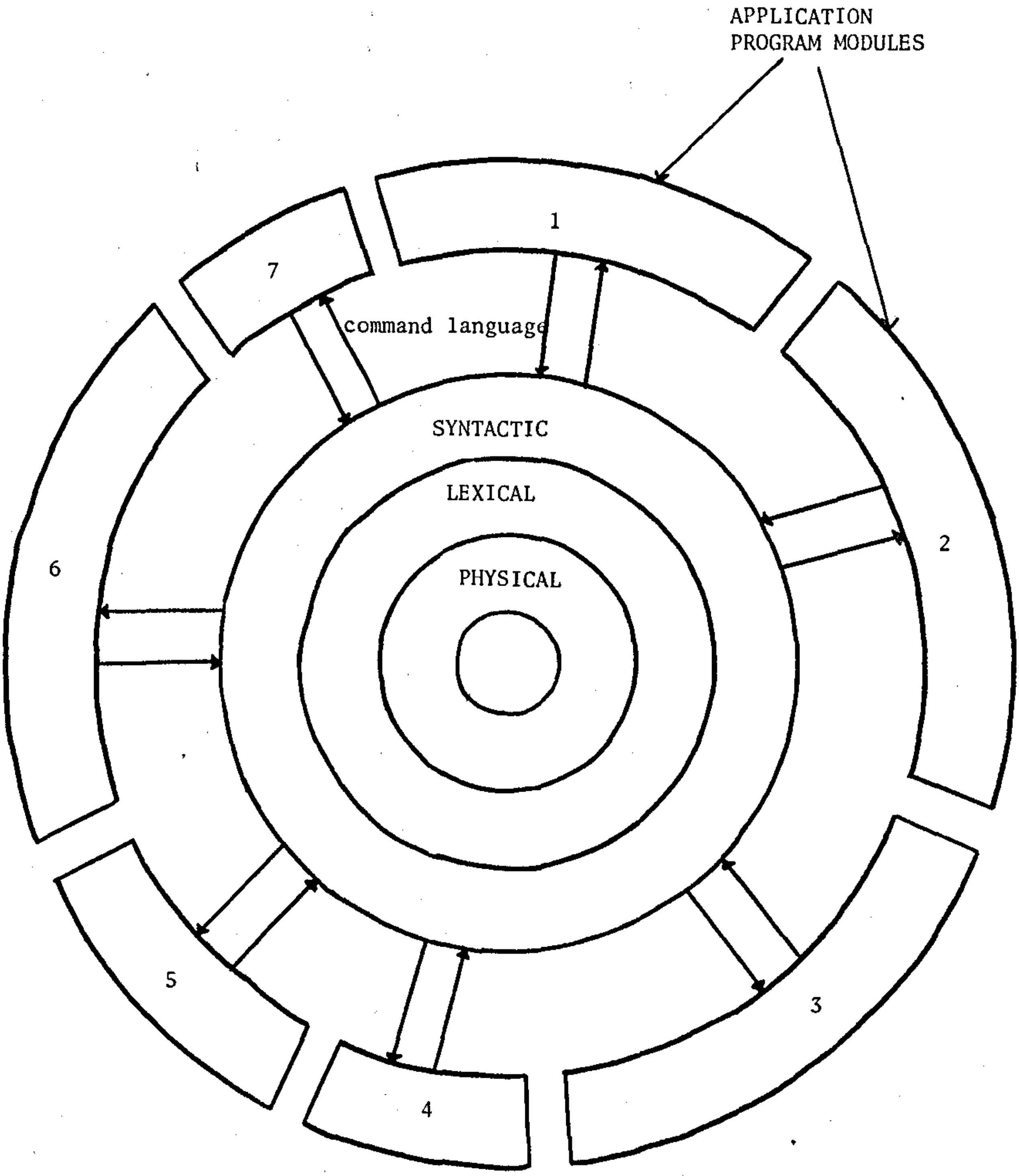


FIGURE 3.4: Relationship Between Command Language and Application Program Modules

4. INTERACTIVE INPUT DEVICES

The discussion here is concerned with some common input devices and the role that each may play in an interactive environment.

4.1 Keyboard

All interactive graphical terminals have a non-graphical input available through an ordinary keyboard similar to the teletype keyboard. All character sets may be expected to include a Fortran character set. Some terminals have keys which are purely 'local', and depression of these does not send any data to the device driver, e.g. the Tektronix 4010 'Page' key erases the current display from the screen. Several types of keyboard usage are possible:

(i) The most obvious use is probably for direct input in response to program input statement:

```
e.g.      READ(KEYBRD,10)FILNAM      ...... requires text
          READ(KEYBRD,20)X           ...... requires a numerical value
```

where KEYBRD is the channel number for the keyboard of the graphical terminal. Experience has shown that the keyboard is the best device for input of 'precise numerical values'. Techniques such as 'thermometer' (see virtual devices later), although conceptually more sophisticated, offer no particular advantages over the keyboard when the required input is in the form of known and precise values. Keyboard input is also used to good effect to input any 'character information' which cannot be predetermined - examples here are file names or headings for plotted output. It is usual to precede the input by an output message which tells the user that the machine is now ready for his input and advises him on the nature of input that is expected.

(ii) Another direct use of keyboard input is as a means of identifying picture parts on the screen. If a name is displayed next to each picture segment, the user may identify parts of the model simply by typing the appropriate name. This technique has particular significance in storage-tube graphics.

(iii) Selected keyboard characters can also be used effectively as cursor terminators/triggers on storage-tube systems. The display software package will usually pass to the application program details about the cursor terminating characters used (e.g. using ASCII or some other code) and this may be used as a switch to control the program. Display of the cursor is usually a sufficient prompt to the user that the input is required.

(iv) The keyboard input of groups of characters (to be interpreted as command mnemonics, say) is well known to anyone who has used a computer interactively for purposes such as text editing or file manipulation (e.g. under Unix operation system on the PDP 11/40). Input prompt characters such as "#" or "/" both orientate the user and signify that input is awaited.

Finally, application systems which use the keyboard almost exclusively as the input device have often been ergonomically preferable as the user does not have to 'grope' from one input device to another. The 'one device' input philosophy preserves a useful 'tactile continuity'. A disadvantage, however, is the need for a (sometimes considerable) user manual and the associated difficulties of user training.

4.2 Cross-hair Cursor

(i) The cursor can be used directly for input of screen coordinates. For example when zooming part of a picture displayed on the screen, a corner of the zoomed window could be indicated by the cursor. Other similar functions which require qualitative judgement (e.g. re-positioning of picture parts) can make direct use of cursor coordinate input.

(ii) Coordinate input can be used for simple picture part identification; but this is fast in operation and easy to organise only in particular circumstances. In the case of a menu, for example, it is easy to define a simple rectangular region inside which the cursor must be if it is to identify a menu option. The identification is simplified when the menu items are horizontal. A set of subroutines for such menu operations are included, for example, in LIGHT.

(iii) Input coordinates can be scaled and combined to produce values for such things as rotation angles, dimensions, etc. This usage may require the display of a scale or dial on which the cursor is positioned. The value associated with the current cursor position (or one of its coordinates) may be displayed alongside. This simply provides visual feedback and it can be usefully termed a 'virtual device'.

4.3 Lightpen

In hardware operations the lightpen simply generates an interrupt when it sees light, but for its application a wide variety of software techniques have been developed.

(i) By means of suitable software (usually employing tracking cross) a lightpen can be used to generate position data and thus be used as a cursor.

(ii) The lightpen is used to identify pictures on the screen. The popular use for this is the identification of menu options (light-buttons). The identification of pictures by lightpen is a powerful technique that is easy to organise; the programmer normally chooses the identifying numbers given to separate pictures in the display file so that they may be rapidly identified later in his program.

(iii) Very powerful use of the lightpen relates to virtual devices. A whole range of virtual devices can be coded using a lightpen as the actual input device. A favourite device for generating numbers, for example, is the 'light potentiometer' or 'thermometer' device, which is in essence a scale on which the pen/cross can be positioned. Other devices exist which are usually designed to generate one input value at a time, the need varying from one application area to another. Within engineering design, for example, the user may wish to do some airthmetic calculation, so a 'virtual calculator' in the form of a pocket calculator keyboard is displayed on the screen and is used in the ordinary way with the lightpen replacing the human finger.

To end this section on interactive input devices, it is important to mention that before any input is allowed, sufficient prompts should be given to the user to signify the form of input expected of him. Such prompts take the form of instructional or other messages or a command option menu. The interactive input should be in some way echoed so that the operator knows it has been accepted. For many major or irreversible operations it is advisable to require first a choice from a CONFIRM/REJECT menu. This reminds the user of the import of his accidently destroying a file structure or erasing a picture from the screen.

CHAPTER 4

GRAPHICS SYSTEM AND DATA STRUCTURE ORGANISATION

1. INTRODUCTION
2. GRAPHICS INPUT/OUTPUT PROCESSES OF AN APPLICATION PROGRAM
3. GRAPHICS OUTPUT PROCESS
4. DATA STRUCTURE FOR INTERACTIVE GRAPHICS
   - 4.1 Basic Requirements
   - 4.2 Simple array representation
   - 4.3 Compound data structure

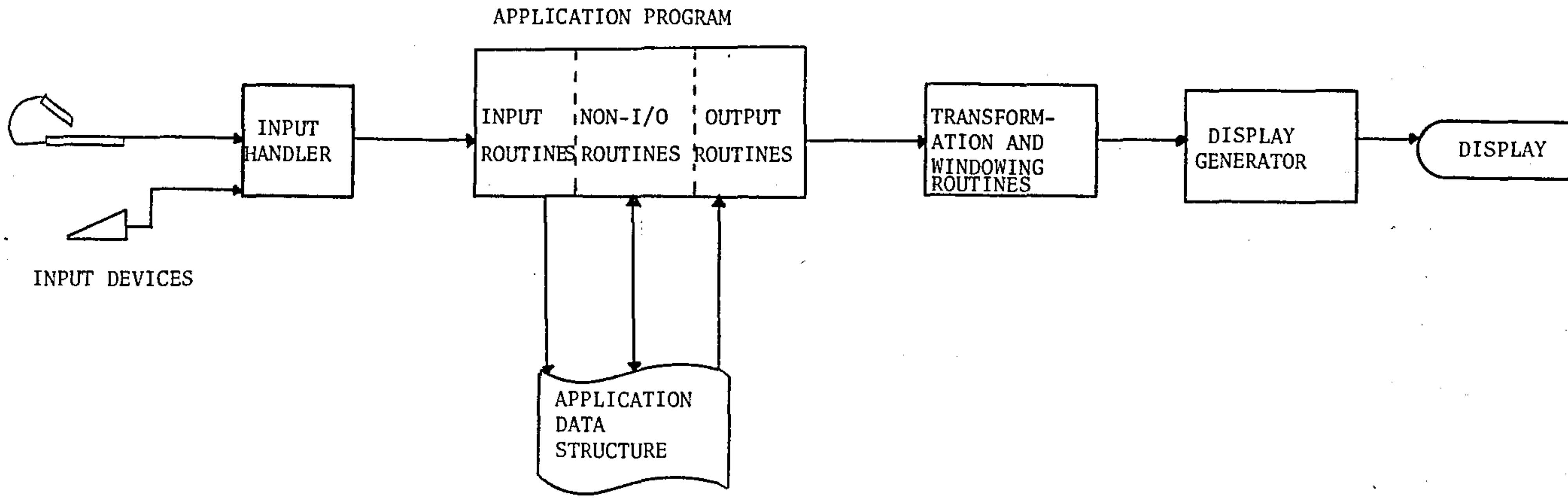
1. INTRODUCTION

The following describes the processes and data essential to a graphics application program with particular reference to the organization of the graphic output process, both for refresh and storage tube display systems. The importance of the graphics application program data structure is also described in some detail.

2. GRAPHICS INPUT/OUTPUT PROCESSES OF AN APPLICATION PROGRAM

A simplified sketch of the processes and data necessary to the operation of an application program is shown in Figure 4.1.

The input handler processes interrupts from the input devices and provides the means for an application program to read data from these devices. The input routines receive data from the input handler, make appropriate changes to the application data structure and pass control to other routines. The non-input/output routines are those portions of the application program that do not directly involve input or output (i.e. computational routines). The output routines define the picture to be displayed, from the application data structure. Effectively they define how these data may be visualised for display purposes. The transformation and windowing routines are capable of scaling, rotating and translating graphic information generated by the output routines. These routines also clip the picture information against a rectangular boundary. A series of different transformations are combined into a compound transformation by concatenation. The display generator generally includes a vector generator and character generator, which convert the transformation and clipped information into signals suitable for the display's deflection systems.



<u>FIGURE 4.1:</u> Simplified Diagram of the Graphics Input/Output Processes

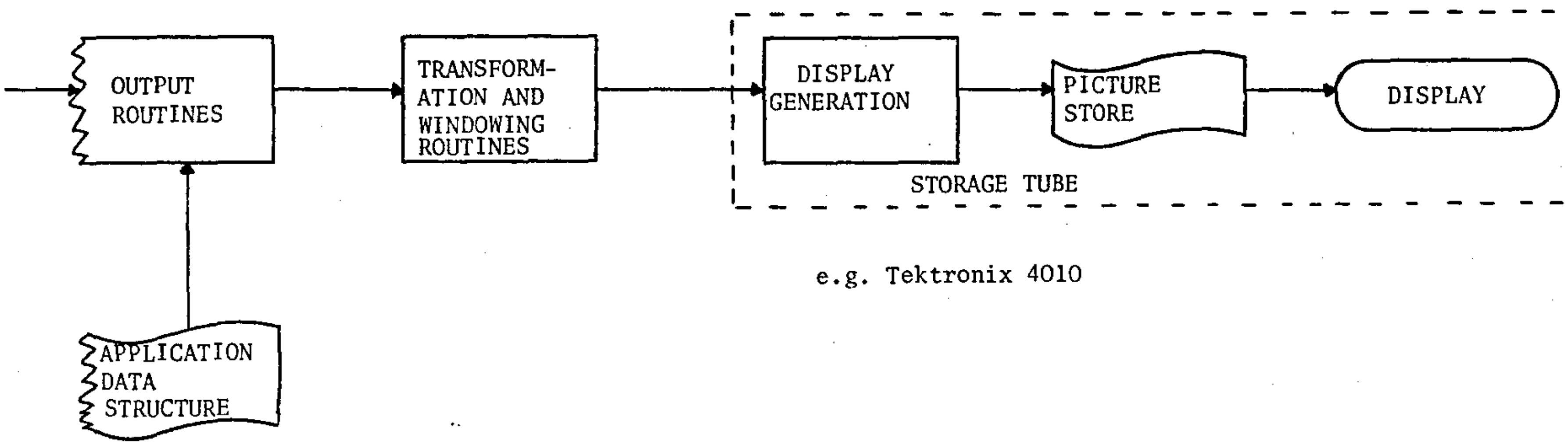
3. THE GRAPHIC OUTPUT PROCESS

If we now consider the output process in Figure 4.1 and assume that the output routines pass graphic data directly to the display screen via the transformation and windowing routines and the display generator, then if these routines are executed once, the picture that they define will flash onto the screen and disappear. But if we arrange for output routines to be executed repeatedly at a sufficiently high frequency the picture will be refreshed and will remain visible. The output routines then perform the function of a 'viewing algorithm' [1], presenting on the screen a continuous view of what is contained in the application data structure. Whenever the data structure is changed, the picture changes accordingly. If we wish to see a different representation of the data, we can substitute a different viewing algorithm by changing the output routine.

This concept of a 'viewing algorithm' is simple but difficult to implement [16]. The problem lies in ensuring that the output routines written by the application programmer execute rapidly enough to keep the picture from flickering. Unless the routines are extremely simple and the data structure quite small, flicker is bound to occur.

However, with storage tube displays, the problem is partly solved because of the inherent storage that cannot be selectively erased. This is known as a 'picture store' and consequently the output process would be modified as shown in Figure 4.2.

With a refresh display, however, it is desirable to provide some means for selectively erasing parts of the picture. This we can do by including a 'transformed display file', that contains results of each transformation process and two sub-processes, one



e.g. Tektronix 4010

FIGURE 4.2: Storage Tube Output Process

to build or generate display file code, and the other to traverse the file for picture generation onto the display. The display file may be divided into any number of logically distinct segments (e.g. Picture Book on the GT42) that may be separately created and erased. This configuration of the output processes for refresh display is shown in Figure 4.3.

It is also possible to have a transformed display file with a storage tube system, but here the display file is not of course used to refresh the display. It can nevertheless perform a very useful function in permitting part of the picture to be changed without the need to transform the whole picture. Instead, just the altered segment is re-constructed, then the screen is cleared and the entire display file is re-transmitted.

Figures 4.2 and 4.3 show some part of the output process enclosed by dotted lines to indicate that these sub-processes may be performed by special hardware. In the case of the storage tube, the display generator, picture store and the display combine together in one hardware package, e.g. the Tektronix 4010.

The GT42 is an example of a programmable refresh display capable of maintaining a segmented display file. It has a small local processor PDP 11/10 to control the display and a simple monitor Picture Book [11] to generate and manipulate the display file and pass back user input. A more advanced form of refresh display would also include a hardware transformation capability e.g. the Vector General.

In each of the above configurations, the application program communicates with the graphics system by means of function calls. Input functions pass data from the input devices to the program, while output functions add line and text to the display file, modify the transformation parameters and manipulate display file segments.

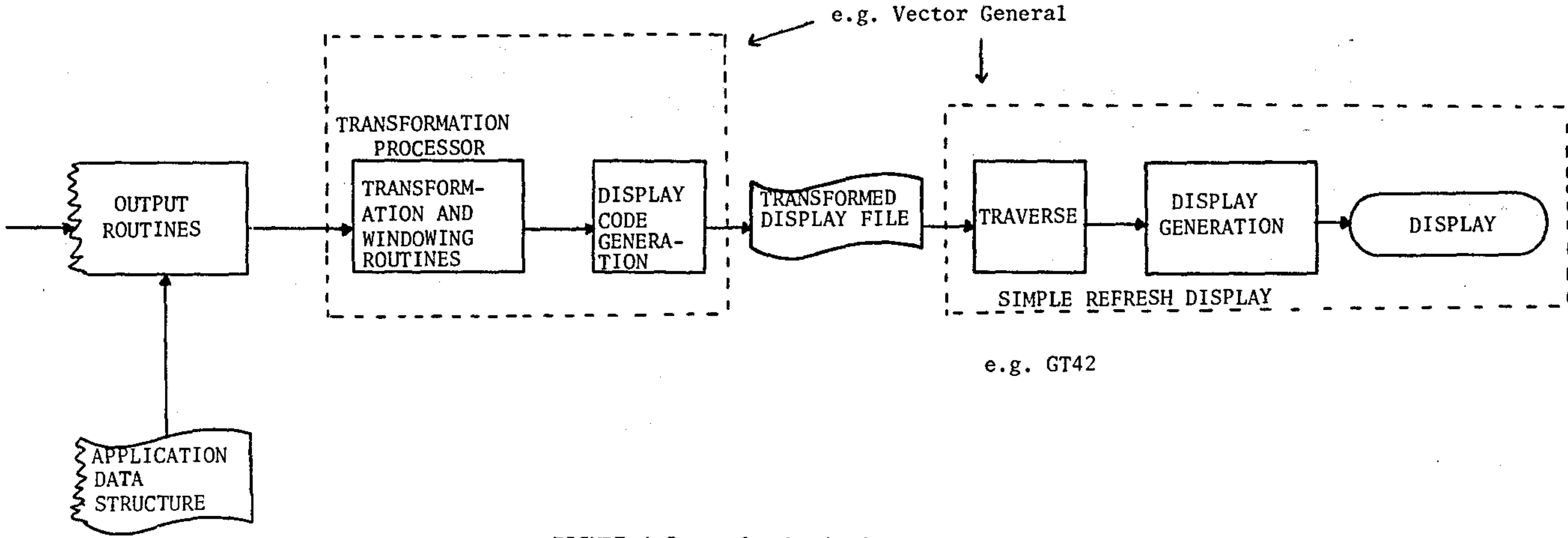


FIGURE 4.3: Refresh Display

4. DATA STRUCTURE FOR INTERACTIVE GRAPHICS

4.1 Basic Requirements

The data structure in a computer graphics system is important for two reasons. Firstly, we make use of various kinds of data organisation, when a graphics system based on display files is built. For example, Picture Book [11] for the GT42 refresh display uses a segmented display file and is conceptually structured into chapters, pages and lines. The package also provides the user with a number of functions for manipulating this structure. This kind of structure must be relatively simple if a display processor is to be able to trace through the file. Secondly, any interactive graphics program must be capable of building and manipulating a database.

The design of such a database should be flexible, and allow searching, accessing and updating to be performed quickly. The degree of structuring used by the application program to describe a picture, depends purely on the application itself.

The picture may have no particular structure, in which case it is best described in terms of a set of point coordinates. In this case it may suffice to use a simple list of the x,y coordinates of the end-points of the straight lines which make up the drawing. Examples of this sort of problem include contour lines and graph plotting. Most pictures, however, have a definite structure; points and lines which represent a separate object, for example, are more closely associated than other points and lines in the display. The most common way of structuring the points and lines in a display is to associate them into sets. The application program then deals, not with points and lines, but with sets; for example, when a transformation (such as rotation or scaling) is applied to a set, it is applied by implication to all the points and lines which make up the set. Situations like this and others require various degrees of complexity to be employed in their data structures, resulting in systems with different degrees of interactive capability. Conventional arrays and vectors, although they are very simple to implement and easy to use, do not perform well in an interactive situation. The reason for this is that they are essentially static data structures, and do not expand or contract during a program's execution. From this we can see that one of the most important qualities to look for in a data structure for interactive use is its ability to change during execution i.e. a dynamic structure. Another important programming concept also relating to computer graphics involves the connectivity between different objects representing some geometric model.

The need to be able to identify, recall, and manipulate these objects requires a well structured data model (database), which is a convenient descriptive representation of the collection of objects on the screen. Each object in the model may be described by a data block (record). A data block consists of a fixed number of contiguous storage words describing a particular entity or figure. Thus, geometric figures represented by data blocks may be selectively created, moved, copied and erased.

4.2 Simple Array Representation

Often we do not need data blocks to be completely flexible, so it is possible to use the concept of array representation of figure elements to serve as a display list. Assume that we are concerned with only straight line segments, and that we wish to use a systematic method of listing the lines for display or for certain basic operations. Two possible representations may be used as illustrated in Figure 4.4.

	XLINE(N,K)		YLINE(N,K)	
LINE NUMBER	K=1	K=2	K=1	K=2
N	X1	X2	Y1	Y2
1	-	-	-	-
2	-	-	-	-
3	-	-	-	-
⋮	⋮	⋮	⋮	⋮

or

LINE	
END1	END2
-	-
-	-
-	-
-	-
-	-

X	Y
-	-
-	-
-	-
-	-
-	-

FIGURE 4.4: Array Representation of Lines

4.3 COMPOUND DATA STRUCTURE

If the data blocks are not stored contiguously in memory, the procedure is more complicated. In this case, we use a list structure in which data blocks or records are chained (connected) together by pointers. A pointer, a word containing an address, is a means of linking one data block with others in the data model. Such a link list may logically connect data blocks which are physically scattered arbitrarily throughout memory. In this way the computer understands the relationship among the objects represented by the data blocks and the relationship between the objects and computation routines. A simple form of a link list representing a physical object to be modelled is composed basically of data blocks, and the basic relationship between the objects are specified by means of pointers, (Figure 4.5a). If the last block in a list has a pointer to the first block in the list, then it is called a ring list, (Figure 4.5b). Usually one data block is designated as the head of the ring, and sometimes pointers from the other rings to the head block are useful.

Another arrangement which is sometimes useful is a set of two-way pointers, as shown in Figure 4.5c.

This consists of a backward pointer corresponding to each forward pointer, so that the program can traverse a sequence of data blocks in either direction.

It is very easy to update a list or ring structure. For insertion of a new data block, all that is required is to create the new block at any convenient place in memory and rearrange the pointer to include it in the list or ring. For deletion, the pointer to this data block to be removed are made to point to the next data block beyond the one to be removed and the unwanted

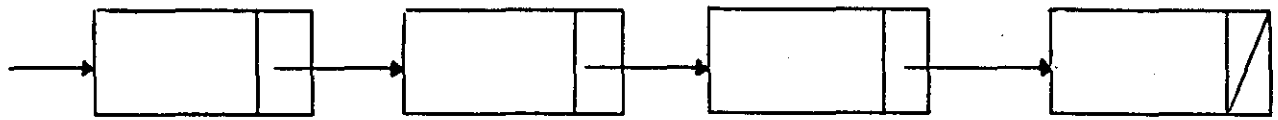


FIGURE 4.5a: Link List

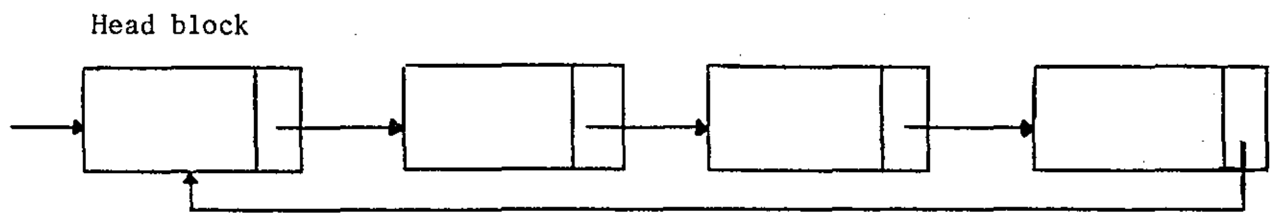


FIGURE 4.5b: Ring Structure

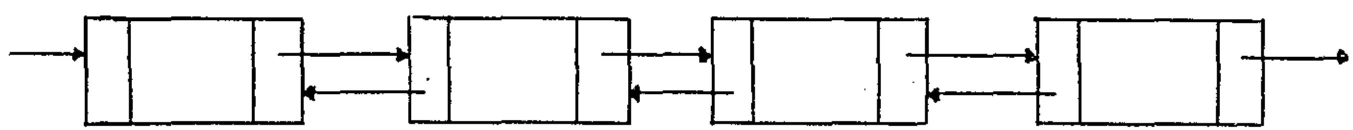


FIGURE 4.5c: Link Structure with Two-ways Pointers

data block may then be removed.

A more complex structure could be built from these basic structures; examples include associative data structure and hierarchical structure.

The associative data structure can be used to relate objects with similar properties even though their data blocks are scattered through storage. We may define certain operators for manipulating the structure e.g. collect objects into sets, assign attributes to objects, delete objects and so forth. We must also devise a way of implementing this structure; for instance we can use rings but this is often inefficient. An alternative method is to use hash-coding [17] i.e. storing and accessing data according to some function (a hash function) of its data content. Some complex data structures use hash coding techniques together with list structures.

In a hierarchical data structure objects are arranged so that certain objects are subdivisions of another at a higher level in the classification scheme. There are two possible forms of this structure:

(i) A tree structure has an identification data block put at the top of the tree; it has pointers to the second level; which in turn have pointers to the third level data blocks, and so on.

(ii) The other hierarchial structure is similar to (i), but it incorporates the concept of a ring. On one ring, there may be branches from any data block to a logically related ring and so on. This structure allows access from any data block to any other via the rings. It is easy to update the structure, since nothing has to be moved in storage; only pointers have to be changed. However, the processing can become quite involved in a deletion operation, especially in making sure that all pointer chains are properly reconnected.

The cost for this more versatile organization is the extra overhead in storage caused by all the pointer chains. The use of very general forms of a compound data structure with a single system for a particular application is inherently inefficient, since it requires excessive storage space and operating time. It has therefore been customary to devise simpler data structures which are tailored to each specific application. In this way storage space, access time, and complexity are minimised, even though considerable design and programming time may be required for development of this data structure.

Typical problems that a generalised structure must be able to resolve are:-

(i) An individual graphic object must be identifiable.

(ii) The relationship, hierarchical or otherwise, between objects must be established.

(iii) Some properties may be shared by different objects, and conversely objects may be allowed to have multiple properties.

(iv) Since drawing may be modified, deleted or expanded in interactive problem solving, data structures must allow for dynamic growth and dynamic association.

The example in Figure 4.6 illustrates a hierarchical data representation of the square (SQ) and circle (CR) using a ring structure. A block, called 'FIG' associates the square and the circle which is also reached from the square block. The centre and radius of the circle are defined in block CR. All blocks may be reached from any given block and the lines are defined in separate blocks (L1 through L4). The end points of a line are also defined as separate blocks and are actually subordinate to these lines.

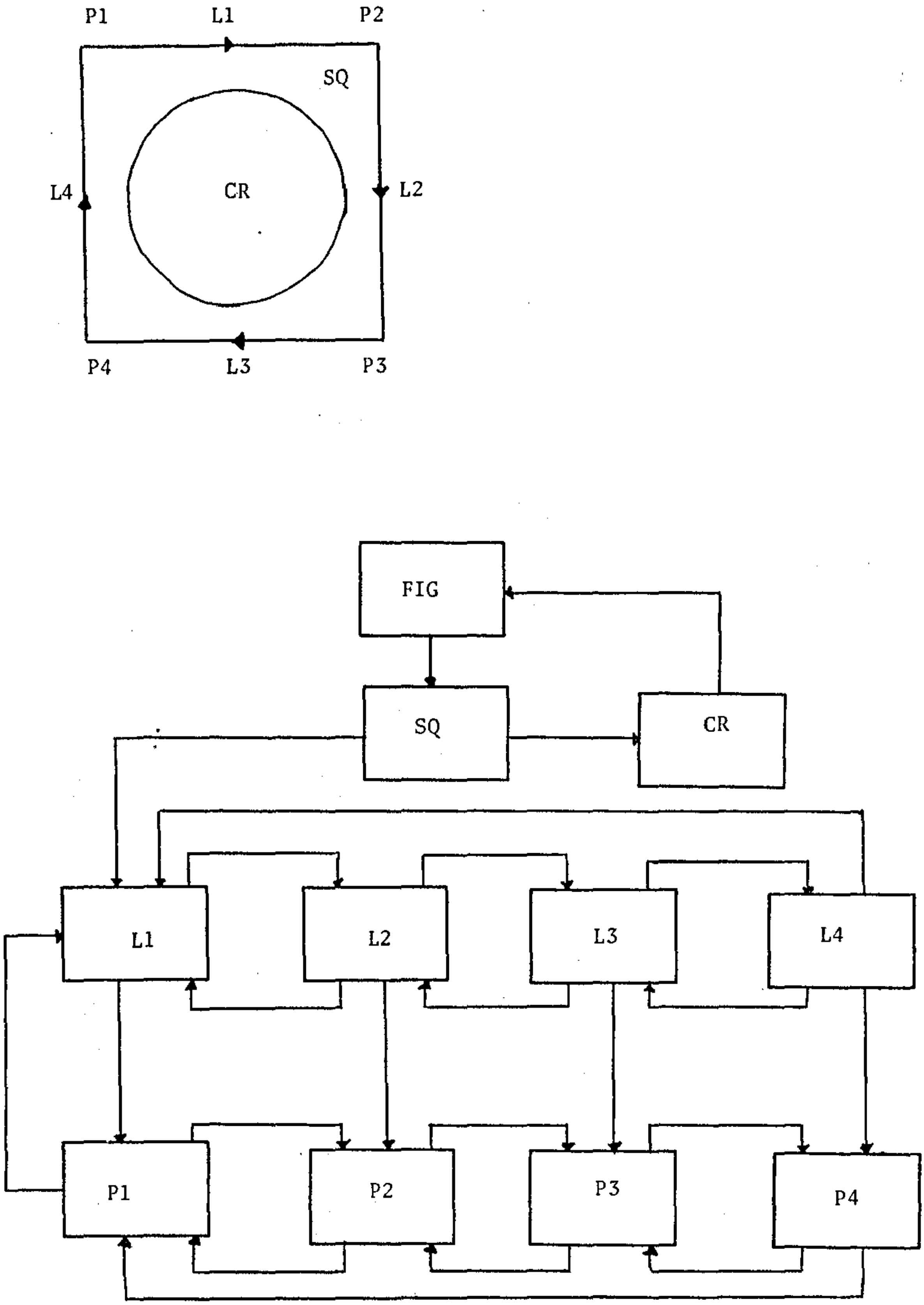


FIGURE 4.6: An Example of Ring Structure

CHAPTER 5

THE GRAPHICS SOFTWARE PACKAGE 'LIGHT'

1. INTRODUCTION

This chapter presents in detail the development and implementation of the graphic software package LIGHT (Loughborough Interactive Graphics system for Tektronix 4010) under the UNIX time-sharing system on the PDP 11/40 mini-computer.

2. THE CURRENT ENVIRONMENT

2.1 Hardware Configuration

The hardware configuration on which LIGHT was developed consists of the PDP 11/40 mini-computer central processor unit (16-bit word) with 60K words of core store. The present installation has two moving head disc drives RK05 each of which provides 2.5 M bytes on a removable disc cartridge. There is also a high-speed paper tape reader PR11, a DECwriter console LA30 matrix printer, a storage tube graphics terminal Tektronox 4010 and a variety of alphanumeric terminals:

7 × KSR33 and 2 × ASR33 teletype;

3 × Newbury Laboratories 7002 VDU's;

1 × Tektronix 4023 VDU.

Shortly after the development of LIGHT, a GT42 refresh display graphic terminal was installed. This consists of PDP 11/10 central processor unit (CPU and store), display processor unit, communication interface, keyboard unit, CRT display unit and a light pen. Figure 5.1 illustrates this hardware configuration, part of which makes our graphics system.

2.2 UNIX Software System

UNIX time-sharing system has been operational since late 1976 on the above-mentioned hardware configuration in this department. It is a general-purpose, multi-user, interactive operating system which provides

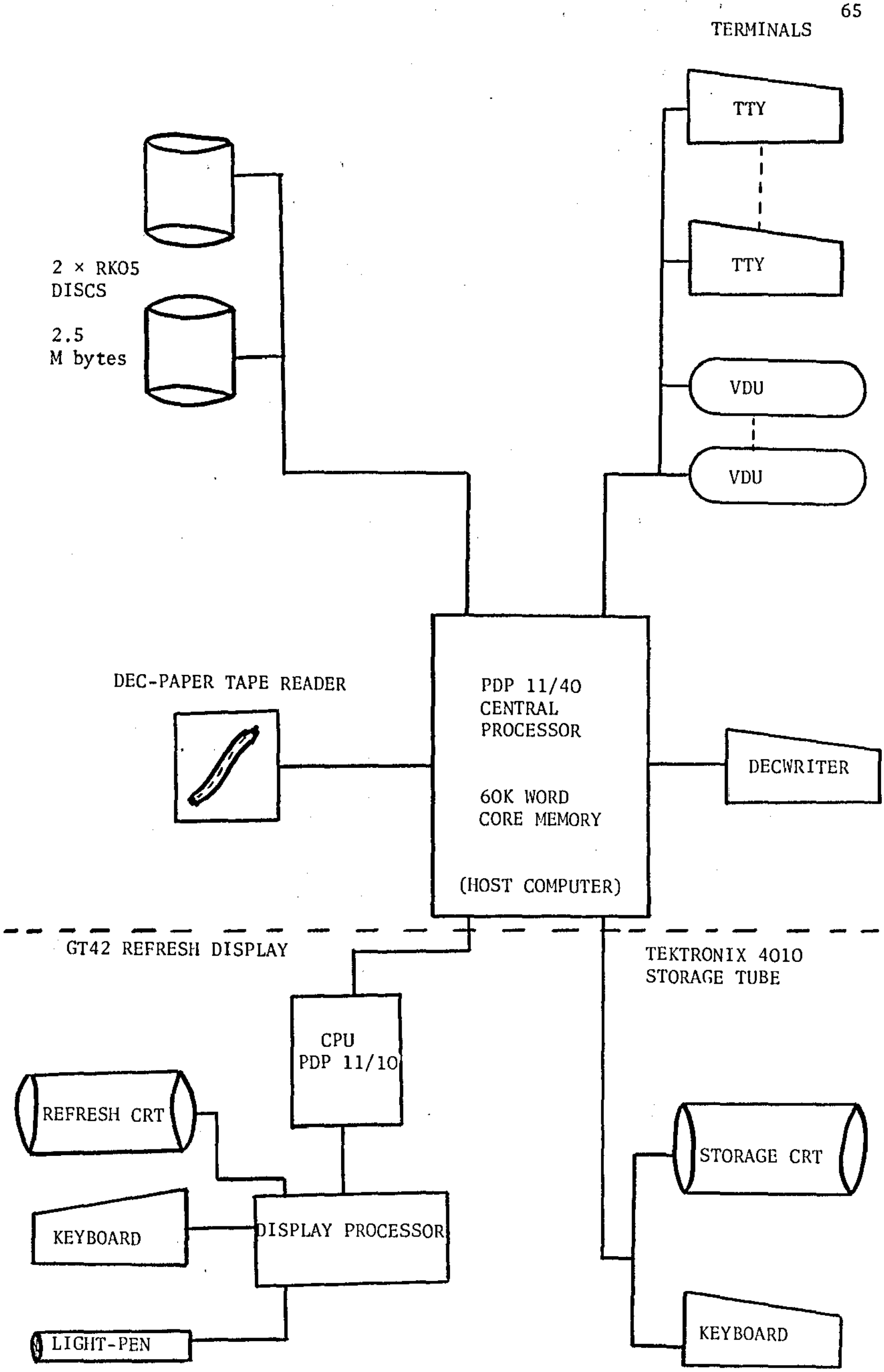


FIGURE 5:1: Hardware Configuration

many facilities of a large system [18]. Among these features are:

(i) a hierarchical file system incorporating demountable volumes,

(ii) system command language selectable on a per-user basis,

(iii) the ability to initiate asynchronous processes.

Besides the system proper, there are a number of major system programs available, of which the following were used extensively in the development of this work:

Assembler which resembles PAL-11R

Text editor based on 'QED'

Linking loader

Fortran compiler.

User communication with UNIX is effected with the aid of a program called 'Shell'. This is a command line interpreter; it reads lines typed by the user and interprets them as requests to execute other programs. Shell is also a command by itself and may be called recursively to execute a series of other commands placed in a file.

The main feature of this system is a versatile, convenient file system with complete integration between disc files and all input/output devices. From the point of view of the user, there are three kinds of files:-

(i) Ordinary file: contains whatever information is placed on it, for example, Fortran programs or textual information.

(ii) Directory: is like an ordinary file except that only the system can modify it. It provides the mapping between the names of files and the files themselves. The file system is a tree-structured hierarchy originating at a root directory. Each user has a directory of his own; he may also create subdirectories to contain groups of files conveniently treated together. Any type of file can occur at any level. At any given time a user process is associated with a particular current directory. When the user program wishes to open or create a file, it gives the system the name of a file in the current directory or it gives a path name which either specifies the absolute location in the tree or the location relative to the current directory. A system directory exists which contains all the programs provided for general use [19]. File system protection consists of associating with the file at time of creation the name of the creator and permitting him to specify whether he and others can read and write the file.

(iii) Special file: each I/O device supported by UNIX is associated with at least one such file. Special files are read and written just like ordinary disc files, but a request to read or write results in activation of the associated device. UNIX I/O system supports a large number of device drivers, which share a great many routines as well as a pool of buffers.

It is not necessary that the entire file system hierarchy resides on the same device (e.g. disc pack). On our installation for instance, the root directory resides on the system disc, and all users' files are contained on another removable disc which is mounted by the system initialisation program.

UNIX occupies 20.8K words of core memory, the rest (39.2K) being available for user programs. The name of any executable program can be used as a command. This name is first searched for in the current directory and if that fails, it is then searched for in a system library. A myriad of commands are available including those for listing directories, moving, copying and deleting files, and changing the current directory. Another feature of the system is the ability to initiate asynchronously executing processes from within a program or from command level. The command interpreter creates a process to execute the command and wait for its completion.

3. DESIGN CRITERIA

In general, the process of building a graphics system can be described as follows:

(i) Choose the language on which to base the system

(ii) Design the functions or language extension

(iii) Write and document the software to perform the graphics functions.

The first two steps constitute the design of what is sometimes called a 'graphics language'. These graphics functions play a vital part in determining the success or failure of the system; and should give the programmer control over the system hardware/software. Some systems are built in the form of a graphics package i.e. as a set of functions or subroutines to be called by application programs written in a high level language. An alternative approach is to design a special programming language; this generally amounts to choosing an existing language, and then extending and modifying it where necessary to perform certain graphics tasks. The first approach was adopted in building up the graphics package LIGHT. This offers greater flexibility; if future experience indicates the need for additional functions, then the appropriate function could be added more easily.

It is essential to make graphics systems not only inexpensive but simple to program. The programmer, whether novice or expert, should find it as easy to write a graphical program as a non-graphical one.

Therefore, the design of the graphics subroutines package 'LIGHT' was influenced by several criteria. The fundamental aim was to provide the user with an immediate, simple but powerful means of writing a graphics application program. Thus, the package was written in a high level language so that it would shield the programmer from the low level features of the hardware. Fortran was chosen because graphics display systems are likely to be used for engineering and scientific applications, and this is the most widely accepted programming language in these fields. In addition, Fortran is also available on virtually every computer, which makes the package to a certain extent portable. Hence, LIGHT was coded in UNIX Fortran which is a subset of ANSI Fortran, and care was taken to minimize the use of non-standard features. However, it is possible to implement the package in a different language, since the set of functions are carefully designed to be independent of any specific language for its implementation.

Early in the design, it became apparent that LIGHT should provide only general-purpose facilities for using the graphic display terminal. For example, the subroutines should neither generate geometric figures (e.g. circles, triangles, logic elements) nor impose constraints on the structuring of modules, since the way in which geometric figures are generated appears to be dependent on applications. However, for these purposes, users can develop libraries of routines that are more in accord with their needs. Another important factor in the design was to assume a set of default conditions that an application programmer would usually require. If he does not take positive action to change them, these remain in effect throughout execution of his program. Whenever default conditions are overridden the newly entered condition prevails from that point on, rather than the default condition.

LIGHT also does not contain any data structure facilities nor does it use a mandatory data structure for picture representation. These are awkward in that they force a user to think in a particular way, and tend to be inefficient, especially in store requirements. This last difficulty is exacerbated by the problems of dumping and restoring from secondary storage. Further, no particular data structure is convenient for even a majority of applications. Another reason for not having a mandatory graphical data structure is that the complexity that would result would prevent use of the software by non-specialists. The following facilities are available under the software graphics package LIGHT:-

1. Initialisation routines for setting the default values
2. The primitive graphics functions for point and line drawing
3. Character and text handling
4. Cursor and menu operations
5. Linear transformation and clipping
6. Simple perspective projection
7. Other utility routines.

Several hardware limitations had dictated the extent of the facilities that were provided by LIGHT. For example, the available disc space was limited because the computer resources were also distributed among up to 14 other time-sharing users working in other areas. In addition, the amount of core store available to the graphics user would necessarily be shared by LIGHT. Also, run-time response was noticeably slow, partly due to the absence of floating point hardware. These and various other constraints, involving both hardware and software, required that the coding of the package be written efficiently. Since graphics is only a part (usually a small part) of most applications it can only expect its fair share of computing resources. Efficiency is not only a function of implementation; it depends also on the design of the user interface. When the software takes the form of a library of subroutines, the function and form of each subroutine is important. Efficiency in terms of computation time is largely a run-time issue although the form of the library must be such that the package can always avoid unnecessary computation. However, programs using floating point (FP) assume that the FP processor is available, so the Fortran compiler generates code using these instructions. On our current configuration which does not have a FP processor these instructions are trapped as 'illegal' and they are interpreted by software routines linked into the object program. The net effect of this is that although the programs run perfectly well they are very slow and tend to impose a large overhead on the rest of the system because of the huge number of interrupts that must be serviced. Additionally the above software routines require disc space and loading time.

4. LOW COST DISPLAY TERMINAL

4.1 The Tektronix 4010 Display Terminal

There is a growing demand for low cost computer graphics for small scale computer users or for the user who would rather have easy access to a somewhat less sophisticated console than have very limited access to a more powerful but far more expensive device. Tektronix 4010 offers this opportunity with its vector capabilities and character writing speed [20]. Such terminals may be used both for time-sharing and for solving conventional graphics problems at an affordable price. This requires a set of functions to transmit line and text information to the terminal in response to function calls generated by the application program.

The real difficulty in using this sort of terminal lies in compensating for its relatively poor performance which includes several deficiencies; poor picture quality, unsatisfactory transmission rate and in most cases, lack of selective erasure capability. The terminal uses serial transmission, both for text and for graphics information. For example, a vector is transmitted to the terminal as a special control character followed by four characters specifying the length of the vector. This use of character transmission greatly simplifies the task of integrating these terminals into an existing time-shared system (e.g. UNIX). The use of serial transmission generally imposes a fairly severe limit on the speed at which pictures can be transmitted, for speeds above 2400 baud (bits/second) are usually beyond the capacity of the transmission line. The problem of lack of selective erasure combined with the low transmission speed and poor time-shared response makes dynamic graphics almost impracticable.

4.2 Operating Modes

The main mode is character (Alpha) mode; in this mode data characters received are interpreted either as characters to be plotted or as ASCII control characters (e.g. carriage return, line feed) depending on the status of the two high order bits. Certain control characters have been given special meanings for performing special functions: for example, 'GS' sets the terminal to graphics mode, 'US' changes the terminal to Alpha mode. In graphic mode, data representing X,Y coordinates are plotted (as points and lines) until the mode is changed back to character/control mode. By ASCII convention, all control characters are distinguished from graphics data characters, by adding two high order (tag) bits in each byte of the latter. The 4010 has only one graphic mode and often uses a pair of characters for some control functions e.g. ESC,FF to erase screen.

5. STRUCTURE OF THE LIGHT PACKAGE

5.1 The Graphics Library

The graphical aspects of an application appear in a program as calls to subroutines to draw lines on the screen, output textual information, raise the cross-hair cursor and develop menus, items of which may be picked from the screen by the cursor. The LIGHT package takes the form of a Fortran-callable library of subroutines, each of which requires a minimal set of parameters. The graphics library was written with few initial applications in mind, so that the main features were immediately and continuously tested. The result is an easily-used system that allows Fortran programmers to produce debugged interactive graphics programs as readily as conventional batch processing programs.

Due to the dynamically changing requirements typical of graphics application programs, it was important to make the facilities offered by UNIX indirectly available to the graphics program through the use of structured library files. Consequently the system would automatically decide which LIGHT subroutines should be loaded with the application program into main store. Effectively the UNIX linking loader would search the library exactly at the point it has encountered the call and only those routines defining unresolved external references are loaded. Therefore it is important that the order of the subroutines in the library must be correct. That is, if a routine from a library references another routine in the library, the referenced routine must appear after the referencing routine in the library. All such references must be resolved before a load module can be executed. Every source file representing a single library subroutine is compiled independently to produce an object file. The group of the object files is then archived into a single library file named 'LIGHT'. This consists of 53 subfiles each of which can be individually updated, replaced or deleted. The library can be maintained by using the UNIX command 'AR' which is an archive and library maintainer [19], allowing the user various options such as copy, append, delete, replace and extract an object file from/to the archive file LIGHT.

5.2 Organization

Functionally, the LIGHT subroutine package is organized into three separate and distinct modules:

(i) LIGHT-UNIX software interface

(ii) LIGHT proper

(iii) Basic (geometric) transformations

Each module has a certain task to accomplish offering the user various facilities that he may require in his graphics application program. These modules can be independently updated or modified without affecting one another. This modular design approach has an important consequence for future extensions to the facilities currently offered by LIGHT. It also provides some degree of device independence. Although this package was intended primarily for the Tektronix 4010 display, it was designed so that other graphics devices can be incorporated with limited programming effort. This was apparent when LIGHT was easily extended to run on the refresh display GT42 in Emulator mode (Appendix 1.8). Moreover, the transformation module is completely device-independent and without any modification it can be used directly with the Picture Book package [11] designed for the GT42 refresh display.

The detailed description and implementation of each module will be presented in the next section. A very large portion of LIGHT subroutines, including all the user-callable routines are entirely written in standard ANSI Fortran, and constitute the 'Front end'. These in turn call other routines, which are not user callable and constitute the device/system dependent part, and are termed the 'back end'. This is the interface between the front end and the device. It also provides the interface between the front end and the machine/operating system as in our case with PDP 11/40/UNIX. Because of this structure, a program containing LIGHT routines can be considered to be in three parts. These are the main program, LIGHT front-end, and LIGHT back-end routines as illustrated in Figure 5.2. The total size of the main program will vary according to its function.

All the information required by LIGHT is conveyed via subroutine arguments and so programs may be no more than a series of Fortran calls. Thus a Fortran programmer can introduce himself to graphics with minimal effort. It is important to emphasise that LIGHT satisfies the basic needs for creating graphics application programs. As mentioned earlier, there is no data structure imposed by the package; thus any data structure may be defined by the calling program.

6. DESCRIPTION AND IMPLEMENTATION OF LIGHT

6.1 LIGHT-UNIX Software Interface

This provides the means of interfacing LIGHT package with the UNIX system. Its primary objective was to facilitate the link between the graphics library and UNIX I/O system. This was desirable in order to handle all I/O of graphical information coded in single ASCII characters; including special control characters to the Tektronix 4010 display terminal. This is not normally possible with the use of Fortran I/O alphanumeric string format.

In addition, under the existing environment the interface also provides some utility routines that have been found to be useful in

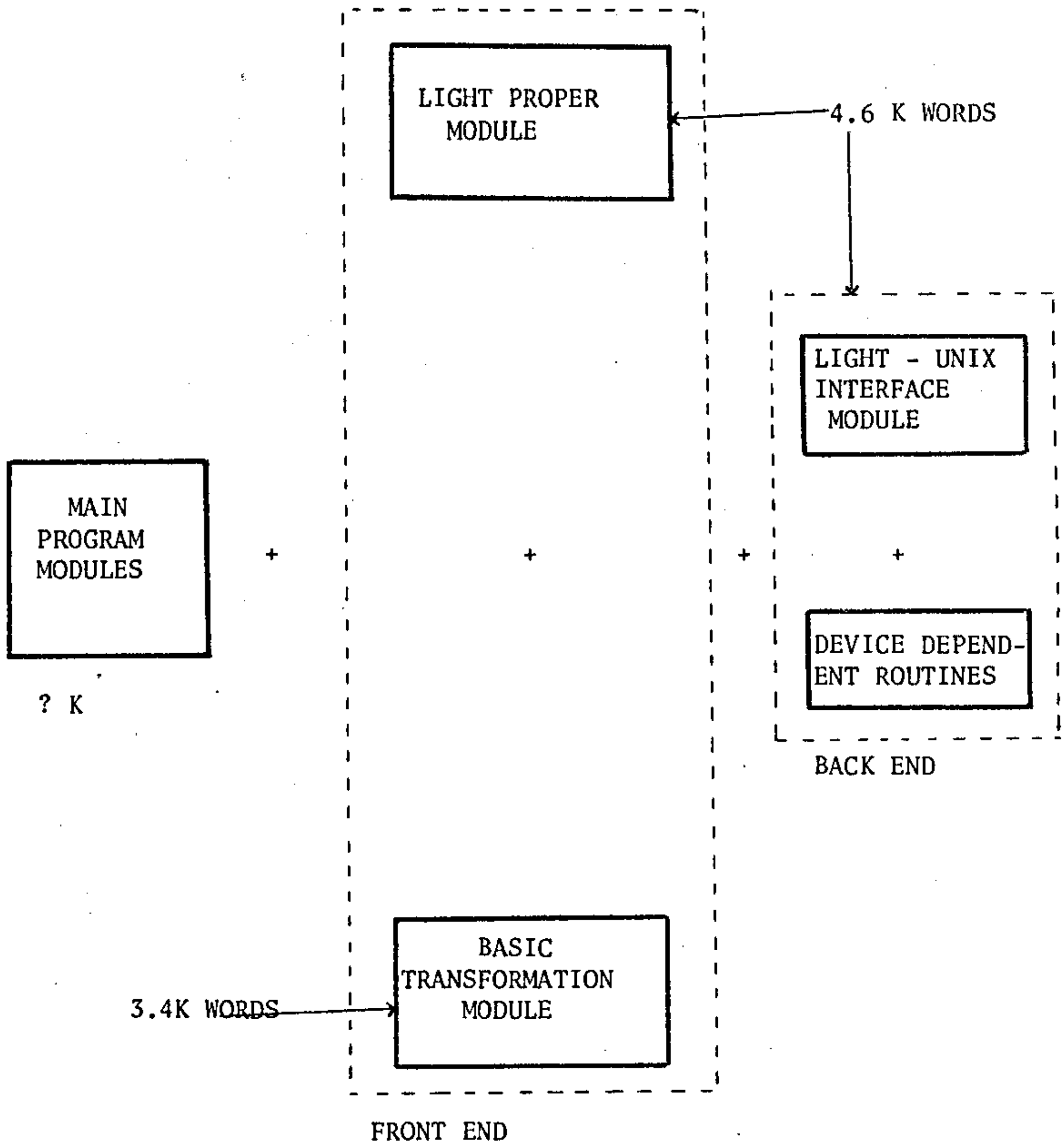


FIGURE 5.2: Structure of LIGHT Software Package

writing graphics programs. They enable the application programmer to exploit some feature of the UNIX file system, through Fortran calls incorporated in his program. This software interface was implemented using a number of UNIX system entries [21]. These allow the UNIX user to communicate easily with the file system. This lowest possible user level is designed to avoid distinction between the various devices and files and between direct and sequential access. No large 'access method' routines are required to insulate the programmer from the system calls; in fact all user programs either call the system directly or use a small library program, only few instructions long, which buffers a number of characters and reads or writes them all at once. Their calling sequence is usable either in assembly or C-language. Assembly language [22] was naturally chosen, as UNIX Fortran permits calls to routines written in assembly code using certain calling sequence conventions. It was not possible to use the C-language portable library, because C-programs can not communicate with Fortran programs in the current system environment.

The calling sequence convention used in coding these assembly routines is as follows:

1. Save register R3,SP (stack pointer)
2. Arguments list (pointer to values) begins at 2(R3).
3. Entry name is name of function or subroutine followed by "."
4. First word after entry point is location of return value.
5. Second word after entry point is pointer to PDP-11 code body
6. Return is expedited by a 'Jump' to the global routine 'retrn'.

The following assembly coded routines (Appendix 1.1) are essential to support the running of LIGHT under UNIX:

(i) Input/Output single character routines

(ii) Cross-hair cursor graphics input routine

(iii) File overlaying routine

(iv) File deletion routine

These form part of the back-end which is machine/operating system dependent.

(i) Input/Output Single Character Routines

These routines namely 'INCHAR' and 'OCHAR' handle the I/O transfer between the LIGHT package and the display terminal through the use of UNIX I/O system calls. As mentioned previously UNIX treats I/O devices as special files in which reading and writing is done just like ordinary disc files. A special scheme is available whereby device drivers may provide the ability to transfer information directly between the user's core image and the device without the use of large buffers. The method involves setting up a character-type special file corresponding to the 'raw' mode. In this mode, every character is passed immediately to the program without waiting for a full line.

The display terminal 4010 is treated normally as a teletype terminal with asynchronous communication interface DL11E which supports most common ASCII terminals. Under UNIX a disc file or device is associated with a file descriptor, an integer between 0 and 9. It is used to identify the file in subsequent read, write or other I/O calls. The file descriptors '0' and '1' are designated for standard input and output respectively. In coding these two Input/Output routines, the following I/O system calls [21] were employed:

GTTY:- This essentially stores the current STATUS information of the terminal whose file descriptor is given in register R0 in a three words argument. The mode of the terminal is contained in the third word which is saved before the status of the terminal is changed to 'raw' mode so that its normal mode can be restored after I/O operations are carried out.

SIGNAL:- This ensures that the terminal would restore to its normal mode when an interrupt signal is generated by some abnormal event, initiated by the user at the terminal or by program error. Normally all such signals cause termination of the program, unless special action has been taken. In these routines the label EXIT1 specifies the address where the interrupt is simulated. The normal status of the terminal is subsequently restored.

STTY:- This converts the mode of the terminal to raw mode by setting the third status word to (octal) 000040, and then restores it to its normal mode after I/O is accomplished.

READ/WRITE:- A single I/O call produces direct transmission between the terminal and user's read/write buffer, so that raw I/O is considerably more efficient when many words are transmitted. These system calls require two arguments:

(1) the user's buffer address, and

(2) the number of contiguous bytes, (in this case, one). The number of characters actually read or written is returned in register R0 .

(ii) Cross-Hair Cursor Graphics Input Routine

It is possible to exploit the Tektronix 4010 as an interactive terminal, since it has a program controllable cross-hair feature and therefore a graphics input capability. To make this easily available to the user, this routine was developed using the UNIX I/O system entries for raising the cross-hair cursor and reading the digitized coordinates. Again the same system calls as in (i) were invoked in raw mode to implement this routine. The entry name of this routine is 'CURSON', which works as follows:-

1. The cross-hair cursor is raised. This requires transmission of two ASCII control characters ESC and SUB(27 and 26 in decimal respectively) to the terminal. The user can then change the cursor control to the desired intersection point.
2. When the user strikes a keyboard character, the character and the coordinate location are sent to the computer. Consequently five bytes of cursor information are read by this routine. These graphics input bytes represent the keyboard character plus a four byte sequence containing high and low order X, and high and low order Y. Each byte contains the two tag bits plus five binary bits. Each byte thus encodes to an ASCII character. Figure 5.3 illustrates this computer response to graphics input from the cross-hair cursor.

After the cursor information is read, the Alpha cursor returns to its home location (top-left hand corner). The status of the terminal is returned to 'cooked' (normal) mode.

(iii) File Overlaying Routine

The limited memory size of our machine has been quite successfully offset by the development of a very easily used overlay capability. The programmer need only divide his program into smaller modules to fit the available core size. These modules could be stored on the disc as 'program files' which may subsequently be called and executed almost as subroutines. When module segments are called from the disc, the new segment with the help of UNIX system entry EXEC is loaded and executed. The system call EXEC overlays the calling process with the named file and

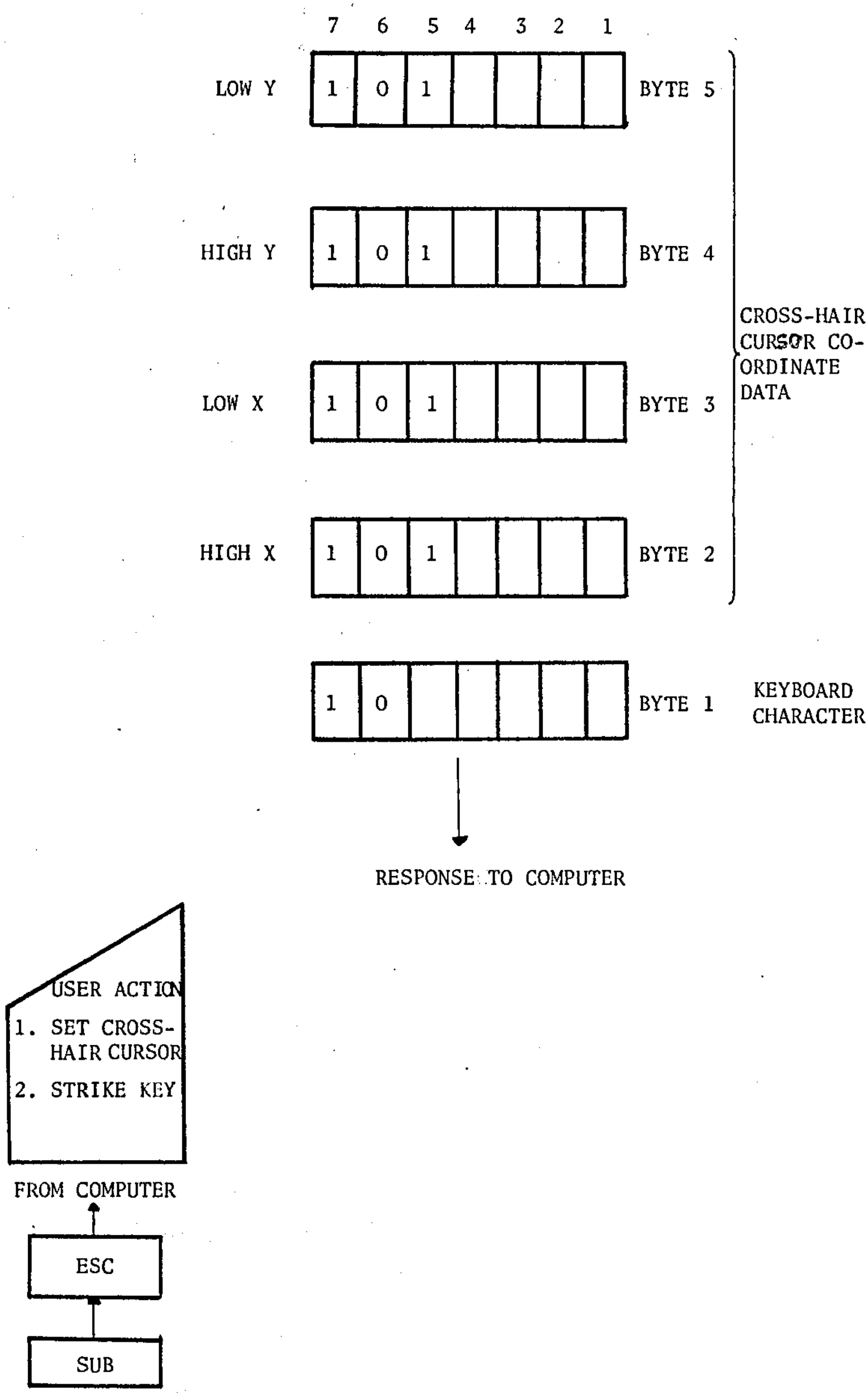


FIGURE 5.3: Response to Computer Command and Cursor Control

then transfers to the beginning of the core image of the file. All code and data in the process using EXEC is replaced from the named file, but open files, current directory and interprocess relationships are unaltered. Only if the call fails, for example when the file can not be found or its execute-permission bit is not set, does a return take place from the EXEC; it resembles a 'jump' machine instruction rather than a subroutine call. The first argument to EXEC is a pointer to the name of the file to be executed. The second is the address of a null-terminated list of pointers to arguments to be passed to the file. The entry name of this routine is 'OVLAY'.

(iv) File Deletion Routine

This is another system entry in the UNIX file system. It removes the entry for the named file passed as an argument from the current directory. This routine makes use of the system call UNLINK. It may be useful in removing intermediate files created by application programs. The entry name of this routine is 'FLRM'.

6.2 LIGHT Proper

Efficiency and flexibility are key points when generating routines to run graphics programs. The LIGHT module provides this capability with a set of Fortran-callable subroutines which form the central part of the LIGHT package. These routines support a wide range of applications in an efficient and cost effective manner. The basic structure is modular, thus providing access to individual features of the graphics hardware and/or software. Through these routines the graphics program has full control of the terminal character/vector generators and the graphic input device. This facilitates the construction of displays from basic graphic elements and program communication with the display and console operator. The library subroutines are divided conceptually into four categories:-

(i) Initialization

(ii) Point and line drawing

(iii) Character and text handling

(iv) Cursor and menu operations.

The functions of the subroutines for each category are summarized in Table 5.1, (see also LIGHT-User Guide, Appendix 1.9).

TABLE 5.1: Basic LIGHT Subroutine Library

Category	Subroutine	Purpose
(i) Initialization:	TXOPEN	Sets up default values for screen coordinates, menu operation and character position.
	TXCLER	Clears the screen ready for next display.
	TXVPRT(X0,Y0,X1,Y1)	Defines a physical 'view-port' on the screen.
	TXWIND(X0,Y0,X1,Y1)	Defines a 'window' in problem space and maps this on to the viewport.
	ALPHMD	Switches the display terminal to 'Alpha' mode for input/output of Alphanumeric information.
	GRPHMD	Switches the display terminal to 'Graphic' mode and returns to the previous beam position.
(ii) Point and Line drawing:		
	TXMOVE(X,Y)	Moves current beam position to scaled point (X,Y) without drawing.
	TXMOVR(DX,DY)	Moves current beam position through a displacement (DX,DY) without drawing.

Category	Subroutine	Purpose
	TXDRAW(X,Y)	Draws a visible straight line from the current position to point (X,Y).
	TXDRWR(DX,DY)	Draws a visible displacement (DX,DY) from the current beam position.
(iii) Character and text handling:		
	TXGET(ICHAR)	Delivers into ICHAR a single character entered by the user from the keyboard.
	TXPUT(ICHAR)	Displays a character whose ASCII code is ICHAR. Interprets TAB as a space and RUBOUT as several superimposed characters.
	TXLINE(STRING,N)	Inputs a line of characters from the keyboard into the array STRING and Echoes to the screen. Deals with TAB and RUBOUT as above.
	MESSAG(TEXT)	Displays characters specified as Hollerith string 'TEXT'.
	TEXTUP(Filename,N)	Displays N lines of textual information previously stored in the named disc file.
	INTGET(I)	Obtains the next input integer from the keyboard.
	SPOUT(TEXT)	Removes spaces, and other non-printable characters from the Hollerith string TEXT.
	DTEXT(X,Y,TEXT,N)	Displays the Hollerith string 'TEXT' at scaled coordinate point(X,Y) on the screen.
(iv) Cursor and menu operations:		
(a) Cursor Control:		
	TXCURS(X,Y,ICHAR)	Sets the cross-hair cursor, reads the cursor coordinate position and the character entered.

Category	Subroutine	Purpose
	CURPOS(X,Y)	Positions the Alpha cursor at the specified point (X,Y).
	CHTOXY(NLINE,NCHAR,IX,IY)	Converts character coordinate (NLINE,NCHAR) to screen coordinate (IX,IY) of the bottom-left hand corner of the character.
	XYVOCH(IX,IY,NLINE,NCHAR,IA,IB)	Converts the screen coordinate (IX,IY) to character coordinate (NLINE,NCHAR).
(b) Menu operation:		
	MNOPEN(X,Y,MNO)	Announces that a menu is to be displayed whose origin (top-left hand corner) is at screen coordinate(X,Y).
	MNTEXT(TEXT,N,MNO)	Displays the next menu item 'TEXT' of N characters.
	MNPICK(I,ICHAR,MNO)	Raises the cursor, allowing the user to pick an item from the menu, and returns the item index in I.
	MNDISP(TEXT,M,LEN,MNO)	Displays a complete menu containing the TEXT of M items
	FRAME(X,Y,NC)	Draws a rectangle round the menu whose origin is at the point (X,Y).

(i) Initialization

These routines (Appendix 1.2) are concerned with setting up the display terminal before any subsequent output is directed to the display by other calls. The Tektronix 4010 contains 1024×1024 addressable points, of which 1024 by 781 are in the viewable area of the screen.

Default values for the display viewport and window are set by TXOPEN to contain 1024×781 addressable points; this was chosen because points just above 780 may be visible but marginal in quality.

Routines TXVPRT and TXWIND are provided for further control of the mapping between the problem space and the screen viewport. These two routines effectively alter the default values and enable the user to specify a rectangular area on the screen within which a picture is drawn. The user is also provided with a routine (TXCLER) to clear the display; this sets the terminal to Alpha-mode and returns the Alpha cursor to its home position. In addition, the terminal can be set to graphic mode (GRPHMD) or to Alpha-mode (ALPHMD) as appropriate. These routines would enable the graphic program to output textual information to the screen in the middle of drawing a picture, thus giving the user full control of the terminal mode.

(ii) Point and Line Drawing

Display images are composed of basic visual elements (primitives) which the terminal can generate. Essentially the only graphical primitives that the programmer needs are functions to define points, lines and displayed text strings. The main criteria in choosing a set of primitive graphical functions are as follows:-

1. Clarity: the functions will often be used by relatively inexperienced programmers, and should be as simple and comprehensible as possible.
2. Convenience: the functions should permit all forms of point- and line-plotting and positioning both by relative and by absolute coordinates.
3. Compactness: the set of functions should not be too large, for this will enlarge the software system. Circles and other

more complex constructions should be provided outside this set.

The basic functions for point and line plotting (Appendix 1.3) are:

TYPE	MODE	
	Absolute	Relative
MOVE	TXMOVE(X,Y)	TXMOVR(DX,DY)
DRAW	TXDRAW(X,Y)	TXDRWR(DX,DY)

These routines in turn call upon two other routines XVPLOT and VPLOT which constitute part of the 'back-end':

(a) XVPLOT:- This mainly converts the user specified coordinates into physical screen (raster) coordinates in integer form. It also checks against any attempt to output coordinates off the screen. If this happens, an error message ('coordinate off screen') is displayed, reminding the user to revise his coordinate setting.

(b) VPLOT:- Graphic plotting information is sent to the terminal in four byte sequences, each containing high and low order Y, and high and low order X. Thus, this routine sets the terminal to graphics mode by sending the ASCII control character GS(29 in decimal), followed by the four ASCII characters carrying the graphic data for vector plotting. After a GS and the initial four bytes have been sent to the 4010, additional bytes that do not change (except for the low X byte) need not be sent; however, low Y bytes must be sent if high X byte has been changed. The low X byte must be sent each time to cause the point or vector to be drawn. Vectors are drawn from the old address to the new address with the exception of the first vector after entering the graphic mode. Figure 5.4 illustrates the method of computing the four bytes. Each number is converted

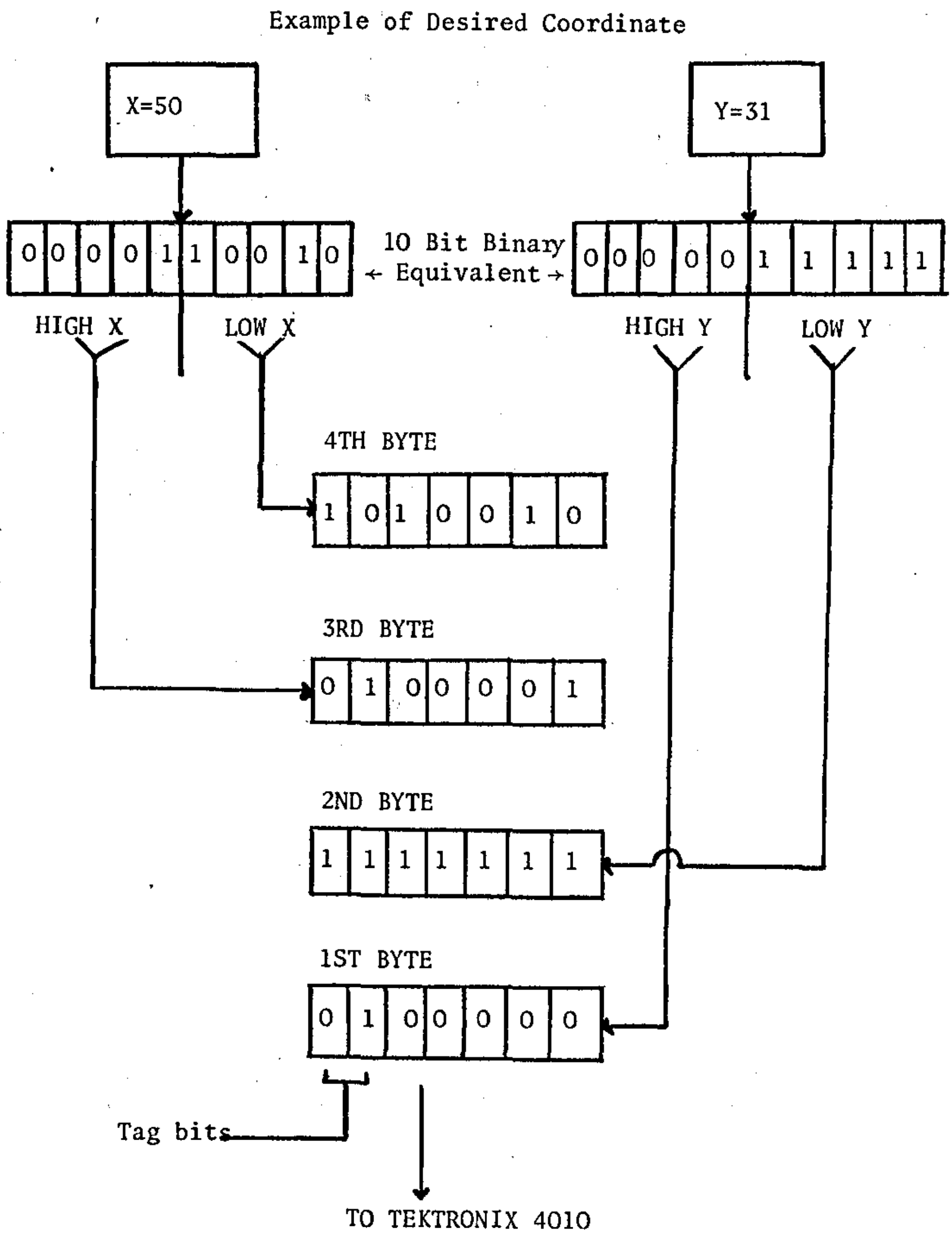


FIGURE 5.4: Computing 4 bytes of data for X=50 and Y=31

to its 10-bit equivalent, which is divided into high and low 5-bits. The bytes are then assembled as shown with two tag-bits added.

(iii) Character and Text Handling

A set of routines (Appendix 1.4) is provided by LIGHT, allowing the user to incorporate standard keyboard interaction in his graphics application program. Keyboard input, for example, can be used to good effect to input textual information which cannot be predetermined. Textual output information can be used to prompt the user to perform a particular action and the application program is instructed to wait for the user's response. Thus terminal users can communicate with an application program by means of alphanumeric Input/Output. The implementation of these routines makes extensive use of INCHAR and OCHAR, which transmit character information directly to the terminal in raw mode. The routines handle simple and composite input/output. Some special characters are also dealt with at the user level, for example, TAB is interpreted as a suitable number of spaces; RUBOUT is displayed as several superimposed characters, the deleted character being removed from the character string.

(iv) Cursor and Menu Operations

LIGHT provides the user with the facility to input and output graphical information. In addition to the standard keyboard interaction, the programmer can also incorporate graphical interaction via the cursor. Cursor input provides identification to the program of selected items on the display screen. The Tektronix 4010 graphic cursor described in section 6.1(ii) was programmed to read five graphic input bytes containing the cross-hair cursor coordinates and the input keyboard character. The conversion from graphic input bytes to numerical coordinates is a straight forward operation which is performed by the routine CURSET (Appendix 1.5) as

X-coordinate = 32 (high X-32) + low X-32

Y-coordinate = 32 (high Y-32) + low Y-32

The Fortran-callable subroutines provided in here fall into two kinds:

(a) cursor control with routines to interpret screen coordinates as character positions.

(b) Menu display and cursor choice therein.

In (a) the routine TXCURS activates the cross-hair cursor by calling the CURSET routine. The returned value of cursor location (X,Y) on the screen is tested for violation of the boundary limit specified by the current viewport, and if so, a warning message ('illegal cursor position') is displayed. The cursor coordinates are subsequently converted from screen to problem coordinates. Two other routines are also available (CHTOXY and XYVOCH) in (a) for relating the character coordinate (NLINE, NCHAR), pointed at by the cursor to screen coordinate (X,Y). Basically, the Y-coordinate is mapped into the line number, and the X-coordinate corresponds to the character within the line. These routines are particular useful for graphical text editing.

The routines in category (b) are concerned with the handling of user defined menus. They enable the programmer to display a menu anywhere on the screen (MNOPEN) and pass control to the terminal user (MNPICK). When a particular item is picked up from the menu by means of the cross-hair cursor, an arrow would be drawn adjacent to the selected item to indicate that the user request is accepted. However, if the horizontal cursor line is placed outside the valid item zone(s) (Figure 5.5) the cursor would return and no arrow would be drawn, allowing the user to try again.

6.3 Basic Transformation

For the purpose of visualisation, graphical information is usually transformed so as to provide a particular view on a display, and further manipulation is sometimes desirable in order to enhance the visualisation in various ways. The transformation facilities in LIGHT may be considered as performing three basic functions:-

(a) Object orientation with respect to the selected coordinate system, e.g. rotation and translation, to obtain a particular view.

(b) Relating the user's coordinate system to the display coordinate system.

(c) Various types of projection and distortion such as perspective, isometric projection, etc. This is mainly used to aid the visualization of objects. There are other visualization techniques, such as hidden line removal [1], intensity modulation and shading.

The transformation functions should be simple to use and efficient in execution. Luckily these two requirements do not conflict, as efficiency in transformation is gained by combining scaling, rotation, translation and perspective projection into a single matrix that applies to the end point coordinates of each line of a given object.

The range of effects that can be produced by transformation is very large and is catered for in its generality by a vocabulary of basic transformations. The basic vocabulary is supplemented by some special routines (themselves using the basic routines) which provide facilities that are generally useful and that can be easily specified. Such routines include axonometric projection (parallel projection) and perspective viewing (point projection) from any viewpoint. The use of homogeneous coordinates (x,y,z,t) to define three-dimensional objects allows either 'affine' or perspective transformations to be applied with equal ease [23]. The three-dimensional point (or vector) corresponding to (x,y,z,t) is

$$(X,Y,Z) = \left(\frac{x}{t},\frac{y}{t},\frac{z}{t}\right) \tag{5.1}$$

The generalised 4×4 transformation matrix based on such a homogeneous coordinate representation is

$$\underset{\sim}{A} = \left[\begin{array}{ccc|c} S_x & C_{12} & C_{13} & P_x \\ C_{21} & S_y & C_{23} & P_y \\ C_{31} & C_{32} & S_z & P_z \\ \hline T_x & T_y & T_z & S \end{array}\right] = \left[\begin{array}{c|c} L & P \\ \hline T & S \end{array}\right] \tag{5.2}$$

which naturally partitions into four separate submatrices:

(1) L(3×3) produces a linear transformation in the form of scaling, shearing and rotation.

(2) T(1×3) row produces translation

(3) P(3×1) column produces perspective transformation

(4) S(1×1) single element produces overall scaling.

When the vector [X Y Z 1] is transformed by the most general 4×4 matrix A it will become the vector [x* y* z* t*] which is usually normalised to [X* Y* Z* 1], as shown mathematically by

$$[X\ Y\ Z\ 1]*\underset{\sim}{A} = [x^*\ y^*\ z^*\ t^*]\rightarrow[X^*\ Y^*\ Z^*\ 1] \tag{5.3}$$

The set of routines (Appendix 1.6) that produce these transformations are summarized in Table 5.2. These routines permit both two- and three-dimensional pictures to be transformed, and assumes the existence of two transformation matrices:

RTM:- the previous reference transformation matrix

TM:- the accumulated transformation matrix resulting from calls to any of the routines - TRANSL, SCALNG, ROTATE, PERSP and PROJCT.

Both these matrices must be declared by the user in his program as

```
COMMON/MATRIX/RTM(4,4),TM(4,4)
```

TABLE 5.2: Transformation Routines

Category	Subroutine	Purpose
1. Linear transformation:	SCALING(SX,SY,SZ)	Superimposes scale changes along the X,Y and Z axes as specified.
	ROTATE(RX,RY,RZ)	Rotates about axes X,Y and Z by the angles RX,RY and RZ (degrees) in this order.
	TRANSL(TX,TY,TZ)	Translate the current point (X,Y,Z) through the displacement TX,TY and TZ.
2. Clipping(windowing):	CLIP(CLINE,XO,YO,X1,Y1,IREJ)	The line CLINE is clipped to the rectangular window XO,YO, X1,Y1.
3. Simple perspective projection:		
	PERSP(PX,PY,PZ)	Sets up a perspective view where PX,PY and PZ are the reciprocals of the viewing distance from the planes YZ, ZX and XY respectively.
	PROJCT(NPLANE)	Projects on a given NPLANE where NPLANE(1,2, or 3) specifies the coordinate plane of projection (i.e. x=0, y=0 or z=0).
4. Transformation control:		
	SAVMAT(A)	Saves a copy of the transformation matrix A into the stack.
	RESTOR(A)	Restores the transformation matrix A from the stack.
	UNITY(A)	Sets the transformation matrix A to unit matrix.
	SETMAT(A)	Sets the reference matrix RTM to A.
	CONCAT(A,B,N)	Postmultiplies matrix(vector) A($N\times4$) by B(4×4) and leaves result in A.

The RTM matrix can be updated by the accumulated matrix TM by concatenation (RTM×TM). The current transformation can be modified, reset, saved or suspended at any stage and TM is available to the user for inspection. The basic transformations are in general non-commutative and so there are inherent perils involved in combining them. Typically, an error in ordering transformations could result in the whole picture being outside the visible area.

1. Linear Transformation

These routines have an important role in building up a picture. If a picture part occurs more than once, in different orientations, it can be defined once and the other instances correctly oriented by use of these routines.

Scaling:- The diagonal terms of the general 4×4 transformation matrix produce local and overall scaling. The routine SCALNG sets the first three diagonal terms which represent the local scaling with scale factors SX, SY and SZ. Consider the following transformation:

$$[X^* \ Y^* \ Z^* \ 1] = [X \ Y \ Z \ 1] \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (5.4)$$

which shows the local scaling effect.

Global scaling may be obtained by using the fourth diagonal element, i.e.,

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & S \end{bmatrix} \qquad (5.5)$$

which has the same effect as

$$\begin{bmatrix} \frac{1}{S} & 0 & 0 & 0 \\ 0 & \frac{1}{S} & 0 & 0 \\ 0 & 0 & \frac{1}{S} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.5}$$

The matrix TM is postmultiplied by the scaling transformation matrix shown in (5.4).

Rotation:- The rotation matrix is an orthogonal matrix. Thus the length of a line joining two points is invariant under this transformation, and in general the size of objects is unchanged by rotations. The routine ROTATE combines the effect of rotations through the angles RX,RY and RZ (degrees) about x,y and z axes in this order in one single transformation matrix (Table 5.3) i.e.,

$$[X^* \; Y^* \; Z^* \; 1] = [X \; Y \; Z \; 1] \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{about x-axis}} \times$$

$$\underbrace{\begin{bmatrix} \cos\phi & 0 & -\sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ \sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{about y-axis}} \times \underbrace{\begin{bmatrix} \cos\psi & \sin\psi & 0 & 0 \\ -\sin\psi & \cos\psi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{about z-axis}} \tag{5.7}$$

The resulting transformation matrix simply reduces, with some trigonometric manipulation, to a matrix whose elements consist of sines and cosines of the sums and differences of the angles as given in Table (5.3). This would involve much less computation time than a concatenation of the individual transformations.

Row \ Column	1	2	3	4
1	$\frac{1}{2}[\cos(\phi+\psi)+\cos(\phi-\psi)]$	$\frac{1}{2}[\sin(\phi+\psi)-\sin(\phi-\psi)]$	$-\sin(\phi)$	0
2	$\frac{1}{4}[\cos(\theta+\psi+\phi)-\cos(\theta+\psi-\phi)+$ $\cos(\theta-\psi+\phi)-\cos(\theta-\psi-\phi)]-$ $\frac{1}{2}[\sin(\theta+\psi)\sin(\theta-\psi)]$	$\frac{1}{4}[\sin(\theta+\psi+\phi)-\sin(\theta+\psi-\phi)$ $-\sin(\theta-\psi+\phi)+\sin(\theta-\psi-\phi)]$ $+\frac{1}{2}[\cos(\theta+\psi)+\cos(\theta-\psi)]$	$\frac{1}{2}[\sin(\theta+\phi)+\sin(\theta-\phi)]$	0
3	$\frac{1}{4}[\sin(\theta+\psi+\phi)-\sin(\theta+\psi-\phi)$ $+\sin(\theta-\psi+\phi)-\sin(\theta-\psi-\phi)]$ $+\frac{1}{2}[\cos(\theta+\psi)-\cos(\theta-\psi)]$	$\frac{1}{4}[\cos(\theta+\psi+\phi)-\cos(\theta+\psi-\phi)$ $-\cos(\theta-\psi+\phi+\cos(\theta-\psi-\phi)]$ $+\frac{1}{2}[\sin(\theta+\psi)-\sin(\theta-\psi)]$	$\frac{1}{2}[\cos(\theta+\phi)+\cos(\theta-\phi)]$	0
4	0	0	0	1

TABLE 5.3: The Elements of the Combined Rotation Matrix (4×4)

The fact that the three-dimensional rotations are non-commutative must be kept in mind. However, a different order of rotations may be performed by using separate calls of the routine.

Translation:- The transformation which translates a point (X,Y,Z) to a new point (X* Y* Z*) is

$$[X^* \; Y^* \; Z^* \; 1] = [X \; Y \; Z \; 1] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ T_x & T_y & T_z & 1 \end{bmatrix} \tag{5.8}$$

The routine TRANSL sets up the elements of the matrix and performs the postmultiplication of TM by this matrix.

2. Clipping

In practice, graphical devices such as the Tektronix 4010 display work with a bounded space; in addition, the user may wish to confine his picture to a prescribed window. We must either ensure by means of scaling that our object lies within the bounded region or we must exclude all parts which lie outside the region. The latter technique is called 'scissoring' or clipping. This window may take any shape but it is usually rectangular.

The routine CLIP provides the clipping of a given line to the rectangular boundary (Figure 5.6) defined by its opposite corner coordinates as (X0,Y0) and (X1,Y1). The coordinates of a given line's end points are first passed as input parameters to the routine, which then returns with the flag IREJ=0 or 1 indicating whether the line is fully rejected or accepted. If IREJ=1 then CLINE will contain the co-ordinates of clipped line end points. The clipping algorithm used in implementing this routine has effectively two parts:-

1. It determines whether the line lies entirely within the window, and if not, whether it can be trivially rejected as lying entirely outside the window (Figure 5.6a). Two function subroutines (IREJCT and JACCPT) handle this test for total rejection or acceptance.

   Let us suppose the line AB joining (a_1,a_2) and (b_1,b_2) is to be clipped by the rectangle with opposite corners (XO,YO) and (X1,Y1). A line is rejected if it lies completely to the left, to the right, above, or below this rectangle. Thus if

$$\left.\begin{array}{ll} & XO \geq \max(a_1,b_1) \\ \text{or} & X1 \leq \min(a_1,b_1) \\ \text{or} & YO \geq \max(a_2,b_2) \\ \text{or} & Y1 \leq \min(a_2,b_2) \end{array}\right\} \qquad (5.9)$$

   the line is rejected. If the line is not rejected by this test, we find whether it crosses a continued edge of the clipping rectangle.

   If $XO > \min(a_1,b_1)$ then the line crosses X = XO

   If $X1 < \max(a_1,b_1)$ then the line crosses X = X1

   If $YO > \min(a_2,b_2)$ then the line crosses Y = YO

   If $Y1 < \max(a_2,b_2)$ then the line crosses Y = Y1

   If none of the above is true, then the line lies totally within the clipping rectangle. Otherwise (i.e. the line crosses one of the continued edges) we conclude a new end point which needs to be investigated more closely.

2. The coordinates of intersection points with the boundary may be computed either by using the simple concept of finding directly the intersection point of two straight lines or alternatively by the iterative method of subdivision [1] of the

line and throwing away the segment which lies off-window. The subdivision method was used here as it was considered to be more efficient. The line is subdivided at its midpoint, yielding two line segments (Figure 5.6b). The test of part (1) is then applied to each segment of the line separately. The search for an end point stops either when both halves of the line are rejected or when the mid point coincides with one of the edges of the window. Clipping to a rectangular two-dimensional region introduces extra flexibility e.g. logically separate pictures occupying different areas of the screen can be prevented from interfering with one another. In addition, it also facilitates zooming down to a number of levels and provides selective viewing of part of a large picture.

3. Simple Perspective Projection

As displays are two-dimensional devices we can draw only two-dimensional projections of three-dimensional objects. The idea of projection onto the plane $z=z_0$ can be expressed in the matrix form:

$$[X^* \; Y^* \; Z^* \; 1] = [X \; Y \; Z_0 \; 1] = [X \; Y \; Z \; 1] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & Z_0 & 1 \end{bmatrix} \qquad (5.10)$$

Note that the transformation matrix has a row of zero coefficients and is therefore singular. This is as one might expect, because if the matrix could be inverted it would mean that we could recover three-dimensional information from the two-dimensional drawing, and this is not usually possible.

The user is provided with a routine PROJCT which would set up a transformation matrix for a specified projection plane.

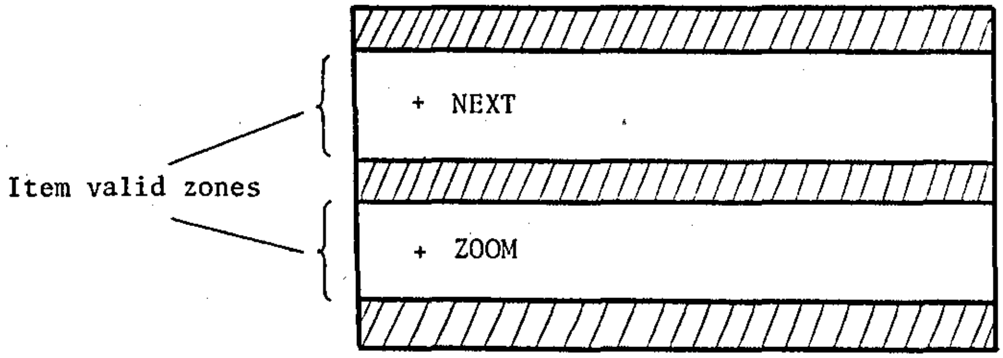


FIGURE 5.5: Menu Item Valid Zones

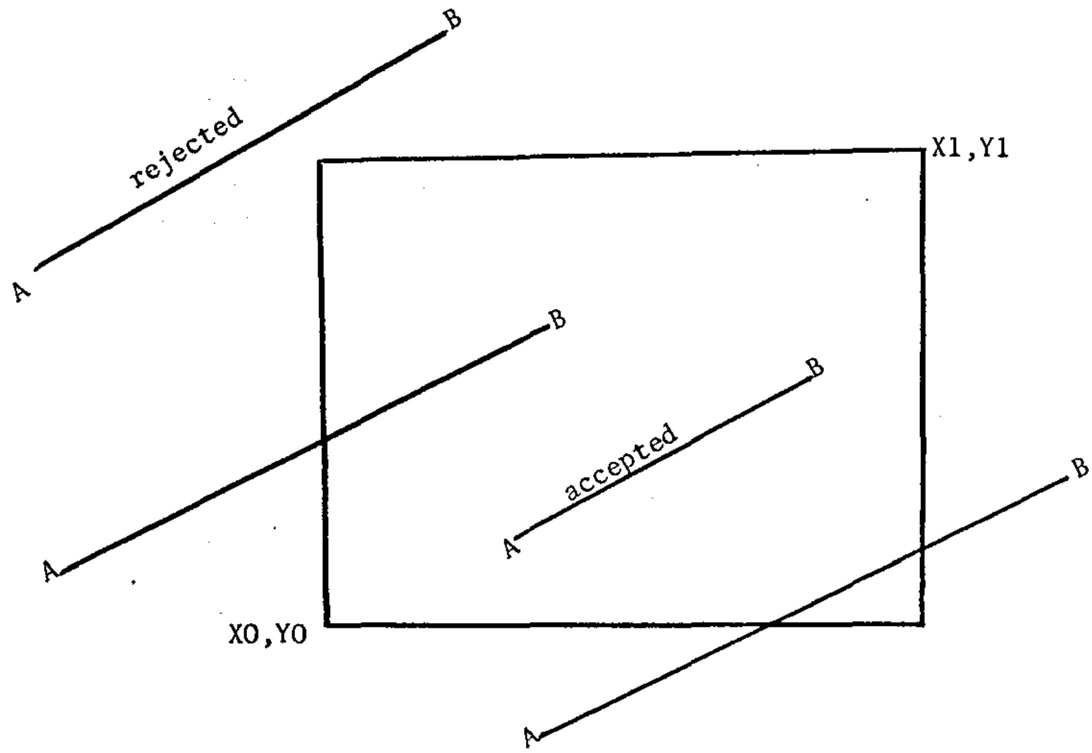


FIGURE 5.6a: Rejection Test

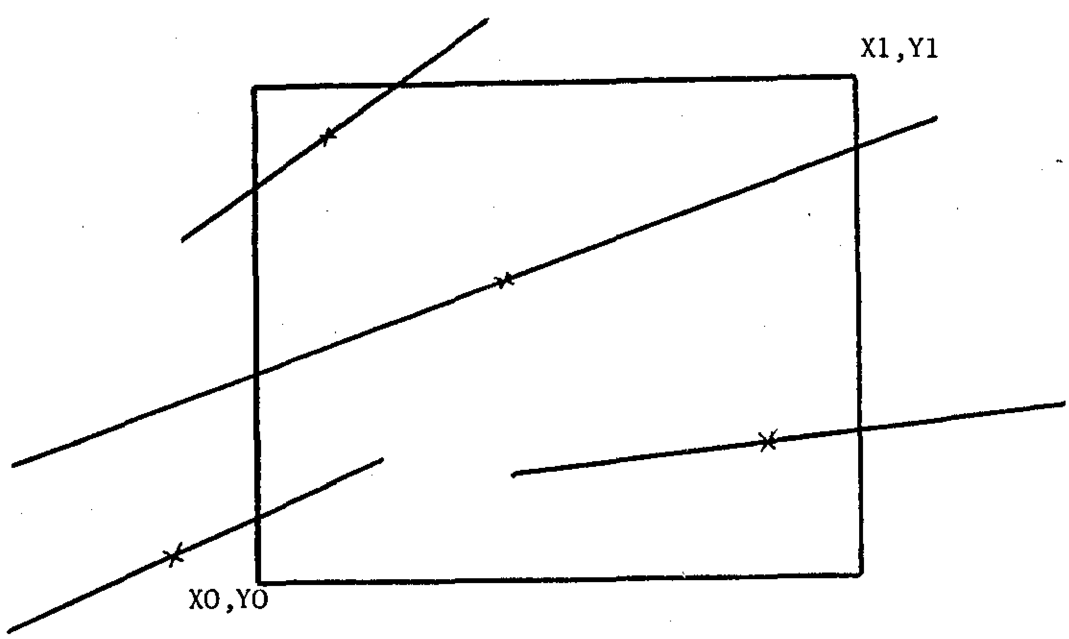


FIGURE 5.6b: Bisection of the Line

We now consider the relevance of the off-diagonal elements of the fourth column of the transformation matrix

$$[x^* \; y^* \; z^* \; t^*] - [X \; Y \; Z \; 1] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = [X \; Y \; Z \; (1+P_z Z)] \qquad (5.11)$$

i.e.,

$$[X^* \; Y^* \; Z^* \; 1] = [\frac{X}{1+P_z Z} \; \frac{Y}{1+P_z Z} \; \frac{Z}{1+P_z Z} \; 1] \qquad (5.12)$$

Under this transformation the origin [0 0 0 1] and the points at infinity on the x and y axes, namely [1 0 0 0] and [0 1 0 0] are unchanged. But the infinite point on the z-axis [0 0 1 0] is transformed into the finite point [0 0 1 P_z], i.e. after normalizing, the point [0 0 1/P_z 1]. Thus, lines which were parallel to the z-direction before the transformation will now appear to pass through the point [0 0 1/P_z 1] which is sometimes called the 'vanishing point' of the perspective transformation.

The user is provided with a call to routine PERSP which applies a perspective transformation. When we project this transformed view onto a plane we obtain 'perspective projection'. For each vanishing point of a perspective transformation there is a corresponding centre of projection which lies on the same axis at the same distance from the origin, but in the opposite direction.

A transformation matrix of the form

$$\begin{bmatrix} S_x & C_{12} & C_{13} & 0 \\ C_{12} & S_y & C_{23} & 0 \\ C_{31} & C_{32} & S_z & 0 \\ T_x & T_y & T_z & 1 \end{bmatrix}$$

followed by a non-perspective projection onto a plane is known as an 'Axonometric projection'. It is in fact a perspective projection with vanishing points all at infinity. When the upper left-hand 3×3 matrix is orthogonal under this transformation the projection is said to be 'trimetric'. If two of the axes in an axonometric projection are equally foreshortened when projected, the transformation is said to be be 'dimetric'. In an 'isometric' projection all these axes are equally foreshortened, and usually at 120^{o} to each other [23]. If the upper left-hand 3×3 is not orthogonal an axonometric projection is said to be an 'oblique' projection.

4. Transformation Control

These routines merely provide some facilities for manipulating transformation matrices. In particular, SAVMAT and RESTOR permit the user to save/restore transformations on/from the pushdown stack (defined by the package) at any time to facilitate display of hierarchical data. For example, a picture containing repeated symbols will involve concatenation of transformations; whenever concatenation is performed, the current transformation must be saved so that it can be restored after displaying the symbol. A one-dimensional array in the form of a pushdown stack is employed to perform the above two operations. The remaining routines perform tasks such as concatenation and initialization of transformations.

7. THE GT42 IN EMULATOR MODE

LIGHT subroutines may also be used in conjunction with the GT42 refresh display in Emulator mode as a Tektronix 4010. The Tektronix 4010 Emulator program [24] accepts input from the DL11E communications interface and converts the characters into alpha or graphic data. This is displayed on the screen by adding the data in serial manner to a display file. The method by which incoming characters, including special-function characters e.g. clear-screen, send cursor position, etc., are converted is the same as that of the Tektronix 4010. The cross-hair cursor and the thumbwheel is simulated on the GT42 as a tracking-cross which may be moved around the screen with the light pen.

PART II

APPLICATIONS

CHAPTER 6

INTERPOLATORY DATA FITTING - IDF

1. INTRODUCTION

Data fitting in general has been used for many years in engineering applications. It occurs in machine tool control, in design problems and increasingly in computer graphical presentation of the numerical solution of physical problems, where numerical results would otherwise be difficult to interpret.

In particular, interpolatory data fitting is a special area of the more general curve fitting process. With many problems, as a result of measurements or calculations, we obtain a set of data points corresponding to a function, and it is usually desirable to pass a 'smooth' curve through these points.

It would be useful to distinguish between curve fitting and curve design. The former involves generation of a smooth curve for an already defined curve shape which is constrained to pass through specified data points. If the data points defining the curve shape contain some random errors, then they are fitted by a curve in some 'best' approximation sense, e.g. using least squares. On the other hand, the curve design problem is either to create ab initio a shape which satisfies some design constraint or to modify an existing mathematically defined shape.

Many methods already exist for interpolating data, ranging from global polynomial interpolation to various piecewise polynomial interpolation schemes, including cubic splines. For curve tracing, excellent results are achieved with these methods. However, they are usually ineffective for interactive curve design. This is due to the fact that control of the curve shape by numerical specification of both direction and magnitude of tangent vectors does not provide the feel required for curve design. In addition, the cubic curve fitting technique specifies a curve of unique order, which does not vary from spline to spline. In order to increase flexibility, more points must be input, creating more splines. An alternative method of curve description has been described by Bezier [23]. This allows greater flexibility in the generation of desired shapes and gives a feel for the relationship between input and output.

As indicated by Forrest [25], the contrast between curve fitting and design is analogous to that between the draughtsman's (physical) splines and french curves.

Physically, a spline curve is obtained by bending a thin metal or wood lath round pins so that the curve passes through the given data points and assumes a shape of minimum internal energy. In this case the draughtsman need only specify the data points and the spline will do the rest of the work. Using a french curve, however, the designer must select from a set of rigid curve templates a particular curve which will pass through a series of points, and the complete curve will be constructed in a piecewise manner from several such selections.

The work reported in this chapter is concerned with the development and implementation of an interactive system using interpolatory curve fitting and in particular, cubic splines. Some applications require accurate and rapid graphical representation of known data where the technique of cubic spline fitting can be useful. For example, it can be particularly effective in an academic environment when used with low cost devices for graphical output (e.g. Tektronix 4010) to display numerical results which must satisfy known mathematical and physical boundary conditions.

Curves can be represented analytically in two basic forms, parametric and explicit. The use of explicit forms is largely confined to planar curves (i.e. two-dimensional) whereas the parametric form is easily extended to three-dimensional space curves.

$$\text{Explicit form:-} \qquad y = f(x) \tag{6.1}$$

$$\text{Parametric form:-} \qquad \left.\begin{aligned} x &= f(t) \\ y &= g(t) \\ z &= h(t) \end{aligned}\right\} \tag{6.2}$$

The parametric representation has several notable advantages over the non-parametric forms. Each set of coordinate values represents a unique point which can be computed by substitution of a single parametric value. It is possible to express a parametric curve in matrix form and to use identical algorithms for computing x,y and z. Thus we can describe the curve in such a way that the form of the mathematical expressions for x,y and z does not change according to the orientation of the coordinate axes. In the explicit form y=f(x) it may be convenient to describe a given curve with the axes in one orientation, though difficulties arise when f(x) is not single valued. The slope of an explicit curve will be infinite or zero if the curve is parallel to one of the axes. The problem of infinite numbers does not usually arise with the parametric representation.

2. THE IDF NUMERICAL ALGORITHMS

2.1 Interpolation

From the classical theory it is known that a unique polynomial of degree n can be passed through n+1 data points. Polynomials also have the advantage of being fairly easy to handle computationally. Also, such polynomials are continuously differentiable up to order n. This high degree of continuity might suggest a pleasing smooth behaviour, but often even the low-order derivatives are of such great magnitude that undesirable oscillations are displayed in the curve. One cause of the failure of polynomial interpolation to represent data properly is the extreme dependency of the entire curve on each individual data point; a slight movement of a point even at one end can radically affect the curve shape. Visually one desires more local dependency. If a data point is altered slightly, the curve should adjust slightly in the neighbourhood of the point and be nearly unaffected away from the point. In an effort to decrease the curve's sensitivity to each data point, the set of functions known as polynomial splines was introduced by I.J. Schoenberg in 1946. Mathematically, a spline is a piecewise polynomial of degree n with continuity of order n-1 at the common joints between adjacent segments.

The following sections contain the mathematical background of the numerical algorithms used in the implementation of the IDF system. These algorithms mainly employ cubic splines; however, two other algorithms are also included for comparison. One uses the Newton form of global polynomial [26] and the other uses a piecewise quintic polynomial [27]. The program listings of these algorithms is given in Appendix 2.1.

2.2 Global Polynomial (Newton Form)

A general method (classical) for constructing a global polynomial $P_n(x)$ which fits a given function or data exactly at a number of arbitrary spaced points (x_i, i=0(1)n) is the Lagrange form

$$P_n(x) = \sum_{k=0}^{n} [f(x_k) \prod_{\substack{j=0 \\ j\neq k}}^{n} \frac{(x-x_j)}{(x_k-x_j)}] \quad . \tag{6.3}$$

or

$$P_n(x) = \sum_{k=0}^{n} f(x_k)\ L_k(x) \tag{6.4}$$

Note the Lagrangian coefficients $L_k(x)$ are independent of the function values and depend only on the set of points x_i. A curve passing through these points using this polynomial is smooth in the sense of being maximally often differentiable. As the number of points increases, these polynomials can oscillate strongly with a direct dependence on the arrangement of the points.

The Lagrange representation has the defect that if other data points were added, then the new higher degree interpolating polynomial could not be obtained by easily modifying the previous one. Lagrange interpolation is also inefficient when several interpolations are required for the same data. A representation which does not have these disadvantages is the Newton form of interpolating polynomial

$$P_n(x) = a_0+(x-x_0)a_1+(x-x_0)(x-x_1)a_2+\ldots+(x-x_0)\ldots(x-x_{n-1})a_n \tag{6.5}$$

a_k is called the k^{th} ordered divided difference and is usually expressed in the form

$$a_0 = f[x_0] \tag{6.6}$$

$$a_k = f[x_0,x_1,\ldots x_k] \qquad k=1(1)n \tag{6.7}$$

By comparison with the corresponding Lagrangian polynomial in (6.3)

$$a_n = f[x_0,x_1,\ldots x_n] = \sum_{k=0}^{n} [\frac{f(x_k)}{\prod\limits_{\substack{j=0\\ j\neq k}}^{n} (x_k-x_j)}] \tag{6.8}$$

In the divided difference form

$$f[x_0,x_1,\ldots,x_n,x_\alpha,x_\beta] = \frac{f[x_0,x_1,\ldots,x_n,x_\alpha]-f[x_0,x_1,\ldots,x_n,x_\beta]}{x_\alpha-x_\beta} \tag{6.9}$$

This leads to a systematic way of calculating the coefficients a_k from the divided difference table:

$$
\begin{array}{llllll}
x_0 & f_0 & & & & \\
 & & \dfrac{f_1-f_0}{x_1-x_0} = f_{01} & & & \\
x_1 & f_1 & & f_{012} & & \\
 & & \dfrac{f_2-f_1}{x_2-x_1} = f_{12} & & f_{0123} & \\
x_2 & f_2 & & f_{123} & & f_{01234} \\
 & & \dfrac{f_3-f_2}{x_3-x_2} = f_{23} & & f_{1234} & \\
x_3 & f_3 & & f_{234} & & \\
 & & \dfrac{f_4-f_3}{x_4-x_3} = f_{34} & & & \\
x_4 & x_4 & & & &
\end{array}
$$

This algorithm is suitable when interpolated values are required at a large number of points since only one evaluation of the table is performed. A single subroutine (NEWTON) was written to implement this algorithm. It involves firstly the computation of the coefficients for a given set of points, followed by a nested multiplication for each interpolation.

The main objection to polynomial interpolation at a large number of points is that the calculation and evaluation of interpolating polynomials become costly and unreliable.

The error of the interpolating polynomial $P_n(x)$ is

$$R_n(x) = f(x)-P_n(x) \tag{6.10}$$

where f(x) is the function (given or implied) which corresponds to the data.

It can be shown that

$$R_n(x) = \frac{1}{(n+1)!}\, f^{(n+1)}(\zeta) \prod_{i=0}^{n} (x-x_i) \tag{6.11}$$

where ζ is some point in the interval 'I' bounded by the largest and the smallest of the numbers x_i and x. If we wish to give an estimate of the error in approximating f(x) by $P_n(x)$ we must know the magnitude of $f^{(n+1)}(\zeta)$ or its upper bound on the interval 'I'. A small interpolation error over 'I' can usually be expected only if

$$\max_{x\varepsilon I} |f^{(n+1)}(x)| \quad \text{and} \quad \max_{x\varepsilon I} |\prod_{j=0}^{n}(x-x_j) \qquad \text{are small.}$$

It follows that the only way to guarantee a small error is to make the interval 'I' small. Since the interval over which f(x) is to be approximated is usually given in advance, this can be accomplished only by partitioning this interval into sufficiently small subintervals and approximating f(x) in each subinterval by a suitable low-order polynomial. This leads to 'piecewise polynomial interpolation'. The entire curve is produced by joining the curve segments together for given continuity conditions.

2.3 Piecewise Quintic Polynomial Interpolation

A method proposed by Maude [27] is developed here for interpolation from a given set of data points in the plane and for fitting a smooth curve to these points. It is based on a piecewise function composed of a set of polynomials applicable to successive intervals of the given points.

Basically, over any interval two local interpolating polynomials are found, and a weighted average is taken, the weight being a function of the independent variables, with suitable smoothing properties.

In the case of a one-dimensional curve, two second order polynomials could be used in the range x_n to x_{n+1} as shown in Figure 6.1

F_n - fitting exactly f_{n-1}, f_n, f_{n+1}

and F_{n+1} - fitting exactly f_n, f_{n+1}, f_{n+2}

The function F_n and F_{n+1} are combined by weight function W to obtain the new interpolating function F over the interval $[x_n, x_{n+1}]$

$$F = W\,F_n + (1-W)F_{n+1} \tag{6.12}$$

where W is a function of x such that

$$\left.\begin{aligned} W(x_n) &= 1 \\ W(x_{n+1}) &= 0 \\ \left(\frac{dW}{dx}\right)_n &= \left(\frac{dW}{dx}\right)_{n+1} = 0 \end{aligned}\right\} \tag{6.13}$$

which ensures first and second derivative continuity in F. The weight function used to yield the desired smoothness is

$$W = 1-3X^2 + 2X^3 \tag{6.14}$$

where $$X = \frac{x-x_n}{x_{n+1}-x_n} = \frac{\Delta x}{\Delta x_n}$$

The coefficients of the quintic polynomials generated by this algorithm for each interval are found as follows:

First consider the two quadratic polynomials F_n and F_{n+1} passing through the points a,b,c and b,c,d respectively as shown in Figure 6.1. Then we have,

$$F_n = a_1 + b_1X + c_1X^2 \tag{6.15}$$

$$F_{n+1} = a_2 + b_2X + c_2X^2 \tag{6.16}$$

Therefore, the polynomial F is constructed by taking the weighted average of (6.15) and (6.16) as given in (6.12). This gives

$$F = a_1+b_1X+(c_1-3(a_1-a_2))X^2+(2(a_1-a_2)+3(b_2-b_1))X^3 + (2(b_1-b_2)+3(c_2-c_1))X^4+2(c_1-c_2)X^5$$

$\therefore$ the quintic polynomial is

$$F = A + BX + CX^2 + DX^3 + EX^4 + FX^5 \tag{6.17}$$

where

$$\left.\begin{array}{ll} A = a_1 & D = 2(a_1-a_2)+3(b_2-b_1) \\ B = b_1 & E = 2(b_1-b_2)+3(c_2-c_1) \\ C = c_1-3(a_1-a_2) & F = 2(c_1-c_2) \end{array}\right\} \quad (6.18)$$

The coefficients in (6.18) are determined from the function values given at the abscissae x_{n-1}, x_n, x_{n+1} and x_{n+2}.

Now by substituting the given function values in (6.15), we obtain

$$\left.\begin{array}{ll} f_{n-1} & = a_1 - b_1\left(\frac{\Delta x_{n-1}}{\Delta x_n}\right) + c_1\left(\frac{\Delta x_{n-1}}{\Delta x_n}\right)^2 \\ f_n & = a_1 \\ f_{n+1} & = a_1+b_1+c_1 \end{array}\right\} \quad (6.19)$$

Similarly, substituting in (6.16), we have

$$\left.\begin{array}{ll} f_n & = a_2 \\ f_{n+1} & = a_2+b_2+c_2 \\ f_{n+2} & = a_2+b_2\left(1+\frac{\Delta x_{n-1}}{\Delta x_n}\right) + c_2\left(1+\frac{\Delta x_{n-1}}{\Delta x_n}\right)^2 \end{array}\right\} \quad (6.20)$$

By solving equations (6.19) and (6.20) for a_1, a_2, b_1, b_2, c_1 and c_2 and substituting in equations (6.18), we obtain

$$\left.\begin{array}{ll} A = f_n & D = -3T_0 \\ B = \frac{D1+R^2 D2}{R(1+R)} & E = 5T_0 \\ C = \frac{RD2-D1}{R(1+R)} & F = -2T_0 \end{array}\right\} \quad (6.21)$$

where

$$\left.\begin{array}{l} D1 = f_n - f_{n-1} \\ D2 = f_{n+1} - f_n \\ D3 = f_{n+2} - f_{n+1} \\ T_0 = \frac{S(1+S)D1+R(1+R)D3-RS(2+R+S)D2}{RS(1+R)(1+S)} \\ R = \frac{\Delta x_{n-1}}{\Delta x_n} \quad \text{and} \quad S = \frac{\Delta x_{n+1}}{\Delta x_n} \end{array}\right\} \quad (6.22)$$

(R=S=1 for equidistant data points along the x-axis)

The coefficients of the quintic polynomial in (6.21) determine uniquely the portion of the curve in the interval $[x_n, x_{n+1}]$. However, in order to determine the two end portions of the curve, an extra point is assumed to exist beyond each end of the curve as shown in Figure 6.2. Effectively, the introduction of these imaginary end points would determine the nature of the end conditions for the whole curve. Consequently, the 'D' values resulting from the new portion must be specified. The algorithm was adapted so that these values are made controllable by the user. Thus three forms of end condition were considered in order to give the user the choice to vary the boundary condition at each end as required. These end conditions are made analogous to those suggested by Spath [28] for cubic splines. These are as shown below:

	End condition	First end	Last end
(i)	CLAMPED ($\frac{dy}{dx}$ is specified)	user must specify the slope (as D1)	user must specify the slope (as D3)
(ii)	RELAXED ($\frac{d^2y}{dx^2} = 0$)	set D1=D2	set D3=D2
(iii)	PARABOLIC ($\frac{d^2y}{dx^2}$ = constant)	$T_0 = 0$ i.e. set D1=2D2-D3 since R=S=1 is assumed	$T_0 = 0$ i.e. D3=2D2-D1 R=S=1 is assumed

A subroutine (PIECWS) was written to implement the above algorithm which fits a smooth curve to a given set of input data points in the x-y plane.

Another method was proposed by AKIMA [29]: this is based on a piecewise function composed of a set of polynomials, each of degree

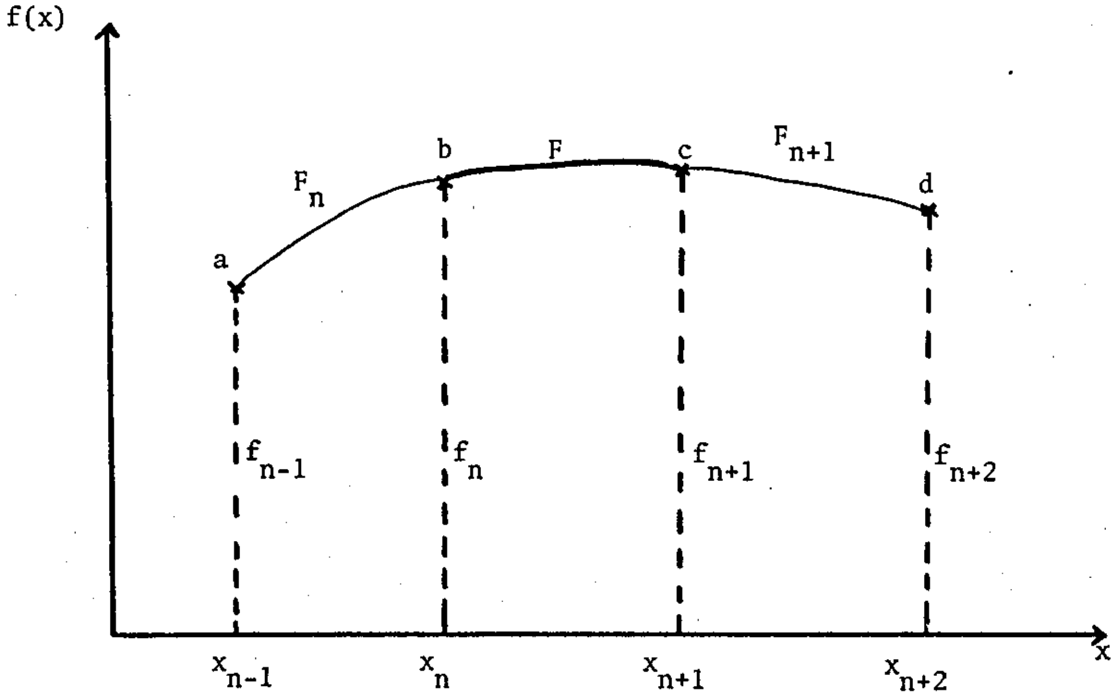


FIGURE 6.1: Four Points Fitted with Two Quadratic Polynomials F_n and F_{n+1}

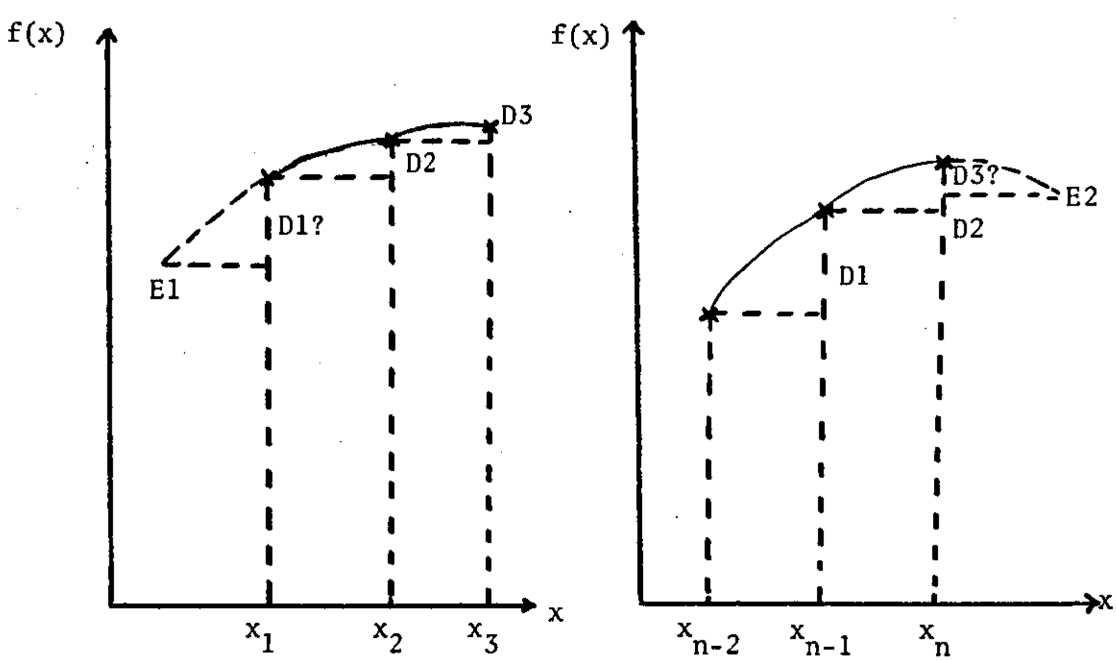


FIGURE 6.2: Addition of Extra Two Points at the Ends of the Curve

three with slope at the junction points locally determined under a geometrical condition. The slope is determined by the coordinates of five points, with the point in question as a centre point, and two points on each side of it; the resulting interpolation is independent of the axes used.

In both methods each individual polynomial is determined locally and additional conditions are needed to fix the two ends. However, the Maude method does not evaluate the slope of the curve at each point and only uses four points locally to determine each portion of the curve.

2.4 Cubic Spline Polynomial Interpolation

A cubic spline is a piecewise cubic polynomial, having the property of continuity of slope and curvature throughout its length. The use of cubic splines means that n-1 cubic polynomials are required for the data points $(x_k, f(x_k))$, $k=1(1)n$. Since a cubic is the lowest degree polynomial which can twist in space and have inflection points, cubic splines are commonly used for curve fitting. The use of low-degree polynomials reduces the computational requirements and reduces numerical instabilities that arise with higher order curves. Also, their use corresponds to minimization of the quantity

$$J = \int (f''(x))^2 dx \tag{6.23}$$

subject to certain boundary conditions. Minimization of J approximates minimization of the integral of squared curvature K along the curve, which has the important physical justification that a thin flexible strip passed through the data points takes a configuration which minimises K.

Consider the cubic spline

$$y_k = f_k(x) = A_k(x-x_k)^3+B_k(x-x_k)^2+C_k(x-x_k)+D_k \tag{6.24}$$

or

$$y_k = f_k(x) = A_k\Delta x_k^3+B_k\Delta x_k^2+C_k\Delta x_k+D_k \tag{6.25}$$

This is the explicit form (non-parametric) of cubic spline function, since y is explicitly dependent on x. Therefore, a two-dimensional (planar) curve can be expressed as in (6.25), provided that the curve progresses from left to right. However, if $\frac{dy}{dx}$ is infinite at any point, this simple form would break down and must be replaced by the parametric form:

$$x_k(t) = A_k\Delta t^3 + B_k t^2 + C_k\Delta t + D_k \tag{6.26}$$

$$y_k(t) = E_k\Delta t^3 + F_k\Delta t^2 + G_k\Delta t + H_k \tag{6.27}$$

The parameter t can be chosen at will as long as it increases steadily as we progress along the curve; but it is usually advantageous to make it approximately proportional to the arc length from the first point to the point in question. The other advantage of the parametric spline formulation is that it extends easily to three-dimensional problems. In such cases the z-coordinate is parametrically:

$$z(t) = P_k\Delta t^3 + Q_k\Delta t^2 + R_k\Delta t + S_k \tag{6.28}$$

The problem is to determine the coefficients of the above equations for each interval, subject to first and second derivative continuity conditions. Now consider an interval $[x_k, x_{k+1}]$. Six possible boundary conditions can be set (Figure 6.3) involving function values, slopes and second derivatives at both ends of the interval. Using the explicit form (6.25) these are given by:

$$\left.\begin{aligned}
y_k &= f_k(x_k) = D_k \\
y_{k+1} &= f_k(x_{k+1}) = A_k\Delta x_k^3 + B_k\Delta x_k^2 + C_k\Delta x_k + D_k \\
y'_k &= f'_k(x_k) = C_k \\
y'_{k+1} &= f'_k(x_{k+1}) = 3A_k\Delta x_k^2 + 2B_k\Delta x_k + C_k \\
y''_k &= f''_k(x_k) = 2B_k \\
y''_{k+1} &= f''_k(x_{k+1}) = 6A_k\Delta x_k + 2\,B_k
\end{aligned}\right\} \tag{6.29}$$

From these boundary equations, the desired coefficients could be expressed in terms of the function values and two derivatives. In the first derivative form, the coefficients are:

$$\left.\begin{aligned} A_k &= \frac{1}{\Delta x_k^2}\left(-2\frac{\Delta y_k}{\Delta x_k} + y'_k + y'_{k+1}\right) \\ B_k &= \frac{1}{\Delta x_k}\left(3\frac{\Delta y_k}{\Delta x_k} - 2y'_k - y'_{k+1}\right) \\ C_k &= y'_k \\ D_k &= y_k \end{aligned}\right\} \tag{6.30}$$

Similarly in the second derivative form the coefficients are:

$$\left.\begin{aligned} A_k &= \frac{1}{6\Delta x_k}(y''_{k+1} - y''_k) \\ B_k &= \frac{1}{2} y''_k \\ C_k &= \frac{\Delta y_k}{\Delta x_k} - \frac{1}{6}\Delta x_k(y''_{k+1} + 2y''_k) \\ D_k &= y_k \end{aligned}\right\} \tag{6.31}$$

Therefore, in order to evaluate the coefficients, we need to determine either the first or the second derivatives by using the continuity conditions at the interior node points.

Considering continuity in the first derivative, and using equations (6.29), we have

at $x=x_k$

$$y'_{k-1}(x_k) = y'_k(x_k) \qquad \text{(for } k=2(1)n-1\text{)} \tag{6.32}$$

which gives the coupling conditions

$$3A_{k-1}\Delta x_k^2 + 2B_{k-1}\Delta x_k + C_{k-1} = C_k \tag{6.33}$$

By replacing the coefficients in equation (6.33) from equation (6.31), it yields the system of linear equations:

$$\Delta x_{k-1}y''_{k-1}+2(\Delta x_{k-1}+\Delta x_k)y''_k + \Delta x_k y''_{k+1} = 6\left(\frac{\Delta y_k}{\Delta x_k} - \frac{\Delta y_{k-1}}{\Delta x_{k-1}}\right) \tag{6.34}$$

for $k=2(1)n-1$, or in vector form:

$$[\Delta x_{k-1} \;\; 2(\Delta x_{k-1}+\Delta x_k) \;\; \Delta x_k] \begin{bmatrix} y''_{k-1} \\ y''_k \\ y''_{k+1} \end{bmatrix} = 6\left[\frac{\Delta y_k}{\Delta x_k} - \frac{\Delta y_{k-1}}{\Delta x_{k-1}}\right] \tag{6.35}$$

for $k=2(1)n-1$.

This represents n-2 simultaneous linear equations for the n unknown y''_k. The two extra unknowns y''_1 and y''_n naturally arise from the absence of a continuity condition at the end points x_1 and x_n. Hence, the system of equations can be solved if the two boundary conditions are given and thus a unique cubic spline curve is determined. Similarly, applying second derivative continuity, we obtain n-2 simultaneous linear equations for n unknowns in y'_k as

$$[\Delta x_1 \quad 2(\Delta x_{k-1}+\Delta x_k) \quad \Delta x_{k-1}] \begin{bmatrix} y'_{k-1} \\ y'_k \\ y'_{k+1} \end{bmatrix} = 3\left[\frac{\Delta x_k}{\Delta x_{k-1}} \Delta y_{k-1} + \frac{\Delta x_{k-1}}{\Delta x_k} \Delta y_k\right] \qquad (6.36)$$

for k=2(1)n-1.

Again, two extra conditions are required to obtain a unique cubic spline curve.

The boundary conditions required by equations (6.35) and (6.36) could take different forms depending on the nature of the problem. Hence, the IDF system contains a set of program subroutines for generating cubic splines, allowing the user various options for specifying interactively the form of the end condition. The forms of end condition available at present are:

(1) Prescribed first or second derivatives

(2) Cyclic or anticyclic behaviour [30]

(3) Variable end conditions

Next let us consider the effect of the various forms of end condition on the system of linear equations (6,35) and (6.36). Since these equations are equivalent formulations of the same problem either form can be used.

(1) Prescribed First or Second Derivatives

(i) First derivative end condition.

In this case the values y'_1 and y'_n at x_1 and x_n are available. These provide the two additional equations needed in (6.35) to solve for $y''_1 \ldots y''_n$. By using equations (6.29) and (6.31) we arrive at the following two boundary equations:

$$\left.\begin{aligned} 2\Delta x_1 y''_1 + \Delta x_1 y''_2 &= 6\left(\frac{\Delta y_1}{\Delta x_1} - y'_1\right) \\ \Delta x_{n-1} y''_{n-1} + 2\Delta x_{n-1} y''_n &= 6\left(y'_n - \frac{\Delta y_{n-1}}{\Delta x_{n-1}}\right) \end{aligned}\right\} \tag{6.37}$$

The set of equations (6.35) together with (6.37) yields a system of linear equations whose $n \times n$ coefficient matrix is tridiagonal, symmetric, and diagonally dominant with positive diagonal elements. The coefficient matrix shown in (6.38) is clearly non-singular, thus guaranteeing the existence and uniqueness of the cubic spline for a given data set.

$$\begin{bmatrix} 2\Delta x_1 & \Delta x_1 & & & \\ \Delta x_1 & 2(\Delta x_1+\Delta x_2) & \Delta x_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \Delta x_{n-2} & 2(\Delta x_{n-2}+\Delta x_{n-1}) & \Delta x_{n-1} \\ & & & \Delta x_{n-1} & 2\Delta x_{n-1} \end{bmatrix} \tag{6.38}$$

The boundary values could be estimated approximately from the given data points defining the curve as

$$y'_1 = \frac{\Delta y_1}{\Delta x_1} \quad \text{and} \quad y'_n = \frac{\Delta y_{n-1}}{\Delta x_{n-1}}$$

The values of the slopes at both end are declared by the user. This form of boundary condition is sometimes termed by Engineers as clamped or encastred.

(ii) Second derivative end condition

Two possible ways of specifying the second derivative boundary values are open to the user.

(a) y_1'' and y_n'' are directly declared by the user. Equations (6.35) are slightly rearranged as shown in (6.39) which has a (n-2)×(n-2) tridiagonal coefficient matrix and can be uniquely solved for $y_2''\ldots\ldots y_{n-1}''$.

$$\begin{bmatrix} 2(\Delta x_1+\Delta x_2) & \Delta x_2 & & \\ \Delta x_2 & 2(\Delta x_2+\Delta x_3) & \Delta x_3 & \\ & \ddots & \ddots & \ddots \\ & & \Delta x_{n-2} & 2(\Delta x_{n-2}+\Delta x_{n-1}) \end{bmatrix} \begin{bmatrix} y_2'' \\ y_3'' \\ \vdots \\ y_{n-1}'' \end{bmatrix} = \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_n \end{bmatrix} \quad (6.39)$$

where
$$B_1 = 6\left(\frac{\Delta y_2}{\Delta x_2} - \frac{\Delta y_1}{\Delta x_1}\right) - \Delta x_1\, y_1'' ,$$

$$B_n = 6\left(\frac{\Delta y_{n-1}}{\Delta x_{n-1}} - \frac{\Delta y_{n-2}}{\Delta x_{n-2}}\right) - \Delta x_{n-1} y_n''$$

and
$$B_k = 6\left(\frac{\Delta y_{k+1}}{\Delta x_{k+1}} - \frac{\Delta y_k}{\Delta x_k}\right) \qquad \text{for } k=2(1)n-1.$$

For the special case where $y_1''=y_n''=0$, the cubic spline is said to have natural or relaxed boundary conditions. This natural spline ending should be used if for example the data is sinusoidal.

(b) Sometimes boundary conditions of the form:

$$y_1'' = uy_2'' \qquad \text{and} \qquad vy_{n-1}'' = y_n'' \quad (6.40)$$

are useful. In this case, though u and v are arbitrary scalars, they are in practice often taken equal to unity, thereby making the second derivatives at x_1 and x_n equal to those at x_2 and x_{n-1} respectively. This is often termed P-spline [30] in which the end segments are parabolic. Another common choice is to take u=v=1/2.

By substituting for y_1'' and y_2'' from (6.40) into equation (6.35), we obtain the slightly modified boundary equations:

$$\left.\begin{aligned} ((2+u)\Delta x_1+2\Delta x_2)y_2''+\Delta x_2 y_3'' &= 6\left(\frac{\Delta y_2}{\Delta x_2}-\frac{\Delta y_1}{\Delta x_1}\right) \\ \Delta x_{n-2}y_{n-2}''+(2\Delta x_{n-2}+(2+v)\Delta x_{n-1})y_{n-1}' &= 6\left(\frac{\Delta y_{n-1}}{\Delta x_{n-1}}-\frac{\Delta y_{n-2}}{\Delta x_{n-2}}\right) \end{aligned}\right\} \qquad (6.41)$$

This means that the coefficient matrix in (6.39) is only modified in two places viz. the element at the top left and the element at the bottom right. The existence and uniqueness of cubic splines with this boundary condition is ensured as long as u and v are chosen so that the matrix remains positive definite. This is guaranteed if

$$\left.\begin{aligned} -(2\frac{\Delta x_2}{\Delta x_1}+2) &\leqslant u < \infty \\ -(\frac{\Delta x_{n-2}}{\Delta x_{n-1}}+2) &\leqslant v < \infty \end{aligned}\right\} \qquad (6.42)$$

The effect of these different boundary conditions on the shape of the curve would be mainly noticeable at the end intervals.

In all the above cases, the Gaussian elimination process [28] was employed to solve the linear system of equations. The elements of the matrix and of the R.H.S. are only calculated at the place where they are required so as to save storage.

(2) Cyclic or Anticyclic Behaviour

Cubic splines with periodic boundary conditions are particularly suitable for the representation of closed smooth curves or a portion of a curve which repeats at intervals.

(i) Cyclic end condition in which the first and second derivatives at one end of the curve have the same values as those at the other end. Mathematically, these conditions are expressed as follows:

$$y_1^{(i)}(x_1) = y_n^{(i)}(x_n) \qquad \text{for } i=0,1,2 \qquad (6.43)$$

Hence from equations (6.37) we obtain

$$2(\Delta x_1+\Delta x_{n-1})y_1''+\Delta x_1 y_2''+\Delta x_{n-1}y_{n-1}'' = 6\left(\frac{\Delta y_1}{\Delta x_1} - \frac{\Delta y_{n-1}}{\Delta x_{n-1}}\right) \tag{6.44}$$

and for k=n-1 in equation (6.35),

$$\Delta x_{n-2}y_{n-2}''+2(\Delta x_{n-2}+\Delta x_{n-1})y_{n-1}''+\Delta x_{n-1}y_1'' = 6\left(\frac{\Delta y_{n-1}}{\Delta x_{n-1}} - \frac{\Delta y_{n-2}}{\Delta x_{n-2}}\right) \tag{6.45}$$

In matrix form, the system becomes

$$\begin{bmatrix} 2(\Delta x_{n-1}+\Delta x_1) & \Delta x_1 & & & \Delta x_{n-1} \\ \Delta x_1 & 2(\Delta x_1+\Delta x_2) & \Delta x_2 & & \\ & \ddots & \ddots & \ddots & \\ & & & & \\ \Delta x_{n-1} & & & \Delta x_{n-2} & 2(\Delta x_{n-2}+\Delta x_{n-1}) \end{bmatrix} \begin{bmatrix} y_1'' \\ y_2'' \\ \vdots \\ \vdots \\ y_{n-1}'' \end{bmatrix} = \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ \vdots \\ B_{n-1} \end{bmatrix} \tag{6.46}$$

where
$$B_1 = 6\left(\frac{\Delta y_1}{\Delta x_1} - \frac{\Delta y_{n-1}}{\Delta x_{n-1}}\right)$$

and
$$B_k = 6\left(\frac{\Delta y_k}{\Delta x_k} - \frac{\Delta y_{k-1}}{\Delta x_{k-1}}\right) \qquad \text{for } k=2(1)n-1$$

The (n-1)×(n-1) coefficient matrix of (6.46) is not strictly tridiagonal and has off-diagonal entries in the corners of the first and last rows. This is a cyclic tridiagonal matrix and is positive definite. (The associated spline is useful for curves in which the first and last points are coincident). This system of equations (6.46) is solved by using the Normalised Periodic Tridiagonal Algorithm developed by Benson [31] from the matrix factorization method.

The cyclic end condition can be used without the knowledge of the initial slope and the initial data point can be at any location on the closed curve.

(ii) Anticyclic end condition is similar to the cyclic condition except that antisymmetry in both slope and curvature is imposed. The antisymmetry condition is expressed as

$$\left.\begin{aligned} y_1'(x_1) &= -y_n'(x_n) \\ \text{and} \qquad y_1''(x_1) &= -y_n''(x_n) \end{aligned}\right\} \tag{6.47}$$

In this case, equations (6.36) which involve unknown y_k' will be used as a basis for the system of equations.

From equations (6.29), we have

$$\left.\begin{aligned} y_1''(x_1) &= 2B_1 \\ \text{and} \qquad y_n''(x_n) &= 6A_{n-1}\Delta x_{n-1} + 2B_{n-1} \end{aligned}\right\} \tag{6.48}$$

Using the second condition in (6.47), we can write from (6.48)

$$B_1 = 3A_{n-1}\Delta x_{n-1} + B_{n-1} \tag{6.49}$$

Substituting from (6.30) into (6.49), we obtain for k=1

$$2(\Delta x_1+\Delta x_{n-1})y_1' + \Delta x_{n-1}y_2' - \Delta x_1 y_{n-1}' = 3\left(\frac{\Delta x_{n-1}}{\Delta x_1}\Delta y_1 - \frac{\Delta x_1}{\Delta x_{n-1}}\Delta y_{n-1}\right) \tag{6.50}$$

and for k=n-1, using (6.36) we have

$$\Delta x_{n-1}y_{n-2}' + 2(\Delta x_{n-2}+\Delta x_{n-1})y_{n-1}' - \Delta x_{n-2}y_1' = 3\left(\frac{\Delta x_{n-1}}{\Delta x_{n-2}}\Delta y_{n-2} + \frac{\Delta x_{n-2}}{\Delta x_{n-1}}\Delta y_{n-1}\right) \tag{6.51}$$

In matrix form, the anticyclic cubic spline $(n-1)\times(n-1)$ coefficient matrix is represented as

$$\begin{bmatrix} 2(\Delta x_1+\Delta x_{n-1}) & \Delta x_{n-1} & & & -\Delta x_1 \\ \Delta x_2 & 2(\Delta x_2+\Delta x_1) & \Delta x_1 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \ddots \\ -\Delta x_{n-2} & & & \Delta x_{n-1} & 2(\Delta x_{n-1}+\Delta x_{n-2}) \end{bmatrix} \tag{6.52}$$

(3) <u>Variable end conditions</u>

The above cubic spline algorithms allow the user to select the same (or related) forms of boundary condition at both ends. The variable end condition provides maximum control of the spline. This enables the user to specify different forms of boundary condition at each end of the curve. This set was suggested in the first place by Nutbourne [30]. In this paper the algorithm is based upon recurrence formulae which are derived by relating the properties of the j+1 interval to the j interval. The algorithm discussed here is again based on a matrix approach which is valid for both the explicit and the parametric forms.

Consider equation (6.36). When it is applied recursively at all interior points (k=2(1)n-1) the resultant coefficient matrix has dimension $(n-2)\times n$ as shown in (6.53).

$$
\begin{array}{l}
\text{End 1} \\ \\ \\ \\ \\ \\ \text{End 2}
\end{array}
\left[\begin{array}{cccccc}
 & & ? & & & \\ \hline
\Delta x_2 & 2(\Delta x_2+\Delta x_1) & \Delta x_1 & & & \\
 & \Delta x_3 & 2(\Delta x_3+\Delta x_2) & \Delta x_1 & & \\
 & & \ddots & \ddots & \ddots & \\
 & & & & & \\
 & & \Delta x_{n-1} & 2(\Delta x_{n-1}+\Delta x_{n-2}) & \Delta x_{n-2} & \\ \hline
 & & ? & & &
\end{array}\right]
\left[\begin{array}{c}
y'_1 \\ \hline y'_2 \\ y'_3 \\ \\ \\ \\ \hline y'_n
\end{array}\right]
=
\left[\begin{array}{c}
B_1 \\ \hline B_2 \\ B_3 \\ \\ \\ B_{n-1} \\ \hline B_n
\end{array}\right]
\qquad (6.53)
$$

where

$$B_k = 3\left(\frac{\Delta x_k}{\Delta x_{k-1}}\,\Delta y_{k-1} + \frac{\Delta x_{k-1}}{\Delta x_k}\,\Delta y_k\right) \qquad k=2(1)n-1$$

The strategy of this algorithm is to construct the two boundary equations required to augment equations (6.53). Each boundary equation is formed independently using a particular end condition. Hence, the algorithm gives the user the ability to have full control over the form of the end condition required. As will be seen later (next section), this capability is provided through the appropriate selection of menu options. This variable end condition is sometimes desirable if only a few points are known or if physical constraints require accurate control of the curve shape at the ends.

The forms of the end conditions provided to set up the boundary equations are given below:

(i) Clamped.

The first derivative is specified to have a certain value as mentioned before. This would simply mean that the two boundary equations are:

$$y_1' = B_1 \qquad \text{and} \qquad y_n' = B_n \tag{6.54}$$

Thus, in matrix form:

$$\left[\begin{array}{cccc} 1 & 0 & & \\ \hline & \ddots & \ddots & \ddots \\ \hline & & 0 & 1 \end{array}\right] \left[\begin{array}{c} y_1' \\ \hline \vdots \\ \hline y_n' \end{array}\right] = \left[\begin{array}{c} B_1 \\ \hline \vdots \\ \hline B_n \end{array}\right] \tag{6.55}$$

(ii) Natural or Relaxed.

This is defined as $y''(x_1)=y''(x_n)=0$.

Using equations (6.29) and (6.30), we obtain the following two equations:

at $x=x_1$

$$y_1' + \frac{1}{2}y_2' = \frac{3}{2}\,\frac{\Delta y_1}{\Delta x_1} \tag{6.56}$$

and at $x=x_n$

$$y_{n-1}' + 2y_n' = 3\,\frac{\Delta y_{n-1}}{\Delta x_{n-1}} \tag{6.57}$$

In matrix form:

$$\left[\begin{array}{cccc} 1 & 0.5 & & \\ \hline & \ddots & \ddots & \ddots \\ \hline & & 1 & 2 \end{array}\right] \left[\begin{array}{c} y_1' \\ \hline \vdots \\ \hline y_n' \end{array}\right] = \left[\begin{array}{c} 1.5 \times \dfrac{\Delta y_1}{\Delta x_1} \\ \hline \vdots \\ \hline 3 \times \dfrac{\Delta y_{n-1}}{\Delta x_{n-1}} \end{array}\right] \tag{6.58}$$

(iii) Parabolic

This requires that the second derivative is constant over the entire end interval. That is,

$$\frac{d^2y}{dx^2} = \text{constant or} \qquad y''(x_1) = y''(x_2)$$

Hence, using equation (6.29) and (6.30), we have

$$2B_1 = 6A_1\,\Delta x_1 + 2B_1$$

$$\therefore \quad A_1 = 0$$

i.e.
$$y_1' + y_2' = 2\,\frac{\Delta y_1}{\Delta x_1} \tag{6.59}$$

Similarly for the other end, we have

$$y''(x_{n-1}) = y''(x_n)$$

$$\therefore\ y_{n-1}' + y_n' = 2\,\frac{\Delta y_{n-1}}{\Delta x_{n-1}} \tag{6.60}$$

In matrix form, we get

$$\left[\begin{array}{cccc} 1 & 1 & & \\ \hline \ddots & \ddots & \ddots & \\ \hline & & 1 & 1 \end{array}\right] \left[\begin{array}{c} y_1' \\ \hline \vdots \\ \hline y_n' \end{array}\right] = \left[\begin{array}{c} 2\,\dfrac{\Delta y_1}{\Delta x_1} \\ \hline \vdots \\ \hline 2\,\dfrac{\Delta y_{n-1}}{\Delta x_{n-1}} \end{array}\right] \tag{6.61}$$

Just to emphasise that the above end conditions do not have to be identical at each end of the spline, any permutation of the above three sets of end condition can be used without affecting the tridiagonal nature of the coefficient matrix. However, the following end condition would produce a non-tridiagonal matrix.

(iv) Q-spline.

This requires that the two adjacent segments at either end have a common cubic equation. This means that the second derivative at the end point and next two points are collinear [30]. The Q-spline predicts the final curvature from the two adjacent values:

this use of higher-order information would be useful in the absence of an alternative choice.

Consider Figure 6.4. From the similarity of the triangles a c d and a b e we have the following equation:

$$[y''(x_2)-y''(x_1)]\Delta x_2 = [y''(x_3)-y''(x_2)]\Delta x_1 \tag{6.62}$$

Substitute into equation (6.62) from equations (6.29) and (6.30); we obtain

$$y_1'+\left(1-\left(\frac{\Delta x_1}{\Delta x_2}\right)\right)y_2' - \left(\frac{\Delta x_1}{\Delta x_2}\right)^2 y_3' = \frac{2}{\Delta x_1}\left(\Delta y_1 - \left(\frac{\Delta x_1}{\Delta x_2}\right)^3 \Delta y_2\right) \tag{6.63}$$

and similarly for the other end

$$-\left(\frac{\Delta x_{n-2}}{\Delta x_{n-1}}\right)^2 y_{n-2}'+\left(1-\left(\frac{\Delta x_{n-2}}{\Delta x_{n-1}}\right)^2\right)y_{n-1}'+y_n' = \frac{2}{\Delta x_{n-1}}\left(\Delta y_n-\left(\frac{\Delta x_{n-2}}{\Delta x_{n-1}}\right)^2\Delta y_{n-1}\right) \tag{6.64}$$

In matrix form:

$$\left[\begin{array}{cccccc}
1 & 1-\left(\frac{\Delta x_1}{\Delta x_2}\right)^2 & -\left(\frac{\Delta x_1}{\Delta x_2}\right)^2 & & & \\
\hline
\Delta x_2 & 2(\Delta x_2+\Delta x_1) & \Delta x_1 & & 0 & \\
\ddots & \ddots & \ddots & \ddots & & \\
& 0 & \ddots & \ddots & \ddots & \\
& & \Delta x_{n-1} & 2(\Delta x_{n-1}+\Delta x_{n-2}) & \Delta x_{n-2} & \\
\hline
& & -\left(\frac{\Delta x_{n-2}}{\Delta x_{n-1}}\right)^2 & 1-\left(\frac{\Delta x_{n-2}}{\Delta x_{n-1}}\right)^2 & 1 &
\end{array}\right]
\left[\begin{array}{c} y_1' \\ \hline y_2' \\ \vdots \\ \vdots \\ y_{n-1}' \\ \hline y_n' \end{array}\right]
=
\left[\begin{array}{c} B_1 \\ \hline B_2 \\ \vdots \\ \vdots \\ B_{n-1} \\ \hline B_n \end{array}\right] \tag{6.65}$$

where $$B_1 = \frac{2}{\Delta x_1}\left(\Delta y_1-\left(\frac{\Delta x_1}{\Delta x_2}\right)^3\Delta y_2\right) \quad ,$$

$$B_n = \frac{2}{\Delta x_{n-1}}\left(\Delta y_n-\left(\frac{\Delta x_{n-2}}{\Delta x_{n-1}}\right)^3 \Delta y_{n-1}\right.$$

and $$B_k = 3\left(\frac{\Delta x_k}{\Delta x_{k-1}}\Delta y_{k-1} + \frac{\Delta x_{k-1}}{\Delta x_k}\Delta y_k\right) \qquad k=2(1)n-1.$$

Again the Q-spline can be used at either or both ends. However, when this ending is used, it would produce a non-tridiagonal matrix, and the tridiagonal Gaussian elimination used so far is no longer suitable.

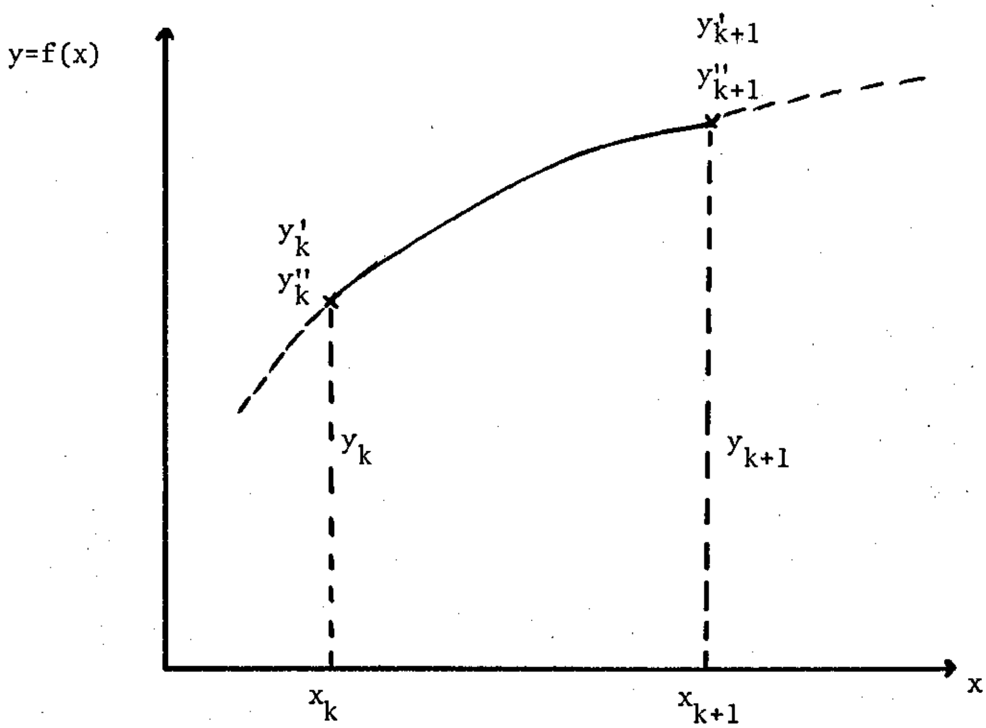


FIGURE 6.3: Fitting Cubic Spline Polynomials

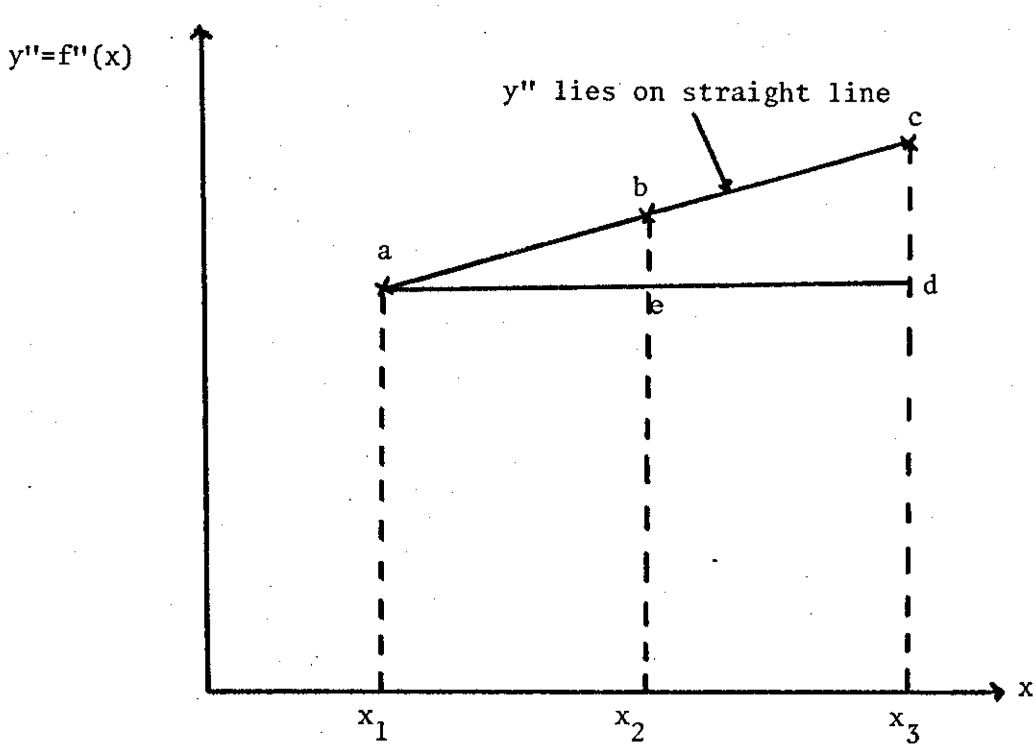


FIGURE 6.4: Q-Spline End Condition

A modified version of Gaussian elimination developed by Evans [32] for quindiagonal systems is used for solving the above equations with all possible combinations of boundary conditions. The algorithm was adapted to avoid any unnecessary computation and storage due to zero elements of the two outer diagonals.

3. THE USER INTERFACE

3.1 An Overall View of the System

The system described here allows a user to enter a set of data points into the computer either via a keyboard or from an on-line file (e.g. disc file), and interactively to specify various ways of fitting, editing and displaying the curve on the Tekronix 4010 storage tube. Thus, the basic problem which is handled by this system is one in which the user presents the machine with a finite set of data points and then manipulates these points until he obtains some curve which is satisfactory to him. The system may also be used as a tool to examine the problem of determining the minimum number of points needed to be able to represent a given input curve. It allows great flexibility in making changes in both input data and output curve interactively. Other uses are also possible as will become apparent in the subsequent discussion.

When designing an interactive system, a very important consideration is the appearance of the system to the user. The system should be designed so that any user can understand what is happening and what is expected of him at each step. The development of an effective but simple man-machine interface is sometimes referred to as the 'human engineering' aspect of the system. Although the IDF system is a special purpose interactive system used by a selected group of users, it is designed with the above aim in mind. As a simple example, words used in menu options should be chosen so as to convey the same meaning to all varieties of user and yet the message should be concise. The system provides checks on user input and reminders or instructional displays on user requests. Consequently, the user's interface is intended to provide the most natural and simple dialogue with the system.

In the design of the IDF system, the user's needs are hopefully being anticipated by presenting him with different options in the form of a menu. Only those options (actions) that are legitimate at a particular step in the data-fitting process are presented for user selection. This avoids the entry of illegal requests which may require elaborate error procedures.

Essentially, from the user point of view, the IDF system consists of a number of logically connected display images as depicted by the flowchart in Figure 6.5. Conceptually, the data-fitting represented by the various displays shown can be described as consisting of three basic phases:

(1) Input specification, which includes the entry of data points and editing.

(2) Numerical Computation required to produce the smooth curve.

(3) Display of the resulting curve and other relevant information.

Each stage would have a number of displays associated with it (some are optional) as illustrated by the dotted lines of Figure 6.5.

Most displays in the system show the following menu options:-

```
+NEXT
+PREVIOUS
+HELP
+RESTART
+EXIT
```

The system response to each individual option is described

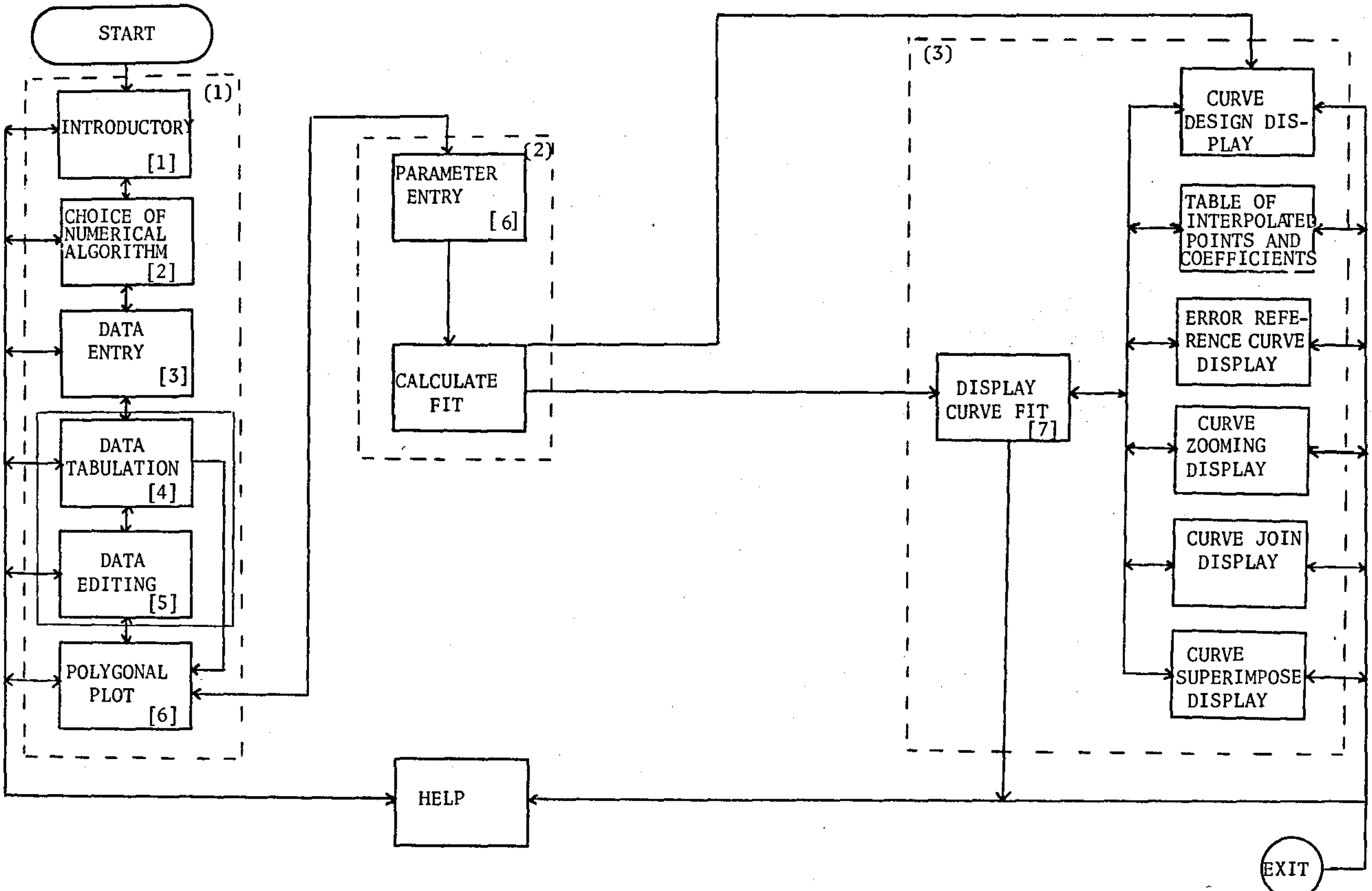


FIGURE 6.5: IDF Displays Flowchart

below:

(a) Move onto the next display. → $\boxed{\text{+NEXT}}$

The current display is erased and the next one in sequence is brought up on the screen.

(b) Move back to the previous display. → $\boxed{\text{+PREVIOUS}}$

The current display is erased and the preceeding one is called.

(c) On-line help. → $\boxed{\text{+HELP}}$

When the user is seeking immediate advice on the system, this option would enable him to obtain the answer without the need to consult an external source. The resulting display HELP (Figure 6.8) provides the user with the following:

(i) It describes briefly the functions of the various displays in their logical sequence as given in Figure 6.5.

(ii) Each display in the list is made user selectable in the same manner as that of a menu option. Hence, it permits the user to transfer control to any point within the data-fitting process, and the appropriate display is brought up on the screen. However, the system does not allow transfer of control to points in the process that are not yet meaningful. For example, if a fit has not yet been computed, a transfer to 'curve fit' display is meaningless. This branching out facility also enables the user to recover the state of the process, if the system terminates due to program error or user's unexpected action. The recovery is simply accomplished by keying in the command 'HELP' which would set up the help display, and the user may resume thereafter.

(iii) It also includes an option to produce a display containing the definition of all menu options used in the system (Figure 6.9).

(d) Reset the current display to its initial state of entry. → +RESTART

(e) Terminate the interactive session and prepare the system ready for next user. → +EXIT

These command menu options are used for directional control over the display sequence, giving the user the justifiable feeling that he is in charge of the system operations. They allow the user freedom to progress through the program normally or go back and review any previous display. This relationship between man and terminal must be such that he finds its use natural, pleasant and efficient.

Other specific options appear in the display menu corresponding to the role of that particular display in the process. These options will be described in some detail in the next section.

When an option is chosen from the menu, an arrow appears next to it, to indicate that a selection has been made. However, the option is not executed until the user has confirmed his choice by pressing the key Y(YES) in response to the message 'CONTINUE Y/N?'. This allows the user to change his mind and prevents inadvertent execution of undesirable options.

3.2 Function of the Various Displays

In what follows we describe the function of the displays corresponding to each of the boxes of the flow chart of Figure 6.5. This is illustrated by photographic reproduction of the screen as seen by the user during a typical interactive session at the terminal. The snapshots will include examples from both the parametric and the explicit methods. The numbering used in the following paragraphs is identical to that of Figure 6.5.

[1] Introductory

This is the first display to appear on the screen (Figures 6.6 and 6.7); it briefly describes the system and gives some general information regarding menu options. This initial display would be important for first time or occasional users and should be read carefully before proceeding to the next display in the sequence. In addition, the help displays (Figures 6.8 and 6.9) may also be regarded as supplementary to the introductory display. These displays effectively form an on-line user's guide to the IDF system.

[2] Choice of numerical algorithm

This display contains a list of currently available algorithms (Figures 6.10 and 6.11). By means of the cross-hair cursor, the user is able to select a suitable algorithm in conjunction with a particular type of end condition (see section 2.4). The boundary values will be declared in the parameter entry display.

The modular design of the system permits the incorporation of other algorithms with minimum programming effort.

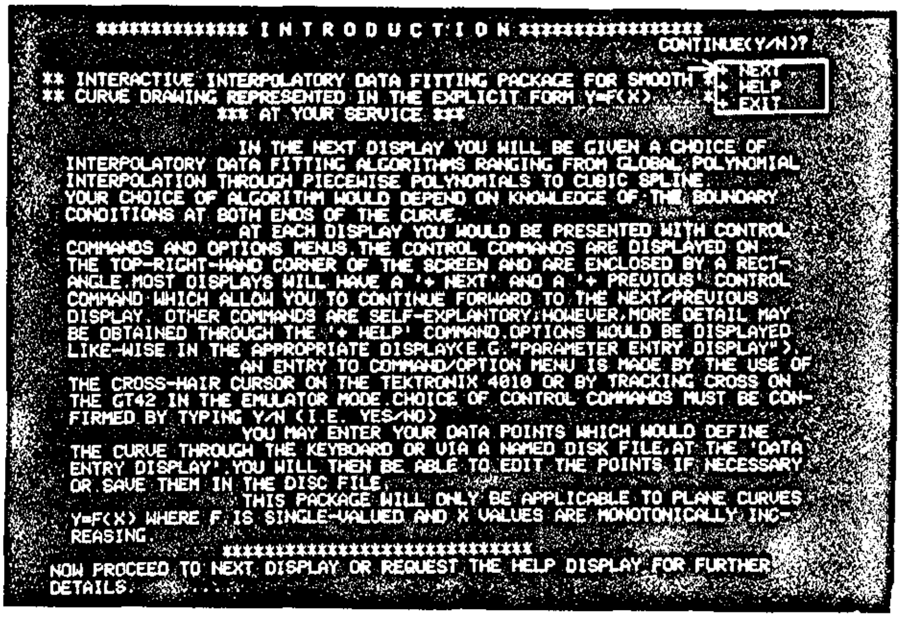


FIGURE 6.6: Introductory Display (Explicit)

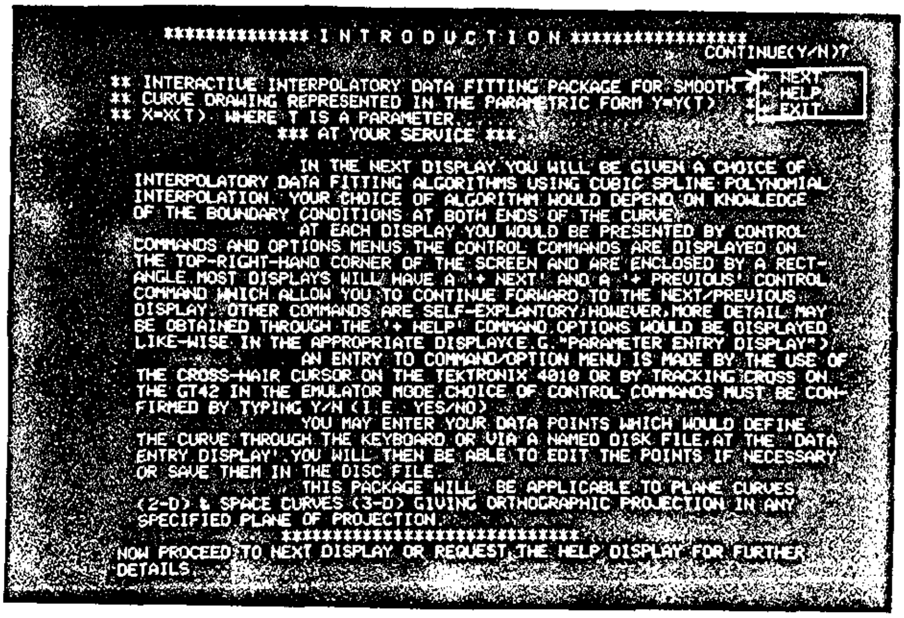


FIGURE 6.7: Introductory Display (Parametric)

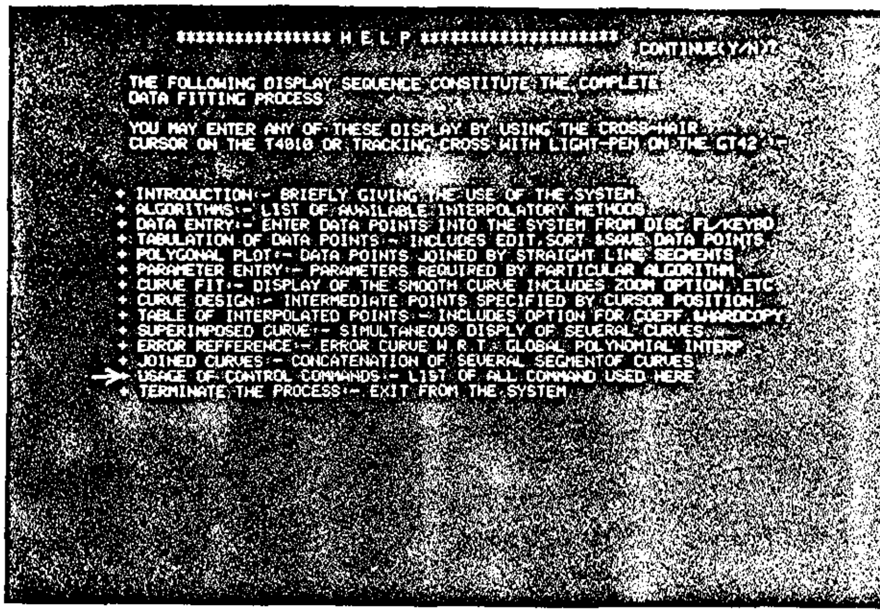


FIGURE 6.8: The HELP Display

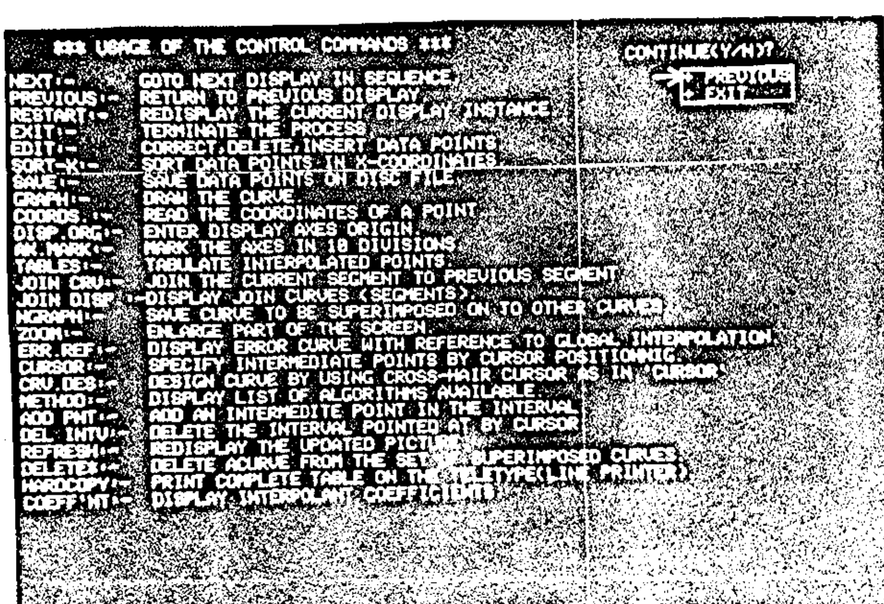


FIGURE 6.9: List of the Menu Options

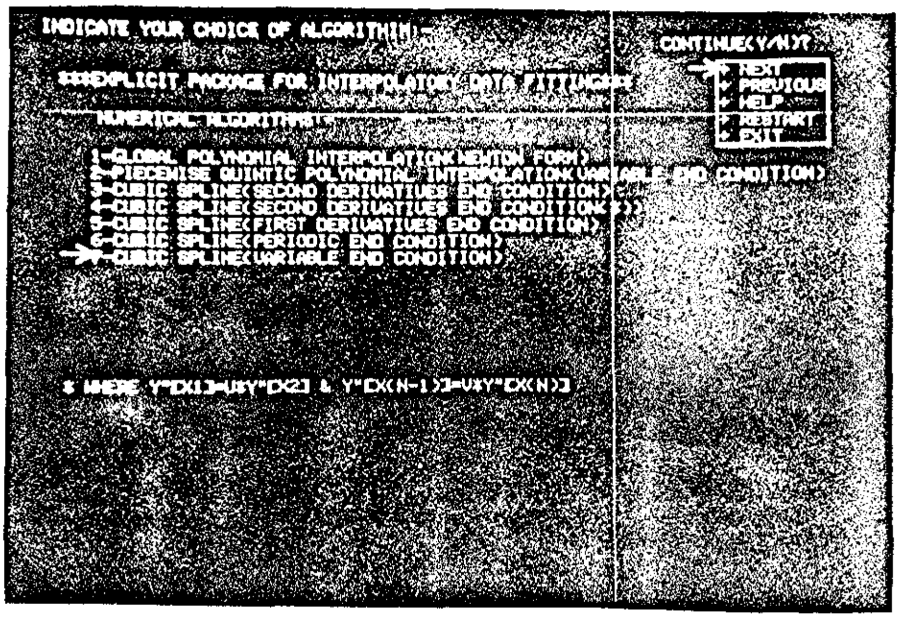


FIGURE 6.10: Choice of Numerical Algorithm and End Condition (Explicit)

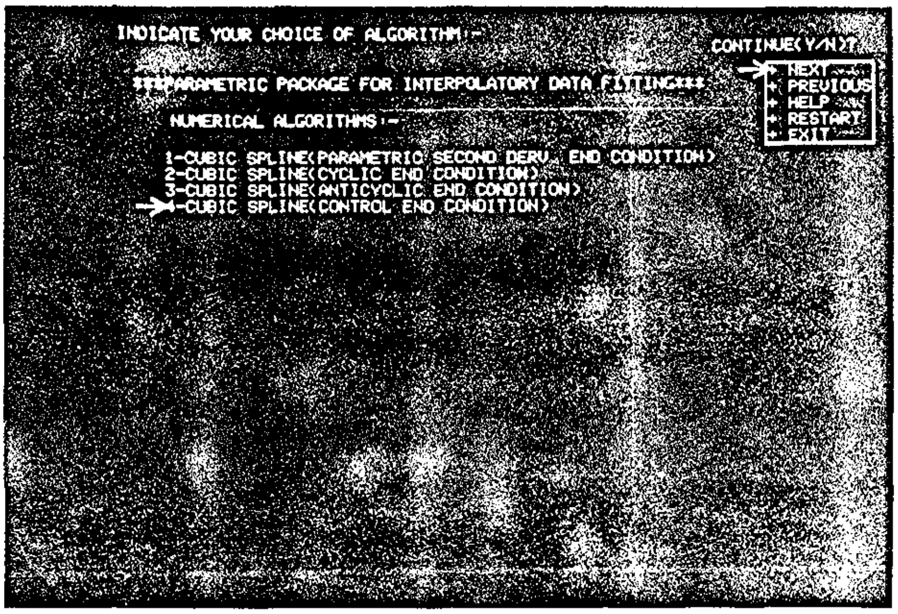


FIGURE 6.11: Choice of Numerical Algorithm and End Condition (Parametric)

[3] Data entry

After choosing the desired algorithm, the next step is to enter the data points into the system. There are two types of data source in this system. The first is external data which is entered either directly from the keyboard or a named disc file. The second is internal i.e. data from the immediately preceding problem. The latter is useful when slight variations of data or changes of algorithm/boundary condition are required.

The user is presented with a 'data specification' menu (Figures 6.12 and 6.13) which contains options on the state, dimensionality and medium of data entry. The state of data options are:

(a) Data is to be entered for the first time. → +NEW

(b) Data is to be used from the immediately preceding program run. → +OLD

The dimensionality is applicable in the case of parametric methods.

(a) Plane curve, requiring (x,y) data. → +2-DIMEN

(b) Space curve, requiring (x,y,z) data. → +3-DIMEN

Data medium entry

(a) Data is entered at the keyboard by typing the x-coordinates separated by commas followed by the y-coordinates (and z-coordinates if applicable). → +KEYBOARD

(b) Data may also be entered from an on-line file, which must be named. This data file would have been created previously by a user program in the form of consecutive x,y pairs (or x,y,z triples). → +DISC FILE

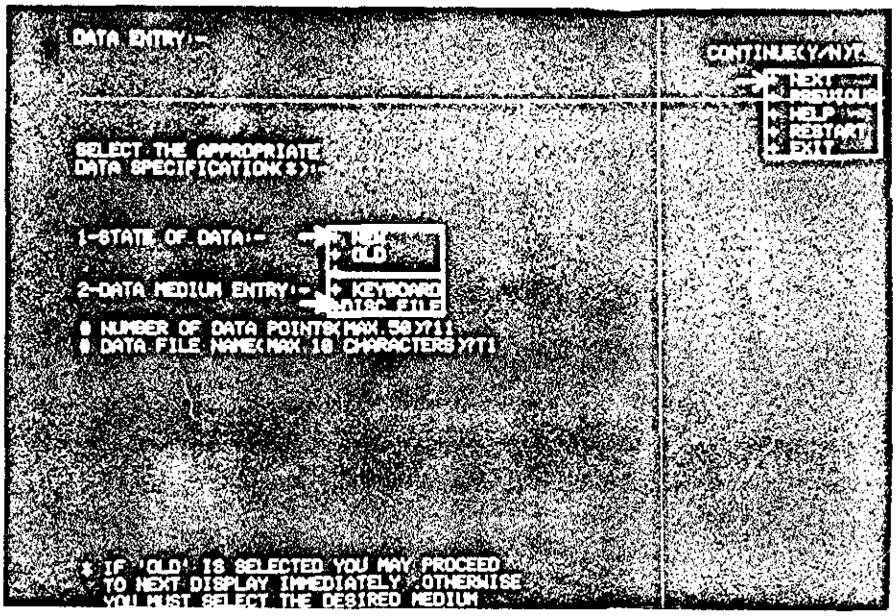


FIGURE 6.12: Data Entry Specification (Explicit)

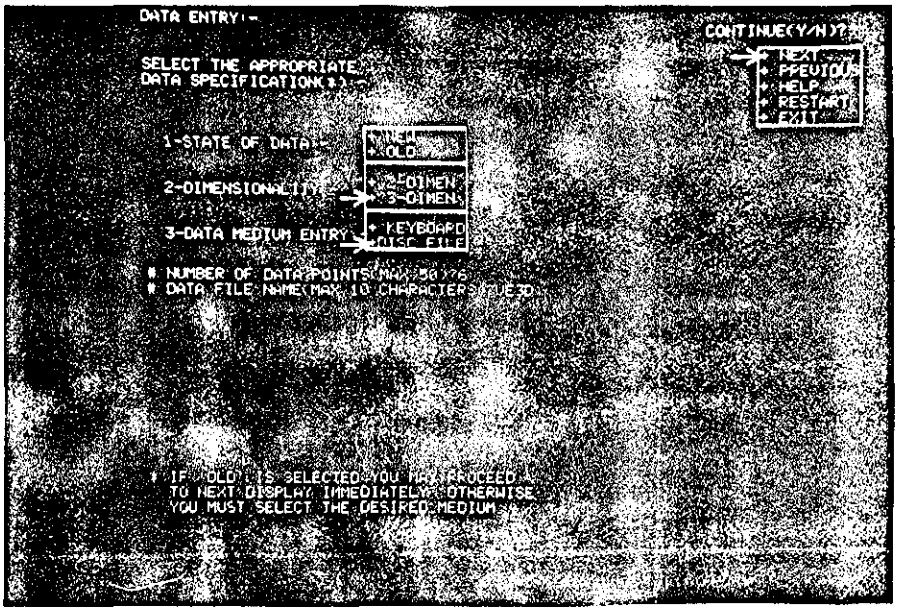


FIGURE 6.13: Data Entry Specification (Parametric)

[4] Data tabulation and editing

Once the data has been entered, the '+NEXT' display in the sequence presents the user with a table of the stored data points (Figure 6.14). This allows the user to check the input for possible errors and correct these before proceeding to the next display.

Often the user may also wish to alter the input data points in some way. These alterations may include, for example

(1) Selection of a subset of the data points.

(2) Insertion of new data points.

(3) Changing the order of the points in the set.

(4) Modification of individual values.

Data manipulation options relating particularly to this display are:

(a) Data point editing. → +EDIT

This generates an auxiliary display, containing a second level of menu options, permitting the user to correct, delete or insert data points in the existing set:

+CORRECT
+DELETE
→ +INSERT

An example of such use is illustrated in Figure 6.15, where two new data points have been inserted. A revised table of the data is shown in Figure 6.16 which appears immediately after the editing operations.

(b) The new data set may be stored on a named disc file → +SAVE

using this option. The name of the file is entered from the keyboard in response to the system request 'FILE NAME?'

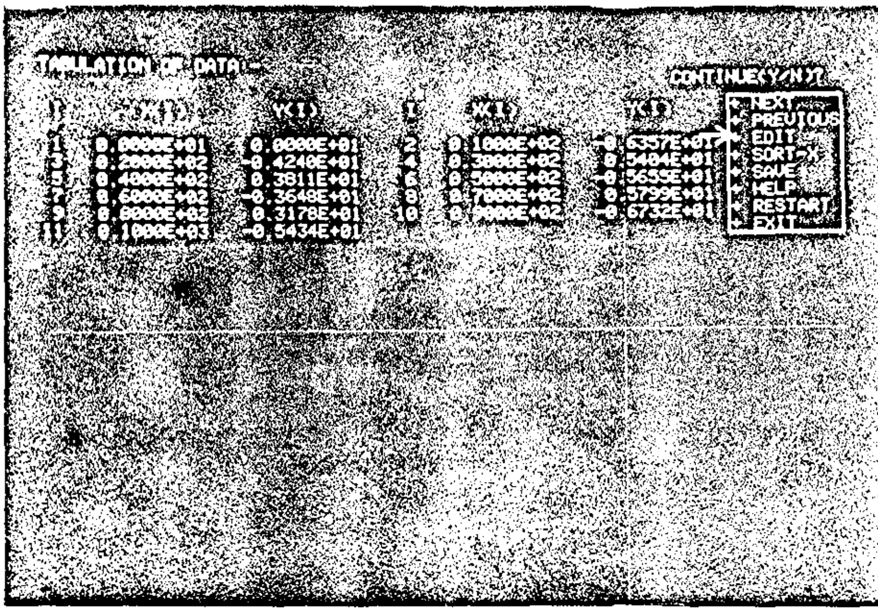


FIGURE 6.14: Table of Input Data Points

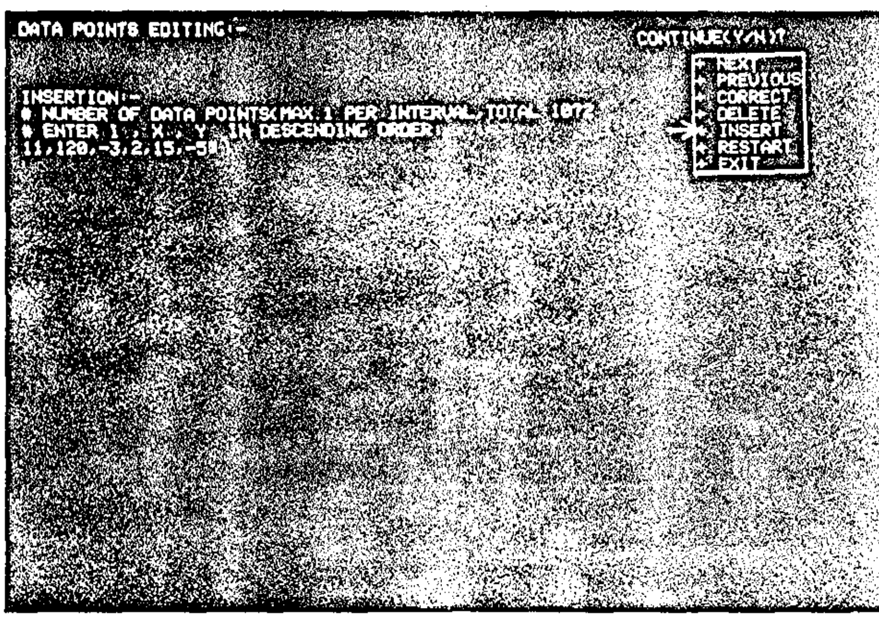


FIGURE 6.15: Data Editing with Insertion

FIGURE 6.16: Table of Data Points After the Insert Operation

(c) This would sort the data in ascending order of x-values. It may be necessary, for example, when using the explicit form y=f(x), for which the x-data must be monotonically increasing. → $\boxed{\text{+SORT-X}}$

[5] Polygonal plot

This is an optional display which is included in the system. It can be ignored by passing control to the '+NEXT' display. It allows the user to produce a graphical plot of the input data points joined by a sequence of straight lines. Note that this is equivalent to fitting polynomials of lowest possible degree (y=A+Bx) to consecutive pairs of points. Although the oscillation of the interpolating function here is minimal, the first derivatives at the interior points are discontinuous, and the curve of course is not smooth. However, the purpose of this display is just to give the user an initial feel of the curve shape. It also allows him graphically to re-check the correctness and order of the input data before invoking the smoothing algorithm.

The curve (in this case the polygon) is drawn within a pre-defined viewport. However, the limits of the plotting area can be controlled by the user interactively. The procedure would involve the use of the following options:

(a) This would allow the user to set up the limits required for the display axes, by typing the minimum and maximum values. As shown in Figures 6.17 and 6.18, the user is already informed about the corresponding data limits prior to his action. In this way he has control of the display process. → $\boxed{\text{+DISP.ORG.}}$

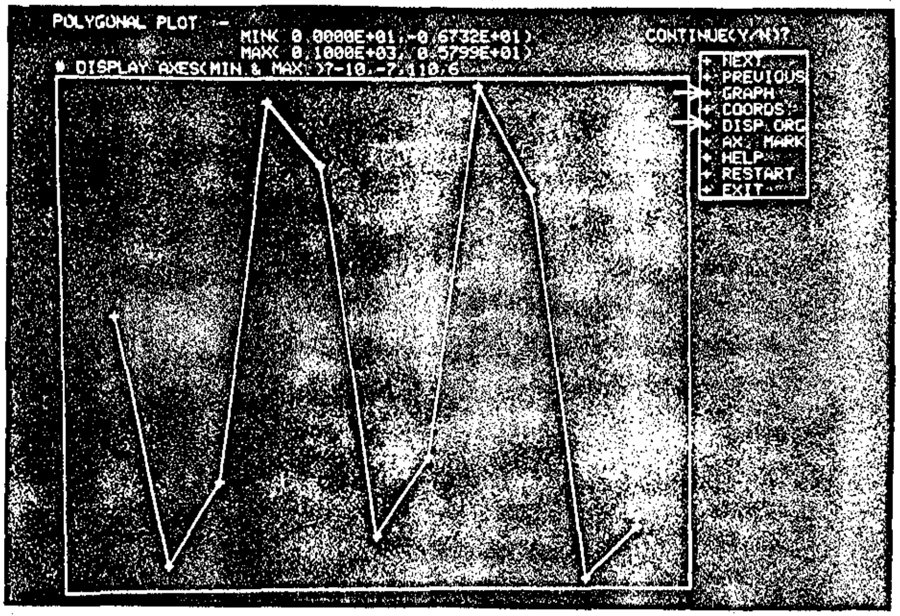


FIGURE 6.17: Polygonal Plot of a Single-Valued Curve

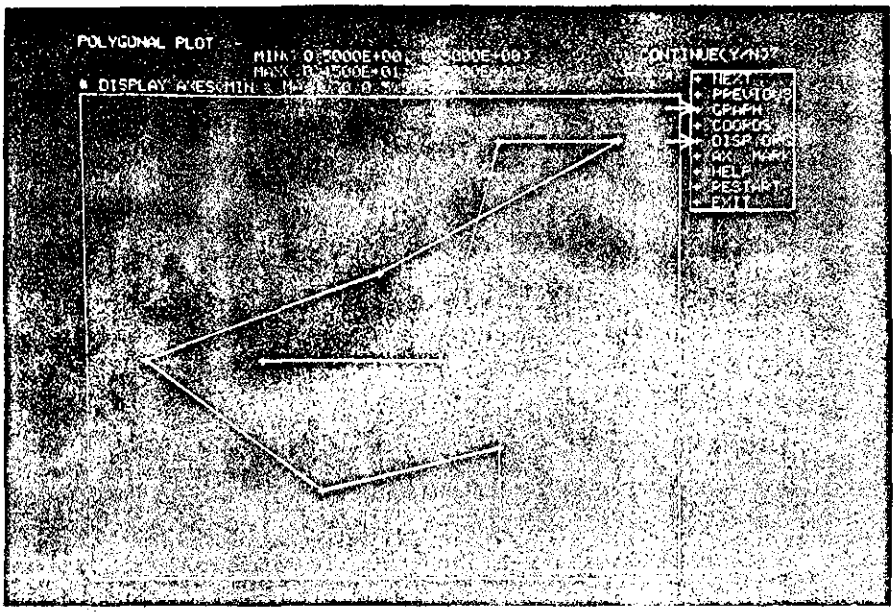


FIGURE 6.18: Polygonal Plot of a Multi-Valued Curve

(b) When this option is selected, the actual drawing operation would take place. The input data points are plotted as plus (+) signs as shown. Note that if this option is selected without prior use of option (a), then the viewport limits would by default be set equal to the input data limits. → $\boxed{\text{+GRAPH}}$

[6] Parameter entry

After the data has been entered, and then modified as desired, the '+NEXT' display on the screen is the parameter entry, shown in Figures 6.19 and 6.20. The user is provided here with a 'parameter specification' menu containing various options on each parameter. These are:

(1) The data type.

In practice, the user could encounter a situation where the required curve has known discontinuities of slope or curvature at certain points. This situation is handled by the present system by allowing the user to partition the given set of points into several subsets (up to 5 at present) at the points of discontinuity. The subsets are specified separately at this stage. Consequently, the system would produce a smooth curve for each segment and join them internally so that the user is able to display the complete joined curve later.

The parameter options provided are:

(a) For a complete (non-partitioned) set of data points. → $\boxed{\text{+COMPLETE}}$

(b) For partitioned data. The first subset must be recognised separately for building the complete joined curve. Any other subset is identified as a subsequent subset. FIRST → $\boxed{\text{+SUBSET}}$ SUBSEQUENT $\boxed{\text{+SUBSET}}$

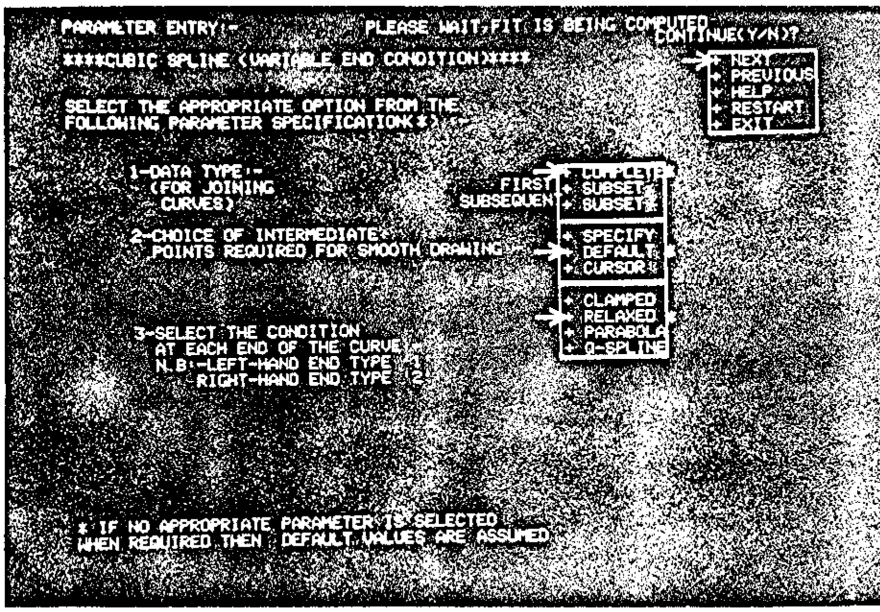


FIGURE 6.19: Parameter Entry Display of Cubic Spline (Explicit)

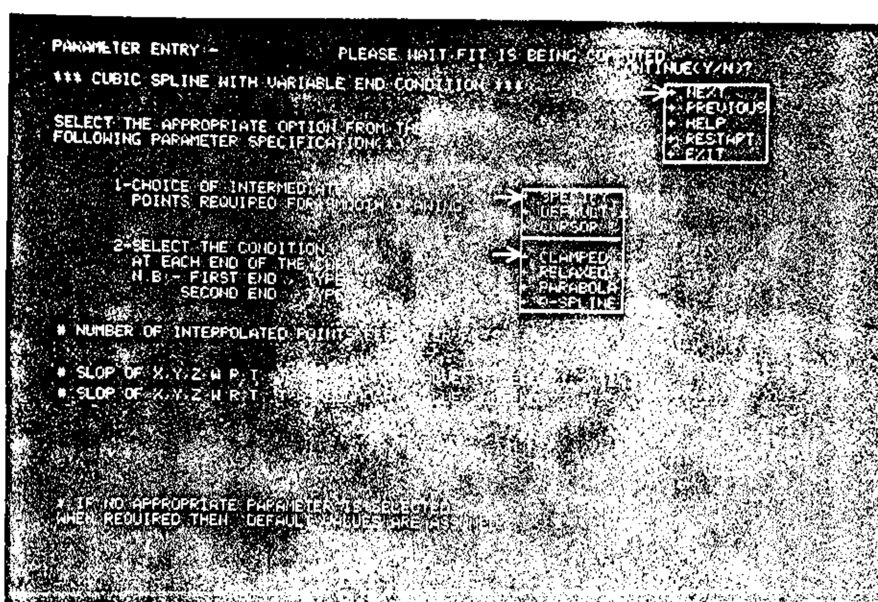


FIGURE 6.20: Parameter Entry Display of Cubic Spline (3-D Parametric)

(2) Choice of intermediate points.

Graphical output devices for digital computers can only be used to a given (raster) resolution. A curve is drawn by specifying a series of raster points which approximate the curve, or by the start point of the curve and a series of increments. In either case the curve is represented by a series of straight line segments.

This option determines the number of straight line segments required in each data interval to draw an acceptably smooth curve. In order to determine this, three choices are provided:

(a) The number of points per interval is specified via the keyboard. This number would be the same for each interval along the curve. However, the use of the same density of points in each interval can sometimes produce unsatisfactory results, since segments with higher curvature obviously require more points than others. → +SPECIFY

(b) This employs a built in algorithm to calculate the number of points necessary for each interval. This method is based on using the chord length of each interval relative to the total funicular polygon of the data points. The algorithm sets internally an upper limit to the total number of points (i.e. number of line segments required to draw the curve. A limit of 200 points is chosen to give a reasonable distribution of points over the display region (viewport) and avoids damaging the screen. A simple formula was derived for this to check the computed number of intermediate points, → +DEFAULT

$$N_I \leqslant \frac{200-N}{N-1} \tag{6.66}$$

where N is the number of data points. Typical values are given by the Table:

N	2	10	50	100	>100
[max N_I]	198	21	3	1	0

(c) This choice allows the user to select visually the intermediate points by means of the cursor, with full control over the smoothness of the trace. → +CURSOR

(3) Select the end condition.

The algorithms listed in 3.2 [2]-Figures 6.10 and 6.11 allow different ways of selecting the boundary conditions. In the most general case ('variable end condition') the available boundary condition options, as described in section 2.4, are →

+CLAMPED
+RELAXED
+PARABOLA
+Q-SPLINE

The user must indicate the appropriate menu entry specifying the boundary condition for each end. In the example shown in Figures 6.19 and 6.20, the same form of condition is used at both ends. In general, however, the condition at end 1 can be specified independently of that at end 2, so that two different forms of condition are possible (e.g. '+CLAMPED' at end 1 and '+RELAXED' at end 2).

Once the user has selected the required parameter, the next step is to invoke the numerical algorithm to compute the interpolated curve by selecting the option '+NEXT' in the main menu. At this stage, the system message 'PLEASE WAIT, FIT IS BEING COMPUTED' is displayed since the number crunching involved would be relatively time-consuming (See Chapter 5).

[7] Curve Fit

This display appears on the screen when the interpolated curve has been computed. The curve is drawn in the same way as illustrated in the polygon plot using the menu options '+GRAPH' and '+DISP.ORG.'. In the parametric case, three menu options are provided for this purpose, namely → +XY-GRAPH +XZ-GRAPH +ZY-GRAPH

in order to enable the user to display an orthogonal projection of a three-dimensional curve. Examples of a two-dimensional curve are shown in Figures 6.21, 6.22 and 6.23, employing the explicit and the parametric form respectively.

A wide range of menu options are provided in this display for anticipating most user requirements in interpreting the resulting curve. These options enable the user to interrogate the computed curve or extract further information either in tabular form or by comparison with other related curves.

(a) Axes marking. → +AX.MARK

This tags the lines enclosing the viewport with scaling marks. Each line is divided into ten equal parts, hence producing a linear scale along the display axes. The actual plotting axes are displayed if the origin (0,0) falls within the defined window (Figure 6.21b). The window limits are displayed at the bottom right-hand corner of the screen.

(b) Reading the coordinates of a point. → +COORDS.

If the user requires to know the coordinates of a specific point in the data window, the cross-hair cursor is located at the desired point. Then by pressing any key on the keyboard, the coordinate

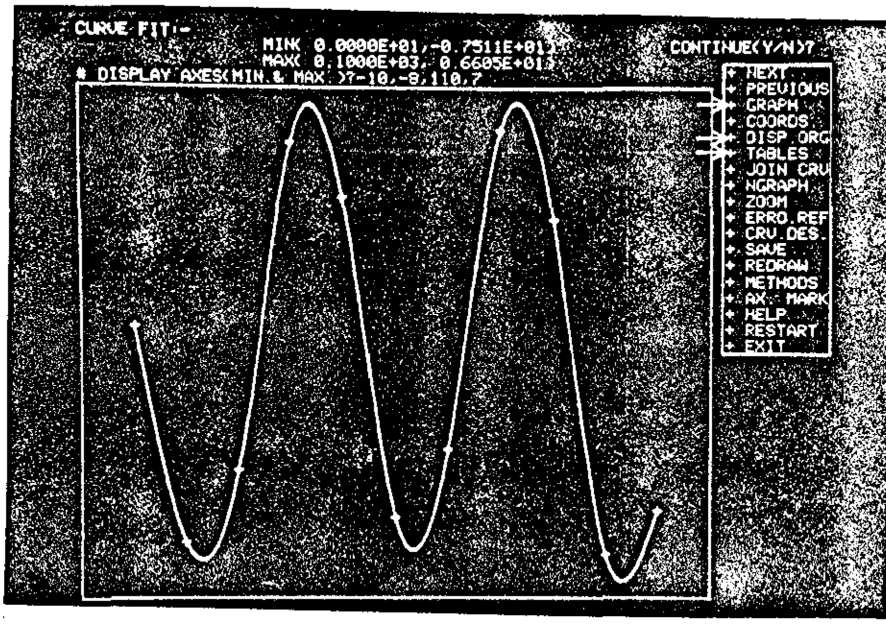


FIGURE 6.21a: A Two-dimensional Smooth Curve using the Explicit Form

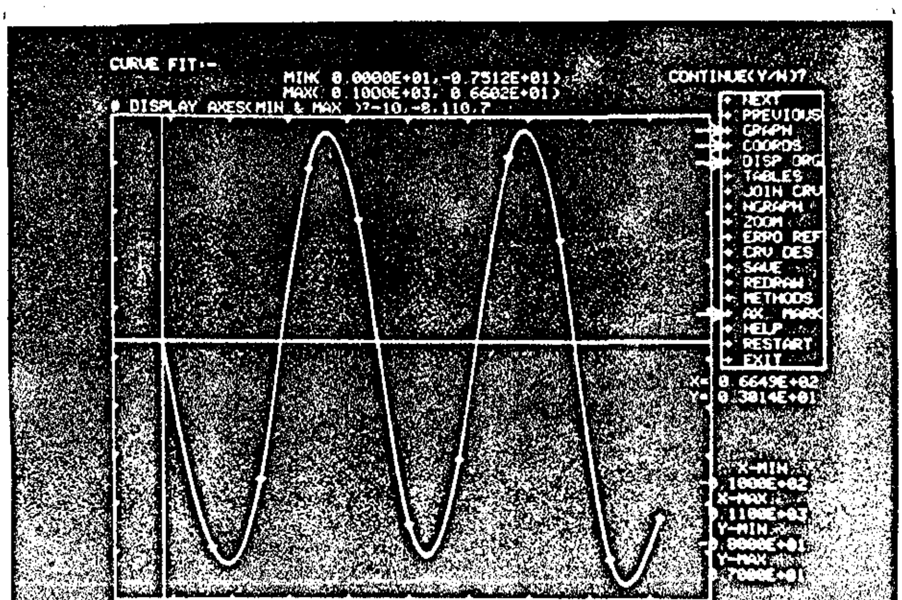


FIGURE 6.21b: Curve Fit Display Showing the Use of '+AX-MARK' and '+COORDS' Options

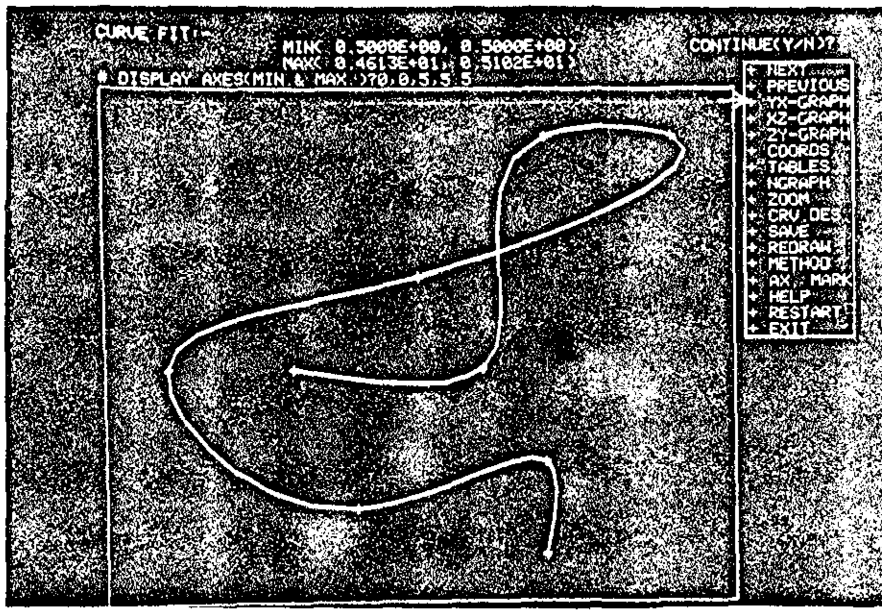


FIGURE 6.22: A Two-dimensional Smooth Curve Using the Parametric form

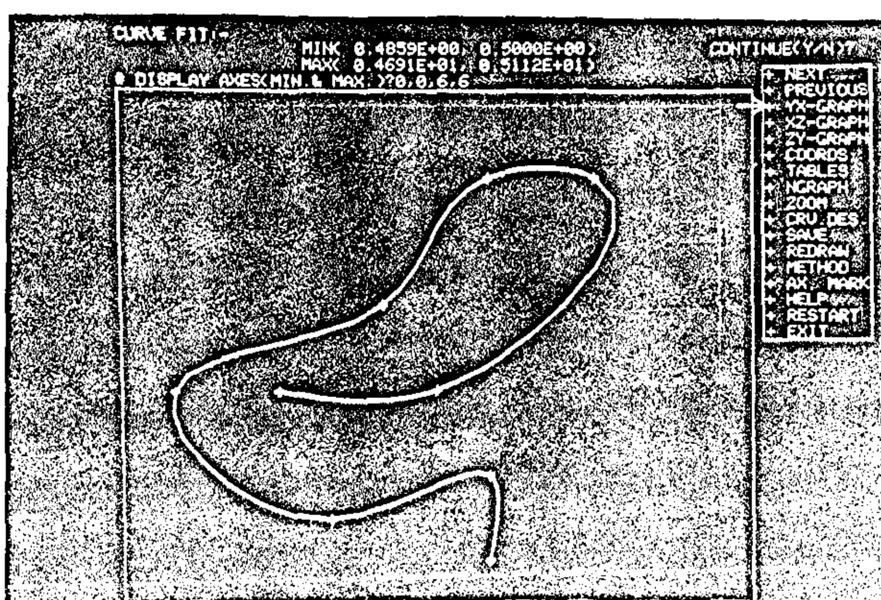


FIGURE 6.23: Same Data Points as Fig.6.22 but in a Different Order

of that point would be displayed (Figure 6.21b). This facility could be useful, for example, if the user wished to introduce other data points and re-examine the curve again.

(c) Saving an interpolated curve on disc file → +SAVE

This option is used to store on a disc file for subsequent use the complete set of scaled points representing the smooth curve. The name of this file is entered through the keyboard in response to a system prompt.

(d) Drawing a curve from a named disc file. → +REDRAW

Interpolated curves previously stored on disc file can be redrawn for viewing together with other displayed curves. This facility may have a number of uses, in particular when the user wishes to investigate the effect of applying different forms of end condition.

(e) Magnifying part of the screen. → +ZOOM

This is used to enlarge part of the screen. The zooming window is defined interactively by cursor input of any two diagonally opposite corners (Figure 6.24a). Once this has been accomplished, the program will perform the necessary clipping and display the zoomed part (Figure 6.24b).

(f) When this option is selected, the user will be → +TABLE presented with an ordered tabulation of the interpolated and input data points (Figures 6.25a and 6.25b). The menu of this tabular display contains the following additional options:

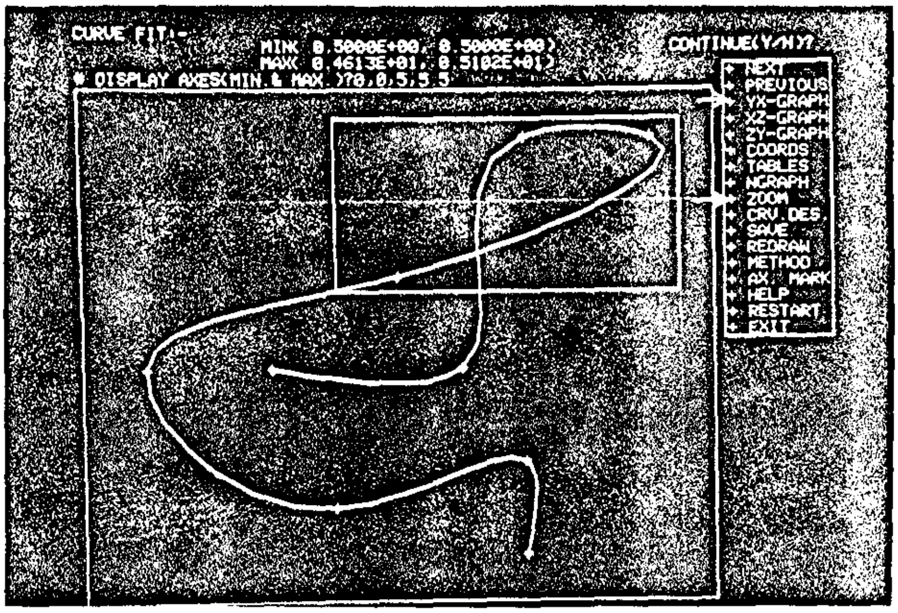


FIGURE 6.24a: Using the '+ZOOM' Option

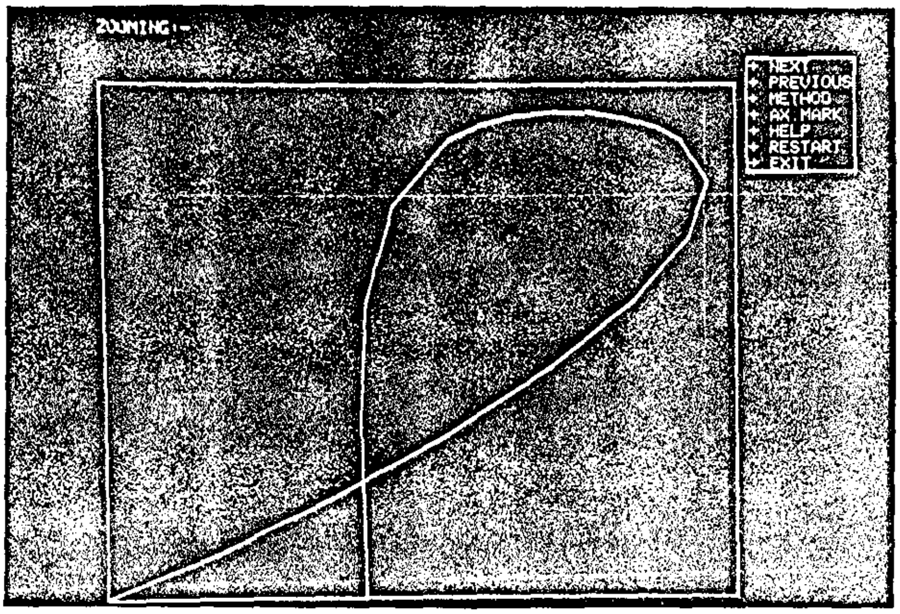
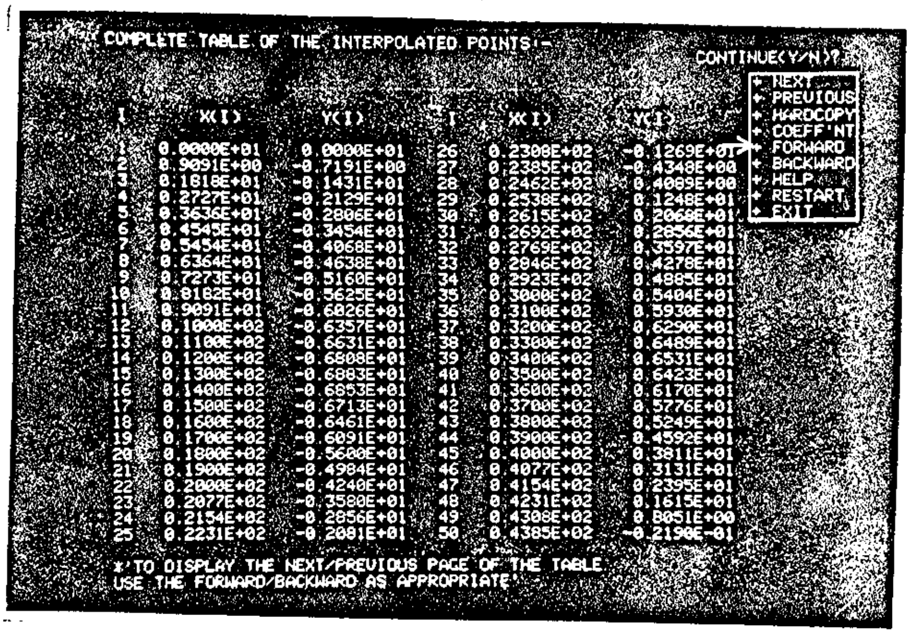


FIGURE 6.24b: Display of the Zoomed Portion



COMPLETE TABLE OF THE INTERPOLATED POINTS:-

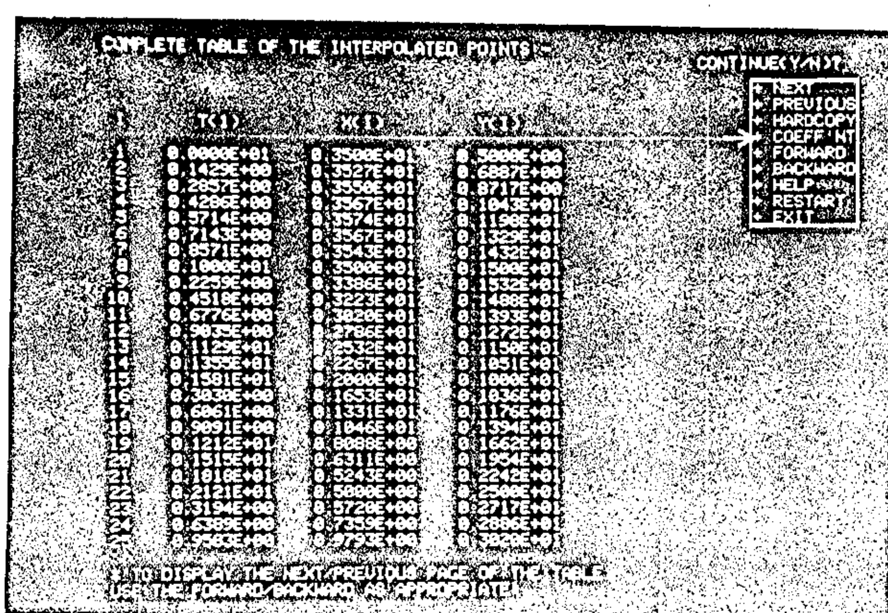
CONTINUE(Y/N)?

I	X(I)	Y(I)	I	X(I)	Y(I)
1	0.0000E+01	0.0000E+01	26	0.2308E+02	-0.1269E+01
2	0.9091E+00	-0.7191E+00	27	0.2385E+02	-0.4348E+00
3	0.1818E+01	-0.1431E+01	28	0.2462E+02	0.4089E+00
4	0.2727E+01	-0.2129E+01	29	0.2538E+02	0.1248E+01
5	0.3636E+01	-0.2806E+01	30	0.2615E+02	0.2068E+01
6	0.4545E+01	-0.3454E+01	31	0.2692E+02	0.2856E+01
7	0.5454E+01	-0.4068E+01	32	0.2769E+02	0.3597E+01
8	0.6364E+01	-0.4638E+01	33	0.2846E+02	0.4278E+01
9	0.7273E+01	-0.5160E+01	34	0.2923E+02	0.4885E+01
10	0.8182E+01	-0.5625E+01	35	0.3000E+02	0.5404E+01
11	0.9091E+01	-0.6026E+01	36	0.3100E+02	0.5930E+01
12	0.1000E+02	-0.6357E+01	37	0.3200E+02	0.6290E+01
13	0.1100E+02	-0.6631E+01	38	0.3300E+02	0.6489E+01
14	0.1200E+02	-0.6808E+01	39	0.3400E+02	0.6531E+01
15	0.1300E+02	-0.6883E+01	40	0.3500E+02	0.6423E+01
16	0.1400E+02	-0.6853E+01	41	0.3600E+02	0.6170E+01
17	0.1500E+02	-0.6713E+01	42	0.3700E+02	0.5776E+01
18	0.1600E+02	-0.6461E+01	43	0.3800E+02	0.5249E+01
19	0.1700E+02	-0.6091E+01	44	0.3900E+02	0.4592E+01
20	0.1800E+02	-0.5600E+01	45	0.4000E+02	0.3811E+01
21	0.1900E+02	-0.4984E+01	46	0.4077E+02	0.3131E+01
22	0.2000E+02	-0.4240E+01	47	0.4154E+02	0.2395E+01
23	0.2077E+02	-0.3580E+01	48	0.4231E+02	0.1615E+01
24	0.2154E+02	-0.2856E+01	49	0.4308E+02	0.8051E+00
25	0.2231E+02	-0.2081E+01	50	0.4385E+02	-0.2190E-01

+ NEXT
+ PREVIOUS
+ HARDCOPY
+ COEFF'NT
+ FORWARD
+ BACKWARD
+ HELP
+ RESTART
+ EXIT

*'TO DISPLAY THE NEXT/PREVIOUS PAGE OF THE TABLE
USE THE FORWARD/BACKWARD AS APPROPRIATE'

FIGURE 6.25a: Table of the Computed Data Points on the Smooth Curve of Fig. 6.21a



COMPLETE TABLE OF THE INTERPOLATED POINTS:-

CONTINUE(Y/N)?

I	T(I)	X(I)	Y(I)
1	0.0000E+01	0.3500E+01	0.5000E+00
2	0.1429E+00	0.3527E+01	0.6887E+00
3	0.2857E+00	0.3550E+01	0.8717E+00
4	0.4286E+00	0.3567E+01	0.1043E+01
5	0.5714E+00	0.3574E+01	0.1198E+01
6	0.7143E+00	0.3567E+01	0.1329E+01
7	0.8571E+00	0.3543E+01	0.1432E+01
8	0.1000E+01	0.3500E+01	0.1500E+01
9	0.2259E+00	0.3386E+01	0.1532E+01
10	0.4518E+00	0.3223E+01	0.1488E+01
11	0.6776E+00	0.3020E+01	0.1393E+01
12	0.9035E+00	0.2786E+01	0.1272E+01
13	0.1129E+01	0.2532E+01	0.1150E+01
14	0.1355E+01	0.2267E+01	0.1051E+01
15	0.1581E+01	0.2000E+01	0.1000E+01
16	0.3030E+00	0.1653E+01	0.1036E+01
17	0.6061E+00	0.1331E+01	0.1176E+01
18	0.9091E+00	0.1046E+01	0.1394E+01
19	0.1212E+01	0.8088E+00	0.1662E+01
20	0.1515E+01	0.6311E+00	0.1954E+01
21	0.1818E+01	0.5243E+00	0.2242E+01
22	0.2121E+01	0.5000E+00	0.2500E+01
23	0.3194E+00	0.5720E+00	0.2717E+01
24	0.6389E+00	0.7359E+00	0.2886E+01
25	0.9583E+00	0.9793E+00	0.3020E+01

+ NEXT
+ PREVIOUS
+ HARDCOPY
+ COEFF'NT
+ FORWARD
+ BACKWARD
+ HELP
+ RESTART
+ EXIT

*'TO DISPLAY THE NEXT/PREVIOUS PAGE OF THE TABLE
USE THE FORWARD/BACKWARD AS APPROPRIATE'

FIGURE 6.25b: Table of the Computed Data Point/on the Smooth Curve of Fig. 6.22

(1) This provides the user with a table of values of the computed polynomial coefficients for each data interval (Figures 6.26a, 6.26b and 6.26c). → $\boxed{\text{+COEFFNT}}$

(2) Roll-by of tabular information. → $\boxed{\text{+FORWARD} \\ \text{+BACKWARD}}$

Sometimes it is not possible to accommodate the whole table on the available screen area. These options are used to control the display of a large numerical table (e.g. data points) by scrolling forward or back through the table page by page.

(3) Hardcopy print of the numerical table can be obtained on one of the existing teletypes if this option is selected. This facility is easily extended to other hardcopy devices. → $\boxed{\text{+HARDCOPY}}$

(g) This option is specifically used for joining partitioned curves, which have been previously defined in the parameter entry display. When the option is first selected, the system will internally append the data points representing the current displayed curve to the other segments, providing the user has already specified the appropriate data parameter. If the option is picked up for the second time, new display will be brought up on the screen, allowing the user to draw the complete joined curve. The two smooth curve segments shown in Figures 6.27a and 6.27b have been joined to produce the curve shown in Figure 6.27c, which includes an internal slope discontinuity. Figure 6.27d represents an enlargement (zoom) of the region surrounding the discontinuity point. → $\boxed{\text{+JOIN}}$

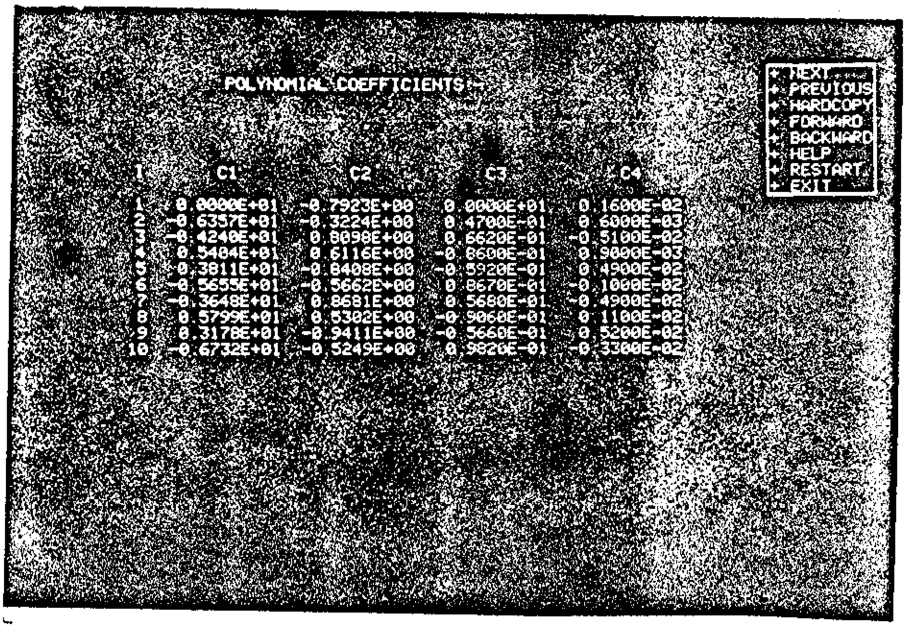


FIGURE 6.26a: Computed Polynomial Coefficients of Fig.6.21a for Cubic Spline

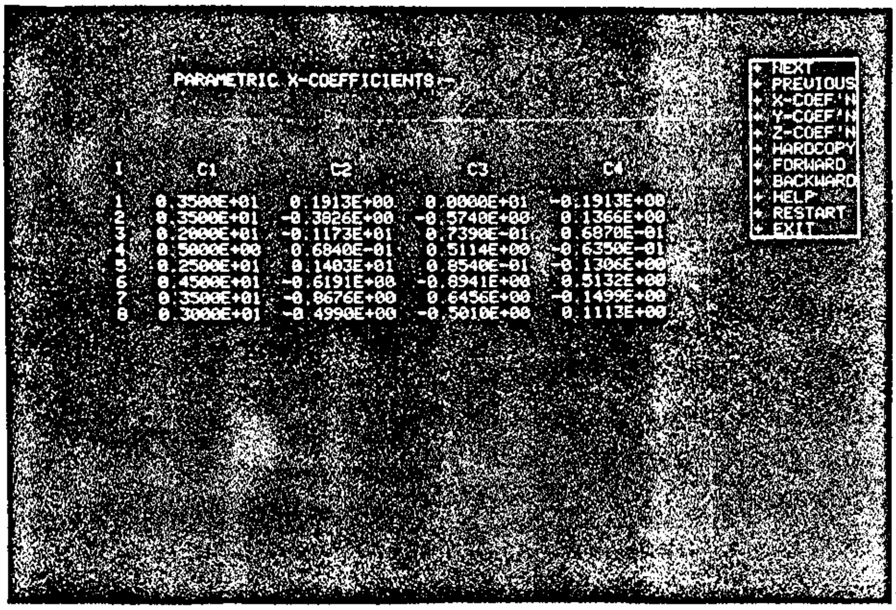


FIGURE 6.26b: Computed Polynomial Coefficients of the Cubic Spline (Parametric) for x of Fig.6.22

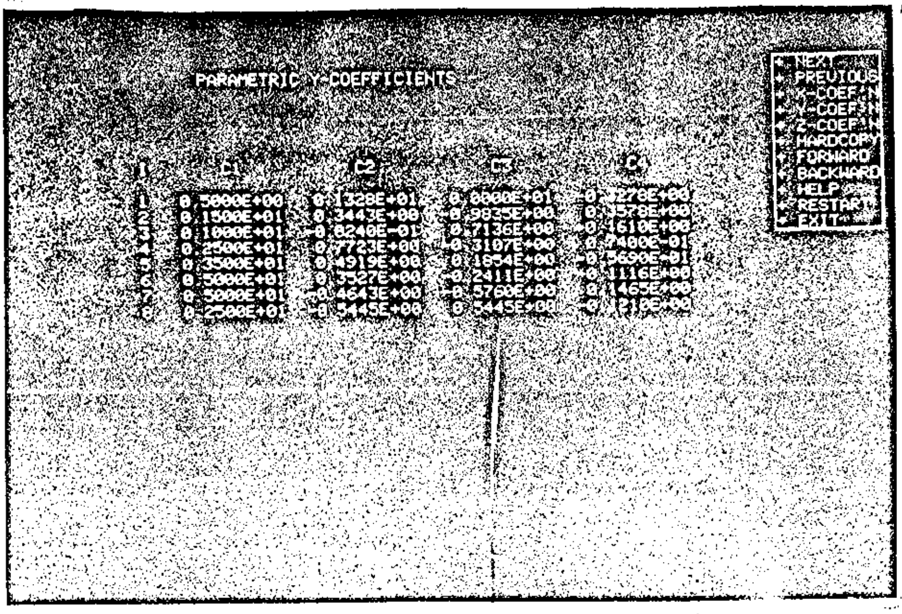


FIGURE 6.26c: Computed Polynomial Coefficients of the Cubic Spline (Parametric) for y of Fig.6.22

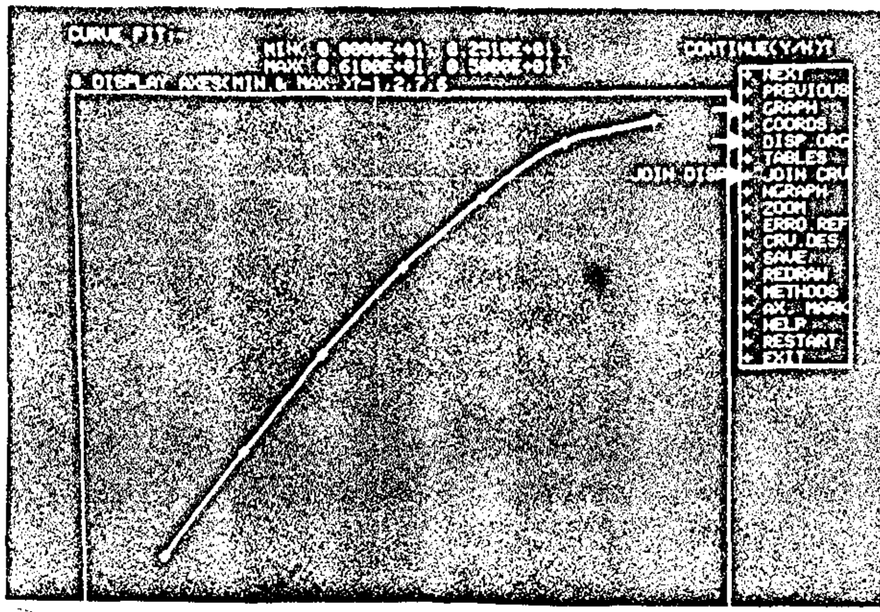


FIGURE 6.27a: Segment 1 of the Joined Curve

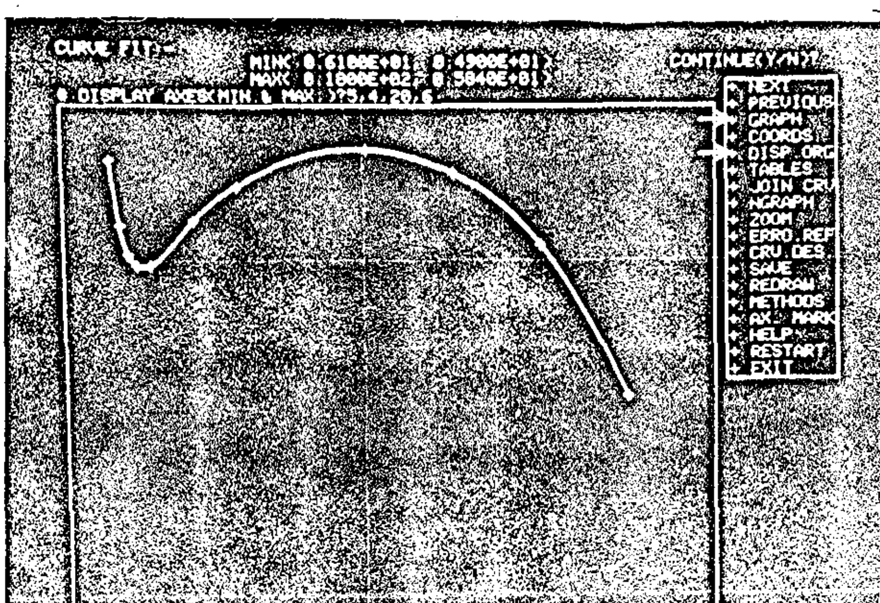


FIGURE 6.27b: Segment 2 of the Joined Curve

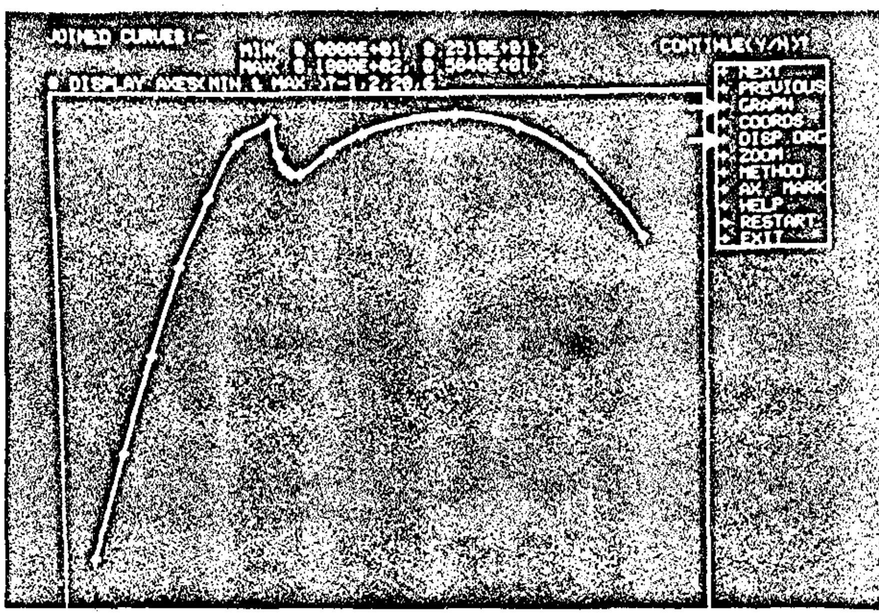


FIGURE 6.27c: Display of the Joined Curve (Segment 1 and 2)

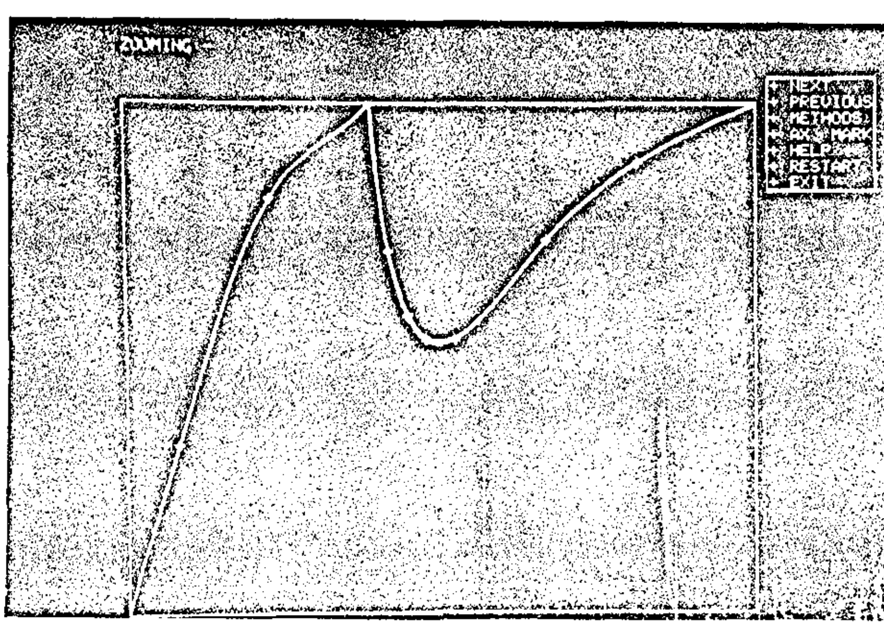


FIGURE 6.27d: A Zoomed Portion of the Joined Curve Near the Discontinuity

(h) A multiple display of a group of related curves can be generated using this option (Figure 6.28). For this purpose, every time a smooth curve of the group is produced this option is selected in order to inform the system of our intention. The superimposed curve display is generated when this option is immediately re-selected. The curves are labelled with consecutive numbers according to their generation sequence. Two additional options are provided: → +NGRAPH

(1) It allows the user to remove a particular curve from the group. This is accomplished once the curve number has been typed in. → +DELETE

(2) This erases the unwanted curve from the screen. → +REFRESH

(i) When this option is selected, an auxiliary display is generated. This display enables the user interactively by cursor input, to select intermediate points in each interval. By this means, the user is able to control the length of the elemental line segments, so that intervals with higher curvature would have shorter segments. Two examples of such displays are shown in Figures 6.29a and 6.29b. The use of this display can be exploited in two basic ways: → +CRV.DES.

(i) Starting from the displayed data points, the curve can be traced by progressing from one interval to another and drawing the line segments.

(ii) An existing traced curve can be visually improved if necessary by reconstructing individual segments.

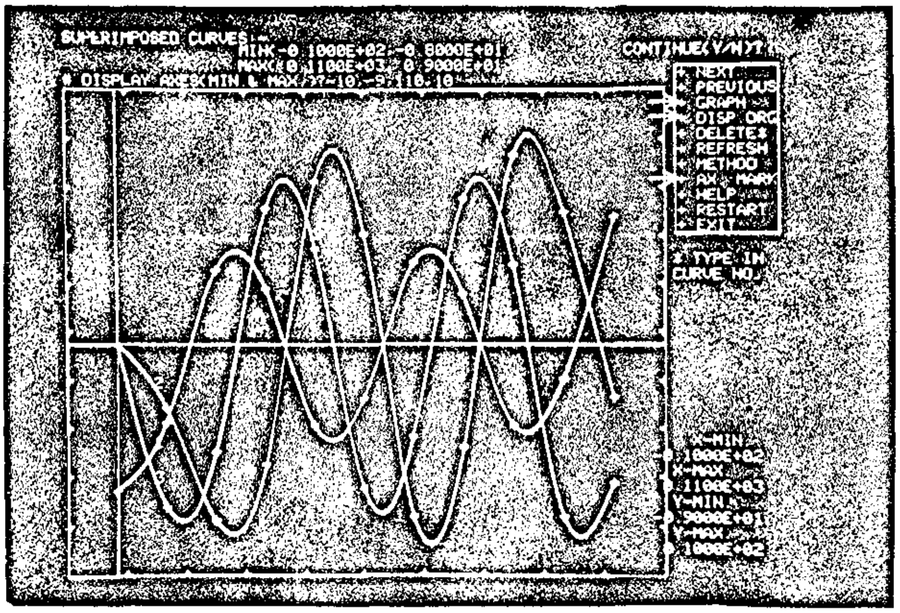


FIGURE 6.28 : Multiple Display of Several Curves

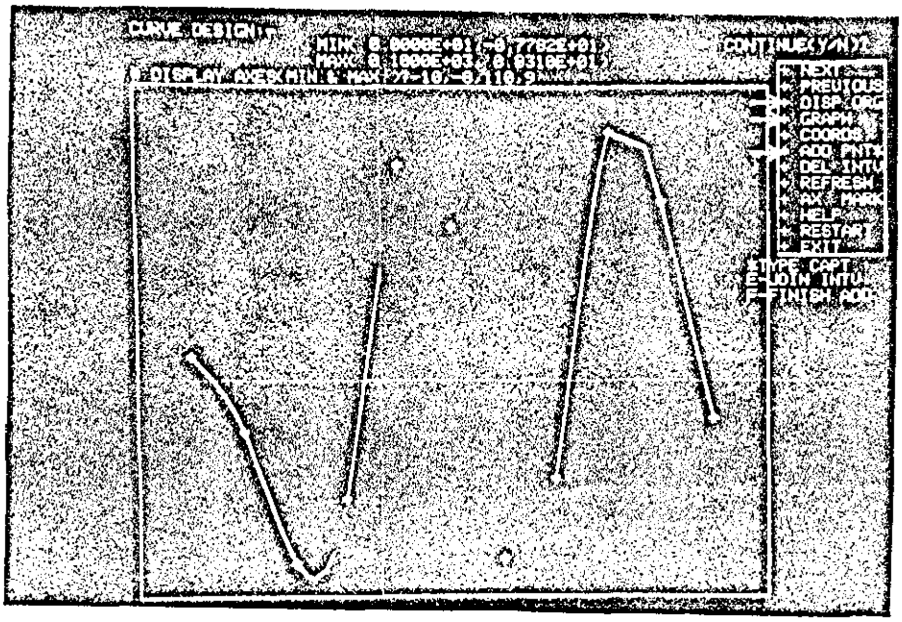


FIGURE 6.29a: Interactive Choice of Intermediate Points by Means of Cursor (Explicit)

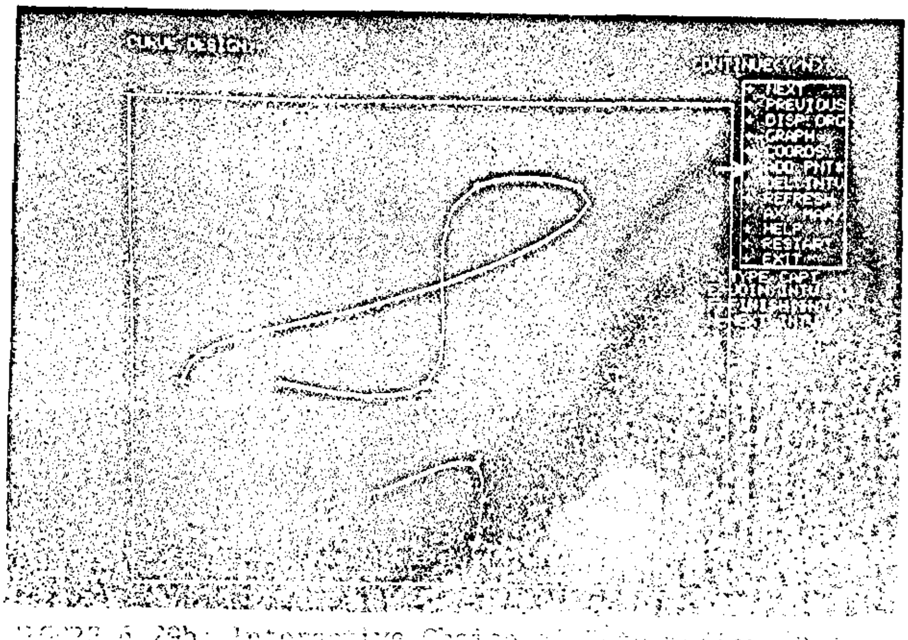


FIGURE 6.29b: Interactive Choice of Intermediate Points by Means of Cursor (Parametric)

The operations required to trace the curve are basically equivalent to the editing process i.e. delete or insert data points. In other words, this is effectively graphical editing of the displayed curve. For this purpose two options are provided:

(1) Once this is selected, the cursor control is returned back to the user enabling him to select intermediate points. This is accomplished by positioning the cursor at the desired intermediate point and by pressing a key (e.g. space bar) on the keyboard. The program would respond by drawing the desired line segment in that interval. In this way the user can progressively trace the curve or add new line segments between two existing points to obtain the required smoothness. → $\boxed{\text{+ADD.PNT}}$

(2) When this option is selected, the user can use the cursor to point at a required interval and virtually remove the entire line segments within this interval. The line segments are physically erased by using the '+REFRESH' option. → $\boxed{\text{+DEL.INT}}$

3.3 Examples

Several examples of interpolated curves are illustrated in Figures 6.30-6.33 to demonstrate the effects of the various algorithms.

(1) Figures 6.30a and 6.30b display the smooth curves passing through the same data points using the global and piecewise quintic polynomial interpolation methods. It can be seen clearly that the former method produces undesirable

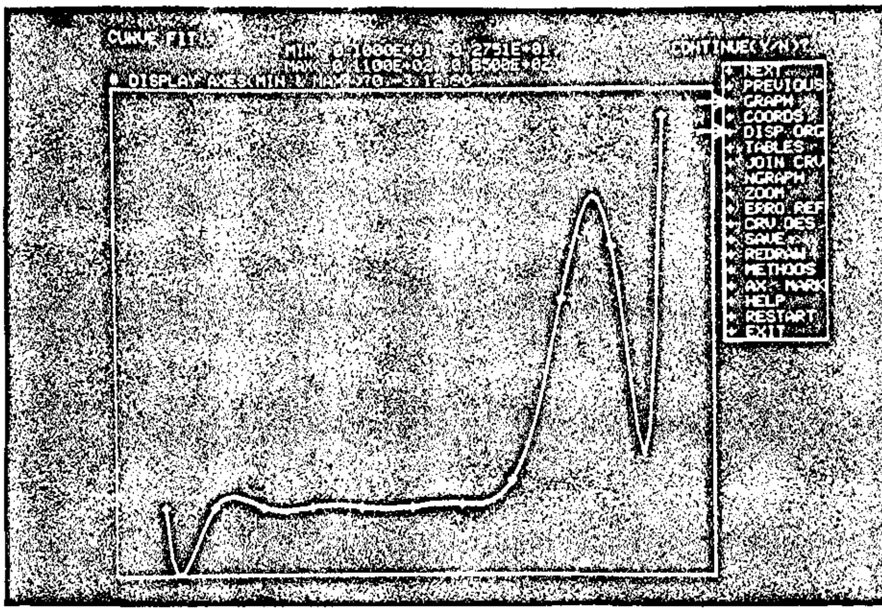


FIGURE 6.30a: Interpolated Curve using a Global (Newton) Polynomial

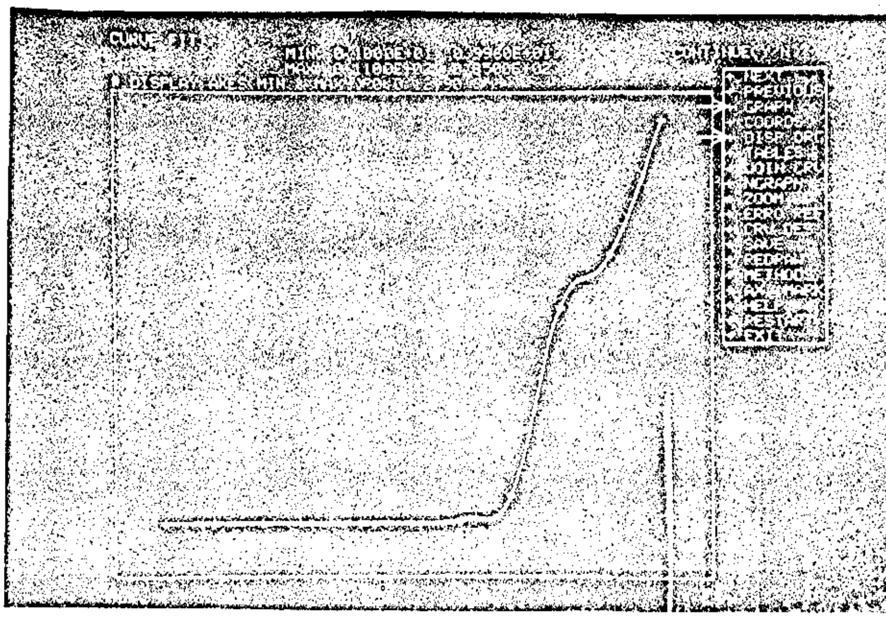


FIGURE 6.30b: The Same Data Points as in Fig.6.30a Using Piecewise Quintic Polynomial Interpolation (Maude)

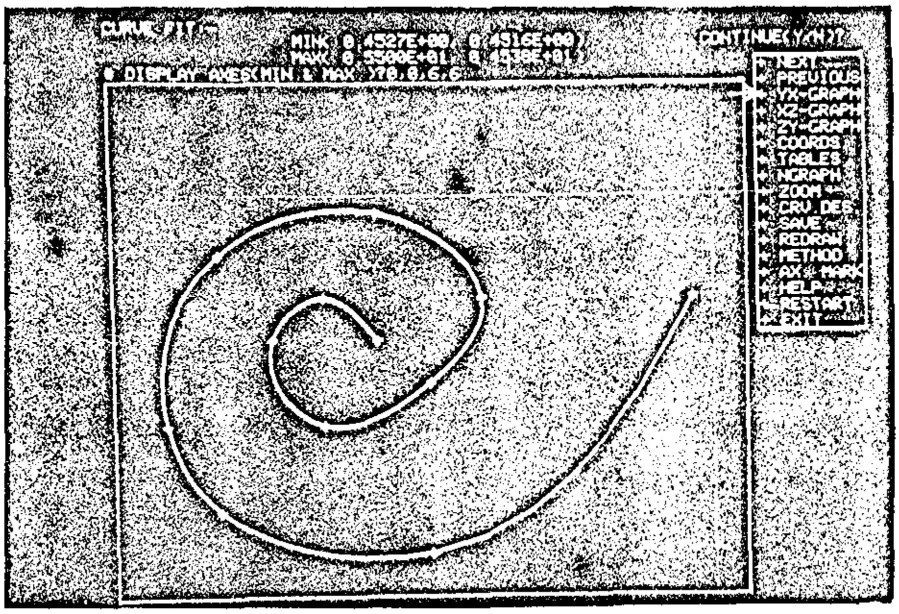


FIGURE 6.31: A Two-dimensional Spiral Curve with Relaxed End Condition

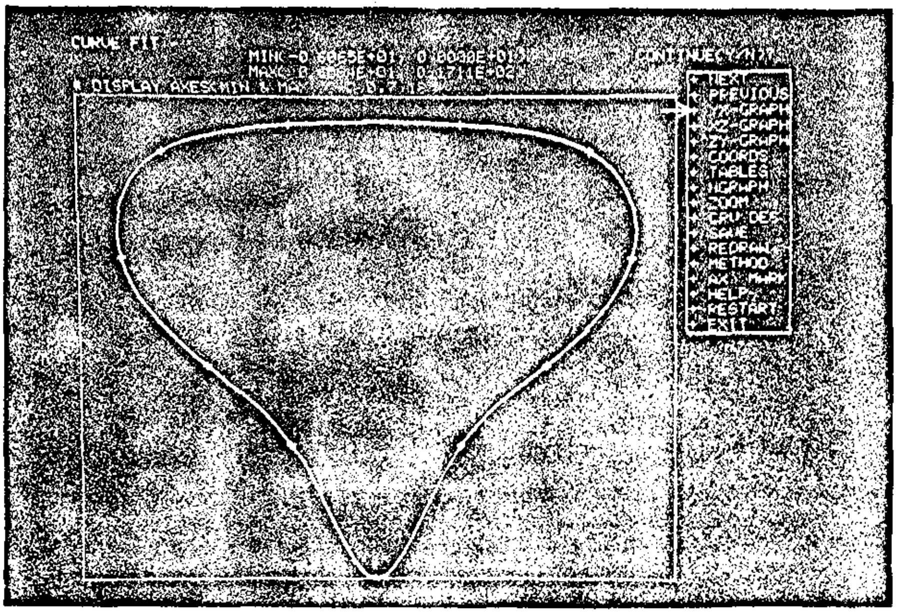


FIGURE 6.32a: Cyclic End Condition

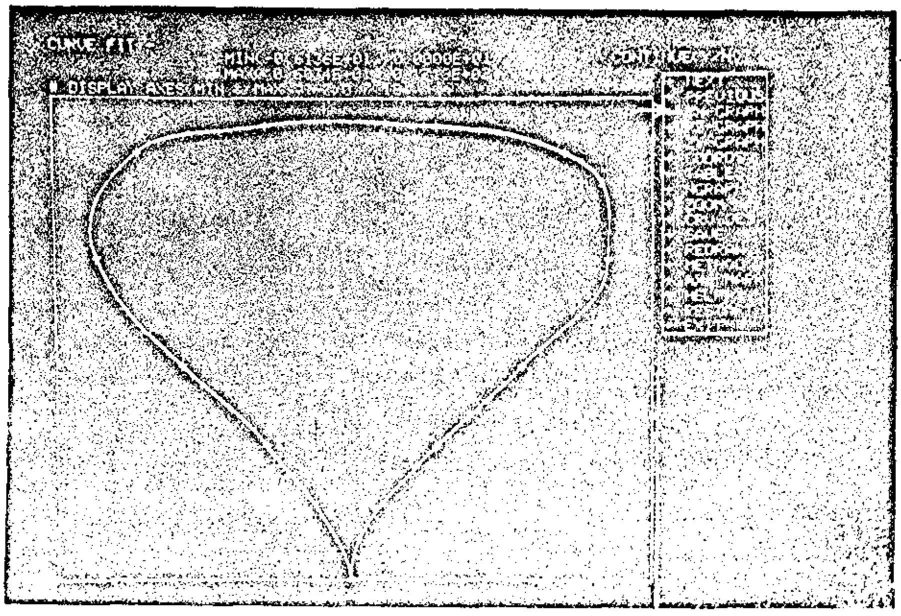


FIGURE 6.32b: Anticyclic End Condition

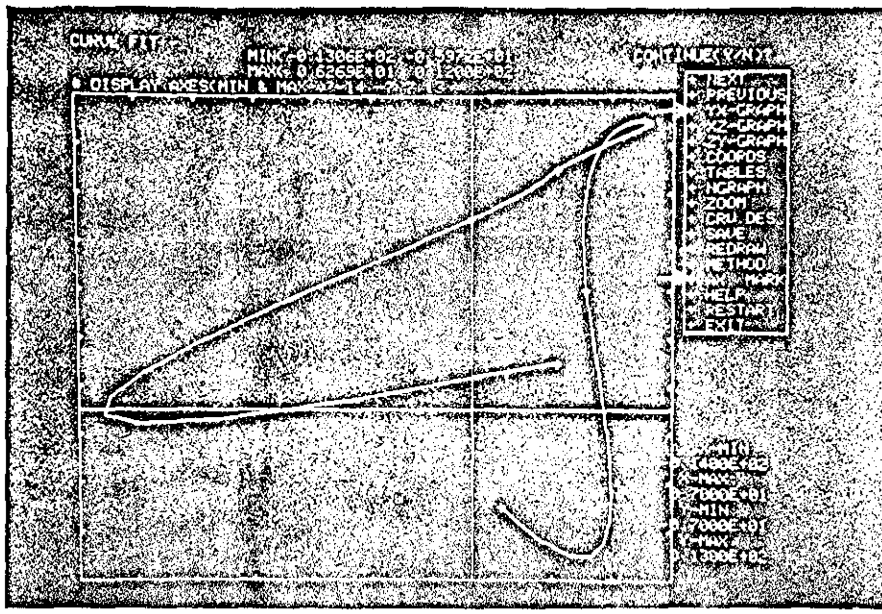


FIGURE 6.33a: Three-dimensional Curve Projected on YX-plane

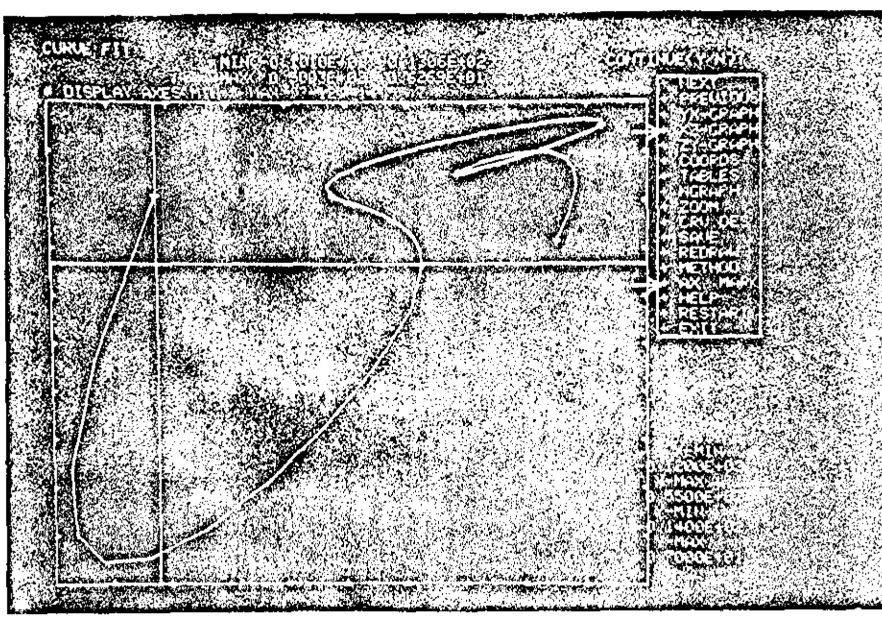


FIGURE 6.33b: Three-dimensional Curve Projected on XZ-plane

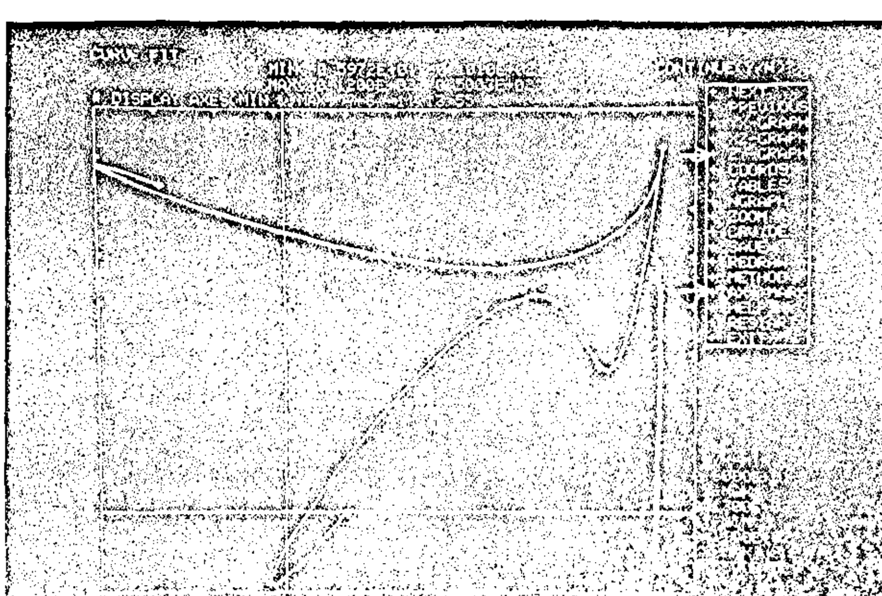


FIGURE 6.33c: Three-dimensional Curve Projected on ZY-plane

oscillations in the curve in particular near the end points.

(2) Figure 6.31 shows the result of using the two-dimensional parametric splines with natural (relaxed) end condition for a multivalued spiral function.

(3) Figures 6.32a and 6.32b illustrate the use of two-dimensional parametric cubic splines with cyclic and anticyclic end conditions respectively.

(4) Finally Figures 6.33a, 6.33b and 6.33c arise from the use of three-dimensional parametric splines with specified parametric slopes at both ends. The parameter entry display for this is shown in Figure 6.20.

4. PROGRAM DESIGN

The program development of this system was initially carried out under RSTS-11 time-sharing system using BASIC-PLUS. At that time no proper graphic software was available locally except a small set of primitive routines written in BASIC-PLUS to drive the Tektronix 4010. Since then UNIX has become operational and the LIGHT package developed; the entire system was rewritten in UNIX-FORTRAN and enhanced further by improving its design features. The system has been designed and implemented in a modular manner, so that additional capabilities can be easily incorporated; for example, new interpolating algorithms or some additional display images that would aid user interpretation of the results. Complete Fortran listings of the individual modules are presented in Appendices 2.2 and 2.3.

4.1 The Interactive Display Routines

In designing such systems we are faced with a man-machine interaction situation involving all the usual ergonomic problems.

A program (routine) is termed to be interactive if it depends for its successful completion on the establishment of a dialogue between user and machine. Therefore, the designer has to resolve the problem of structuring the program so that a dialogue will take place. The form of this dialogue will be written into the program as an algorithm, the alternatives within that algorithm being selected as the result of information either supplied by the user, or deduced within the program itself. In an interactive program certain decisions involve a simple binary choice (yes or no), whilst others require the selection of a subset of actions from a given set of alternatives. What we have, then, is a complex algorithm involving man and machine at various stages. This will require the breaking down of each stage into its constituent parts and looking at the ordering of these parts. Following this line of thinking, the IDF system basically consists of a number of interactive display routines which generate the display images and provide the interactive capability at various stages of the data-fitting process illustrated in Figure 6.5.

The experience of developing these routines under-lined the value, indeed the necessity, of an extensive use of subroutines. Although the complete package was written single-handed, the independent nature of the subroutines would have permitted a group of programmers to write and debug individual routines in parallel, once the definitions had been made.

Program communication between these routines involves the normal use of the subroutine parameter mechanism or a COMMON data area. The interactive facilities of individual routines may be extended or modified without affecting the others, providing each interface remains the same. Every interactive display routine sets up its own menu. The items (options) of the menu represent the various capabilities that a particular routine is programmed to handle. The text of the menu is defined as a hollerith string of characters in an array (e.g. MNTXT), declared as LOGICAL*1, in a DATA statement. This array type declaration in UNIX-FORTRAN resembles the BYTE type declaration used in DOS-FORTRAN [33].

It is often necessary to define a second menu in certain display routines so that a group of independent options are set in a separate menu which in turn is distinguished from the main menu (e.g. in parameter entry). This avoids the display of a large number of menu options in a single menu, which could confuse the terminal user.

The routine, having set the various default values, clears the screen (CALL TXCLER) in preparation for the next display image. Following this, it outputs the display title and often some information text instructing the user briefly on the various options or messages required for keyboard entry. The menu is then displayed by issuing calls to the LIGHT menu handling routines, for example:

```
CALL  MNOPEN(875.,715.,1)
CALL  MNDISP(MNTXT,5,10,1)
CALL  FRAME(870.,732.,5)
CALL  MNPICK(J,ICHAR,MNO)
```

As a result the user is prompted by the immediate appearance of the cross-hair cursor on the screen giving him control over the execution path of the routine through the selection of menu options. The user would now be ready to make his own choice from the menu, while the interactive routine is waiting to respond. The routine distinguishes between the two menus, when necessary, from the returned integer value in MNO(1 or 2). The routine treats the two menus in slightly different ways. For example, all options picked up from the main menu must be confirmed by the user. This avoids or at least minimises incorrect menu selection, which may cause the deletion of intermediate data files or overwriting the COMMON data area. On the other hand, options selected from the second menu have only a local effect and need not be confirmed. Their effect can be cancelled by selecting alternative options.

As soon as the user has confirmed his selection, the routine branches to the appropriate code (or calls the appropriate routine) depending on which option of the menu is picked up. This simple transfer of control is often effected by a computed GOTO statement e.g.

```
GOTO(10,20,30......),J
```

where J is the option sequence number within the menu. Therefore the routine may invoke some special purpose subroutines to perform certain tasks which correspond to the user's request. If, however, the user has selected an option that requires a new display image to be brought up on the screen (e.g. '+NEXT' or '+PREVIOUS'), then control is returned to the main program calling sequence.

4.2 Program Module

The IDF package is divided into several program modules, each module being autonomous and self-contained. The reason is that the size of the core currently available is not large enough to incorporate the entire package as a single executable module. In fact, a constraint which was imposed at an early stage of the design was the ability to run the system on a machine having 16K words of user area. This of course improves the portability of the package.

The absence of a direct overlaying scheme under UNIX, has led us to employ the simple facility provided by the LIGHT package through the subroutine call 'OVRLAY' discussed in Chapter 5.

Basically, each module has its own main program segment (driver program) monitoring user interaction with the system. This in turn controls the flow of logic between a set of display routines. The main program also passes control to other modules when required via the overlay routine, e.g. CALL OVRLAY(MODL2). Where MODL2 is the file name of the module which is to be brought from the disc into core and executed. Therefore, at any one time during the running of the system only one module is occupying the user area. A typical organisation of such a modular structure is illustrated in Figure 6.34.

The organization of the package into separate modules, each independently performing a given set of tasks, has greatly simplified the actual development and implementation work. In particular, owing to the tree structure of the UNIX file system directory, it was possible to arrange the program source codes of each module as it develops under a separate subdirectory. This helps us to test and debug individual modules thoroughly before they are included in the package and re-tested.

Data communication between modules is maintained via COMMON data areas. When the in-core module is overlayed by the new module, the COMMON data areas are no longer accessible to the new module, since all code and data of the old module is virtually replaced by the file containing the new module. Therefore, it was vital to save the COMMON data on a disc file before the new module is brought into core, and restore it immediately afterwards. These data files are created internally by the system at run time and eventually removed when the user decides to terminate the session by selecting the option '+EXIT'. Evidently, the reading and writing of these data files during the overlay operation could effect the response time of the system, and will also occupy a certain amount of disc space. However, these files have a useful function in providing a back-up facility in the

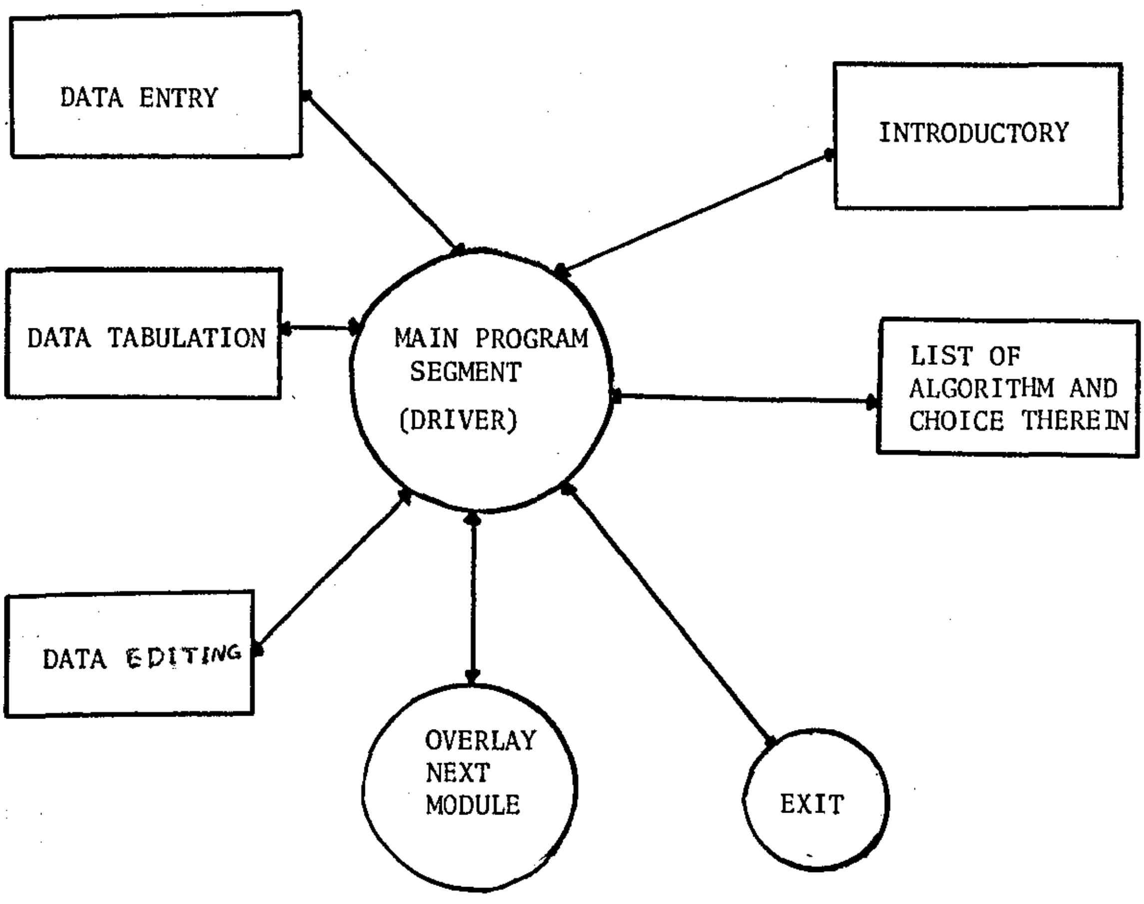


FIGURE 6.34: Program Organization of Module 1 of the IDF System

event of user as well as program errors which can cause premature termination of the session. As mentioned in section 3, the user may recover the display image and subsequently resume the program by typing the command HELP at the keyboard. In this way, the data in the files are correctly despatched into program variables and stored back again into the COMMON data areas.

There are basically two labelled COMMON data areas used to communicate between the different modules of the system:

(1) DATASUP: This contains the input information supplied by the user concerning the curve under investigation. The data points defining the shape of the curve are held in one-dimensional array (X,Y, [Z]), allowing up to 50 points to be specified. The one-dimensional integer arrays (L,IH) are also used to contain the linked list pointers. These are initialised and used in conjunction with the data point arrays to form a linked list data structure which is utilized in the data point edit operations. An array NPI is also used to hold the number of intermediate points required in each interval for actually drawing the smooth curve. Other variables are also used to contain input information such as the index of the selected algorithm, boundary values,... etc.

(2) CURVEFIT: This contains the output results of the numerical algorithm applied. The complete set of the interpolated points representing the smooth curve are stored in the one-dimensional array (XCORD,YCORD,[ZCORD]). The computed polynomial coefficients are held in the two-dimensional array COEF.

All modules share common library subroutines archived in 'EPLIB' file. Program listings of the subroutines are given in Appendix 2.4. The functions of these subroutines fall into two categories:

(1) read and write operations of the common data areas into and from the data files.

(2) utility display routines, including for example:

- setting the display window and viewport.
- marking the axes.
- reading the display origin coordinates from the keyboard.
- reading input cursor coordinates off the screen.

4.3 Overlay Support

The present overlay structure requires that each single module should have a main program segment as shown in Figure 6.34. Note that here the 'driver program' (for each module) is essentially the same and this is obviously wasteful. A much better structure would be to have a control module or root segment module (as shown in Figure 6.35) which monitors user action on the screen and automatically overlays other modules when their associated options are picked up from the menu. Each overlayed module consists of a number of interactive display routines as shown.

An investigation was made to find out the possibility of achieving such an overlay structure under UNIX. Since UNIX is a process-based operating system, in order to have the control module and any other module in core at any one time, we require two independently executing processes each having a separate core image. A new process can come into existence only by use of the system call FORK [21]. If FORK is executed by the control module as a process (parent), a new process (child) is created and its core image is a copy of that of the parent.

In the child, control returns directly from the FORK (i.e. to the instruction following the FORK), while in the parent, control is passed to the next instruction (i.e. a skip return):

```
e.g.    SYS  FORK        /CALL FORK OPERATION
        BR   CHILD       /RETURN HERE FOR CHILD
        :                /RETURN HERE FOR PARENT
```

At label CHILD in the above example, we may execute code in the new process. Should we wish to overlay this process with a new child process, we execute an EXEC system call as part of the CHILD code, for example

```
        CHILD:  .
                :
                SYS EXEC
```

Parameters supplied with the EXEC system call specify the image to be overlayed. This image is overlayed and control passed to it. Meanwhile the parent process is continuing to run 'in parallel'. In the present case, we need to cause the parent to wait for the child to terminate since the parent is doing no more than interfacing with the user at command level. This is effected by using the system call WAIT. Hence the code for overlaying is:-

```
                :
                SYS FORK         /CALL FORK OPERATION
                BR  CHILD        /RETURN HERE FOR CHILD
                SYS WAIT         /HERE FOR PARENT WAIT
                :                /FOR CHILD TO TERMINATE
        CHILD   SYS EXEC         /OVERLAY NEW MODULE
```

Hence, the child process core image can be overlayed by the appropriate module and control passed to it.

The remaining problem is to provide a means of accessing data from the COMMON area. Since processes are normally swapped in and out of core and they are relocatable and totally independent, the COMMON data area in the control module (parent) is no longer accessible by the overlayed module (child).

There are operating systems which have a built in overlay mechanism for handling such an overlay structure. As an example, the RSX-11 [34] assumes that the basic program unit executing under its environment is a task, which may consist of a program module or a set of program modules. The overlay structure consists of a single root segment and any number of overlay segments which share memory with one another. Any one of these overlays may likewise give rise to a number of tasks which further overlay one another Figure 6.36. In this example, A,B and C overlay each other as do the tasks belonging to A or B or C. All tasks at a given level (indicated by the arrows) share a COMMON area of memory. Considering any task at any level, it is possible for that task to access global data in any segment on a path between that task and the root. For example, in Figure 6.36 task D may reference any global data in task A or the root segment. Ultimately all tasks may reference global data in the ROOT segment and hence this may be used to store, amongst other things, shared data. The Fortran COMMON block may be included in this shared region, making it accessible to all other modules. Unfortunately, with the present environment (i.e. under UNIX), it is not possible to have a shared region between modules to hold the COMMON data. However, communication between the two processes can be accomplished by either using data files as before or incorporating the system entry 'PIPE' which allows communication between processes.

4.4 Data Structure

Since this application has offered the user the facility of manipulating interactively the data points representing the curve, it requires a dynamically changing list of points. For this purpose, a

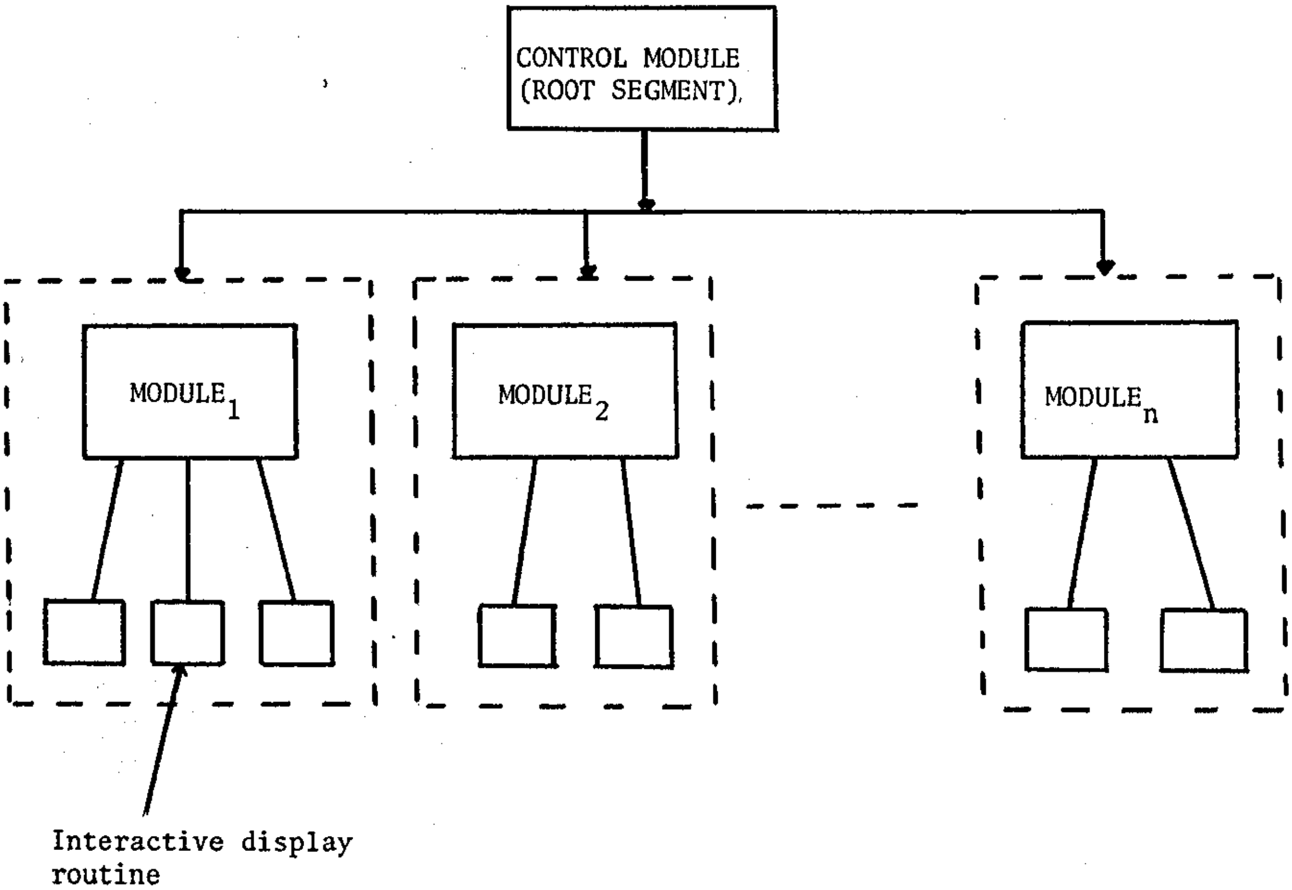


FIGURE 6.35: The Proposed Overlay Structure

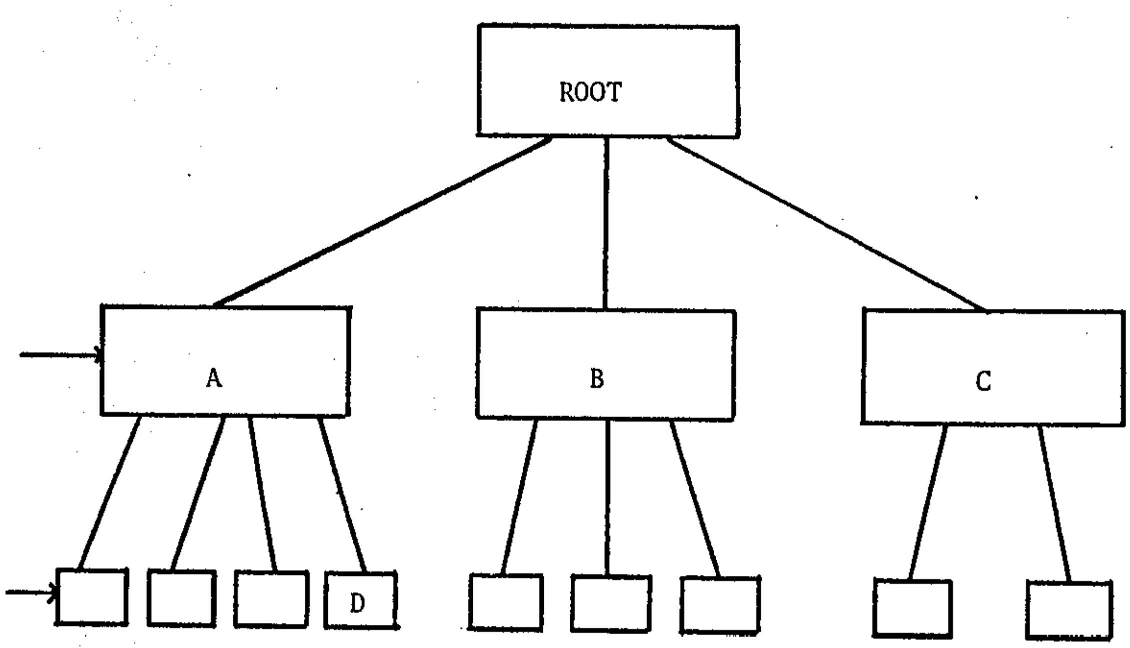


FIGURE 6.36: Overlay Segments

simple form of linked list data structure was tailored for this application, so that anywhere in the list data points can be inserted, deleted or replaced by other points without disturbing the rest of the structure.

These capabilities were provided as shown earlier in the data editing and the curve design displays. In the former, the user was able to alter freely the list of the input data points displayed in tabular form, whereas in the latter he was given the ability via the cursor to change the number of intermediate points required in each interval directly on the displayed curve.

Therefore, both the input data points and the output points (computed smooth curve) are associated with pointers forming a linked list data structure. Basically, each element of the list consists of three (or four in the case of 3-D curves) data items as shown in Figure 6.37.

When a data structure is implemented, it is not only necessary to design the structure of the data, but also the algorithms which can retrieve and manipulate the data and its internal relationships. An important choice to be made in designing such a system is the balance between structure and algorithm, because the two are usually complementary.

The linked lists chosen for this application are partitioned into sublists, in order to avoid searching the whole list to look for a particular element when performing a certain operation.

Consider first, the structure used in organizing the input data, as shown pictorially in Figure 6.38. The integer array IH holds the pointer to the start of each sublist which consists of a maximum of ten elements. A free list pointer is used to point at the list of unused elements. If a new element is required, it is taken from the free list and conversely if an element is deleted from the list, it is returned to the free list.

A function subroutine INDEX is provided to do the mapping between the sequence number of the data point in its display tabular form and that of its actual location in the array list. This function returns an integer value indicating the location of the required element in the list. Three subroutines are also provided to handle the three basic edit operations on the list, namely deletion, insertion and correction.

Now consider the linked list associated with the output data points. This has essentially the same structure except that the last element of each sublist is not connected to the first element of the next sublist, as shown in Figure 6.39. Basically, each sublist consists of an input point followed by the intermediate points for that interval. The operations carried out on this list are concerned mainly with the deletion and insertion of intermediate data points. Routines are provided to handle these operations, keeping track of the free list and doing the garbage collection.

Finally, a data structure was also implemented for appending the segments of joined curves together. As this required a careful use of a set of array pointers showing precisely the start and end of each segment and their associated intermediate points. This structure is shown in Figure 6.40. The real arrays XJ and YJ contain the complete set of points for each segment. The integer arrays JJ3, JJ4 and IPNTR hold various items of information including number of intermediate points, start address of each segment and so on.

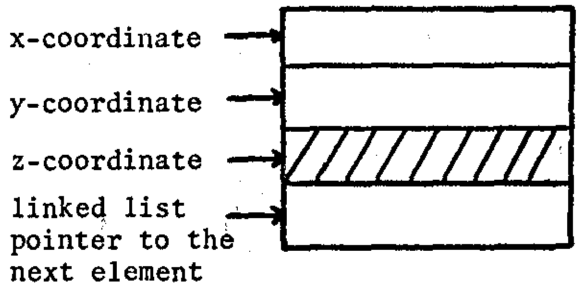


FIGURE 6.37: An Element of the Linked List Data Structure

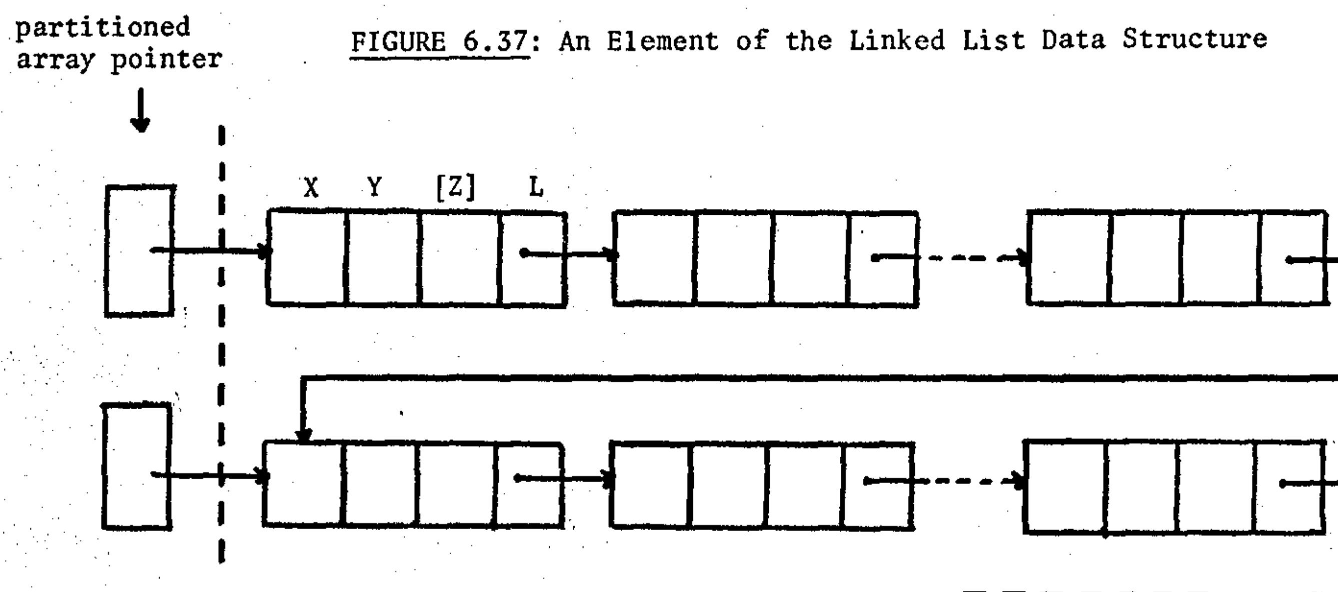


FIGURE 6.38: Data Structure Used in the Data Point Editing Display

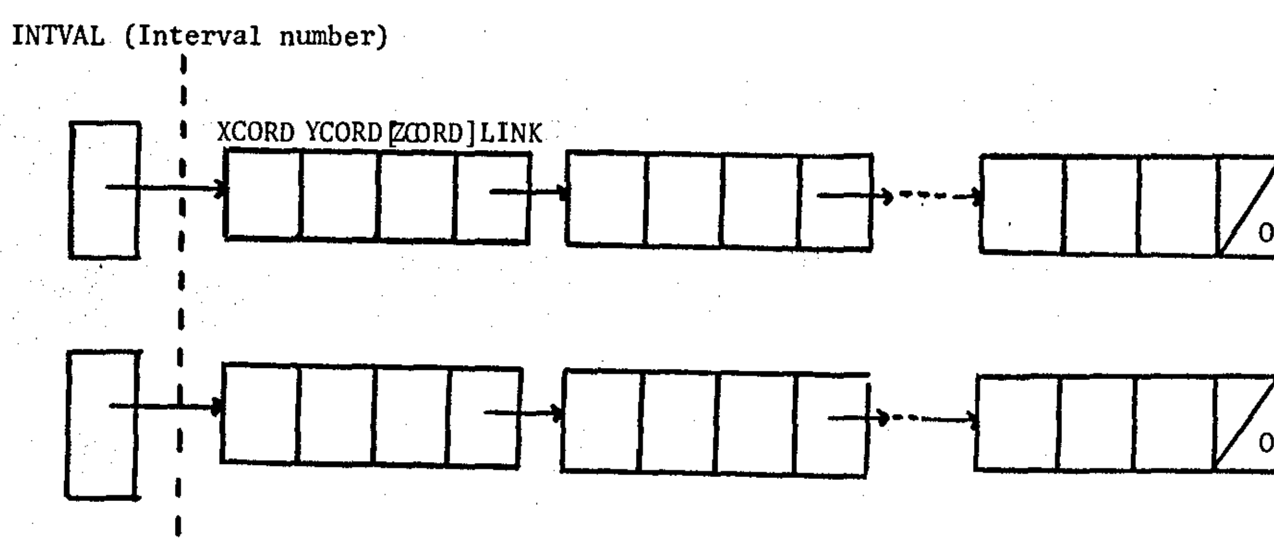


FIGURE 6.39: Data Structure Used in the Curve Design Display

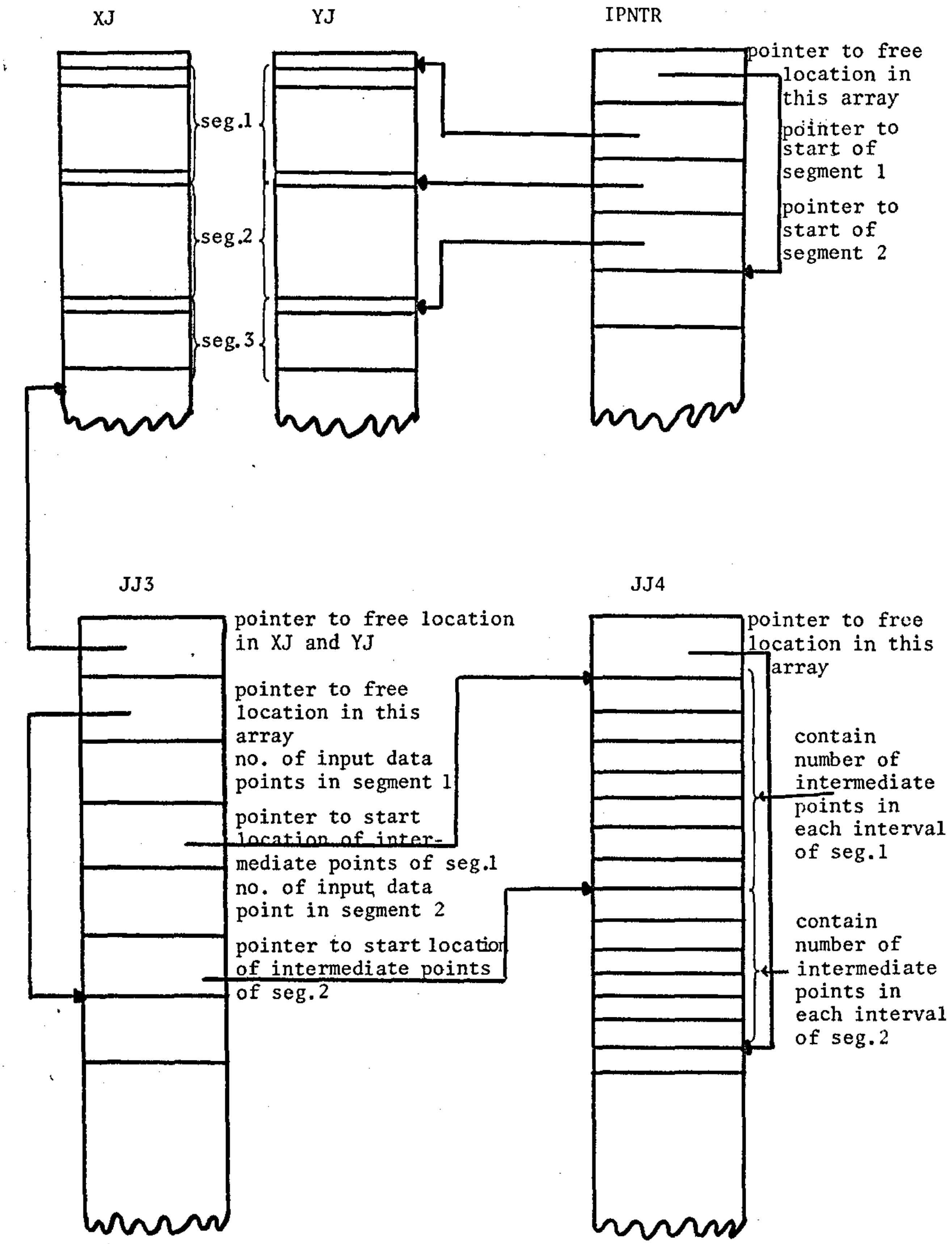


FIGURE 6.40: Joined Curve Data Structure

CHAPTER 7

INTERACTIVE CONTOUR TRACING - ICT

1. INTRODUCTION

1.1 Application Area

The representation of surfaces in the two-dimensional plane by means of contour maps is in widespread use in science and engineering. Such maps have both quantitative and pictorial value and thus are frequently used for presentation of final results and as a research tool.

The most common example is a contour map representing elevation as a function of position in a two-dimensional geographical region. Other position-dependent variables that are commonly represented in the form of contour maps are temperature (isotherms) and pressure (isobars). In some applications, contour maps may be used to facilitate visualisation of data even though an equation may exist that describes this data, for example, a plot of the equipotential lines around an electric dipole. Also, contour lines can be used to represent functions which are involved in some optimization process.

1.2 Contour Lines

Contour lines are usually defined as the lines of intersection between a given surface and a family of parallel surfaces, usually horizontal planes. This definition is ambiguous if the given surface has a horizontal portion at the same elevation as one of the horizontal planes. In order to overcome this difficulty Morse [35] defined the following types of contour lines (Figure 7.1):

(i) A positive (negative) contour line is a line connecting points all of the same elevation such that points adjacent to one side of the line are at higher (lower) elevation and points adjacent to other sides are at the same elevation or at a lower (higher) elevation. The usual contour line found on a contour map is the union of a positive and negative contour line, both of the same elevation. A line is called a normal contour line if it is either a positive or a negative contour line.

(ii) A maximum (minimum) contour line is a line connecting points all of the same elevation such that all points adjacent to either side of the line are at a lower (higher) elevation. Maximum and minimum contour lines are degenerate cases of the normal contour (Figure 7.2).

The above definitions take care of all possible pathological situations in contouring. However, since these degeneracies are extremely rare and are not easily reproduced by finite computation of the function, they are not usually incorporated in a general-purpose contouring algorithm. The following two non-degenerate properties of contour lines are assumed in the subsequent work.

(a) Different contour lines never cross.

(b) Normal contour lines which do not intersect the boundary of the map are closed curves.

1.3 The Contouring Problem

Contouring is increasingly being implemented by computer with the aid of graph plotters and CRT displays. The object of contouring is to draw curves of constant value of a dependent variable (z), projected into the plane of two independent variables (x,y). Each contour is approximated by piecewise-continuous lines and the basic problem is that of locating and linking these line segments.

The information (raw data) supplied will normally consist either of a finite number of values of the dependent variables (z_i) at a set of locations (x_i,y_i) - Figure 7.3 - or an explicit or implicit mathematical expression relating z to x and y e.g. $z = \exp\{-1/4\,[(x-5)^2+(y-5)^2]\}$.

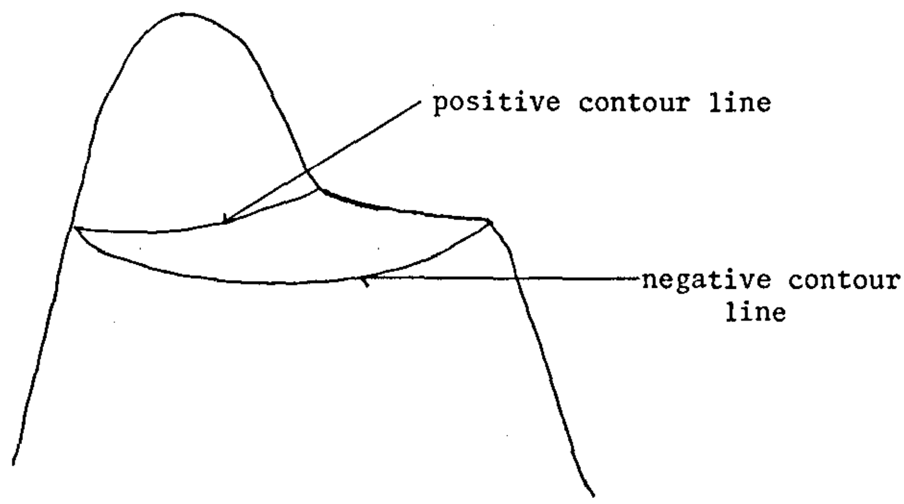


FIGURE 7.1: Resolving the Ambiguity

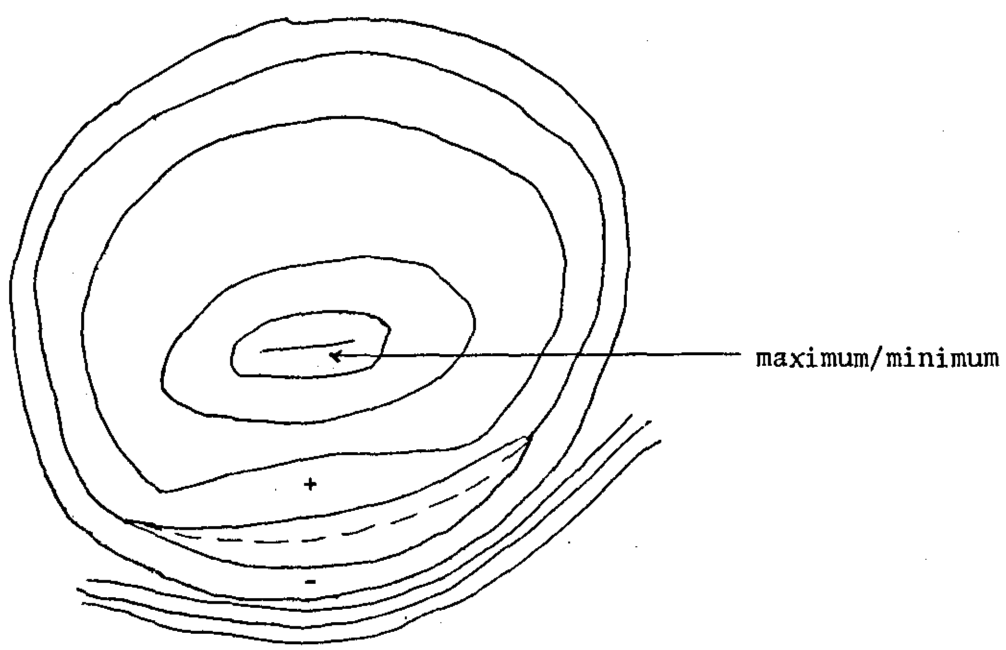


FIGURE 7.2: Degenerate Contours

In general, to trace a set of contour lines $c_j=f(x,y)$ for $j=1,2,\ldots,m$ over a desired region, it is the usual practice to have a reference system or grid in order to keep track of the contours and the regions which have been explored. For this purpose at least two distinct approaches are possible:

(i) Partitioning the region into triangles whose vertices are the known finite set of data points. This triangulation involves joining neighbouring data points by straight lines to form triangular plane segments (Figure 7.4). An algorithm must be included to form an optimal partitioning of the region into triangles. An optimal partition, Pitteway [43], is one in which for any point within any triangle, that point lies at least as close to one of the vertices of the triangle as to any other data point. The triangles must be assigned so that they are as nearly equilateral as possible. The addition of this possibly time-consuming algorithm to the program constitutes the major disadvantage of the method. However, it may be useful where the data locations are fixed (though non-equispaced), such as a set of permanent observatories or recording stations. The triangles then can be established once for a number of contour applications.

(ii) Superimposing a regular mesh on the region where it is used as a reference system. Most contouring algorithms make use of a rectangular grid (or net) imposed on the desired region (Figure 7.5). These algorithms are conceptually simpler and usually faster than those based on other reference systems.

The overall problem of drawing contour lines from a given arbitrary set of data points involves two logically different stages:-

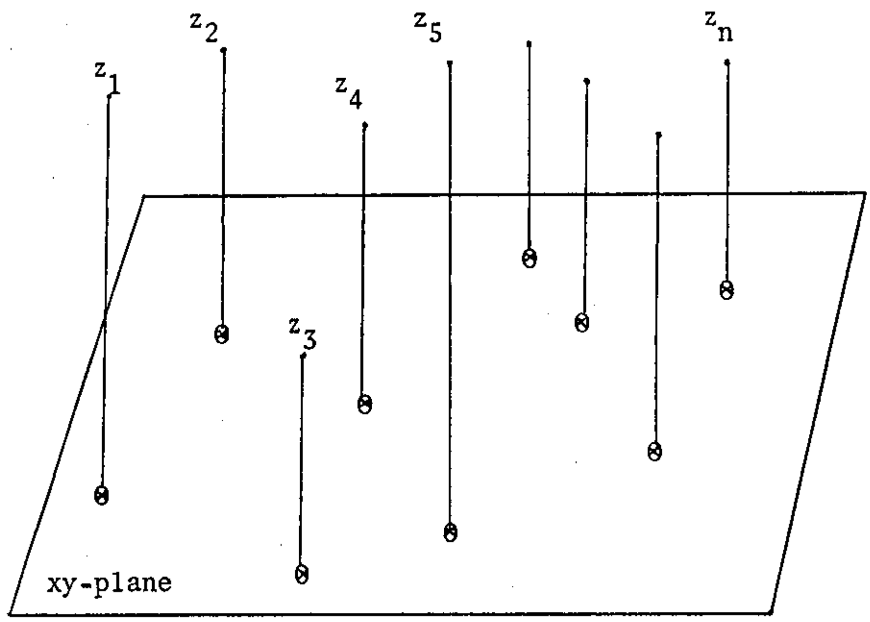


FIGURE 7.3: Finite Set of Data Points Given At Arbitrary Locations

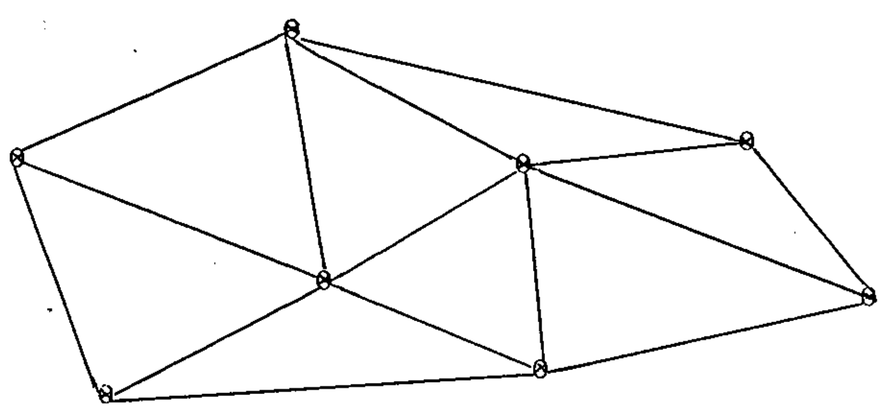


FIGURE 7.4: Triangular Partitioning

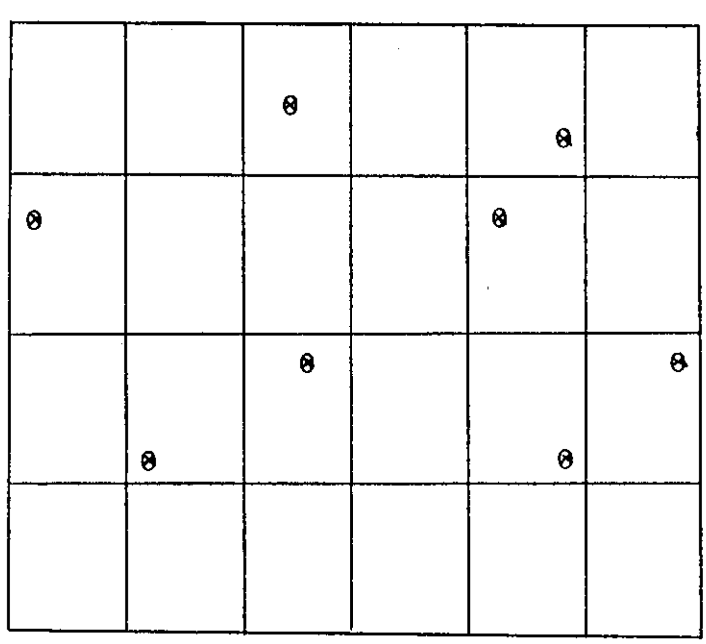


FIGURE 7.5: Regular Rectangular Mesh Superimposed Over a Finite Set of Data Points

(1) Interpolation, where for given n values $z_1, z_2, \ldots, z_n$ at the positions (x_1,y_1), $(x_2,y_2) \ldots (x_n,y_n)$, it is required to estimate a value of the function $z = f(x,y)$ at an arbitrary new location.

(2) Contour tracing, where for a given function $f(x,y)$ which can be calculated at any arbitrary location, it is required to draw a set of contour lines $f(x,y) = c_j$ for prescribed c_j, $j=1,2,\ldots,m$.

When a rectangular grid is used, the two stages would involve the interpolation from the raw data to the grid points and then the contouring of the grid data.

An algorithm recently developed by Schagen [42] is based on the triangulation approach and interpolates from a set of arbitrary data points, using a generalised least-squares technique. The program has been implemented in Fortran on the ICL 1904S using an incremental plotter with GINO-F graphic software.

The work reported in this chapter is mainly concerned with the development and implementation of an interactive contour tracing algorithm based on a rectangular grid using a graphic display terminal (Tektronix 4010) and the local graphics package, LIGHT. The algorithm would trace the curve $f(x,y) = c$ in a region in which there is a method of calculating the function. Several methods exist for estimating the function value at mesh and intermediate points from a set of arbitrary data points. For example Mclain [39] uses a weighted least-squares technique followed by local bicubic spline interpolation. Another method suggested by Powell [41] makes use of triangulated piecewise quadratic surface fitting.

1.4 Regular Grid Techniques

For a given two-variable function $z = f(x,y)$ over a rectangular domain R defined as $a \leq x \leq b$, $c \leq y \leq d$, the region of interest is discretised into a rectangular grid. This produces a subdivision of the region into smaller rectangles with four adjacent nodes of the grid as vertices. Denoting each node by (i,j), a matrix $[z(i,j)]$ can be defined. This regular grid provides an interface between the interpolation and contouring stages mentioned above, where the nodal values could be generated from the raw data at the interpolation stage.

Once the grid values of the function are produced, the drawing of contour lines can be accomplished. Some of the details of a particular contouring algorithm may depend on the output device used for drawing the lines, but most of the principles involved are device-independent. No matter what output device is employed, the solution could be achieved by two logical steps:-

(i) Computation of the coordinates of all the intersection points of each contour line and the edges of every rectangle.

(ii) Suitable linking of these points i.e. the organization of the drawing of all contour segments.

These require two types of interpolation:

(1) On the edges of each rectangle (i.e. step (i))

and (2) Within the rectangles themselves (i.e. step (ii)).

The contour tracing algorithms can be divided into two categories depending how the above techniques are employed:

(a) Grid-scan Techniques

In this, step (i) above is completely exhausted as a whole, by scanning the entire region to find the intersection points. Then, using some criterion, to be established, the points found in step (i) are linked during the execution of step (ii). The methods using this technique require the storage of all intersection points. However, the storage of the entire matrix z(i,j) can be avoided if all m contours are treated simultaneously, but the problem of labelling the different curves may become difficult to program. The simplest method is to search for intersection points by exploring all the rectangles in a chosen order e.g. row by row, and join each intersection pair in every rectangle. This is very simple to program but is impracticable and time-consuming. The slowness is due to the disjointed series of drawing movements involved and is exacerbated when a plotter is used. To minimize plotter pen motion, it is essential to plot each contour level continuously by ordering the intersection points in their natural sequence along the contour. Advantages of the method are rapid computation and simplicity of programming.

(b) Contour-following Techniques

One 'entry' point of the contour line is located and the line is followed until it exits from the region. The two steps mentioned earlier are thus performed together; i.e. continuous generation of segments of a single contour line. There are, accordingly, no difficulties in labelling the different lines, since they are found in succession. This technique requires the storage of the entire matrix z(i,j), if the function is empirical, or not easily calculated, but storage of the intersection points is not needed. The procedure would terminate upon closure of the line or intersection with region boundaries. The logic of such a method is necessarily more complex than the grid-scan technique, and may therefore require additional computer overheads. The scheme involves some difficulties concerning the identification of different branches of the same contour level, usually without any common point. In order to follow all the branches, another search is necessary to find at least one point for each of them.

Briefly, the procedure commences with a scan until a contour crossing is located. The contour is then followed to completion, and the scan continued.

(i) Calculate and draw contour line segment in the rectangle where a crossing is located.

(ii) If the contour is closed or has reached the boundary, return to scan for the next contour crossing; else continue.

(iii) Determine into which of the adjoining rectangles the contour exits. Return to (i) for this new rectangle.

A method of preventing the redrawing of a contour when another portion of it is encountered by the scan must be included in the logic. This has been accomplished by making use of an auxiliary array to identify previously contoured lines. The fundamental advantages of contour-following techniques are the increased speed of plotting and the ease of adding contour labels.

2. REVIEW OF SOME EXISTING METHODS

The purpose of this section is to outline briefly the main features of some existing contour drawing algorithms which are based on a regular grid and to highlight their differences. All these algorithms basically retain the two logical steps (Section 1.4) which are required to draw the contour lines but the approach in each step varies slightly from one algorithm to another. Most of the algorithms below substantially employ a contour-following technique.

2.1 Cottafava and Le Moli [36] 1969

This method incorporates a preliminary step, searching for the points of intersection with edges, but without calculating or storing the coordinates of these points, (requiring only that the matrix $z(i,j)$ be stored). It finds the intersections by systematically exploring the edges of each rectangle. An edge, say, AB of the rectangle is intersected by the contour line $z = c$ if c lies between the function values of the two adjacent nodes, i.e. if

$$(Z_A-C).(Z_B-C) \leqslant 0 \ . \tag{7.1}$$

The results of this test are stored in the array $z(i,j)$ by recording for each pair of adjoining edges two binary variables specifying whether the corresponding intersections exist (Figure 7.6). In order to follow the contour line through the intersected edges, the algorithm has to accomplish the following two tasks:

(i) The behaviour of the line in the interior of the rectangle and in the adjacent ones must be examined. If an edge of a rectangle (i,j) is intersected by a level line, then at least one other edge of the rectangle (i,j) is intersected by the same contour line. A fixed order of edge inspection is used throughout the contour tracing. The following steps are made to follow the line:

  (1) Search in the chosen order for a rectangle edge with a stored intersection.

  (2) The stored intersection is cancelled to avoid meeting it again.

  (3) The intersection coordinates are calculated by linear interpolation.

  (4) The analysis continues for the rectangle adjacent to the intersected edge by repeating the same procedure from (1).

In this way an ordered point set is constructed. The above procedure has examined the contour line from two viewpoints: the first is topological and involves verifying the existence of intersections and following the line; the second is numerical and involves calculating the coordinates along the edges. Therefore, the contour is defined by a set of points on the edges.

(ii) The behaviour of the line with regard to the whole domain must also be examined. This is divided into two parts:

   (1) Search for starting point from which to follow the contour line in (i). This is accomplished by scanning all the edges in the chosen order until a stored intersection is found and then following the branch until it stops. Scanning for new starting points on other branches is resumed from the interruption.

   (2) Search for the stop point of a line. If the contour line is closed the stop point can be obtained as a consequence of step (2) in (i). The following convention can be established: a line stops when no intersection is found during execution of step (1) in (i) or the region boundary has been reached. In the latter case, further investigation of the adjacent rectangles external to this boundary is considered to be necessary. This method is quite general and allows one to follow any contour line.

2.2 <u>Rothwell [37] 1972</u>

This is a simplified version of the algorithm described above. Often, a generalised tracing procedure is unnecessary and time consuming when some properties of function $z(x,y)$ are known a priori, e.g. a level line has only one branch. In such cases, no preliminary operation is required and consequently the need for intersection storage is avoided. The procedure is as follows for a single contour line:

(i) Search the grid lines for a set of points for which $f(x,y)=c$. To do this, some form of approximation to the function has to be made between the nodes.

(ii) Having obtained the set of intersection points, join these points in some way to produce the contour, which in general will be in several discrete sections.

The four edges of the region are searched for the start of an open contour. The contour is then traced through the region until another edge is found. When all the open contours have been traced, the interior of the mesh is searched for closed contours which are drawn similarly. As observed earlier, no intersection storage was needed. However, such storage has the following advantages:

(1) It avoids finding an intersection repeatedly.

(2) It allows following of the contour line to be easily stopped.

(3) It allows different branches to be distinguished. The Cottafava and Le Moli step (ii) was required in order to identify different branches and the start and end of the contour line.

2.3 Robinson and Scarton [38] 1972

The procedure suggested here tackles the problem in slightly different form, but still retains the same structure. The object of this algorithm is to determine the possibility of finding more than one relative maximum occurring in z(i,j), which could give rise to several disjoint and highly convoluted curves at a given contour level. This algorithm is set up to accomplish the following tasks:

(1) Examine the complete array z(i,j) at each contour level so that all curves (branches) are found.

(2) Remember the position of all previously found and plotted lines so that the contours are not repeatedly redrawn.

(3) Find successive points along a contour line and keep them in their proper order so that complicated contours are drawn correctly.

(4) Have some means of deciding when all points in a contour have been found, and whether the curve is open or closed.

The algorithm consists mainly of two parts which are carried out simultaneously. First, the rectangles are scanned to find the start point of a contour line, and the position is found more accurately using quadratic polynomial approximation on the four neighbouring points. The line is then followed by threading through the adjacent rectangle. The quadratic approximation which is incorporated here allows the use of a coarser grid of points and gives much improved results over linear interpolation, particularly in the neighbourhood of saddle points.

The second part which is executed simultaneously is concerned with setting up a separate memory array. Each element of this integer array is identified with a rectangle by means of a simple code and contains all the necessary information concerning previously found contours. In particular, each element tells whether or not any contour points have been previously found along an edge of the corresponding rectangle. If any have been found, the total number of points and their locations are also stored. For each contour level, the whole memory array is initially set to indicate that no contour points have been found. Whenever a contour point is found along the edge of a rectangle, the corresponding element in the array is immediately recalculated to indicate this fact and to indicate on which side it was found. The information in the memory array is used for two purposes:

(1) If a point is found, while scanning a rectangle to find a beginning point of a contour line, then the memory is checked to see if the point has previously been found. If it has, the point is ignored and the scanning proceeds. In this way, a contour line is only found and drawn once.

(2) When extending a contour line from rectangle to rectangle, the array is checked for previously found points in the new rectangle. If such points are found, one of several alternatives will occur:

(a) the routine will first check all remaining sides of the new rectangle for contour points. If one can be found which has not been previously used, the contour line is simply extended in that direction.

(b) If no new points can be found, the algorithm checks to see if it is in the same rectangle as the beginning point of the line. If so the contour is assumed to be closed with the last point joining on to the first. If it is not the initial rectangle, the contour is assumed to be open, and is then extended backwards from the starting point until no further points can be found.

2.4 Mclain [39] 1974

This avoids the program difficulties of recording which parts of which line have already been drawn as discussed in the above algorithms. The approach here, is a more direct one than is usually adopted above, e.g. Cottafava and Le Moli use an elaborate procedure for threading the contours. This procedure works as follows:

The area under consideration is divided up into small rectangles within which the contour lines can be assumed to be uncomplicated, with no extremely sharp corners. Within each rectangle, to draw a contour line of height c of a function f(x,y), a zero of f(x,y)-c=0 is first found along one of the sides. The contour is then traced through the rectangle by a series of steps in one of eight directions (N,NE,E,SE,etc.). This effectively results in further subdivision of the region into still smaller rectangles called cells (Figure 7.7). The direction of the next step to be taken is chosen from one of three which depend on the direction of the previous step. The three steps (1,2,3) to be considered are one in the same direction as the previous and each of the other two in a direction at 45^{o} to either side of this. For each point 1,2,3 the absolute value of f(x,y)-c is calculated and the point with minimum value found. This is the point chosen for the next step. This process is repeated until the contour line passes out of the rectangle. A record is kept of the x and y coordinates of the exit point. This is repeated for all roots along the side of the rectangle, first checking them against the current list of exit points. (If they are members of this list then they are not used as starting points for tracing the contour inside the current rectangle).

It may be observed that this algorithm fails to follow the contour line correctly in some circumstances. Consider for example a function which has a surface cross section shown in Figure 7.8, in the region of the indicated height. Applying the Mclain algorithm we have the choice of three function values - one on either side of the point where the true contour line crosses the surface i.e. (1), (3) and one (point (2)) in the bottom of a 'valley' to the right (Figure 7.8a). Since the algorithm only considers the absolute magnitude of the difference between the function values and contour line being drawn, point (2) is wrongly chosen. Once this step has been made, the algorithm may then have the choice of three values as in Figure 7.8b. Clearly point (1) is the point with smallest absolute difference and so will be chosen. Consequently, rather than the contour being traced, this valley is mistakenly followed.

A solution which has been suggested [44] to this problem requires that the algorithm must detect the point when it starts following the

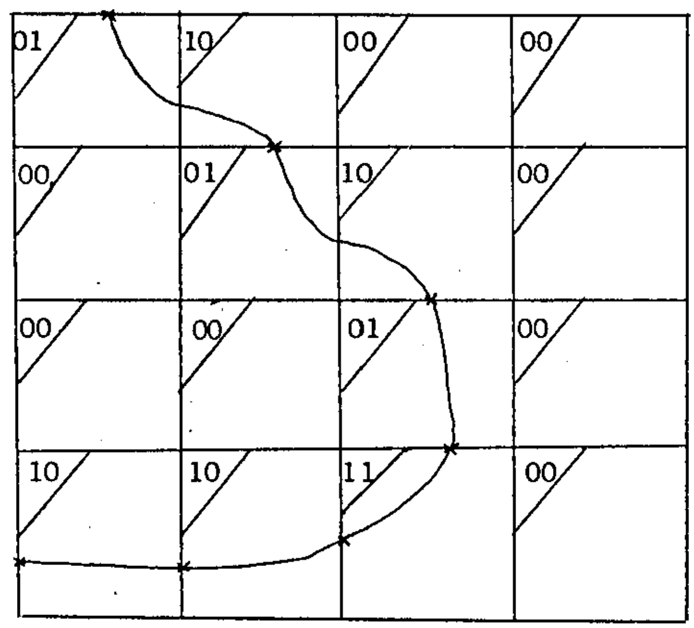


FIGURE 7.6: Cottafava and Le Moli Method

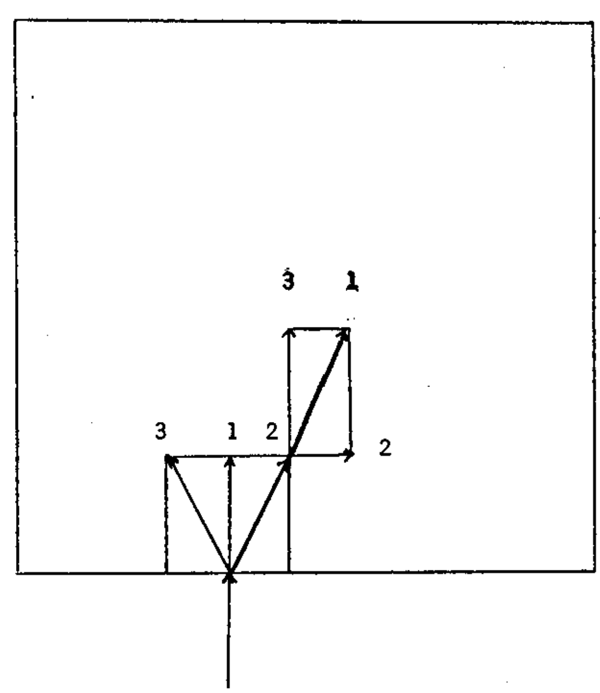


FIGURE 7.7: Mclain Algorithm

valley and stops the tracing at this point. The other end of the contour is then found when the other roots along the side of the rectangle are examined, and the rest of the contour will be traced from that end. It is possible to accomplish this at the stage where the algorithm has to choose between the three values of f(x,y)-c in the case when these values have the same sign. This involves quadratic interpolation of f(x,y)-c on the points (1),(2),(3) with local coordinates 0,+1,-1 respectively. A valley is detected if the roots of the quadratic are complex or if the absolute value of the smaller root is not sufficiently close to zero.

2.5 Sutcliffe [40] 1975

Basically, this algorithm adopts a similar strategy to that suggested by Mclain above. Both methods trace the portion of the contour line directly within each rectangle without the need to perform some preliminary operation or record information regarding the entire region. Furthermore, a common feature of both algorithms is that during the actual drawing process, each rectangle is effectively subdivided into smaller rectangles which are referred to as cells (Figure 7.9). Only those cells in the neighbourhood of the contour line are examined. However, the main difference between the algorithms relates to the way in which the contour is traced.

(i) The starting point of the contour line in this algorithm is taken as the middle of an interval which brackets the root of f(x,y)-c=0. Mclain, however, determines the starting point by finding more accurately the root of the equation f(x,y)-c=0 along one side of the rectangle.

(ii) The direction of the line segments to be drawn are determined according to the sign change of the function values at the corners of the cell. As a result a straight line is drawn to the middle of the cell side which shows a sign change of the

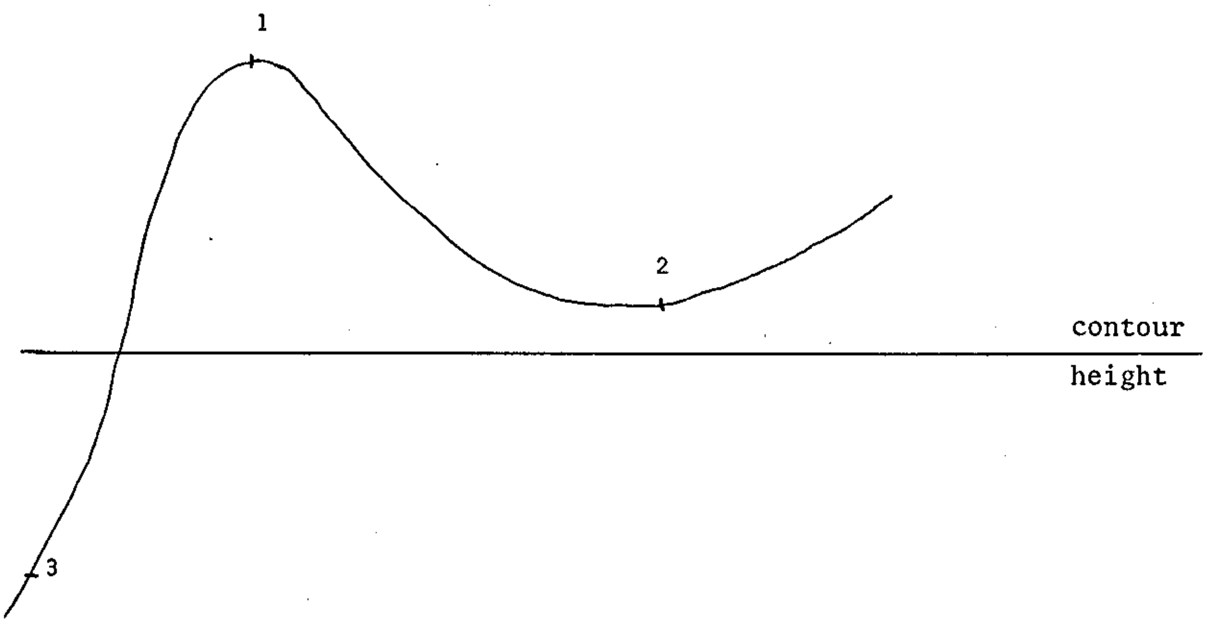


FIGURE 7.8a: Situation (showing choice of points) Where Error Will Occur

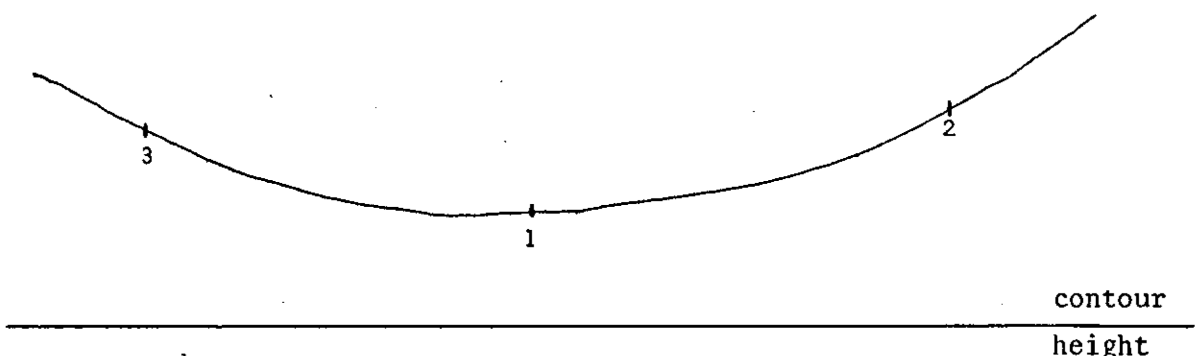


FIGURE 7.8b: Typical Choice of Points After Error Has Occurred

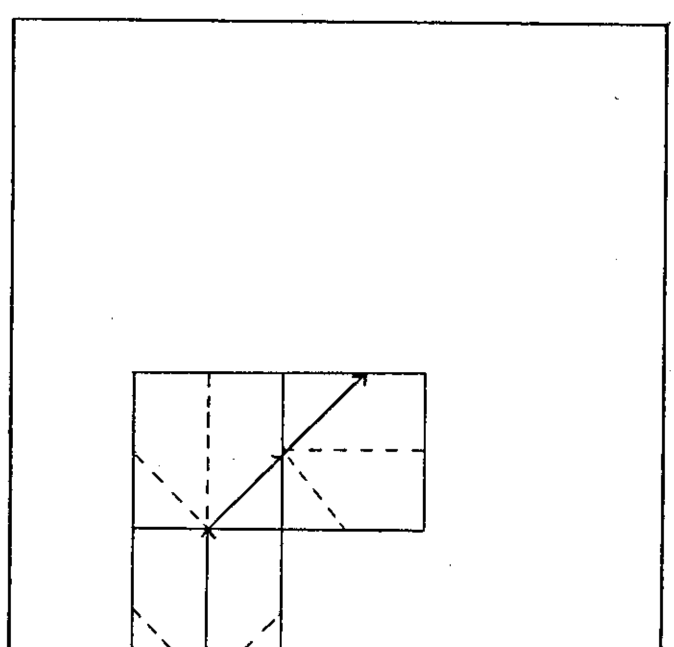


FIGURE 7.9: Sutcliffe Algorithm

function at its ends. Subsequently, this side is used as a baseline for the next cell and the process is repeated. This method avoids the problem which can arise in the Mclain algorithm where the contour is incorrectly traced in the neighbourhood of a valley.

3. DEGENERACY PROBLEM

Having reviewed some of the existing algorithms, it is essential to see how they tackle the problem of degeneracy that might occur. Degeneracy in this context is connected with the assumed function discretization, rather than any pathological function behaviour. The problem of degenerate nodes (corners) and degenerate cells described by Cottafava and Le Moli can be dealt with as follows.

3.1 Node Degeneracy

This arises when a contour line passes exactly through a grid point (Figure 7.10). Cottafava and Le Moli, Rothwell and Crane [45] all avoid this problem by scanning all the nodes and altering the associated values by a small amount so that the contour never passes through a node. Robinson and Scarton, on the other hand, use a slightly more sophisticated method (with more accurate root information) and make the next rectangle to be examined the one which is diagonally opposite the present one. However, Sutcliffe suggested that it is considerably easier to treat a zero value as if it is positive. This has the same effect as the scan above, as it can be simply achieved by writing a sign-finding function to produce only minus one (negative) or plus one (positive). Thus degenerate corners are effectively avoided.

3.2 Cell Degeneracy

When a cell is intersected more than twice by the contour line, it is usually referred to as a degenerate cell (Figure 7.11). It should be noted that at most one intersection with each edge is assumed using test (7.1). Therefore a rectangular cell has no more than four intersections and in general there can be only two or four intersections. In the latter case, the problem of linking the four points arises. We may also observe the following:

(1) We can not be sure which configuration is the correct one when the function values are only known at the four vertices. However, some interpolation criterion is usually applied within the interior of the degenerate cells.

(2) In any degenerate cell, the function must have a minimum, maximum or saddle point. In the last case, the configuration in Figure 7.11b is the correct one if the contour level equals the value of the function at the saddle point.

(3) In the complete domain there will be very few degenerate cells if (as is usually the case) the basic cell size is small.

The problem of degenerate cells is slightly more complex than in (3.1). Cottafava and Le Moli argue that degenerate cells occur very rarely and that, when they do, it is unlikely that the case in Figure 7.11b is the correct interpretation. Consequently, their method interprets the situation as either (a) or (c), choosing between them by the way the rectangle is examined. Rothwell similarly only looks at cases in Figure 7.11a and Figure 7.11c, finds the direction of the two possibilities and chooses the one where the direction changes the least from the previous step. He also comments that no strategy can be accurate in all places, due to lack of information, and this was the one he found to be the best. Sutcliffe adopts a similar strategy to that of Rothwell. The curve tracing

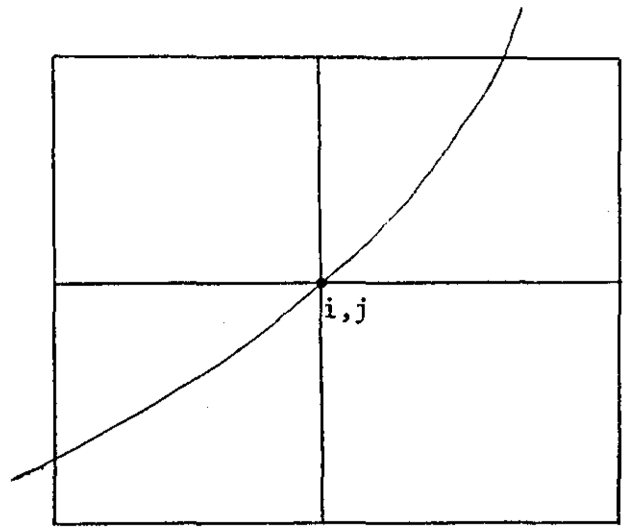


FIGURE 7.10: Degenerate Node (i,j)

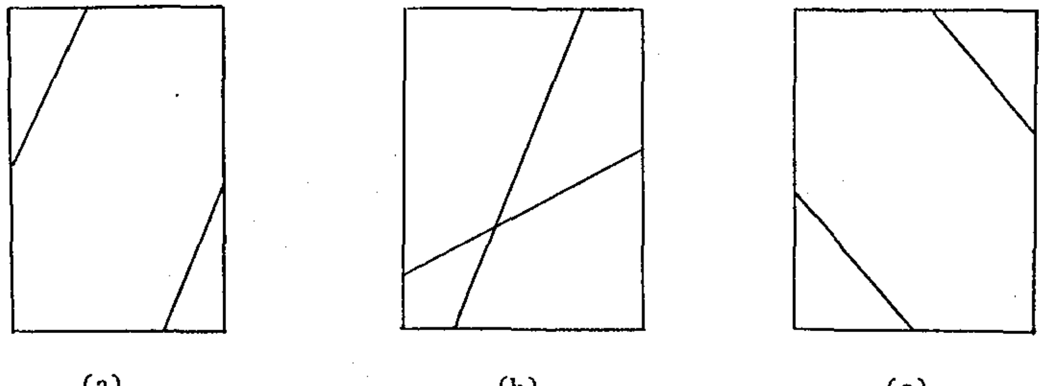


FIGURE 7.11: Degenerate Cells

is continued in the same direction as before if a degenerate cell is encountered: i.e. if the previous step was normal to the cell side, then the next step is straight across the cell, or if the previous step was diagonally across the cell, the next step should be a continuation of that diagonal.

4. AN IMPROVED METHOD

4.1 Choice of the Interactive Method Used

After the preliminary study of the different methods available, an algorithm was developed and implemented for interactive use on the display terminal. As we have seen in the review, many methods for contour tracing are based on a rectangular grid but their approaches vary greatly. For example, the method by Cottafva and Le Moli requires a preliminary operation to be performed on the entire region under consideration before actually commencing the drawing of contour lines. This produces a set of ordered intersection points which are finally fitted by a smooth curve or simply joined by straight lines. On the other hand, the methods suggested by Mclain and Sutcliffe adopt a more direct approach, where the contour line is traced within each grid rectangle without the need to examine the whole region and the use of extra storage. These latter methods have a more localised approach which gives the terminal user a feel of control over the tracing. Therefore this type of method is more easily adopted for interactive use. Furthermore, it is also possible to provide the user with the ability to control some of the basic parameters such as the choice of the domain boundary, the smoothness of the line drawn, choice of contour level.... etc. Details of these parameter specifications become more apparent in the subsequent sections. The algorithm under discussion here belongs to this contour-following category of methods and represents an interactive refinement of the Mclain/Sutcliffe approach.

4.2 Description of the Algorithm

This algorithm traces the contour line f(x,y)=c in a domain over which there exists a method of calculating the function at any given point in the xy-plane. The region under consideration is divided up into rectangles, each of which is defined by:

$$XMIN \leq X \leq XMAX, \qquad YMIN \leq Y \leq YMAX$$

Each rectangle is further subdivided into small equilateral triangular cells during the tracing process.

Now consider the procedure which traces the portion of a given contour level within a single rectangle of the region.

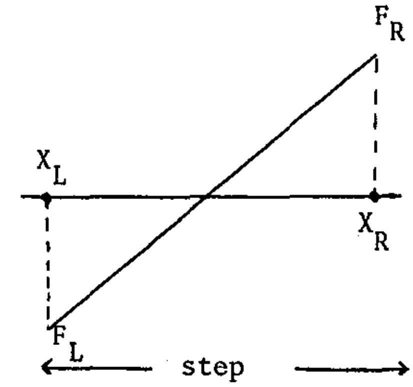
(i) The four sides of the rectangle are examined in some chosen order, searching for an interval (Figure 7.12) of length 'step' which brackets the root of f(x,y)-c=0. The value of step is set equal to 1/2(xstep + ystep), where xstep and ystep can be specified interactively by the user, preferably such that each rectangle is spanned by an integral number of xsteps or ysteps. [Note that other choices of 'step' could be used, e.g.

(a) min(xstep, ystep)

(b) xstep if an x side of the rectangle is being examined
ystep " " y " " " " " " "

The choice used here is more robust than (a) and is basically simpler to implement than (b)]. For each rectangle side, the search is performed by using repeated bisection and making use of the fact that the function will have opposite signs at either side of the root. Thus if a root is detected it will eventually be located to accuracy equal to 'step'. This accuracy is then improved further by (inverse) linear interpolation. Therefore, for the side on which x varies:

$$X_{root} = \frac{X_R . F_L - X_L . F_R}{F_L - F_R} \qquad (7.2)$$



Similarly, for the side on which y-varies:

$$Y_{root} = \frac{Y_R . F_L - Y_L . F_R}{F_L - F_R} \qquad (7.3)$$

Having determined the intersection point, the display beam is moved to this position to start tracing the line within the current rectangle. The interval 'step' is also used as starting baseline for the equilateral triangular cell (Figure 7.13a) which is constructed on this side.

(ii) Once the baseline has been determined the contour line would be followed by threading through the triangular partitioning which is constructed dynamically. The coordinate point of the third vertex (t) is found simply from:-

$$\left.\begin{aligned} X_t &= \frac{x_\ell + x_r}{2} \\ Y_t &= y_\ell + \frac{\sqrt{3}}{2} \text{xstep} \end{aligned}\right\} \qquad (7.4)$$

where 'step' is the length of each side of the cell (Figure 7.13b).

The sign of f(x,y)-c at point 't' would determine uniquely which side of the cell the contour line will cut. This is simply indicated by a sign change along side 'ℓt' or 'rt'. The crossing point is again calculated using linear interpolation, and a straight line segment is drawn from the previous beam position to this new point (p in Figure 7.14a). This side (rt, say) is now used as the baseline for the new cell 'tsr', where the coordinates of s are determined by reflection of ℓ in the side tr (Figure 7.14b), i.e.

$$\left.\begin{aligned} x_s &= x_r + x_t - x_\ell \\ y_s &= y_r + y_t - y_\ell \end{aligned}\right\} \qquad (7.4)$$

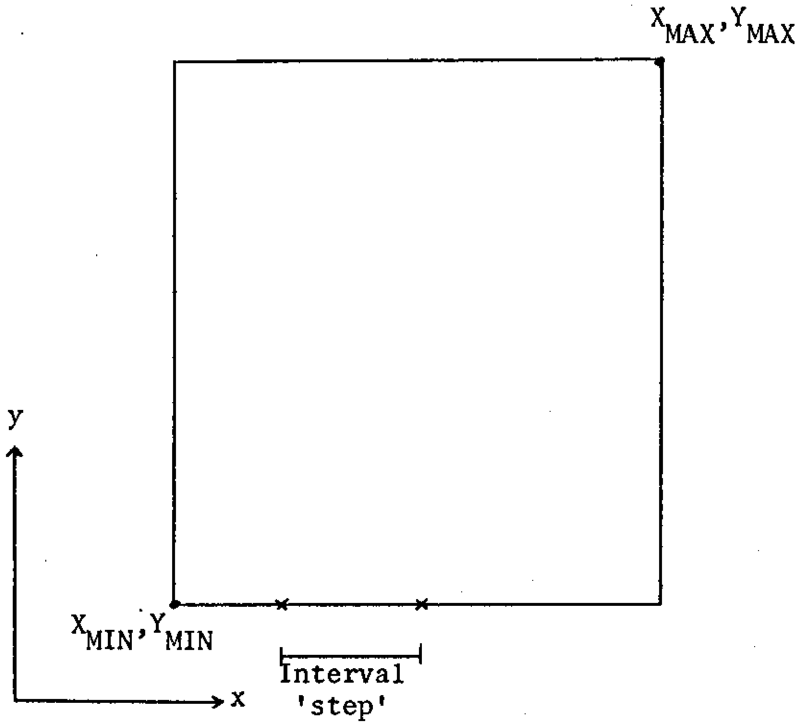


FIGURE 7.12: A Single Rectangle of the Region Showing the Interval Along One Side

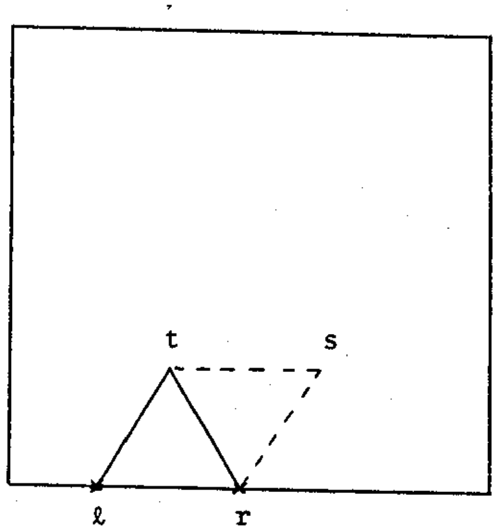


FIGURE 7.13a: Construction of the Triangular Cell on the Baseline

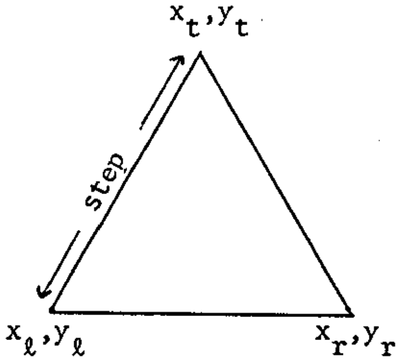


FIGURE 7.13b: A Typical Triangular Cell

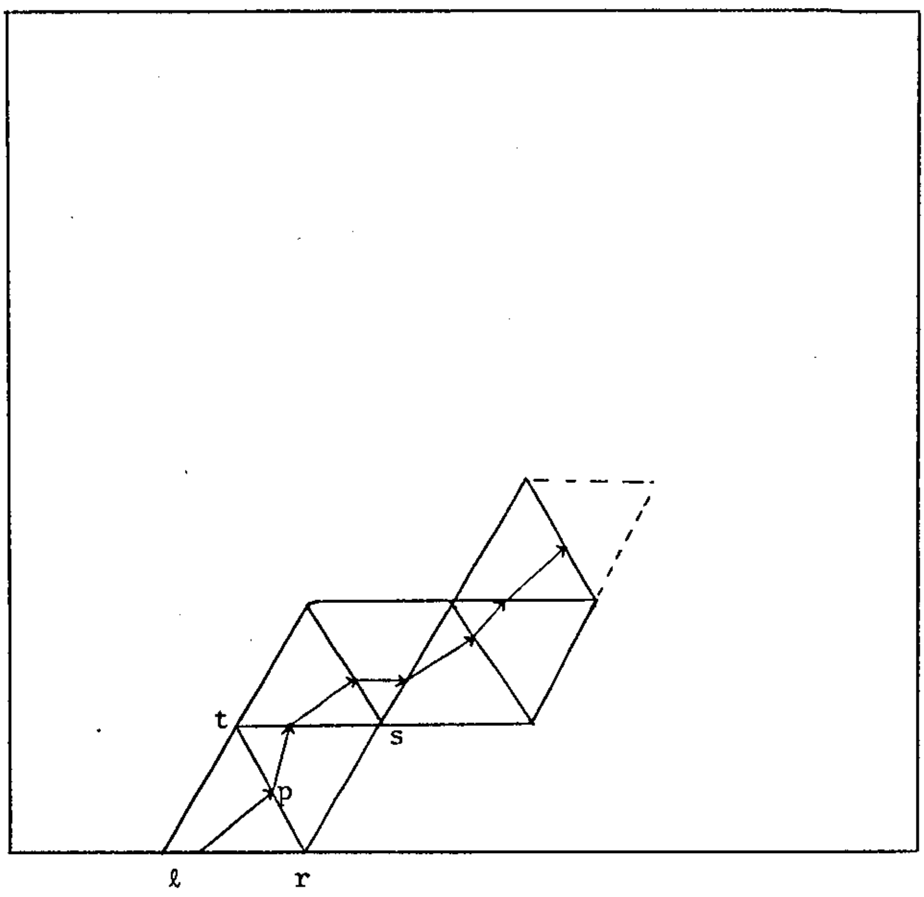


FIGURE 7.14a: Tracing the Contour Through the Triangular Cells

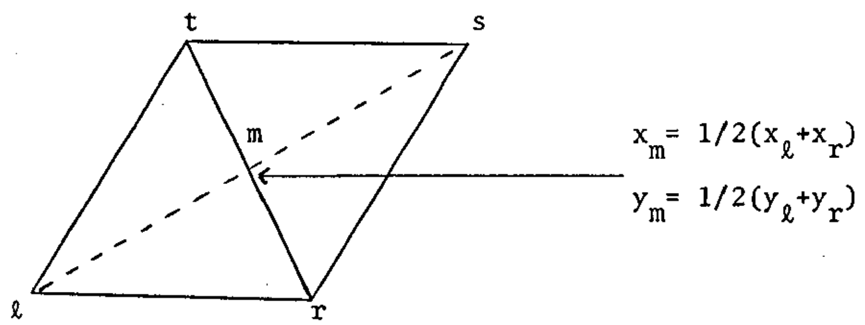


FIGURE 7.14b: Reflection of ℓ in the Side t_t

The triangulation process is repeated until the edge of the rectangle is reached. This triangular subdivision with interpolation permits the lengths of the beam/plotter movements to be relatively large without any severe changes in direction. Thus a relatively smooth continuous contour line is produced within the rectangle.

(iii) When the edge of the rectangle is reached the point of exit is recorded. Each of the other sides of the rectangle are also searched for an interval containing the root which, when found, is checked against the list of recorded exit points for previously drawn contours in order to avoid tracing the line twice. If the root is not in the exit list then the program can enter the drawing loop as described above. Steps (i) and (ii) are repeated for each rectangle in the region. However, within the interactive environment provided, the user is able to select the appropriate rectangle from the mesh by means of the cross-hair cursor (see next section).

4.3 Advantages

This algorithm has achieved a number of improvements over those suggested by Mclain and Sutcliffe. These enhancements relate to the speed, smoothness and treatment of degeneracy and include the following:

(i) Degenerate cells are avoided. In the Sutcliffe method, with a rectangle cell (Figure 7.15) ambiguity arises when all four sides show a sign change of $f(x,y)-c$, and further decisions are required to resolve such ambiguity. On the other hand, with the triangular cells the side to which the contour line would be drawn is uniquely determined once the sign of the third vertex is found.

In general the use of elemental triangles ensures a unique piecewise planar approximation of $f(x,y)$. This is not so when rectangular cells are used.

(ii) There is substantial reduction in the number of function evaluations needed while tracing the line. To check this assertion, a second version of the algorithm was implemented using rectangular cells, and run with several test functions. A comparison of the triangular and rectangular version is given in Table 7.1 which shows in nearly all cases a difference in function evaluations of at least 20%. Note further that the rectangular version also incorporates linear interpolation and would incur slightly fewer function evaluations than the Mclain/Sutcliffe method at the same level of discretization.

This improvement becomes more significant when each function evaluation requires a large amount of computation. The overall speed of the program operation is therefore noticeably improved, particularly when using a small machine and sharing resources with other users.

(iii) The degree of smoothness of contour lines becomes very important when graphic displays are used. The limited display area and screen relation require that the line drawn must be quite smooth in order to avoid overlapping contour lines which are close to each other. When the Sutcliffe algorithm was tried, it produced a curve with a noticeable ripple, which can be reduced only by making the cell size as small as possible, as the algorithm does not compute the intersection point accurately, but assumes that the mid-point is good enough. However since our method employs linear interpolation to compute the point of intersection, the resulting contour line has an acceptable smoothness even when the cell size is not particularly small.

The way in which the curve is traced directly by these methods is considered to be more efficient than using an interpolation scheme such as a higher degree polynomial interpolation for fitting a set of ordered points defining the contour line. Even the use of splines can cause some problem especially at segments of very high curvature, which can occur particularly in the neighbourhood of stationary points. Although splines can alleviate part of the problem, they can still run into difficulties near for example saddle points, if insufficient discretization has been used.

	Function	Contour Level	Region (Min & Max X,Y)	□	△
1.	$(y-x^2)^2+(1-x)^2$	30	(-1,2),(2,10)	164	140
		25	"	162	133
		20	"	160	138
		15	"	160	134
		10	"	156	130
		5	"	126	105
2.	$(y^2+x^2-1)^2+(x+y-1)^2$	0.1302	(0.9,-0.4),(1.2,-0.2)	60	56
		0.1102	"	54	46
		0.0902	"	42	34
		0.0702	"	32	24
		0.0502	"	20	15
3.	$\exp-((x-5)^2+(y-5)^2)$	0.1	(5,3.5),(6.5,4.5)	52	39
4.	$\exp-((x+y-11)^2+(x-y)^2/10)$	0.2	(6.5,4),(8,5)	36	27
5.	$8(4y+7)^3-9(4y+7)^2(2x+3)+3(4y+7)(2x+3)$	500	(0,-0.5),(6,0.5)	78	62

TABLE 7.1: Comparison in the Number of Function Evaluations Using the Rectangular & Triangular Subdivision Schemes

(iv) The Sutcliffe method assumes that the region under consideration should be divided into rectangles which are of a size such that the contour line only crosses any edge once. In practice, a situation could arise where for a chosen subdivision of the region, a contour line does cross an edge twice. This could happen for example in the neighbourhood of a saddle point (Figure 7.16). Further discretization would incur considerable overheads and greatly increase the computing cost. However, the present algorithm takes care of such cases without resorting to additional root-finding procedures. (The usual sign test at the nodes of the rectangle would not yield any roots in this example). For example, as shown in Figure 7.16, if the sides of rectangle A(I,J) are intersected more than once, the algorithm keeps a record of the sides and traces the line inside this rectangle from each exit point.

5. THE INTERACTIVE DISPLAY PROGRAM

The algorithm developed here is made available to the user via the graphic display terminal (Tektronix 4010) in an interactive mode. Communication with the algorithm is maintained through a set of commands presented as menu options which comprise the user interface. Its aim is to provide the user with a direct and simple means of controlling the execution path of the algorithm in order to satisfy certain requirements. This kind of capability permits the user to take different courses of action while operating the program until the required result is obtained.

The final objective is to enable the user to draw a set of contour lines $f(x,y)=c$ of a given function $z=f(x,y)$ over a region, with interactive control of the following:

(1) The boundary of the region within which the set of contour lines are required.

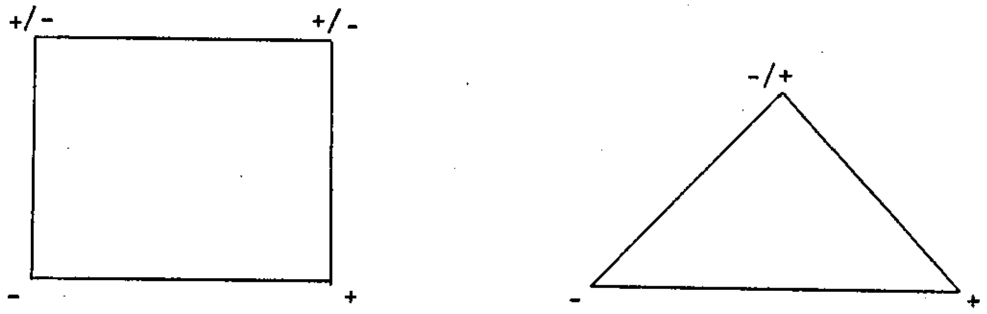


FIGURE 7.15: Sign Changes at the Vertices

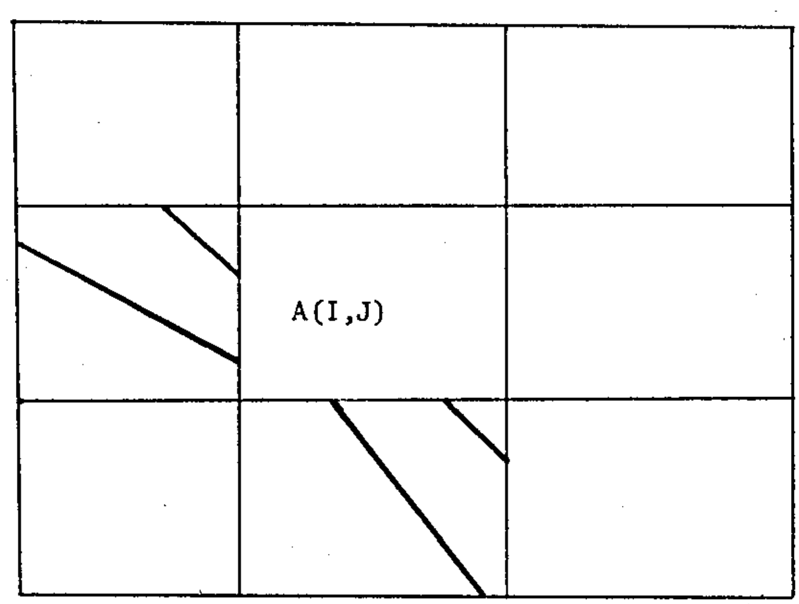


FIGURE 7.16: Contour Line Crosses at Least One Rectangle Side Twice

(2) The contour levels.

(3) The size of the rectangular grid and the triangular cells which subdivide each rectangle.

(4) The threading of the contour line from one rectangle to another, using the cross-hair cursor. The user can alternatively request this process to be completed automatically by the program

(5) The zoom window which defines the part of the display requiring magnification.

We now illustrate how the user can, with the aid of the keyboard and cross-hair cursor, control the algorithm to determine the boundary of the region for a given function. The program permits the user to start with two points which define the rectangular region. He would then be able to examine the region for any trace of the contour lines. If no trace is found, he could try a different set of boundaries and re-examine the region. The user could then iterate the above process until the boundary is satisfactorily defined for contouring.

The graphics terminal provides two modes of communication between the user and the tracing algorithm. The first is the keyboard through which the user can enter boundary coordinates, grid information and the various contour heights. The second means of communication is the cross-hair cursor, which is used to select options from the menu and to identify appropriate rectangles for the algorithm. When an option is picked up, the program would invoke the appropriate routine which may prompt the user for keyboard entry and subsequent injection of the new parameter values into the algorithm.

Essentially, the interactive program consists of two main display modes, each having a particular role during the user session at the terminal. The two displays are:

(i) Contour parameter entry:

The ICT program is activated by typing the program name 'CONTOUR' at the terminal. As soon as the program runs, the screen would be cleared and the first display (Figure 7.17) appears. This contains short introductory remarks and two separate menus, each containing a set of options.

(a) Program control menu.

This has three options whose functions are mainly concerned with controlling the display image and the logic flow in the program. These options are similar to those mentioned in the IDF package (Chapter 6).

→ +NEXT
+RESTART
+EXIT

(b) Contour parameter menu.

When a parameter option is selected, the user would be prompted by a message (appearing at the bottom of the screen), and the program waits for the user to make a keyboard entry in response to this message. If the options marked with an asterisk (*) are not selected, the program will set their values by default.

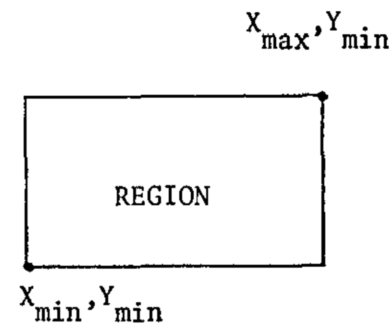
(1) The height of the contour line initially to be traced can be specified at this display. However, this entry could be postponed until the next display, so selection of this option is not compulsory. The function displayed in these photographs (Figures 7.18-7.19) is → +CON.LEVEL

$$z = (y-x^2)^2 + (1-x)^2$$

(Rosenbrock's 'Banana' function).

(2) This is a compulsory item which the user must select in order to define the boundary of the region. The → +REGION

coordinates (X_{min}, Y_{min} and X_{max}, Y_{max}) of the rectangular region are then keyed in response to a program prompt on the screen.



This has the effect of moving a window over the region within which the function can be evaluated and its contours displayed. The viewport size is fixed at 750×750 screen rasters.

(3) The discretization of the region into a rectangular grid is controlled through this option by specifying the size of each side of the rectangle (X-WIDTH and Y-WIDTH). The program allows an upper limit of 30×30 rectangles into which the region could be subdivided. In most cases this fine discretization of the region is unnecessary, since the algorithm further discretizes each rectangle into smaller triangles whose minimum size is constrained by screen resolution. However, the program has a built-in default action if the user bypasses the above option. This default subdivides the region into 15×15 squares of side 50 rasters, representing an adequate discretization. → $\boxed{\text{+G.WIDTH}}$

(4) This option would effectively control the size of the triangle and hence the smoothness of the contour line. The size of X or Y-step must be chosen such that → $\boxed{\text{+X-Y STEP}}$

STEP = WIDTH/<integer>

The default setting in this case is

STEP = WIDTH/10

(5) The grid generated would be drawn on the next display → $\boxed{\text{+DISP.GRD}}$ (Figure 7.18), if the user has picked this option. The mesh would provide the user with some visual aid when interactively tracing the contour. In addition, it would be useful for reading the coordinates of the grid nodes when trying, for example, to re-define the region boundary.

(ii) Contour lines display:

This is the next main display generated by the program and → represents the final outcome of the algorithm. Two menus appear on the screen, containing some options which appeared (and were described) in the previous display. The bottom menu effectively allows the user to vary the contour level in order to display the set of contours associated with the function in the region under consideration. Also, if necessary, the user can vary the degree of smoothness. (Figure 7.20b).

+CON.LEVEL
+X-STEP
+Y-STEP

The contour lines could be displayed in two different ways:

(1) Interactively. → $\boxed{\text{+CURSOR}}$

This option is usually used in conjunction with the grid net drawn as in Figure 7.18 where the tracing of the contours is carried out interactively by means of the cursor. Here, the cursor is used as a pointing device for identifying the appropriate rectangle within which the contour (if it exists) is to be drawn. This is accomplished by positioning the cross-hair cursor intersection point anywhere inside the rectangle and by pressing any

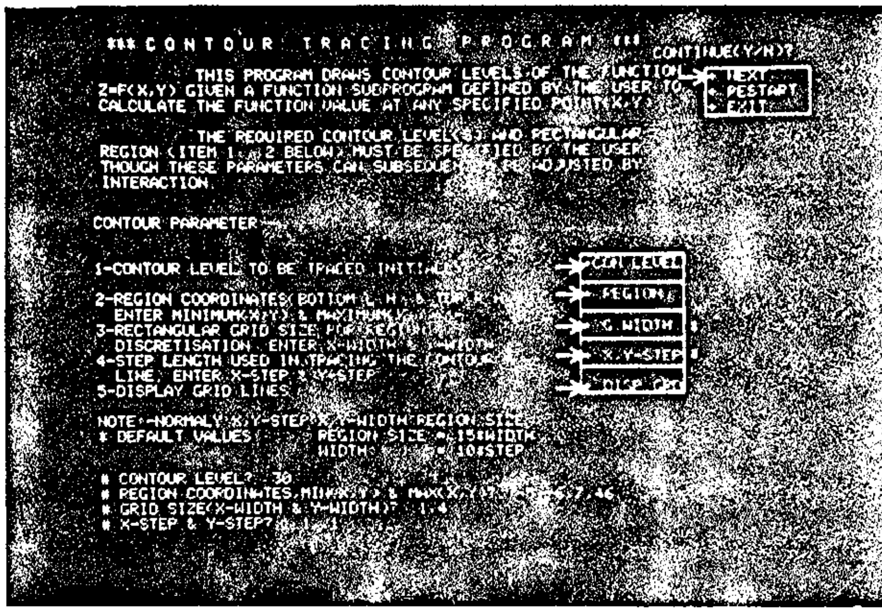


FIGURE 7.17: Contour Parameter Entry Display

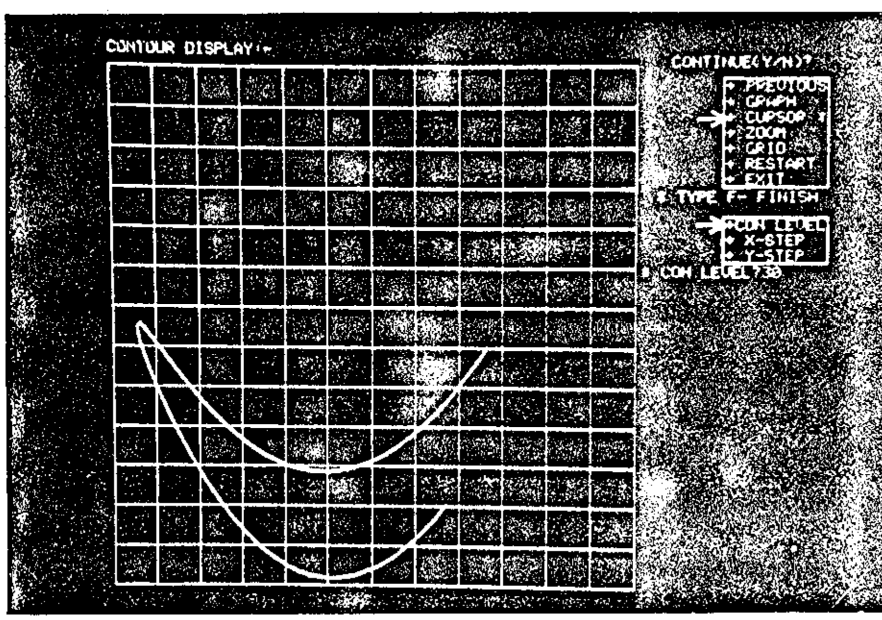


FIGURE 7.18: Tracing the Contour Line by Means of the Cross-Hair Cursor

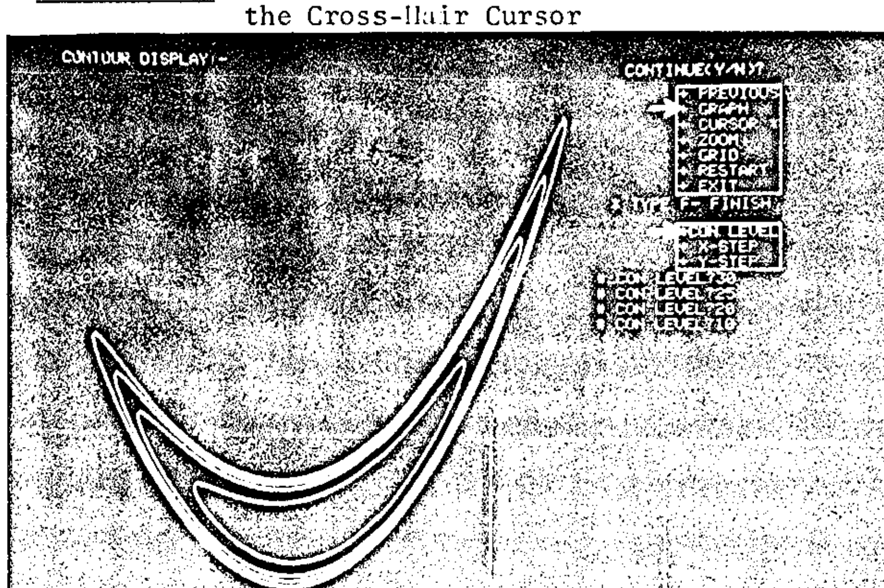


FIGURE 7.19: Contour Display of Rosenbrock's Function

$$f(x)=(y-x^2)^2+(1-x)^2$$

key, say the space bar. If, however, the contour does not pass through this rectangle, the cursor control will be immediately returned back to the user and he could try again. Once a rectangle is found to contain a portion of the contour line, the extrapolation of the line becomes easy and natural, as the user can recognise visually the direction of the contour. This scheme provides the user with the capability for searching through a given region until a trace of the line is found, and if necessary adjusting the boundary. It would also give the user the flexibility to stop tracing a particular contour level and proceed to the next one.

(2) Automatically.

The algorithm would thread the contour line through the rectangles of the region, when this option is picked up. The two examples shown in Figure 7.19 and Figure 7.20a are produced using this option. The contour lines represent the following two explicit functions: → $\boxed{\text{+GRAPH}}$

$$f(x,y) = (y-x^2)^2 + (1-x)^2 \quad \text{--------- Fig.7.19}$$

$$f(x,y) = (y^2+x^2-1)^2 + (y+x-1)^2 \quad \text{------- Fig.7.20}$$

As can be seen, the present algorithm produces visually smooth curves, and (in Figure 7.19) copes with sharp corners. It is also clear from the example in Figure 7.21 that this algorithm has successfully overcome the situation where the contour crosses the edge of the rectangle twice, without the need to discretize the region further. This situation has risen in Figure 7.21 near the saddle point at the contour level 0.1102. The use of this option is recommended when the user wishes to produce the final display of the set of contours, perhaps followed by photographs or other forms of hardcopy. Finally, an additional menu option '+ZOOM' is provided,

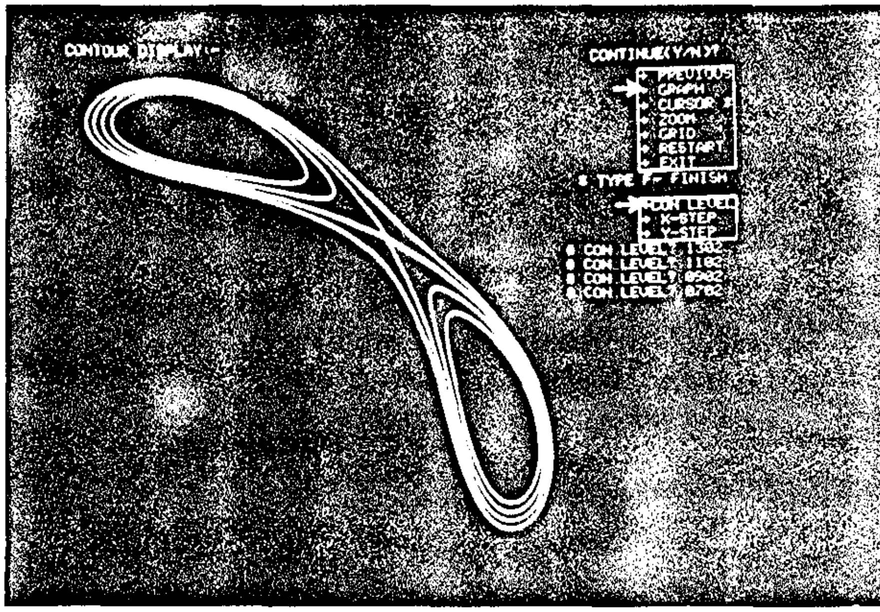


FIGURE 7.20a: Contour Display of the Function $f(x)=(y^2+x^2-1)^2+(y+x+1)^2$

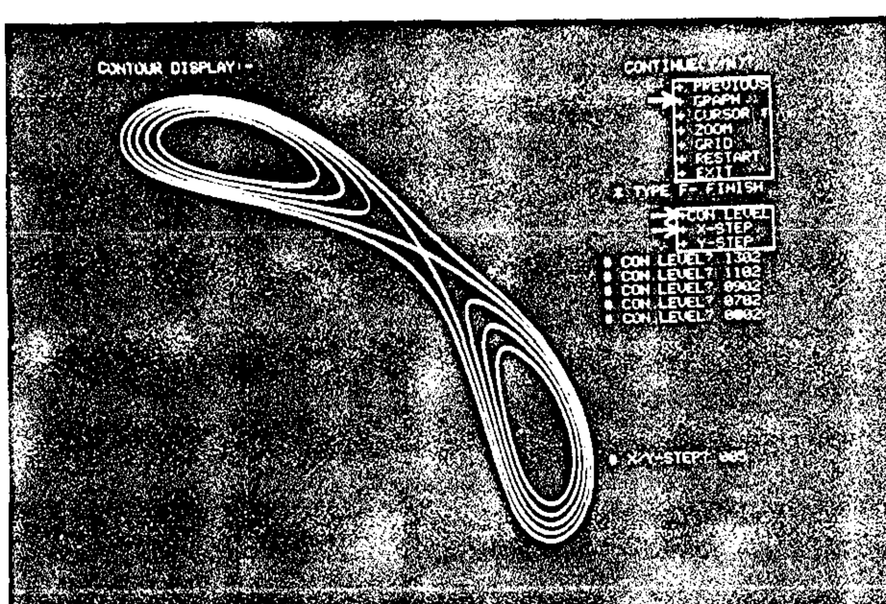


FIGURE 7.20b: Same as Fig.7.20a with Additional Contour Level and Variation of x/y Step

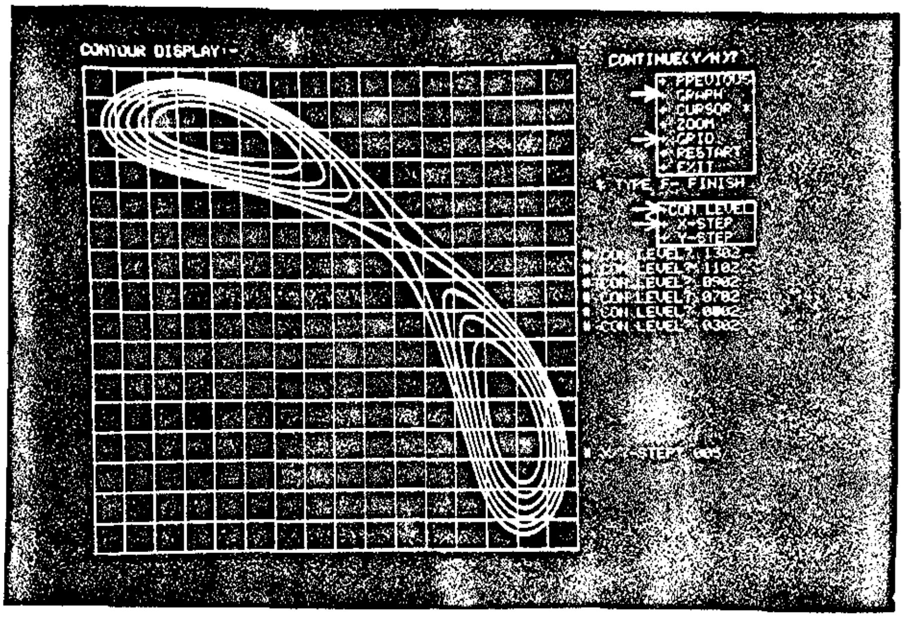


FIGURE 7.21: Grid Lines Superimposed Showing the Contour Level .1102 (near the saddle point) Crossing the Rectangular Edge Twice

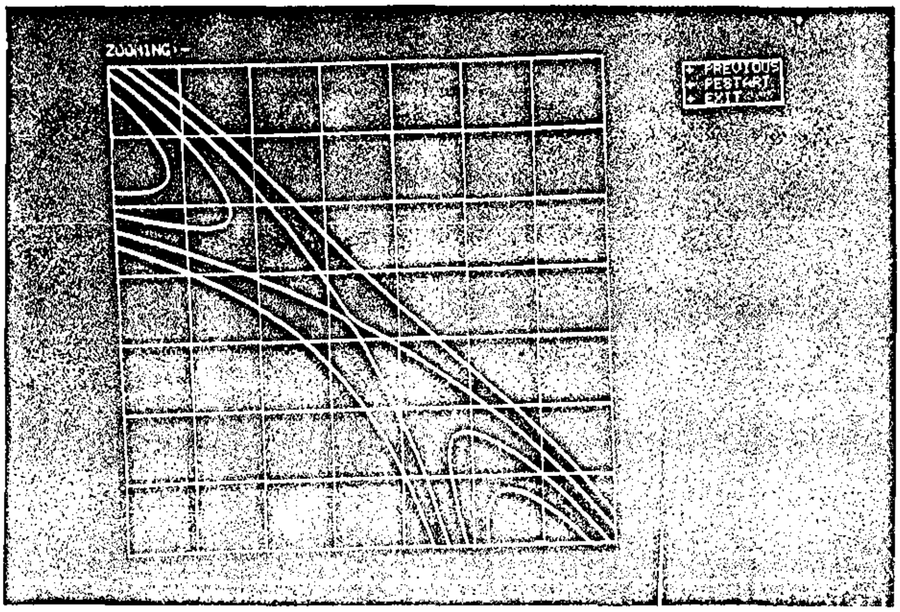


FIGURE 7.22: The Zoom Display

allowing the user to blow up a selected portion of the region in order to gain a closer view of that portion (Figure 7.22). The zoomed area is determined by pointing (with the cursor) at two rectangles situated at opposite corners of the window.

6. PROGRAM IMPLEMENTATION (Interactive Contour Tracing)

The ICT program was implemented on the PDP 11/40 minicomputer operating under the UNIX system. The program is entirely written in Fortran IV, uses the graphic software package LIGHT and consists of two sets of subroutines controlled by a main segment.

(i) The contouring algorithm subroutines (Appendix 3.1)

(ii) The user interface subroutines (Appendix 3.2)

The first set consists of four main subroutines together with three subsidary ones, forming the entire code of the main algorithm. The general flowcharts are shown in Figure 7.23 giving the logic flow of the algorithm. The second set consists mainly of two interactive display routines which organise the menus and handle the user interaction. These routines in turn call nine other subsidary subroutines used for setting up the prompting messages, interpreting user input and computing the appropriate scaling factors.

The program design is highly modular, due firstly to the division of the two completely separate tasks the program has to perform, and secondly, the subroutine nature which enables future modification to be incorporated easily.

In order to create the executable version of the program the user is required to include a Fortran function subroutine which returns the value of the function at any given point (X,Y). For example, if the function under investigation has an explicit form such as the one given in the examples above, the function subroutine would be

```
FUNCTION F(X,Y)
   F=(X*X+Y*Y-1)**2+(X+Y-1)**2
RETURN
END
```

However, if the required function has values specified at some arbitrary data points, then the expression in the above example will be replaced by a body of code which would evaluate the function at a given point. Note that whatever form the body of the function would have, the function name used is F(X,Y), this being the actual name called by the ICT program.

Having, specified the function evaluation routine, the user is provided with the 'shell' file which he could run to generate the executable module of the program 'CONTOUR'. The following UNIX keyboard commands are used for this purpose

```
% SH JCLTRI              ; Create the contour program
% CONTOUR                ; Run ICT program
```

Another version (Appendix 3.3) of the algorithm was implemented for comparison purposes, and uses rectangular cell subdivisions while tracing the contour line. For this the UNIX commands are

```
% SH JCLREC
% CONTOUR
```

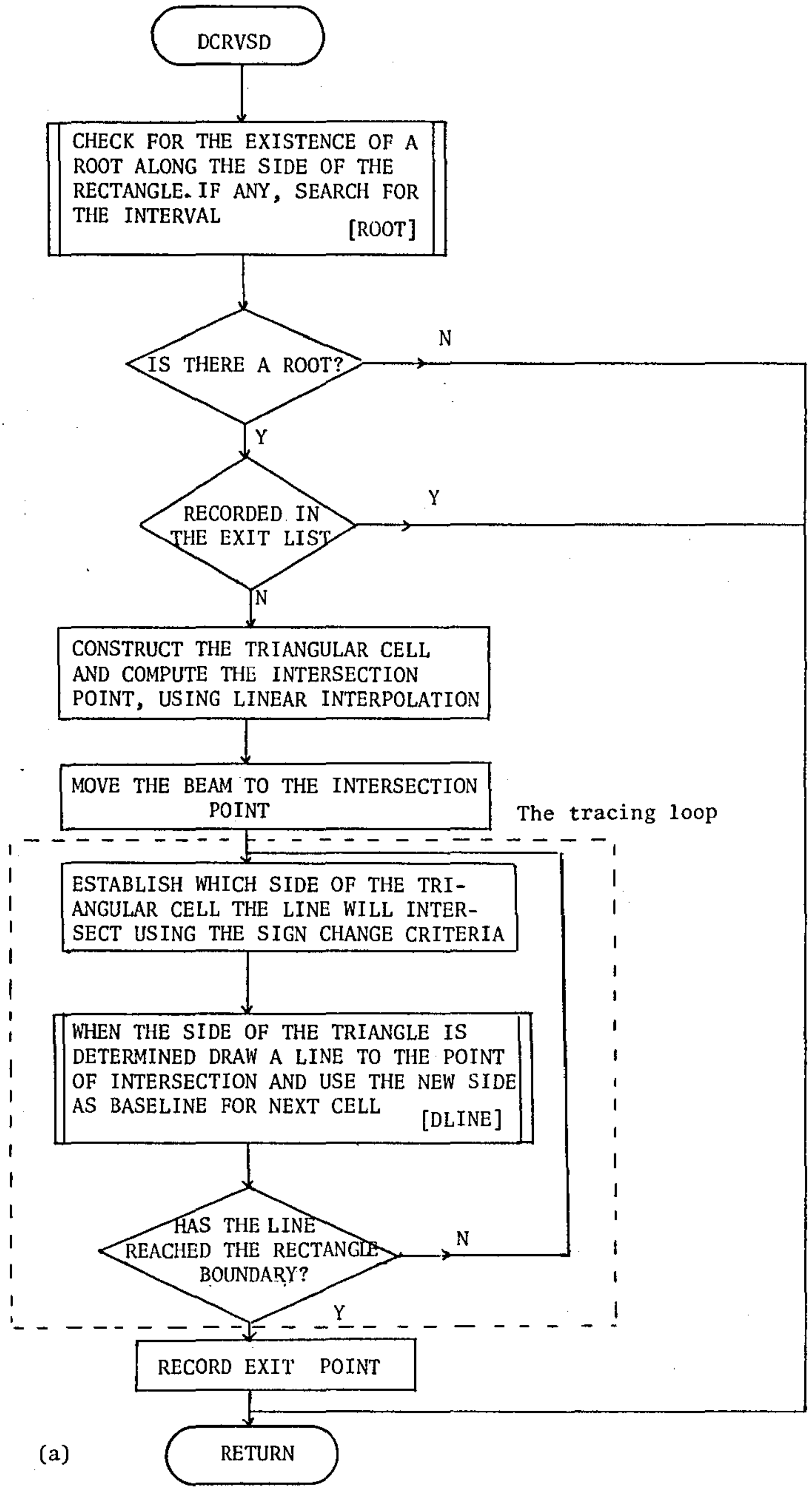


FIGURE 7.23: General Flowchart of the Contour Tracing Algorithm

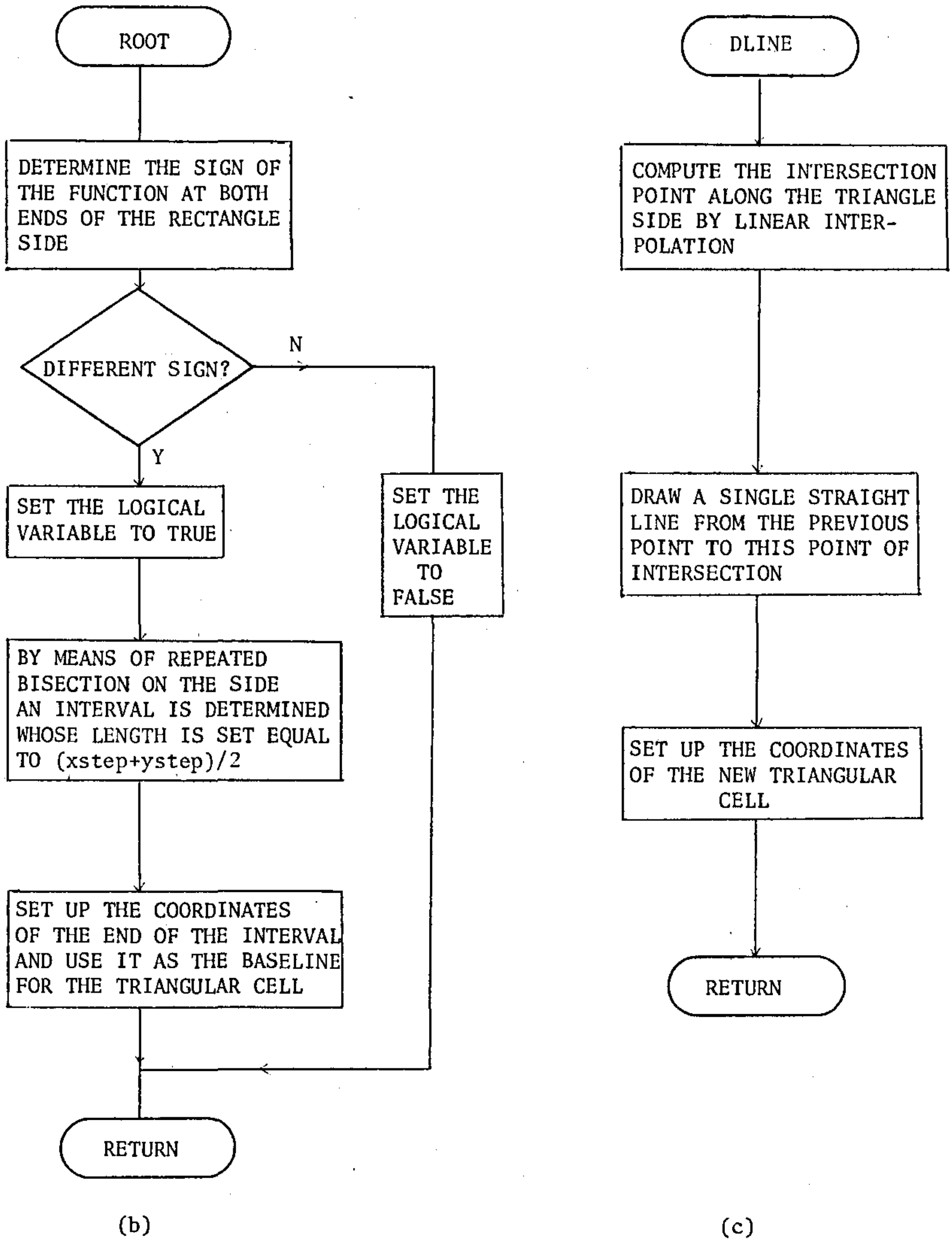


FIGURE 7.23 (continued)

CHAPTER 8

TRIANGULAR MESH GENERATION - TMG

1. INTRODUCTION

A generalised finite element algorithm was proposed [46] for the solution of elliptic boundary-value problems using non-uniform triangular mesh systems.

The above work is mainly concerned with triangulations in the plane with particular reference to the numerical solution of the Poisson equation, and the subsequent graphic display of the generated mesh and its associated equipotential lines.

The program developed by Cardew [46] is an adaptation of Winslow's [47] for generating non-uniform triangulation with selective zoning in different areas. The versatility with which the problem boundaries can be represented in the mesh is one important feature; further, there is the unique facility for grading the mesh to suit special areas in the plane of solution.

A wide variety of problems in mathematical physics can be formulated as the classical problem of solving the Poisson or Laplace equation. For example, the distribution of electric flux in a general, anisotropic medium is modelled by Poisson's equation:

$$\mathrm{Div}(\varepsilon\nabla\phi) = -\rho \tag{8.1}$$

where ε is the permittivity, and ρ is the electric field charge density (source density). The boundary conditions usually have two forms:

$$\left.\begin{array}{lll} & \phi = \phi_i(x,y) & \text{'Dirichlet conditions'} \\ \text{and} & \underline{n}.(\varepsilon\nabla\phi) = 0 & \text{'Neumann conditions'} \end{array}\right\} \tag{8.2}$$

where $\underline{n}$ is the normal to the boundary at (x,y). Another example is Laplace's equation for the electric field in conducting media:

$$\mathrm{Div}(\sigma\nabla\phi) = 0 \tag{8.3}$$

where σ is the conductivity.

If ε is constant, equation (8.1) may be rewritten in terms of operator ∇^2 as:

$$\nabla^2\phi = \rho' \tag{8.4}$$

The usual classical approach to solving the boundary-value problem by the method of finite-differences is to discretize the region into a uniform rectangular grid (Figure 8.1).

Using Taylor series for obtaining the approximations to the operator ∇^2 in the above, the equation $\nabla^2 u=\rho$ may be replaced by the five point difference equation

$$[u_1+\alpha^2 u_2+u_3+\alpha^2 u_4-2(\alpha^2+1)u_0] + O(h^2) = \rho\alpha^2 h^2 \tag{8.5}$$

where $h_x=h$ and $h_y=\alpha h$. More accurate formulae can be found through the use of more complex molecules; for example, a nine-point approximation on a square net involves an $O(h^4)$ local truncation error. However, using uniform equilateral triangulation with a seven-point molecule, the error term is also $O(h^4)$. With orthogonal X and Y axes the 'equilateral net' is defined by two intersecting sets of lines (Figure 8.2), the L and K lines, corresponding to the rows and columns of a square net. The logical origin of the triangulation is situated at the bottom-left corner and the convention is that there should always exist two neighbouring triangles at this point. If the spacing between adjacent points on the same row is h, then for the equilateral net the spacing between adjacent rows is αh where $\alpha=\sqrt{3}/2$. For $\alpha\neq\sqrt{3}/2$ the net will still be topologically equivalent to the equilateral net. Using Taylor expansions about O (Figure 8.3), the seven-point difference approximation to $\nabla^2 u=\rho$ on a equilateral grid may be written as:

$$-\sum_{i=1}^{6} u_i+6u_0+3\,\frac{h^2}{2}\rho = \frac{9h^4}{64}[\nabla^2\rho] + O(h^4) \tag{8.6}$$

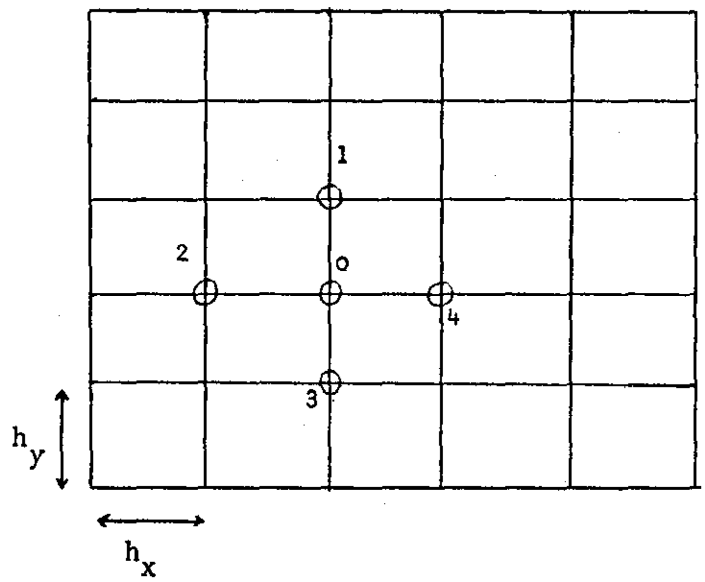
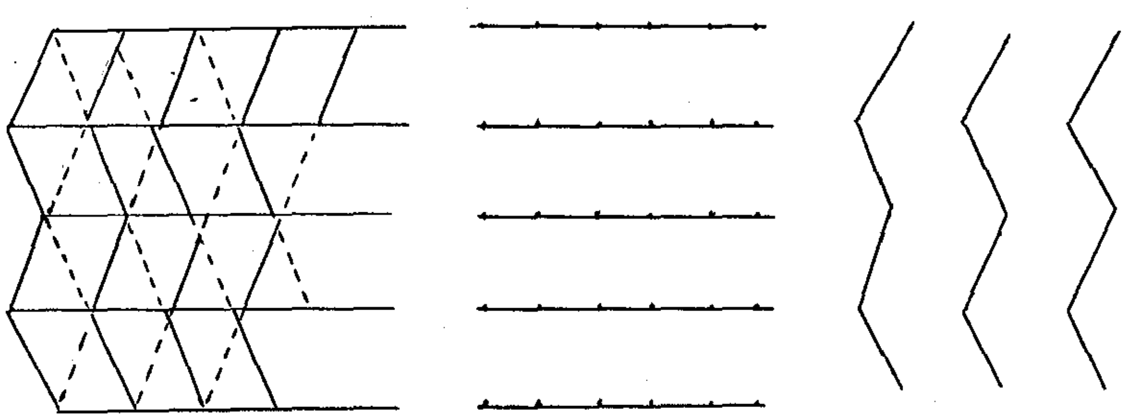


FIGURE 8.1: Five-Point Difference Molecule



(a) Network (b) Rows (L-lines) (c) Column (K-lines)

FIGURE 8.2: Triangular Network and Its Basic Elements (Row and Column)

A uniform mesh is inefficient when the problem requires a more accurate solution in some regions than in others, since the smaller spacing required leads to wasteful computation over the less important regions. The use of an irregular mesh is therefore desirable. Moreover, with this flexibility it would be possible to arrange that mesh points lie precisely on the boundary.

In the following sections we state the basic assumptions, difference equations and methods of solution used in the development of the program. Finally, a full account will be given with supporting examples of how to run the program and the subsequent interactive display program for observing the result.

2. RITZ FINITE-DIFFERENCE EQUATIONS

The basic assumptions of the finite-difference method made here are:

(i) the boundaries and interfaces of the region R are approximated by straight-line segments.

(ii) the region is triangulated

(iii) the values of ϕ are defined at triangle vertices, and ϕ is assumed to vary linearly over each triangle.

(iv) ε and ρ are assumed to be constant over each triangle.

The type of triangulation used here is topologically regular; i.e. it is topologically equivalent to an equilateral triangle array in which six triangles meet at every interior mesh point. Since any polygonal region can be triangulated, the method can be applied to regions of any shape and will produce a mesh in which boundaries and interfaces lie entirely on mesh lines. This causes a considerable simplification of boundary conditions.

Consider the numerical solution [48] of a generalised form of Poisson's equation within a two-dimensional domain R with given boundary condition. In particular, consider the second order, linear elliptic partial differential equation expressed as

$$-\frac{\partial}{\partial x}\left(a\,\frac{\partial\phi}{\partial x}\right) - \frac{\partial}{\partial y}\left(c\,\frac{\partial\phi}{\partial y}\right) + k\phi = f \tag{8.7}$$

where $(x,y)\varepsilon R$, $a(x,y)$, $c(x,y)>0$.

The boundary condition is

$$\alpha\phi + \frac{\partial\phi}{\partial n} = \beta \tag{8.8}$$

where n is the direction of the normal derivative and α,β are piecewise continuous functions along the exterior closed boundary of R. The solution ϕ,k and f are assumed to be continuous within R.

A two-dimensional finite element may be defined as a polygonal subdomain of the domain R of a boundary-value problem. The potential within this subdomain is approximated by a polynomial interpolated through the n vertices of the subdomain. The order of this interpolated polynomial is determined by n, e.g. a 'linear element' is a triangular subdomain in which the potential is approximated by a linear polynomial. The approximate solution u of the partial differential equation is represented by a polynomial function (usually linear) over each element with matching conditions on the inter-element boundaries. Each polynomial is defined by a number of coefficients or by the function value and its derivatives at certain points. Then by use of a classical result from the calculus of variations, we know that the solution of (8.7) is given by the function which minimizes the integral

$$I(v) = \frac{1}{2}\int\int_R \left\{a\left(\frac{\partial v}{\partial x}\right)^2 + c\left(\frac{\partial v}{\partial y}\right)^2 + kv^2 - 2fv\right\}dxdy + \int_\Gamma \left(\beta v+\frac{1}{2}\alpha v^2\right)\,ds \tag{8.9}$$

over all functions with at least piecewise continuous first derivatives which satisfy the same boundary conditions as ϕ. If we substitute u for v in the integral (8.9), I(u) becomes the sum of the integrals of the polynomials over the finite elements within the domain R.

Given a triangulation of the region R based on the rows and columns shown in Figure 8.2, over which u is required at each triangle vertex, we can derive a set of difference equations from (8.9) by using the Ritz varational method. Basically, we express u as the continuous piecewise linear sum

$$u = \sum_i u_i \alpha_i (x,y) \tag{8.10}$$

where u_i is the value of u at the mesh point (x_i, y_i) and $\alpha_i(x,y)$ is a pyramid function with the properties:

(i) $\alpha_i(x_i, y_i) = 1$

(ii) $\alpha_i(x,y) = 0$ for any point (x,y) lying on, or beyond the hexagonal perimeter 123456, (Figure 8.3).

(iii) α_i varies linearly in each adjacent triangle.

We minimize I(u) by setting $\frac{\partial I}{\partial u} = 0$ at each mesh point, thus obtaining a system of linear equations in the parameter u_i of the form:

$$\sum_{q=1}^{6} \omega_{pq}(u_q - u_p) + b_p = 0 \tag{8.11}$$

for all points $P \varepsilon R$ where b_p is a combination of boundary and source terms and can be expressed as,

$$b_p = \sum_{T_q=1}^{6} \rho_{T_q} A_q \tag{8.12}$$

A_q is the area of the triangle T_q (Figure 8.4) and ω_{pq} is called the coupling coefficient between the points P and Q and depends on the geometric and material properties of the two triangles (T_{q+1}, T_{q-1}) having PQ as a common side. It could be shown that

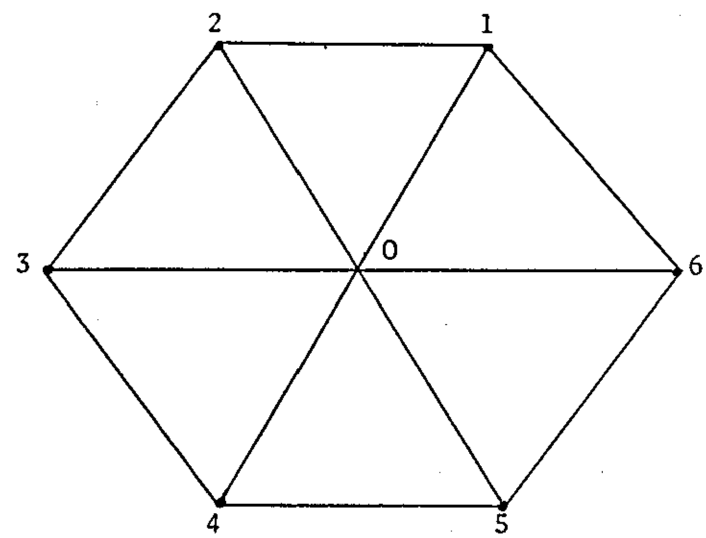


FIGURE 8.3: The Computational Molecule

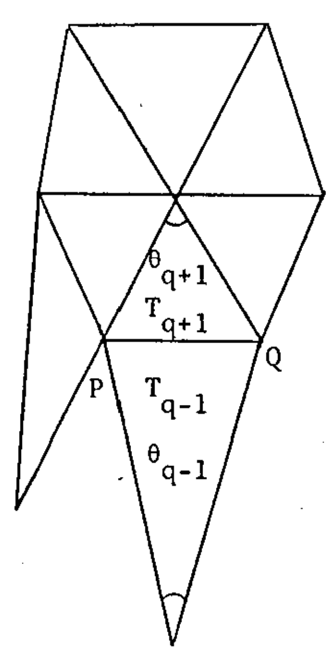


FIGURE 8.4: Couplingof Triangular Elements

$$\omega_{pq} = \frac{1}{2}(\varepsilon_{T_{q+1}} \cos\theta_{q+1} + \varepsilon_{T_{q-1}} \cos\theta_{q-1}) \tag{8.13}$$

Equation (8.11) is the Ritz finite-difference equation using a general seven-point molecule.

If the triangulation is uniform and equilateral and if the source term ρ is piecewise constant over the region of solution, equation (8.11) reduces to the seven-point regular hexagonal equation (8.6) derived from the Taylor series.

Complete generality is present in the Ritz approximation (8.11) in the following respects:

(i) the mesh may vary in any manner, though it must remain topologically regular.

(ii) the source function ρ can vary in a piecewise manner between subregions of the problem.

(iii) the coefficient functions may also vary in a piecewise manner between regions.

If the mesh is approximately equilateral in certain zones, then the discretization error there will be $O(h^4)$. Clearly, there is more computational effort involved with using (8.11) than with the five-point or seven-point equation resulting from Taylor series. This would appear to be the disadvantage of using a non-uniform net. However, the prime motive in employing (8.11) resides in the simplification introduced in the treatment of:

(a) Composite material media

(b) Complex boundaries

and (c) Singular field points, where the mesh spacing may be reduced around such points.

3. ITERATIVE SOLUTION OF THE RITZ EQUATION

The linear system of equations generated by repeatedly applying equation (8.11) to all mesh points can be expressed as:

$$\underline{Au} = \underline{b} \tag{8.14}$$

Since each point of the network is coupled only to its neighbouring points $\underline{A}$ is a large order, sparse, positive definite, symmetric matrix containing the coupling coefficients ω_{pq} derived from (8.13) as elements banded around the main diagonal. In general the more complex the finite-difference molecule is, the wider the bandwidth becomes. The vector $\underline{u}$ is the required solution at the mesh points and $\underline{b}$ is a vector whose elements are combinations of boundary and source terms.

Methods for inverting the system of linear equations may broadly be classified as either Direct or Indirect. In the direct method a finite number of computational steps are required for evaluating the solution, whilst in the indirect method an initial 'guess' is made for the solution which is subsequently improved upon in successive iterations. These cycles are terminated when the changes in u from one cycle to the next are negligibly small (convergence).

The iterative methods of solution are almost invariably much simpler to program and often preferred to direct methods because they can cope with the sparsity of the matrix more effectively. Only the non-zero coefficients are used in the numerical algorithm involved and thus the computer storage usage is minimal. Further, the problem of rounding error growth is avoided. The iterative scheme used in this program involves splitting the matrix A in the following manner

$$\underline{A} = \underline{D} - \underline{L} - \underline{U} \tag{8.15}$$

where $\underline{D}$ is a diagonal matrix and $\underline{L}$ and $\underline{U}$ are strictly lower and upper triangular matrices respectively. Thus, rewriting the linear system of equation (8.14) as

$$(\underline{D}-\underline{L})\underline{u} = \underline{U}\,\underline{u} + \underline{b} \tag{8.16}$$

and introducing the iteration counter n, the Gauss-Seidel iterative scheme can be defined:

$$\underline{u}^{(n+1)} = \underline{D}^{-1}(\underline{L}\,\underline{u}^{(n+1)} + \underline{U}\,\underline{u}^{(n)} + \underline{b}) \tag{8.17}$$

The convergence of this method can be improved by introducing the parameter ω into (8.17), which would result in the well known successive over-relaxation (S.O.R.) method as

$$\underline{u}^{(n+1)} = \underline{u}^{(n)} + \omega\underline{D}^{-1}(\underline{L}\,\underline{u}^{(n+1)} + \underline{U}\,\underline{u}^{(n)} - \underline{D}\,\underline{u}^{(n)} + \underline{b}) \tag{8.18}$$

or alternatively,

$$\underline{u}^{(n+1)} = \underline{L}_{\omega}\underline{u}^{(n)} + \underline{b}_{\omega} \tag{8.19}$$

where ω is the over-relaxation parameter chosen to accelerate the convergence and

$$\underline{L}_{\omega} = (\underline{D}-\omega\underline{L})^{-1}\,\{\omega\underline{U} + (1-\omega)\underline{D}\}$$

The matrix is usually generated from a simple computational molecule (Figure 8.3) rather than stored directly in the computer memory. Thus, only the approximations $\underline{u}^{(n)}$ are stored (requiring one vector of storage) and a mechanism (program) simulates the molecule's traverse of the network in some order during each iteration. For regular points of the net, the coupling coefficients are just stored once while for irregular or boundary points, each point is tagged and the associated coupling coefficients stored as lists.

By choosing a suitable local ordering such as in Figure 8.3, and returning to equation (8.11) the corresponding S.O.R. formula is

$$u_p^{(n+1)} = u_p^{(n)} + \omega\{(\sum_{q=1}^{6}\omega_{pq}u_q^{(n,n+1)} + b_p)/\sum_{q=1}^{6}\omega_{pq} - u_p^{(n)}\} \tag{8.20}$$

In each iteration cycle, the mesh points are swept in sequence with the $(n+1)^{th}$ iterate replacing the n^{th} iterative component at u_p as soon as it is calculated. For symmetric, positive definite matrices, the method is known to converge. The main point of interest is the choice of the over-relaxation factor ω which minimises the spectral radius $\rho(L_\omega)$ and thus maximises the convergence rate. The optimum value of ω is

$$\omega_{opt} = \frac{2}{1+\sqrt{(1-\lambda^2)^{\frac{1}{2}}}} \tag{8.21}$$

where λ is the spectral radius $\rho(\underline{D}^{-1}(\underline{L}+\underline{U}))$ of the related Jacobi operator. For a given ω (less than ω_{opt}), we can express λ as

$$\lambda = \frac{\omega+\mu(L_\omega)-1}{\omega\sqrt{\mu(L_\omega)}} \tag{8.22}$$

where $\mu(L_\omega) = \dfrac{||\delta^{(n+1)}||}{||\delta^{(n)}||} = \left[\dfrac{\sum_{i=1}^{n}(\delta_i^{(n+1)})^2}{\sum(\delta_i^{(n)})^2}\right]^{\frac{1}{2}}$

The error norm $||\delta||$ can be evaluated by forming the sum of the squares of the residuals (errors) at all mesh points on the completion of each iteration.

An under-relaxation parameter β is introduced in the actual formulae used in the program. This would prevent ω_{opt} from changing too rapidly from one iteration to the next [47]. This is desirable since large changes in the value of ω_{opt} may seriously perturb the spectral radius $\mu(L_\omega)$. The final formula used for the automatic estimation of the ω_{opt} is

$$\omega_{opt} = \beta\left(\frac{2}{1+\sqrt{1-\lambda^2}}\right) + (1-\beta)\omega_{opt'} - \beta\rho_0 \tag{8.23}$$

where $\omega_{opt'}$ is the previous value of ω_{opt} and ρ_0 is a further damping term.

4. AUTOMATIC MESH GENERATION

4.1 Non-uniform triangulation

The generation of the non-uniform triangulation is performed through three main steps:

(i) Generation of boundary tags and coordinates along all external and interface boundaries described in the input data.

A tagging word (24 bits) is allocated to each mesh point (Figure 8.5a) in which is packed certain information relating to the disposition of the point w.r.t. the boundaries. The structure of this word is given by:

(1) VTU and VTL are the region numbers of upper and lower cells associated with the point I (Figure 8.5b).

(2) C is the boundary code, where

C=0 if I is not on a boundary

C=2 if I is on a boundary that is to be relaxed

C=3 if I is on a non-relaxed boundary.

(3) S is a side tag with the following interpretation:

S=0 if neither of the VU,VL lie along boundaries

S=1 if VU lies on a boundary

S=2 if VL lies on a boundary

S=3 if both VU,VL lie on a boundary.

(ii) Assignment of region numbers to all cells.

On completion of the first step (boundary points), a scan is made along each row of the mesh and the following action is performed: when the first point on this row is reached having a vertical side tag or the first point on the row above having a lower side tag, all following cells lying between the current mesh row and the one above are tagged with the current region number up to the next 'tagged side'; no further tagging is performed until another side is found. Whenever a point is found with a side tag then this tag is erased.

(iii) Generation of internal mesh coordinates.

In order to reflect the relative importance and material properties of each region, the mesh for a given problem should also be composed of regions which can be zoned to different average mesh spacings, with the mesh spacing in each region varying smoothly. Thus the coordinates of the unspecified internal points are found through solving a 'pseudo-potential' problem [47]. That is, the zoning problem is formulated as a potential problem, with the mesh lines playing the role of equi-potentials. The triangular mesh generated is composed of three sets of straight lines (Figure 8.6) intersecting each other at 60°, of which any two sets are sufficient to define the mesh.

Let one of these two sets be associated with a function $\phi(x,y)$ and the other with a function $\psi(x,y)$, each satisfying the Laplace equations

$$\left.\begin{aligned} \nabla^2\phi &= 0 \\ \nabla^2\psi &= 0 \end{aligned}\right\} \qquad (8.24)$$

over each region with boundary conditions determined by the interface and boundary zoning. Solving (8.24), the intersecting 'equipotentials' ϕ=constant and ψ=constant, together with the third set drawn through the intersection points, form the desired triangle mesh. By inverting equations (8.24) and writing them in terms of $x(\phi,\psi)$ and $y(\phi,\psi)$, we find that (8.24) are transformed into inverse Laplace equations:

$$\left.\begin{aligned} \alpha x_{\phi\phi}-2\beta x_{\phi\psi}+\gamma x_{\psi\psi} &= 0 \\ \alpha y_{\phi\phi}-2\beta y_{\phi\psi}+ y_{\psi\psi} &= 0 \end{aligned}\right\} \qquad (8.25)$$

where $\alpha=x^2_{\psi}+y^2_{\psi}$, $\beta=x_{\phi}x_{\psi}+y_{\phi}y_{\psi}$ and $\gamma=x^2_{\phi}+y^2_{\phi}$.

With meshes of a mild non-uniformity, equations (8.25) can be simplified, into a linear form i.e.

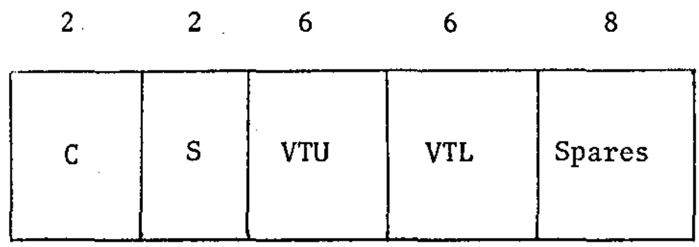


FIGURE 8.5a: The Tag Word

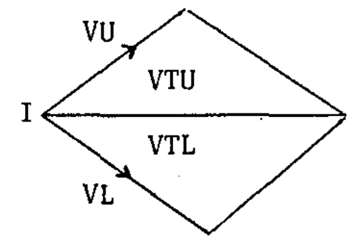


FIGURE 8.5b: A Typical Cell Configuration with Point I

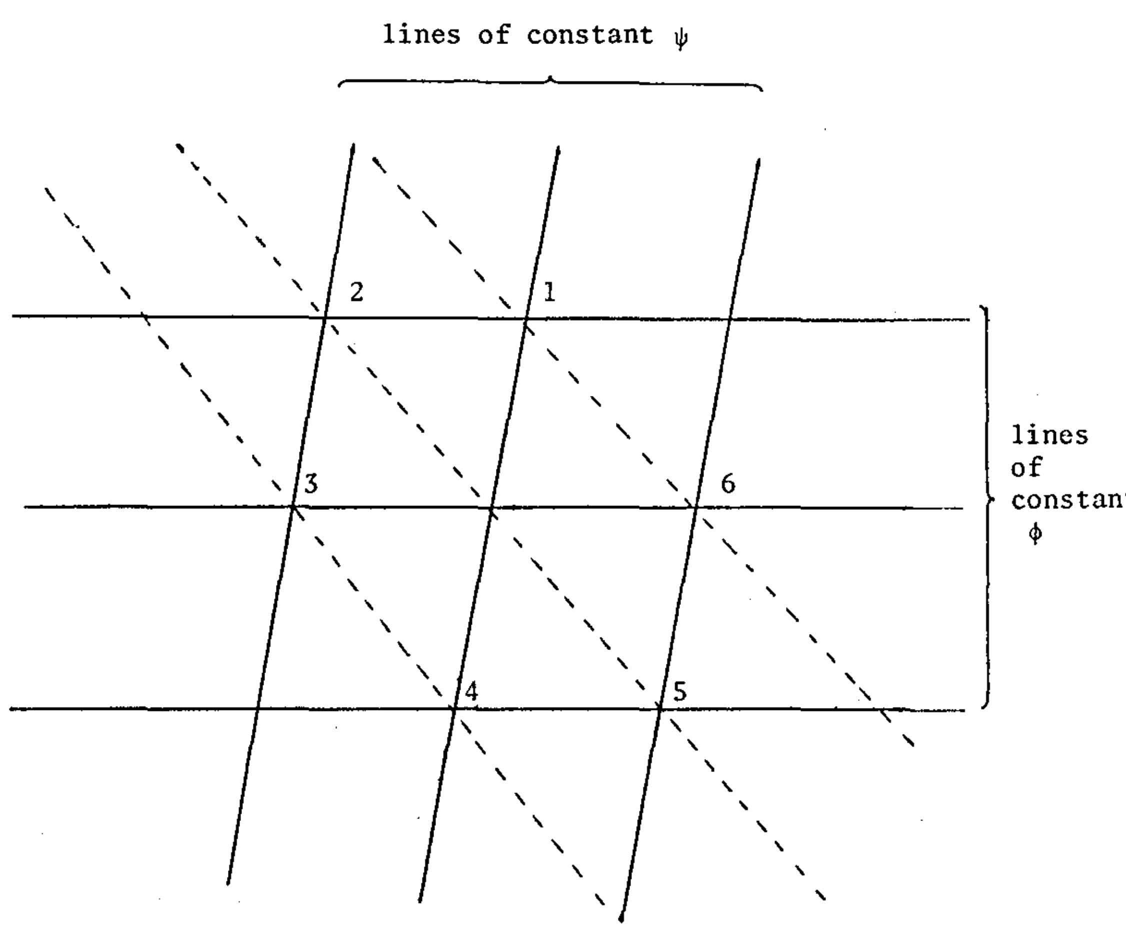


FIGURE 8.6: Pseudo-Equipotential Lines

$$\left.\begin{aligned} x_{\phi\phi} + x_{\psi\psi} &= 0 \\ y_{\phi\phi} + y_{\psi\psi} &= 0 \end{aligned}\right\} \qquad (8.26)$$

The difference equations corresponding to these two equations are:

$$\left.\begin{aligned} &\text{case (a):} \quad x_0 = \frac{1}{6}\sum_{i=1}^{6} x_i \,, \qquad y_0 = \frac{1}{6}\sum_{i=1}^{6} y_i \\ &\text{case (b):} \quad x_0 = \frac{1}{4}\sum_{i=1}^{4} x_i \,, \qquad y_0 = \frac{1}{4}\sum_{i=1}^{4} y_i \end{aligned}\right\} \qquad (8.27)$$

i.e. x_0, y_0 are simple averages of neighbouring coordinates. The type of zoning (case a or b) is specified in the input data for a region by a parameter z=1 or 2. The above two difference equations are solved by the method of successive point over-relaxation in an analogous nammer to that of the Ritz equation (8.11).

4.2 Logical Diagram and the Boundary Input Data Format

A useful apparatus for arranging the disposition of boundary points is the logical diagram. The concept of the logical mapping is useful in preparing the way for a description of the method of choosing the mesh layout for non-uniform triangulations.

This section is intended to outline briefly the use of the logical diagram and the format for the description of the boundary points to define the region geometry which is input to the program. A typical example is shown in Figure 8.7. The rules in the preparation of this diagram are as follows:

(i) Choose an equilateral net of sufficient degree of fineness to suit the problem in hand.

(ii) Represent any straight boundary section along any desirable logical direction in this mesh. The length and orientation of this path will determine both the number of mesh points and the zoning of the mesh.

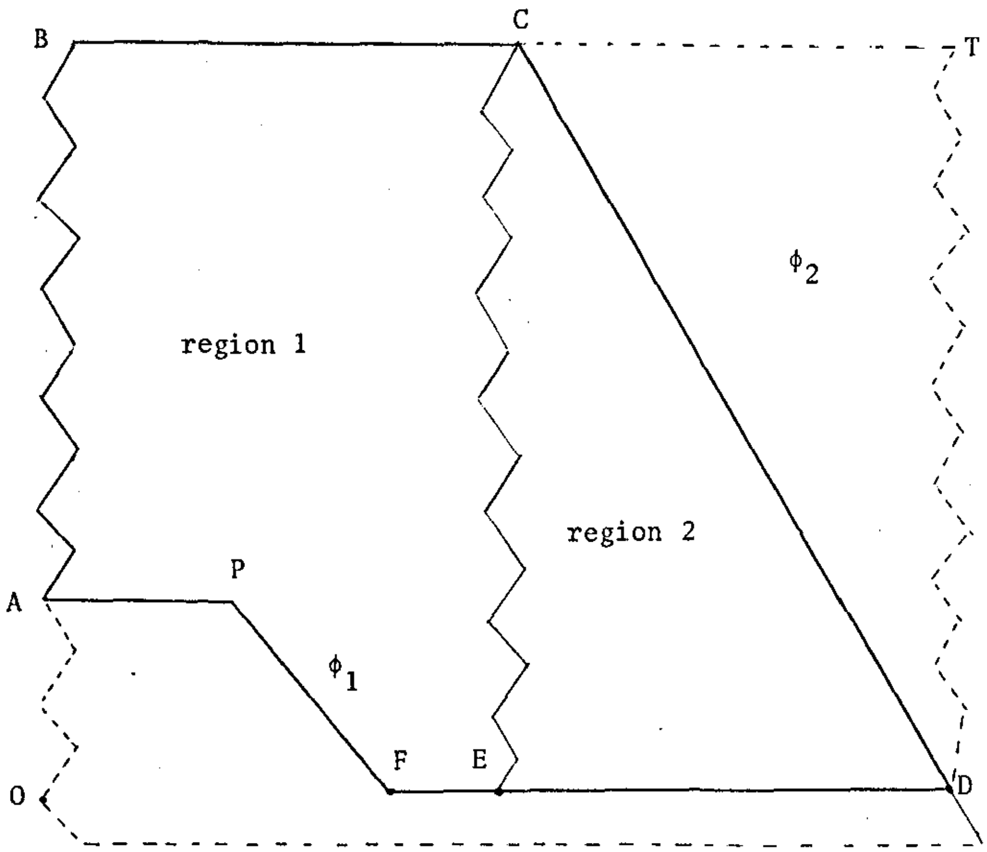


FIGURE 8.7: Logical Diagram

(iii) Simulate any 'Reflection' (Neumman) condition for symmetry by introducing a row of 'Dummy' cells on one side of the section, e.g. FED of Figure 8.7. When the external boundaries of the region do not fit precisely a rectangular outline, an imaginary rectangular boundary is marked to complete the region.

(iv) Curved sections (AF in Figure 8.7) can be represented by using the 'logical arc' facility provided in the program. The centre of the arc is defined in the input data by the following quantities L_0, K_0, θ_0 where (L_0, K_0) is the intersection point at the 'K' line and 'L' line through O and θ_0 is given as

$$\theta_0 = \frac{\text{no. of mesh sides along the logical slant line}}{\text{Total no. of mesh sides along the logical arc}}$$

(v) Code C=0 for any internal point,

C=1 is used for any boundary point which is to be relaxed,

C=2 for a boundary point which is not to be relaxed.

The format for the description of each boundary point is (ℓ, k, y, x, ϕ, C), where ℓ and k are row and column numbers in the range (0,LMAX and 0,KMAX), y and x are the coordinates of the boundary point, ϕ is the potential at that point and C is the code.

The boundary input data to the problem is arranged in the following manner:

If a boundary has the same constant code and potential (ϕ, C) for a large number of consecutive points in the input data, then rather than repeating the code and potential at each point it would be useful to signal only 'changes' in parameters. A convention was adopted in which the first point to be presented for a boundary is preceded by a section with a code of unity and potential of zero. With this convention it is only necessary to define (ℓ,k,y,x) at this point. If the section following the first point has the same code and potential then it is only necessary to give the coordinate of the point at the end of this section; and so on for the remaining points. However, if at any stage the type or potential of the intermediate points (or the end points) of a following section changes, then prior to giving the coordinate of the next point, one introduces an alphabetic sentinel 'B' followed by the quantities C_I, C_E, ϕ_E where:

C_I: is the code for the intermediate points

C_E: is the code for the end point

ϕ_E: is the potential at this point

The following example illustrates the use of this scheme for defining the geometry of a given region. In region 2 (Figure 8.7) the input data are defined by the following card images:

1st card: 2 Z_2 ε_2 ρ_2

This card specifies the properties of region 2 where ε_2, ρ_2 are the region constants and Z_2 the zoning code (1 or 2, see section 4.1(iii)).

2nd and subsequent cards:

$$\ell_E k_E y_E x_E B\ 1\ 2\ \phi_2 \ell_C k_C y_C x_C B\ 2\ 2\ \phi_2 \ell_D k_D y_D x_D\ G$$

Here the potential along ED is assumed zero. There is no need to repeat the first point to indicate closure of the region: the terminator G is used to signal this to the program. The data describing the arc AF of region 1 consists of an alphabetic sentinel A followed by the first point, the centre and the last point i.e.

$$A\ell_F k_F y_F x_F \ell_O k_O \theta_O \ell_A k_A y_A x_A$$

A routine called GEOMETRY reads in the cards defining the geometry, in a free format so that spacing between the separate items on a card is immaterial and so also is the number of items on the card. The only restriction on the layout of the data is that there should be at least two space characters between individual fields.

5. THE TMG-PROGRAM

The program was originally implemented on the ICL 1904S and the plotting part was performed off-line on an incremental plotter using the ICL graphic library routine HGPLOTT. Due to the limited computing resources available with a PDP 11/40 and the large amount of computing time required by the TMG program, it was not posssible to implement a fully interactive version of TMG on this machine. As a result the program is partitioned into two separate jobs.

(i) the numerical computation is performed in Batch mode on the ICL machine,

(ii) the plotting part is then carried out on the PDP 11/40 machine using the LIGHT package on the graphic display and allowing the user a limited amount of interaction with the displayed picture.

5.1 Description of the Batch Program and Its I/O Data

This program would carry out all the number crunching part of the process of generating the triangular mesh. It consists of the master or steering segment 'POISSON' and a number of subroutines performing the various computational tasks required by the program (see program listing Appendix 4.1). The main course of action required in the processing of each task is illustrated in the block flowchart of Figure 8.8. A number of distinct problems can be treated in one execution of this program. For each problem the user must include the input data in the following order:

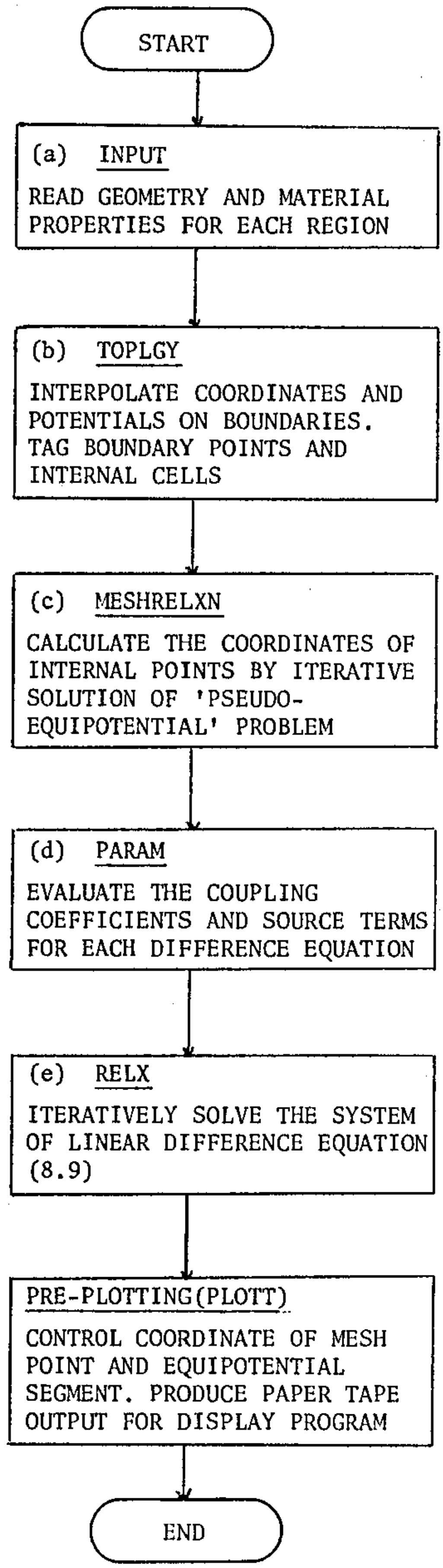


FIGURE 8.8: The Off-line Batch Program - POISSON

(a) Control data: these consist of five control cards carrying the following information in free format:

Card I — Number of problem.

Card II — Title for the current problem.

Card III — (1) a logical parameter (PLOTMESH) which assumes the value 'True' or 'False' according as the mesh lines are to be displayed or not.

(2) a count (NEQPTL) of the number of equipotential lines to be displayed.

Card IV — (1) number of regions in the geometrical description of the problem (NREG).

(2) material and geometrical properties to be ascribed to each region. This is usually fixed in number (equal to 4) e.g. region number, zoning code, permittivity, charge density.

Card V — The logical size of the mesh, KMAX,LMAX followed by the mesh parity.

(b) Geometric data: this consists of an unspecified number of cards, each containing the material and geometric properties of each region followed by the boundary points and their potentials. The type of information and the data format of this input is fully described in section 4.2 above.

The numerical results are dispatched to two types of output media:

(i) line printer output which includes print out of some intermediate results regarding the acceleration factor and convergence rate together with coordinates of mesh points and their associated potentials for each row.

(ii) paper tape punch contains the coordinates of the mesh points and the equipotential line segments. The distinction between mesh and equipotential data is made by generating a code for each type. All mesh lines are tagged with code 1, whilst equipotential lines are tagged with code 2,3,4... depending on the level. This paper tape data will be employed in the display program (see next section).

The main subroutines of the program depicted by each block of the flowchart of Figure 8.8 are:-

(a) INPUT.

Reads in the number of mesh rows and columns and the material and geometric data. The latter is read in free format by the routine GEOMETRY.

(b) TOPLGY.

Steers the processing of the geometric input data in the following manner. Between each pair of boundary points an entry is made to the routine BSET which interpolates the coordinates of the intermediate points and ascribe tags to these points and the last point of each section. The subroutine CODE is used by TOPLGY for examining any changes in the type of boundary code. If a marker denoting a curved (arc) section is encountered then entries are made to ARCSET and ARC, which in turn distribute the intermediate points by equal increments in polar angle.

Following the generation of a region boundary, an entry is made to SETREGION for the tagging of internal mesh cells associated with this region. TOPLGY continues with the next region until finally all regions of the problem have been completely defined.

(c) MESHRELXN.

The linear difference equations (8.27) are simultaneously solved by successive point over-relaxation. At each point of the relaxation sweep a check is made that the current point does not lie within a dummy region or on any boundary. If the point is internal then the region in which it lies is determined by extraction of the region tag for the upper cell at this point.

(d) PARAM.

This controls the process of evaluation of the coupling coefficients and source terms. Since the couplings and sources are linear combinations of the basic geometric and material properties within the cells, it will be obvious that a given coupling coefficient can be assembled by the contributions from cells at different stages of the sweep. All the evaluations are performed by the subroutine TERMS.

(e) RELX.

The system of linear Ritz difference equations (8.11) are solved by successive point over-relaxation. Initially, a starting value is chosen for the acceleration factor ω_{opt} (1.5) and the iteration counter ITN is set to zero. Following this a sweep is performed through all the points of successive rows. The sweep commences at the point (0,0) and proceeds to the end of the first row; this is repeated for the next row and so on. When the mesh sweep is completed the iteration counter ITN is updated and an entry is made to the routine SOR for a re-estimate of ω_{opt} equation (8.23).

(f) The pre-plotting routine PLOTT.

This routine organises and controls the generation of mesh and equipotential lines. The fundamental routines employed are respectively

LINET and EQUPLT. Initially, a sweep is made through the mesh and the maximum and minimum coordinates (used for setting the display viewport later) and potentials are determined. The required equipotential points are evaluated. Following this, a forward sweep is made for all points on the first row (L=0), and the coordinates of the mesh lines are set off to be punched on the paper tape for later use with the display program. Still on the same row, a cellwise, backward sweep is performed; for each cell the vertical sides are also sent to the paper tape punch. These sweeps are repeated for all subsequent rows.

5.2 The Display Program

This essentially displays the triangular mesh and the equipotential lines of the problem region from the output paper tape data generated by the Batch program. It also provides a limited interactive capability with the display picture through the selection of menu options.

Prior to running this program, the data on the paper tape must be read in and stored on disc file the name of which is specified by the user. For this purpose a small C-program (Appendix 4.2) was written for reading the paper tape data produced by the ICL machine and storing it on disc file on the PDP 11/40 machine. The program also removes unwanted characters from the data (e.g. null character, carriage return, parity bits) so that the format of the data stored is acceptable by UNIX-Fortran. In order to read the data, the paper tape is mounted on the PDP 11/40 paper tape reader and the following UNIX PIPE command [19] is used:

```
%PRTAPE</DEV/PR>AFILE
```

(A pipe is simply a way to connect the output of one program to the input of another program, so the two run as a sequence of processes).

Here, the PRTAPE is the executable C-program module, and AFILE is the name of the disc file containing the final data.

The display program (Appendix 4.2) consists of the main segment and three subroutines responsible for the plotting and controlling of various displays which include:

(i) Triangular mesh of the problem region.

(ii) Equipotential lines with or without the mesh.

(iii) Zooming part of the display picture.

The display program is generated by running the existing shell file:

%.SH JCL

where JCL is the shell file containing all necessary UNIX command for compiling and linking the program with the LIGHT library subroutines. The executable module is named 'FINITE'; so the user would be required next to type in:

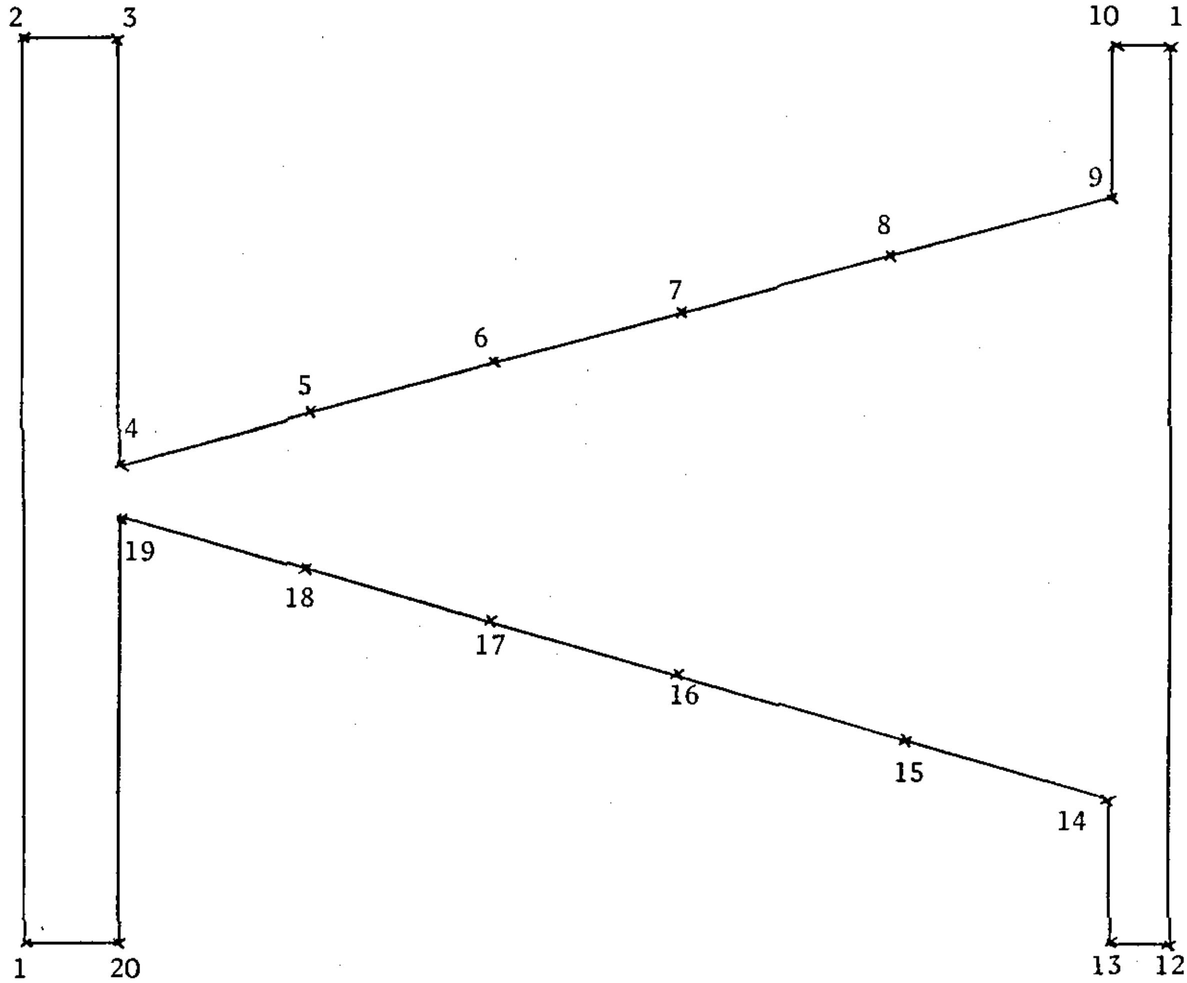
%.FINITE

and the program will run under his control.

The sequence of displays the user would encounter while running the program is best illustrated by means of an actual example. The problem region investigated in this example is concerned with the field distribution of an electron gun with its geometry and boundary potentials given in Figure 8.9. As soon as the program runs, the display image shown in Figure 8.10 appears on the screen. This display contains a short introductory remark followed by instructions for the user to make keyboard entries. The input data required by the program are:

(1) Data file name

(2) The limit of the data to be mapped on to the pre-specified viewport (i.e. minimum and maximum of x,y values).



x boundary points

	ℓ,k,y,x	ϕ	Cards	Input cards
1.	0,0,-25,-5	1500	1.	B 2 2 1500 0 0 -25 -5 20 0 25 -5
2.	20,0,25,-5	1500	2.	20 3 25 0 B 2 2 1525 13 3 1.5 0
3.	20,3,25,0	1500	3.	13 11 2.5 8 15 12 3.83 13.67
4.	13,3,1.5,0	1525	4.	15 18 6.5 25 18 19 10.17 37.33
5.	13,11,2.5,8	1525	5.	18 25 17 62 B 2 2 1500 20 25 25 62
6.	15,12,3.83,13.67	1525	6.	20 27 25 65 0 27 -25 65
7.	15,18,6.5,25	1525	7.	0 25 -25 62 B 2 2 1475 2 25 -17 62
8.	18,19,10.17,37.33	1525	8.	2 19 -10.17 37.33
9.	18,25,17,62	1525	9.	5 18 -6.5 25 5 12 -3.83 13.67
10.	20,25,25,62	1500	10.	7 11 -2.5 8 7 3 -1.5 0
11.	20,27,25,65	1500	11.	B 2 2 1500 0 3 -25 0 G
12.	0,27,-25,65	1500		
13.	0,25,-25,62	1500		
14.	2,25,-17,62	1475		
15.	2,19,-10.17,37.33	1475		
16.	5,18,-6.5,25	1475		
17.	5, 12,-3.83,13.67	1475		
18.	7,11,-2.5,8	1475		
19.	7,3,-1.5,0	1475		
20.	0,3,-2.5,0	1500		

FIGURE 8.9: Geometric Data for the Electron Gun Problem

Most of the menu options used throughout this program have the same function as those described in previous applications (e.g. IDF package).

Once the user has completed the necessary keyboard entries, he can proceed to the '+NEXT' display. This is the main display of the program, and enables him to perform the following actions, through the selection of menu options provided:

(a) When this option is picked up the triangular mesh generated would be displayed (Figure 8.11). Effectively the program reads the stored data one line at a time and draws the mesh. When the entire mesh is displayed, the program returns control to the user by the appearance of the cross-hair cursor; then the user may wish to proceed to take the next action. → +MESH

While the program reads the data, it stores the equipotential line coordinates on an array list so as to avoid the time-consuming operation of reading the data file again.

(b) This option will cause the equipotential line to be displayed. If the user wishes to plot these lines separately (Figure 8.12) he can do so by selecting the option '+RESTART' (which clears the screen) prior to this option. On the other hand, if combined displays of both the mesh and equipotential lines (Figure 8.13) are needed then the user would select this option directly. → +EQUIP'L

(c) This option allows a portion of the region to be enlarged. For example, Figure 8.14 shows that a zooned window is defined (by using the cursor and → +ZOOM

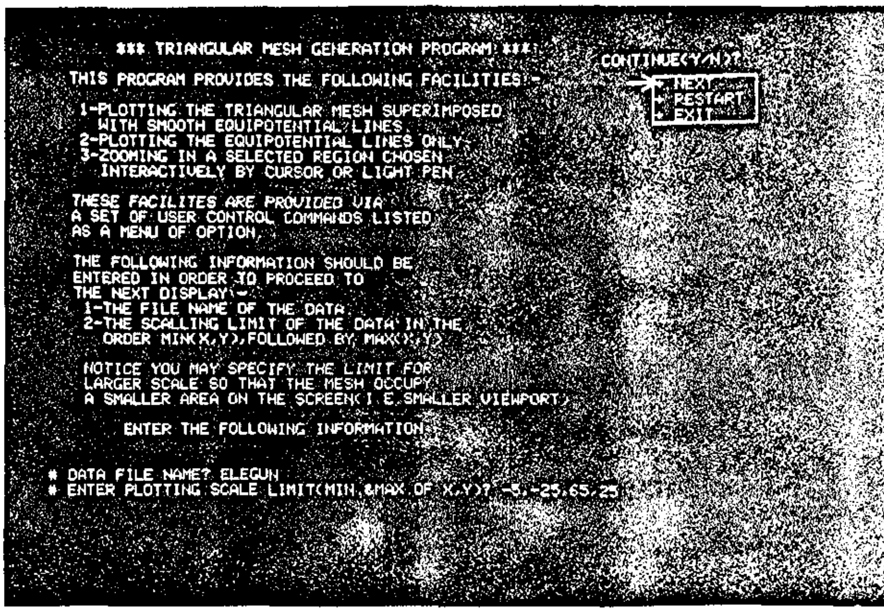


FIGURE 8.10: Introductory Display

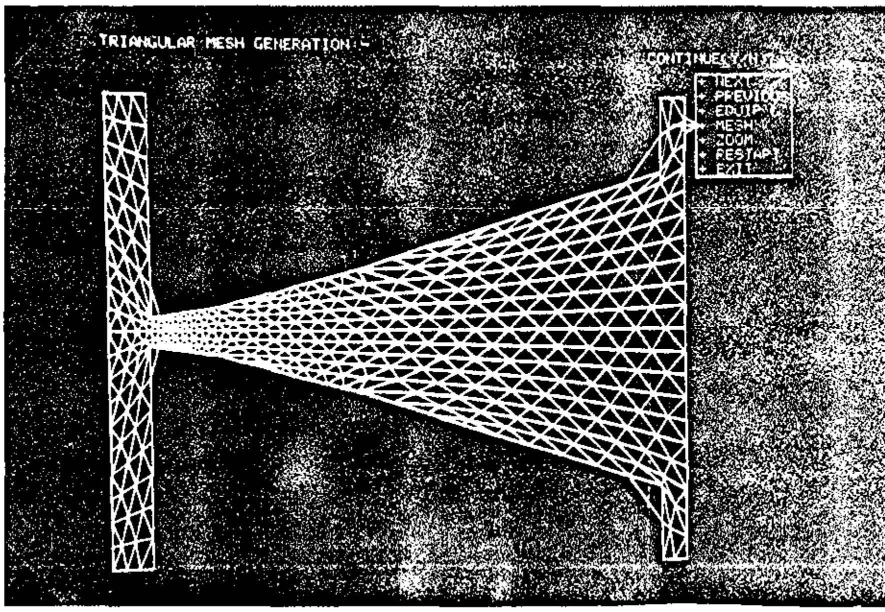


FIGURE 8.11: Triangular Mesh of the 'Electron Gun' Problem Region

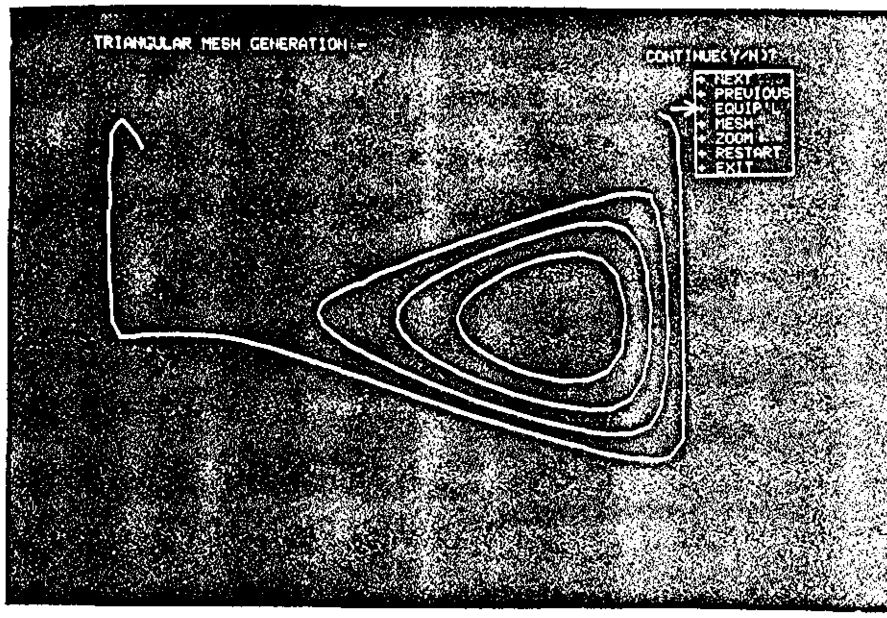


FIGURE 8.12: Equipotential Lines

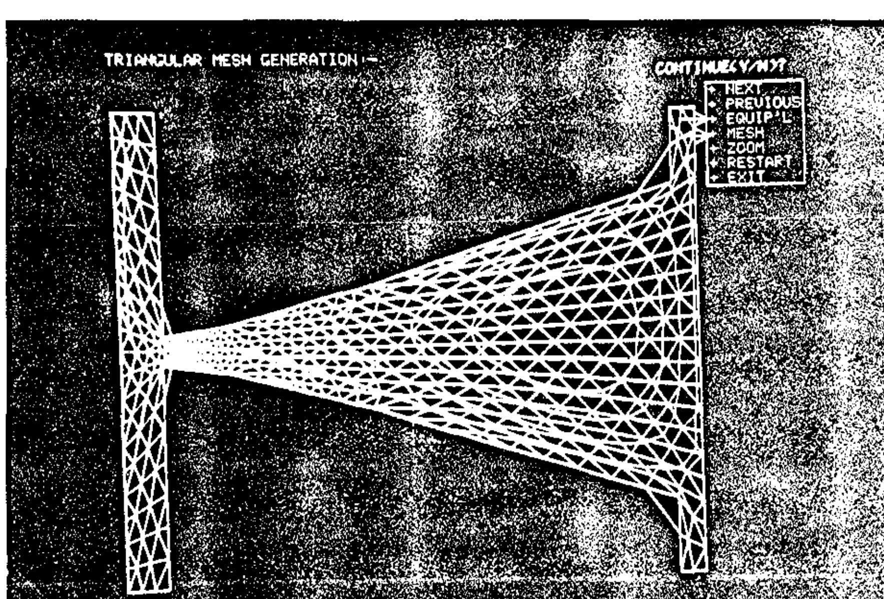


FIGURE 8.13: Triangular Mesh Superimposed By The Equipotential Lines

selecting any two opposite corners of the rectangle) and Figure 8.15 shows the blown-up display of the window selected.

Another example is also shown in Figure 8.16 and Figure 8.17 of the field distribution in the region between two charged parallel conductors of finite width but infinite lengths.

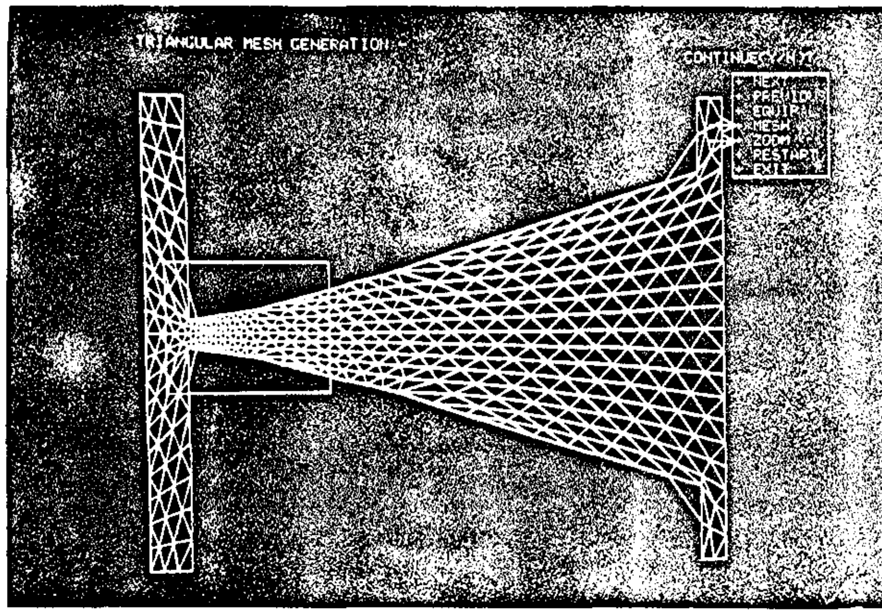


FIGURE 8.14: Choosing the Zooming Window

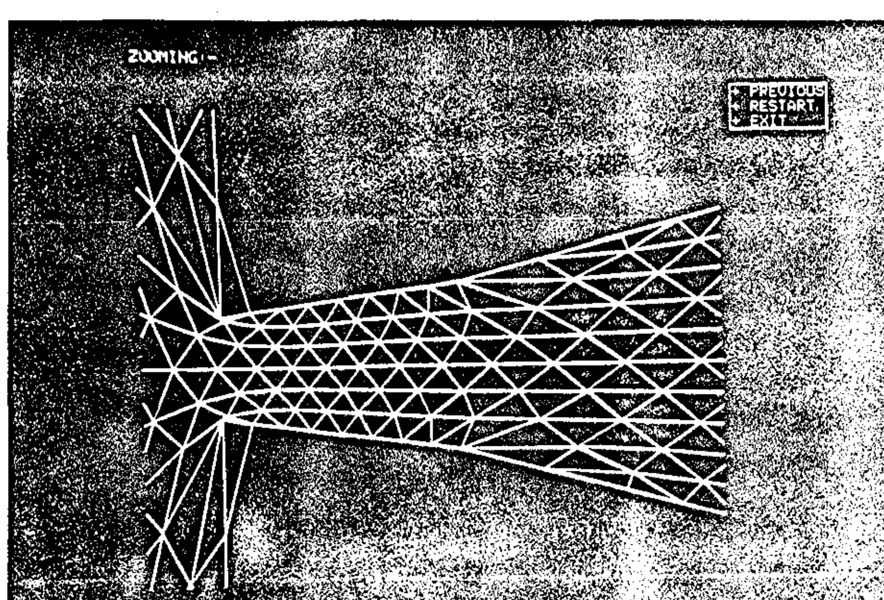


FIGURE 8.15: The Zoomed Portion of the Triangular Mesh

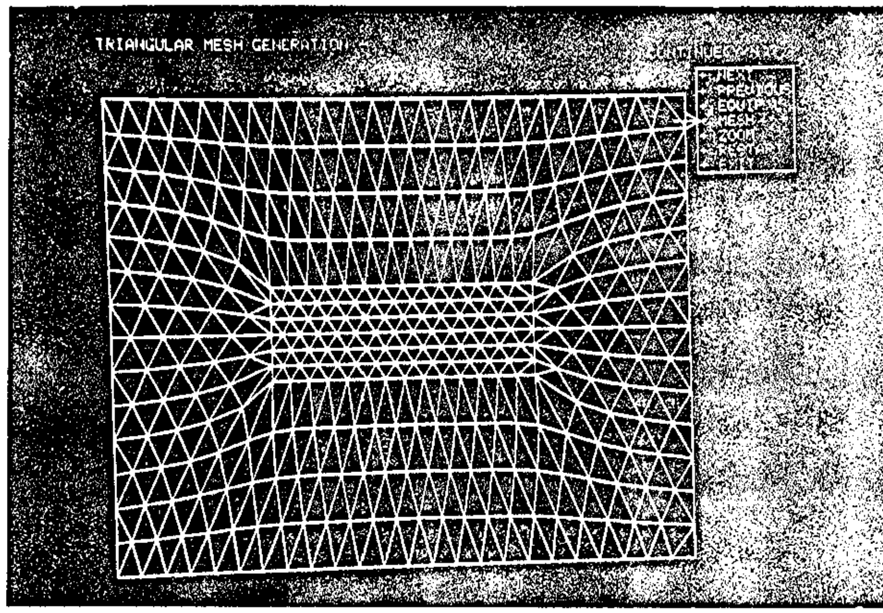


FIGURE 8.16: Triangular Mesh of the Two Charged Parallel Conductors

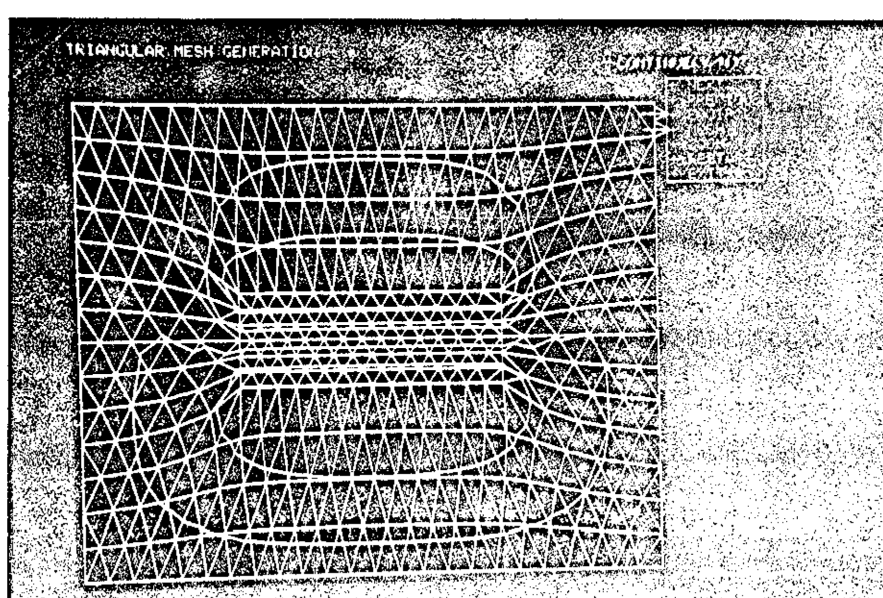


FIGURE 8.17: Same as the Above Example with Equipotential Lines

CHAPTER 9

SUMMARY AND CONCLUSIONS

An initial study was made of the current state of hardware/software facilities in interactive computer graphics. This was a useful exercise in its own right, but was also motivated by the limited graphics support which was locally available at the start of this work. General-purpose graphics packages such as GINO-F were being marketed commercially but usually required too many resources for our mini-computer environment. It was apparent that in-house graphics software was needed to support a wide range of computer graphics applications, typically in the area of scientific computing, and to form the basis for developing a graphics laboratory. Consequently, the graphics software package LIGHT was then implemented as a set of Fortran-callable library subroutines, capable of driving the graphics display (Tektronix 4010) from the PDP 11/40 under UNIX time-sharing system.

The scope of LIGHT was extended beyond the basic requirements to include three-dimensional transformations, simple perspective projection, menu operation and text handling capabilities ; it was also made accessible from the GT42 refresh display operating in Emulator mode. At the same time the total core requirements of the LIGHT library were designed to be less than 8K words.

Fortran was chosen because graphics systems are likely to be used for scientific and engineering applications. The use of Fortran also makes the package (and the application programs) portable except for the 'back end' which contains some assembly code and the terminal dependent routines. Because of the modular design of the package and the inclusion of the back end code in a separate module, the only code modification which would be needed in a different configuration is performed on this module alone.

The Tektronix 4010 cross-hair cursor and keyboard can be used efficiently as input devices. LIGHT has provided various input graphics capabilities activated from the keyboard or by a cursor position with single-character command or through menu selection, each of which can be transmitted to the computer program during execution. Thus the application program can be written to incorporate such user interaction in a natural manner.

Window/Viewport transformations have been introduced to ease the problem of coordinate system mapping. Graphic displays are generated by defining a coordinate system, scales, labelling and text generation. The scales are determined either automatically in order to fit a selected portion of the data or explicitly by the user. Where the data falls outside the screen viewport, a clipping (scissoring) routine is used to exclude all such extraneous elements.

A variety of three-dimensional transformations have also been implemented, including rotation and perspective projection. The generation and storage of transformation matrices by LIGHT effectively offers the application programmer a system of transformation control.

The present capability of the graphic software can be extended to include hidden line removal and graphical data structure facilities; these could be valuable assets in a computer graphics laboratory. It would also be possible to implement a display-file based version of LIGHT. Here, the display file would not of course be used to refresh the display. However, it can serve a useful purpose in permitting part of the picture to be manipulated. Furthermore, some of the facilities provided by LIGHT can be easily incorporated into the locally available Picture Book package [11] used in conjunction with the GT42 either directly (transformation routines) or with slight modification (e.g. menu handling routines).

LIGHT has shown its usefulness in a range of applications and for a wide variety of users. There is a good case for such a general purpose and efficient graphic software package which is accessible from a high-level language. The following application areas were exploited during the course of this work and made extensive use of LIGHT in an interactive environment.

(i) Interpolatory Data Fitting - IDF

This provides a user interface for solving a certain class of data-fitting problems and is oriented toward the non-programmer. The 'easy-to-use' design of the interaction embodied in IDF, and the inclusion of the ability to specify various end conditions give it the combination of simplicity and effectiveness necessary to a useful interactive problem-solving device. It consists of two separate packages (Explicit and Parameteric), each containing a number of overlayed modules and a library of numerical algorithms featuring mainly cubic spline interpolation methods. This set of algorithms has been enhanced and adapted for interactive use. Additionally, the user is offered a command menu, through which he would have full control over the execution path of the package, with on-line help and a large number of utility options for thorough examination of the interpolated curve.

Several cubic splines were generated using spline algorithms for a variety of boundary conditions. This smooth interpolatory curve fitting system can serve a useful purpose for creating and displaying graphical information. When straight line vectors are used to connect the interpolated points, rapid and relatively inexpensive graphical output can be produced and modified interactively by the user. The controlled end condition specifications allow easier control over the final shape of the curve. However, this technique does not rival the methods of interactive curve design developed by Bezier [49] and Riesenfeld [50]. However, when a few known data points are available, and control of the curve through these points is desirable, the above type of algorithm can be useful.

Some possible extensions and improvements that are possible in the IDF system are:

(1) Extending the localised polynomial method to incorporate different weighting functions and the ability to select these weights interactively.

(2) The addition of further algorithms, for example least squares cubic spline approximation.

(3) For users with 'noisy' data the suggestion made in (2) could be developed further into a separate Generalised Data Fitting package having the same philosophy and structure as IDF but incorporating for example least squares and minimax algorithms.

(4) The addition of more display modes that would aid user interpretation of the results. For example, one could display the first and second derivatives at the data points and at prescribed intermediate points.

(ii) Interactive Contour Tracing - ICT

The basis of this program is a tracing algorithm for drawing the contour f(x,y)=constant in a region over which there is a method of calculating the function. After some preliminary study of the available methods based on a regular mesh, an algorithm was developed from those which used a more localised tracing technique. This algorithm was adapted for interactive use on a display terminal, allowing the user to trace contours automatically or interactively. Some fundamental modifications were incorporated, providing improvements in smoothness, efficiency and the treatment of degeneracy.

The generality and capability of this program can be extended further by providing the user with the option of tracing a set of contour lines for a given set of arbitrary points (or grid points). This would obviously require the inclusion of a routine which would estimate the function value of a given point (x,y) from the set of data points specified initially by the user.

(iii) Triangular Mesh Generation - TMG

This program provides a graphics display and limited interactive facilities for the triangular mesh generation algorithm developed previously in conjunction with the solution of the Laplace or Poisson partial differential equation in an arbitrary two-dimensional region. Some development work was needed for modifying the original ICL 1900 program and its output before subsequent interactive use on the PDP 11/40. This provides a display of the triangulated region, and the equipotential lines using interpolation techniques on the solution values at the mesh nodes. It also allows zooming into any subregion of interest.

If more computing power were available on the PDP 11/40, the program would be entirely run on this machine with a fully interactive capability built into it. For example, it would then be possible to input and edit the geometric data describing the boundary of the region by means of cursor and keyboard entries. Furthermore, intermediate computational results could be displayed on request for examination by the user.

A number of other application programs were successfully developed by other users using LIGHT, for example:

(1) Displays for critical path analysis in an interactive graphics environment by Hargrave [51].

(2) Computer simulation of boundary layer growth and wake propagation on compressor cascade by Jamani [52].

The published photographs of the screen were produced from a domestic camera. However, a hardcopy device such as a pen-plotter would be a useful asset to the system.

For the modest investment in terms of both money and software development, the graphics systems developed in the course of this work have provided successful and useful tools for the average user. Moreover, they represent the basis of a graphics laboratory which could be used for future innovation in this field and its applications.

The original aim of this project was centred on the data-fitting application of graphics. However, it will be observed that the scope of the work which is reported in this thesis has expanded considerably from the original idea. Further, it seems natural to suggest that there are other classes of numerical problem which can also benefit from the same kind of treatment. These may include:

(1) Partial differential equations. Research on interactive graphical systems to represent the solution of partial differential equations would be very rewarding. Regions could be displayed and modified interactively and the corresponding results indicated graphically as well.

(2) Numerical linear algebra. This area exemplifies problems whose intermediate results can be graphically presented for quick comprehension by the users. The condition of the coefficient matrix involved in the solution of systems of linear equations is a critical factor in the proficiency of various algorithms to obtain a satisfactory solution. There are various ways to examine the condition of a matrix.

Geometrical consideration yields a hyperellipsoid whose axes are inversely proportional to the eigenvalues of the matrix. Thus an elongated hyperellipsoid indicates bad conditioning whereas a near hypersphere indicates a well conditioned matrix. It is felt that an on-line user will be able to recognise and interpret a graphically presented hyperellipsoid more readily than a list of numbers.

REFERENCES

[1] W.M. NEWMAN and R.F. SPROULL,

'Principles of Interactive Computer Graphics',

McGraw-Hill Computer Science Series, New York, 1973.

[2] DEC - GT42,

'User-Guide Manual',

Digital Equipment Corporation, Massachusetts, 1975.

[3] GINO-F MK2 Manual,

CADC Cambridge, 1976.

[4] P.R. DIMMER,

'Graphics Facilities',

Interactive computer graphics for engineers,

Leicester CAD Group, 1976.

[5] D.P. BRADLY,

'Basic Graphics Packages',

Interactive computer graphics for engineers,

Leicester CAD Group, 1976.

[6] GPGS,

'Users Tutorial Reference Manual',

University of Mymegen, October 1975.

[7] DISPLA,

'Mini Manual',

ISS Co. 1973.

[8] EVANS and SUTHERLAND,

'Picture System',

User Manual, 1974.

[9] CUPID,

'User Guide' Vol.11,

P.O. Research Department, Jan.1972.

[10] PLOT-10,

'Advanced Graphics Users Manual',

Information Display Products, Tektronix, Mar.1972.

[11] DEC - GT42,

'Picture Book',

Reference Manual, Digital Equipment Corporation, 1973.

[12] G.A. BUTLIN,

'Interactive Input',

Interactive computer graphics for engineers,

Leicester CAD Group, 1976.

[13] N.W. WISEMAN, H.U. LEMKE, and J.O. HILES,

'A New Approach to Graphical Man-machine Communication',

IEE International CAD, 1969.

[14] J.D. FOLLEY and V.L. WALLACE,

'The Art of Natural Graphical Man-machine Conversation',

Proceeding IEEE, Vol.62, No.4, 1974.

[15] D. ROSS,

'The AED approach to generalised computer aided design',

Proceedings ACM National Meeting, 1967.

[16] W.M. NEWMAN and R.F. SPROULL,

'An approach to graphics system design',

Proceeding of IEEE, Vol.62, No.4, 1974.

[17] R. WILLIAMS,

'A survey of data structures for computer graphics systems',

Computing Surveys, Vol.1, No.1, 1971.

[18] D.M. RITCHIE and K. THOMPSON,

'The UNIX time-sharing system',

Bell Laboratories, CACM, Vol.17, No.7, 1974.

[19] UNIX Programmer's Manual,

COMMANDS, Internal Document, Section 1.

Computer Studies Department, Loughborough University of Technology.

[20] *'Tektronix 4010 Computer Display Terminal'*,

User Manual, Tektronix Inc., 1972.

[21] UNIX Programmer's Manual,

SYSTEM CALLS, Internal Document, Section 2.

Computer Studies Department, Loughborough University of Technology.

[22] UNIX Assembler Reference Manual,

Internal Document, Section I.

Computer Studies Department, Loughborough University of Technology.

[23] D.F. ROGERS and J.A. ADAMS,

'Mathematical elements for computer graphics',

1976.

[24] I. McNEIL,

'Tektronix Emulator Manual',

DEC, 1976.

[25] A.R. FORREST,

'Current development in the design and production of three-dimensional curve objects',

Proc.Roy.Soc., London, 1971.

[26] P.W. WILLIAMS,

'Numerical Computation',

1972.

[27] A.D. MAUDE,

'Interpolation - mainly for graph plotter',

Computer Journal, Vol.16, No.1, 1973.

[28] H. SPATH,

'Spline algorithms for curves and surfaces',

1974.

[29] H. AKIMA,

'A new method of interpolation and smooth curve fitting based on local procedure',

JACM, Vol.17, No.4, 1970.

[30] A.W. NUTBOURNE,

'A cubic spline package, Part 2',

Computer-Aided Design, Vol.5, No.1, 1973.

[31] A. BENSON,

'The numerical solution of partial differential equations by finite difference methods',

Ph.D. Thesis, 1969.

[32] D.J. EVANS,

'Algorithm for the solution of quindiagonal systems of linear equations',

Computer Studies Department, Loughborough University of Technology. Unpublished Notes.

[33] DEC - DOS,

'Fortran Compiler and Object Time System',

Programmer's Manual, 1973.

[34] RSX-11,

'Multi-tasking Operating System',

Digital Equipment Corporation.

[35] S.P. MORSE,

'Concepts of use in contour map processing',

Communications of ACM, Vol.12, No.3, 1969.

[36] G. COTTAFAVA and G. LE MOLI,
'Automatic Contour Map',
Communications of ACM, Vol.12, No.7, 1969.

[37] M.A. ROTHWELL,
'A computer program for the construction of Pole figures',
J.Appl.Cryst. 1971.

[38] E.L. ROBINSON and H.A. SCARTON,
'Contor: A Fortran subroutine to plot smooth contours of a single-valued arbitrary three-dimensional surface',
Journal of Computational Physics, 1972.

[39] D.H. McLAIN,
'Drawing contours from arbitrary data points',
The Computer Journal, Vol.17, No.4, 1974.

[40] D.C. SUTCLIFFE,
'An algorithm for drawing the curve f(x,y)=0',
The Computer Journal, Vol.19, No.3, 1975.

[41] M.J.D. POWELL,
'Piecewise quadratic surface fitting for contour plotting',
Theoretical Physics Division, A.K.A.E.A. Research Group, Harwell, 1973.

[42] I.P. SCHAGEN,
'Contouring for arbitrary positioned data points',
Computer Studies Department, Loughborough University of Technology, Internal Report No.60, 1977.

[43] M.L.V. PITTEWAY,

'Computer Graphics Research in an Academic Environment',

Datafair 73 Conference Proceedings, Vol.2.

[44] D.C. SUTCLIFFE,

'A remark on a contouring algorithm',

The Computer Journal, Vol.19, No.4, 1975.

[45] C.M. CRANE,

'Contour plotting for functions specified at nodal points of an irregular mesh based on an arbitrary two parameter coordinate system',

The Computer Journal, Vol.15, 1972.

[46] G.E. CARDEW,

'A Ritz finite difference program for static field problems',

University of Sheffield, M.Sc. Thesis, 1971.

[47] A.M. WINSLOW,

'Numerical solution of the quasilinear Poisson equation in a non-uniform triangular mesh',

The Journal of Computational Physics, 1967.

[48] D.J. EVANS,

'The analysis and application of sparse matrix algorithms in the finite element method',

The Mathematics of Finite Elements and Applications,

J.R. Whiteman, 1973.

[49] P.E. BEZIER,

'Example of an Existing System in the Motor Industry: The Unisurf System',

Proc.Roy.Soc.(London), Vol. A321, pp.207-218, 1971.

[50] R.F. RIESENFELD,

'Berstein-Bezier Method for the Computer Aided Design of Free-Form Curves and Surfaces',

Ph.D. Thesis, Syracuse University, 1973.

[51] T. HARGRAVE,

'Displays for critical path analysis in an interactive graphics environment',

M.Sc. project 1977, Loughborough University of Technology.

[52] S. JAMANI,

'Computer simulation of boundary layer growth and wake propagation on compressor cascades',

B.Sc. Final Year project 1978, Loughborough University of Technology. (Private Communication).

APPENDICES

APPENDIX 1

LIGHT - PROGRAM LISTING

```
/                           ****************
/                           * APPENDIX 1.1 *
/                           ****************
/
/ LIGHT-UNIX INTERFACE ROUTINES
/ -------------------------------------------
/
/
/
/THIS ROUTINE WOULD OUTPUT A SINGLE CHARACTER TO THE SCREEN
/FORTRAN CALL:- R=OCHAR(I)......FUNCTION CALL
.GLOBL OCHAR.
.GLOBL RETRN
OCHAR.:
        VALUE                   /LOCATION OF RETURN VALUE
        .+2                     /POINTER TO EXECUTION CODE
        MOV  2(R3),R1           /POINTER TO ARG. LIST
        ADD  $2,R1              /POINTS AT THE 2ND WORD OF THE INTEGER
        MOV  R1,CHAR            /SET CHAR ADRESS
        MOV  $1,R0              /SET FILE DESCRIPTOR
        SYS  GTTY;STATUS        /GET TERMINAL STATUS
        MOV  STATUS+4,OMODE     /SAVE OLD MODE OF TTY
        SYS  SIGNAL;2;EXIT1     /CONTROL C
        BIC  $26,STATUS+4       /CHARACTER MODE
        BIS  $40,STATUS+4       /SET RAW MODE
        MOV  $1,R0
        SYS  STTY;STATUS        /SET TERMINAL MODE
        MOV  $1,R0              /FILE DESCIPTOR 1 FOR WRITE
        SYS  WRITE              /WRITE CHAR
CHAR:   0                       /BUFFER ADRESS
        1
        MOV  R0,VALUE           /RETURN NO.OF CHAR WRITTEN
EXIT1:  MOV  OMODE,STATUS+4     /BACK TO OLD TERMINAL STATUS
        MOV  $1,R0
        SYS  STTY;STATUS
        JMP  RETRN
.BSS
VALUE:  .=.+2                   /SPACE FOR RETURN VALUE(AN INTEGER)
STATUS: .=.+6                   /TERMINAL STATUS ARG.
OMODE:  .=.+2                   /OLD TTY STATUS MODE
/
/THIS ROUTINE INPUT A SINGLE CHAR FROM TEKTROINX 4010
/FORTRAN CALL:- ICHAR=INCHAR(X)   WHERE X IS DUMMY PARAMETER
.GLOBL INCHAR.                  /DEFINITION OF ENTRY
.GLOBL RETRN                    /REFERENCE OF RETURN
INCHAR.:                        /ENTRY POINT
        VALUE                   /LOCATION OF RETURN VALUE
        .+2                     /POINTER TO EXECUTION CODES
        MOV $0,R0               /SET FILE DESCRIPTOR
        SYS GTTY;STATUS         /GET TERMINAL STATUS(MODE)
        MOV STATUS+4,OMODE      /SAVE OLD TERMINAL MODE
        SYS SIGNAL;2;EXIT1      /CONTROL C
        BIS  $40,STATUS+4       /SET IT TO RAW MODE
        BIC  $10,STATUS+4       /SWITCH ECHO OFF
        MOV $0,R0               /FILE DESCRIPTOR
        SYS STTY;STATUS         /SET TERMINAL MODE
        MOV $0,R0               /PASS FILE DESCRIPTOR
        SYS READ                /READ A CHARACTER
        VALUE+2                 /ADDRESS OF CHAR JUST READ
        1                       /NO. OF CHAR READ
        MOV $0,R0
```

```
EXIT1:  MOV  OMODE,STATUS+4        /RESET TO ENTRY TTY MODE
        SYS  STTY;STATUS           /SET TERMINAL MODE AS BEFORE
        JMP RETRN                  /RETURN TO MAIN PROCEDURE
.BSS
VALUE:  .=.+4                      /CONTAIN CHAR JUST READ
STATUS: .=.+6                      /STATUS WORD
OMODE:  .=.+2                      /OLD TTY MODE
/
/SET UP CROSS-HAIRS CURSOR ROUTINE
        .GLOBL  CURSON. /ENTRY NAME
        .GLOBL RETRN    /RETURN
CURSON.:
        VALUE
        .+2
        MOV     R3,SAVER3                  /SAVE REGISTER 3
        MOV     $1,R0                      /SET FILE DESCRIPTOR
        SYS     GTTY;STATUS                /GET THE CURRENT TERMINAL STATUS
        MOV     STATUS+4,OMODE             /SAVE TERMINAL MODES
        BIC     $36,STATUS+4               /SET TERMINAL MODES
        BIS     $40,STATUS+4               /SET RAW MODE
        MOV     $1,R0
        SYS     STTY;STATUS                /SET THE NEW TERMINAL STATUS
        MOV     $1,R0
        SYS     WRITE;SETCURSOR;2          /SET UP CROSS HAIR CURSOR
        MOV     $5,R5                      /SET COUNT TO FIVE
1:      MOV     $1,R0
        SYS     READ;GETCURSOR;1           /READ A CHAR
        TST     (R3)+                      /SET ARGUMENT LIST POINTER
        MOV     (R3),R1
        TST     (R1)+
        MOVB    GETCURSOR,(R1)             /PUT ARGUMENT LIST
        SOB     R5,1B                      / LOOP FIVE TIMES
        MOV     $1,R0
        SYS     READ;GETCURSOR;1           /INPUT FIVE CHARACTERS
        MOV     $1,R1
        SYS     WRITE;HOME;6               /SET ALPHA CURSOR HOME
        MOV     OMODE,STATUS+4             /RESTORE ORIGINAL MODE
        MOV     $1,R0
        SYS     STTY;STATUS
        MOV     SAVER3,R3                  /RESTORE REGISTER
        JMP     RETRN

        .DATA
SETCURSOR:
        .BYTE   27.,26.                    / FOR SETTING UP CURSOR
HOME:   .BYTE   29.,55.,127.,32.,64.,31.;/FOR RETURING TO ALPHA MODE
        .BSS
VALUE:  .=.+4
STATUS: .=.+6                              /WORK SPACE TO SAVE TERMINAL ST.
OMODE:  .=.+2                              /WORK SPACE TO SAVE TERMINAL MODE
SAVER3: .=.+2                              /WORK SPACE TO SAVE R3
GETCURSOR:
        .=.+2                              /HOLD INPUT CHARACTER

/
/OVERLAY PROGRAMS MODULES
        .GLOBL OVLAY.
        .GLOBL RETRN
OVLAY.:
        VALUE
        .+2
        MOV R3,R1                  /PASSING PARAMETER
```

```
        TST  (R1)+
        MOV  (R1),NAME          /PASS THE NAMED FILE
        MOV  (R1),ARGS          /SET ARGUNENT TO NAME
        SYS EXEC                /OVERLAY THE CALLING PROCESS
NAME:   0
        ARGS
        JMP RETRN
.DATA
ARGS:   0;0
.BSS
VALUE:  .=.+2
/
/       REMOVES A NAMED FILE FROM CURRENT DIRECTORY
        .GLOBL FLRM.
        .GLOBL RETRN
FLRM.:
        VALUE
        .+2
        MOV  R3,R1              /PASSING THE PARAMETER
        TST  (R1)+
        MOV  (R1),NAME
        SYS UNLINK              /REMOVE NAMED FILE FROM CURRENT
                                /DIRECTORY
NAME:   0
        JMP RETRN
.BSS
VALUE:  .=.+2
```

```
C                        ****************
C                        * APPENDIX 1.2 *
C                        ****************
C
C INITIALISATION ROUTINES
C -----------------------
C
C
C
C **********INITIALISE THE DISPLAY DEVICE**********************
C
      SUBROUTINE TXOPEN
C THIS SETS THE DEFAULT VALUES FOR DISPLAY AREA & SCALES OF
C X,Y.THIS ALSO DOES SOME INITIALISATION CONCERNING SCREEN
C COORDINATES & MENU OPERATIONS.
C N.B.THIS SUBROUTINE MUST BE ISSUED AS THE FIRST CALL ON THE
C PACKAGE.
      COMMON/AREA/SCALE(12)
      COMMON/TOPLHC/IXORIG,IYORIG
      COMMON/MNGRPH/MODE,LPOS,CHPOS
      COMMON/MATRIX/CTM(4,4),TM(4,4)
      INTEGER CHPOS
C SET UP COORDS.OF TOP L.H CHARACTER & SCALLING
      SCALE(1)=0
      SCALE(2)=0
      SCALE(5)=0
      SCALE(7)=0
      SCALE(4)=780
      SCALE(8)=780
      SCALE(10)=780
      SCALE(12)=780
      SCALE(3)=1023
      SCALE(6)=1023
      SCALE(9)=1023
      SCALE(11)=1023
C SET UP MENU ORIGIN & CHARACTER  LINE POSITION
      IXORIG=0
      IYORIG=780
      MODE =1
      LPOS=0
      CHPOS=0
C SET TRANSFORMATION MATRIX TO UNIT MATRIX
      CALL UNITY(CTM)
      CALL UNITY(TM)
      RETURN
      END
C
C***************ERASE THE SCREEN************************
C
        SUBROUTINE TXCLER
C       THIS CLEARS THE SCREEN
C OUTPUT TWO ASCII CHARACTERS
C ERASE SCREEN AND RETURN TO ALPHA MODE
        S=OCHAR(27)
C GOES HOME
        S=OCHAR(12)
        RETURN
        END
C
C *************** SET A DEFINED AREA AS VIEWPORT*************************
C
```

```
      SUBROUTINE TXVPRT(X0,Y0,X1,Y1)
C THIS LIMIT THE EXTENT OF A DISPLAY TO A SELECTED AREA
C OF THE SCREEN.IF PARAMETERS OUTSIDE AREA CALL IS IGNORED.
      COMMON/AREA/SCALE(12)
C CHECK FOR ANY VIOLETION
      IF(X0.LT.0.OR.X1.GT.1023.OR.X0.GE.X1.OR.Y0.LT.0.OR.Y1.GT.780
&       .OR.Y0.GE.Y1) GOTO 1
      SCALE(1)=X0
      SCALE(2)=Y0
      SCALE(3)=X1
      SCALE(4)=Y1
      SCALE(11)=X1-X0
      SCALE(12)=Y1-Y0
      RETURN
1     CALL CURPOS(200.,500.)
      CALL ALPHMD
      CALL MESSAG("DRAWING OFF THE SPECIFIED VIEWPORT^")
      RETURN
      END
C
C ******************SET A WINDOW FOR GIVEN SCALLING OF THE DATA********
C
      SUBROUTINE TXWIND(X0,Y0,X1,Y1)
C THE SCALES ARE SET SUCH THAT X0 TO X IS MAPPED ONTO X-AXIS
C OF DEFINED SCREEN AREA,SIMILARLY WITH Y0 TO Y1
      COMMON/AREA/SCALE(12)
      IF(X0.GE.X1.OR.Y0.GE.Y1) GOTO 1
      SCALE(5)=X0
      SCALE(6)=X1
      SCALE(7)=Y0
      SCALE(8)=Y1
      SCALE(9)=X1-X0
      SCALE(10)=Y1-Y0
      RETURN
1     CALL CURPOS(200.,500.)
      CALL ALPHMD
      CALL MESSAG("DRAWING OFF THE SPECIFIED WINDOW^")
      RETURN
      END
C*************SET THE DISPLAY IN ALPHA MODE****************
C
        SUBROUTINE ALPHMD
C OUTPUT A SINGLE ASCII CHARACTER
        S=OCHAR(31)
        RETURN
        END
C
C**************PUT THE TERMINAL IN GRAPHIC MODE*************
C
        SUBROUTINE GRPHMD
        COMMON/REMXY/IXX,IYY
C RETURN TO PREVIOUS POSITOIN AND SET DISPLAY TO GRAPHIC MODE
        CALL VPLOT(0,IXX,IYY)
        RETURN
        END
```

```
C                        ****************
C                        * APPENDIX 1.3 *
C                        ****************
C
C POINT AND LINE DRAWING ROUTINES
C ------------------------------
C
C
C
C ************** DRAW A DARK OR BRIGHT VECTOR***********************
C
        SUBROUTINE VPLOT(I,IX,IY)
C THIS IS DEVICE DEPENDENT ROUTINE
        COMMON/OLDBT/IOLDBT(4)
        DIMENSION IV3(4)
C CONVERT COORDINATES X,Y INTO 4 BYTES INFORMATION
        IX1=IX/32
        IY1=IY/32
C SET CHARACTER STRING FOR GRAPHIC
        IV3(1)=32+IY1
        IV3(2)=96+IY-32*IY1
        IV3(3)=32+IX1
        IV3(4)=64+IX-32*IX1
C DARK / BRIGHT VECTOR?
        IFLG=0
        IF(I.EQ.0)GOTO 3
C HIGH Y
        IF(IV3(1).NE.IOLDBT(1))S=OCHAR(IV3(1))
C LOW Y
        IF(IV3(2).EQ.IOLDBT(2)) GOTO 4
        IFLG=1
        S=OCHAR(IV3(2))
C HIGH X
4       IF(IV3(3).EQ.IOLDBT(3)) GOTO 5
C LOW Y SENT?
        IF(IFLG.EQ.1) GOTO 6
C SEND LOW Y & HIGH X
        S=OCHAR(IV3(2))
6       S=OCHAR(IV3(3))
5       S=OCHAR(IV3(4))
        GOTO 11
3       S=OCHAR(29)
        DO 2 J=1,4
C OUTPUT POINT COORDINATES
2       S=OCHAR(IV3(J))
11      DO 7 K=1,4
7       IOLDBT(K)=IV3(K)
        RETURN
        END
C
C**************SET CHARACTER POSITION ON THE SCREEN***************
C
      SUBROUTINE XVPLOT(I,X,Y)
C MOVE(I=0) OR DRAW(I=1) TO (X,Y)
C AND ALSO SETS (LPOS,CHPOS) CHARACTER POSITION ON THE SCREEN
      COMMON/AREA/SCALE(12)
      COMMON/MNGRPH/MODE,LPOS,CHPOS
      COMMON/REMXY/IXX,IYY
      INTEGER CHPOS
C WINDOWING
      IX=SCALE(11)*(X-SCALE(5))/SCALE(9)+SCALE(1)
```

```
      IY=SCALE(12)*(Y-SCALE(7))/SCALE(10)+SCALE(2)
      CALL XYVOCH(IX,IY,LPOS,CHPOS,IA,IB)
      MODE=1
C CHECKS THE COORDINATES X,Y NOT OFF THE SCREEN
      DIF1=X-SCALE(5)
      DIF2=X-SCALE(6)
      DIF3=Y-SCALE(7)
      DIF4=Y-SCALE(8)
      IF(DIF1.LT.0..OR.DIF2.GT.0..OR.DIF3.LT.0..OR.DIF4.GT.0.)GOTO 2
1     IF(I.EQ.0) GOTO 3
      IXX=IX
      IYY=IY
3     CALL VPLOT(I,IX,IY)
      RETURN
C OUTPUT WARRING MESSAGE
2     CALL VPLOT(0,10,650)
      CALL ALPHMD
      WRITE(6,60)
60    FORMAT("COORDINATES OFF SCREEN")
      IF(I.EQ.0) GOTO 3
      CALL GRPHMD
      GOTO 1
      END
C
C*************** MOVE THE BEAM TO ABSOULTE X,Y**************
C
        SUBROUTINE TXMOVE(X,Y)
        COMMON/RXY/RX,RY
C RESET COMMON VARIABLES
        RX=X
        RY=Y
C OUTPUT DARK VECTOR
        CALL XVPLOT(0,X,Y)
        RETURN
        END
C
C*********** MOVE RELATIVE ******************************
C
        SUBROUTINE TXMOVR(DX,DY)
        COMMON/RXY/RX,RY
C UPDATE COMMON VARIABLES FOR RELATIVE MODE
        X=RX+DX
        Y=RY+DY
        RX=X
        RY=Y
C OUTPUT DARK VECTOR OF LENGTH DX,DY
        CALL XVPLOT(0,X,Y)
        RETURN
        END
C
C **************DRAW ALINE TO X,Y FROM CURRENT BEAM POSITION*********
C
        SUBROUTINE TXDRAW(X,Y)
        COMMON/RXY/RX,RY
C RESET COMMON VARIABLES
        RX=X
        RY=Y
C OUTPUT BRIGHT VECTOR
        CALL XVPLOT(1,X,Y)
        RETURN
        END
C
```

```
C ************ DRAWS A LINE IN RELATVE MODE *********************
C
      SUBROUTINE TXDRWR(DX,DY)
      COMMON/RXY/RX,RY
C UPDATE COMMON VARIABLES FOR RELATIVE MODE
      X=RX+DX
      Y=RY+DY
      RX=X
      RY=Y
C OUTPUT BRIGHT VECTOR OF LENGTH DX,DY
      CALL XVPLOT(1,X,Y)
      RETURN
      END
```

```
C                         ****************
C                         * APPENDIX 1.4 *
C                         ****************
C
C CHARACTER AND TEXT HANDLING ROUTINES
C -----------------------------------
C
C
C
C
C************* GET A SINGLE CHARACTER FROM THE SCREEN******************
C
      SUBROUTINE TXGET(ICHAR)
C DELIVER A SINGLE CHARACTER FROM SCREEN EXPANDING
C ANY ABBREVIATION AS REQUESTED IN /ABBREV/
      COMMON/ABBREV/NSPEC,FSTCH,PTRS(1)
      INTEGER PTRS,P,COLON,SPECHD,FSTCH
      INTEGER*1  C(1)
      EQUIVALENCE(C(1),NSPEC)
      DATA P,COLON/0,58/
      IF(P.EQ.0)GOTO 1
C P NONZERO MEANS DELIVER NEXT CHAR FROM ABBREV(STOP AT :)
4     ICHAR=C(P)
      P= P+1
      IF(ICHAR.NE.COLON) GOTO 99
C REACHED COLON
      P =0
C NORMAL USE INCHR
1     ICHAR=INCHAR(X)
      IF(NSPEC.EQ.0) GOTO 99
      DO 2 J=1,NSPEC
2     IF  (C(FSTCH+T).EQ.ICHAR)GOTO 3
C NO MATCH
99    ICHAR=IREM(ICHAR,128)
      RETURN
3     P=PTRS(J)
      GOTO 4
      END
C
C ************OUTPUT ASINGLE CHARACTER TO THE SCREEN******X**********
C
        SUBROUTINE TXPUT(ICHAR)
C     SEND C TO THE SCREEN,TAB IS INTEPRETED AS A SUITABLE NUMBER
C     OF SPACES RUBOUT IS PRINTED AS FULL BLOCK OF DOTS
      COMMON/MNGRPH/MODE,LPOS,CHPOS
      COMMON/IO/IN,IOUT
      INTEGER CR,FF,TAB,RUBOUT,SPACE,BACKSP,A(4),US,CHPOS
      DATA CR,FF,TAB,RUBOUT,SPACE/13,12,9,127,32/
      DATA BACKSP,A(1),A(2),A(3),A(4),US,LF/8,36,73,88,72,31,10/
      I=ICHAR
      IF(MODE.NE.0) S=OCHAR(US)
      MODE=0
      I=IREM(I,128)
      IF(I.EQ.BACKSP)CHPOS=CHPOS-2
      IF(I.NE.FF) GOTO1
      LPOS=0
      GOTO2
1     IF(I.NE.CR) GOTO 3
2     CHPOS=-1
3     IF(I.NE.LF) GOTO 31
      CHPOS=-1
```

```
      GOTO 31
100   LPOS=LPOS+1
      CHPOS=CHPOS-1
C     TEST IF ON SCREEN
31    IF(LPOS.GT.34.OR.CHPOS.GE.74) WRITE(IOUT,10)
10    FORMAT("CHAR.OFF THE SCREEN")
5     CHPOS=CHPOS+1
      IF(I.NE.TAB) GOTO 4
C     TAB TO NEXT MULTIPLE OF 8
      S=OCHAR(SPACE)
      IF(MOD(CHPOS,8).NE.0) GOTO 5
      GOTO 99
4     IF(I.NE.RUBOUT) GOTO 6
      DO 21 I=1,4
        S=OCHAR(A(I))
        S=OCHAR(BACKSP)
21    CONTINUE
      I=SPACE
6     S= OCHAR(I)
C     GENERATE A LF AFTER A CR (LF'S ARE OTHERWISE IGNORED)
      IF(I.NE.CR) GOTO 99
      I=LF
      GOTO100
99    RETURN
      END
C
C************INPUT A LINE OF TEXT FROM THE SCREEN***********
C
        SUBROUTINE TXLINE(STRING,N)
C INPUT ALINE OF CHARACTER FROM THE SCREEN INTO STRING(N)
C ECHO AND DEAL WITH RUBOUTS
C END OF INPUT WITH LF ,CR,EOT
C STRING GET ALL CHARS INPUT INCLUDE TERMINATOR
        INTEGER*1 STRING(N)
C PTRS IS USED TO HOLD APPARENT OFFSET OF CORRECT CHAR
        INTEGER PTRS(72),CR,BACKSP,RUBOUT,SPACE,LF,EOT,TAB
        DATA CR,BACKSP,RUBOUT,SPACE,LF,EOT,TAB
&                /13,8,127,32,10,4,9/
        NECHO=0
        NEXTIN=1
C MAIN LOOP
100     CALL TXGET(K)
        IF(K.EQ.RUBOUT) GOTO 101
        STRING(NEXTIN)=K
        PTRS(NEXTIN)=NECHO
        NEXTIN=NEXTIN+1
        IF(K.EQ.CR.OR.K.EQ.EOT.OR.K.EQ.LF) GOTO 99
        IF(K.EQ.TAB) GOTO 102
C ORDINARY CHARARCTER,ECHO AND LOOP(IF ENOUGH SPACE)
        NECHO=NECHO+1
        CALL TXPUT(K)
        IF(NEXTIN.LT.MIN0(N+1,72)) GOTO 100
C FINISHED FOR SOME REASON
99      STRING(NEXTIN)=0
        RETURN
C TAB-LOOP OUTPUTTING SPACES
102     NECHO=NECHO+1
        CALL TXPUT(SPACE)
        IF(MOD(NECHO,8).NE.0) GOTO 102
        GOTO 100
C RUBOUT
101     IF(NEXTIN.EQ.1) GOTO 100
```

```
      NEXTIN=NEXTIN-1
      NBACK=NECHO-PTRS(NEXTIN)
      DO 91 I=1,NBACK
91    CALL TXPUT(BACKSP)
      CALL TXPUT(RUBOUT)
      CALL TXPUT(BACKSP)
      DO 92 I=1,NBACK
92    CALL TXPUT(SPACE)
      GOTO 100
      END
C
C ************OUTPUT A GIVEN MESSAGE(END IN ZERO CHARACTER)*************
C
      SUBROUTINE MESSAG(TEXT)
C START THE MESSAGE WITH % FOR OUTPUTING TO THE NEXT LINE
C TERMINATE THE TEXT WITH '^' AS THE END OF THE MESSAGE
      COMMON/MNGRPH/MODE,LPOS,CHPOS
      COMMON/IO/IN,IOUT
      INTEGER*1 TEXT(1)
      INTEGER PCENT,HAT,CHPOS
      DATA LF,PCENT,HAT/10,37,94/
      CHPOS=-1
      DO 1 I=1,129
        J=TEXT(I)
        IF(J.EQ.HAT) RETURN
        IF(J.EQ.PCENT)J=LF
 1      CALL TXPUT(J)
C ATTEMPT TO OUTPUT A MESSAGE MORE THAN 128 CHARACTERS
      WRITE(IOUT,10)
10    FORMAT("ATTEMPT TO OUTPUT A MESSAGE MORE THAN 128 CHARACTERS")
      RETURN
      END
C
C***********OUTPUT TEXT UP ON THE SCREEN FROM A FILE**************
C
C THIS  DISPLAY TEXT FROM THE FILE NAMED
C A HOLLERITH STRING AS FOR SETFIL
      SUBROUTINE TEXTUP(FILE,N)
      COMMON/IO/IN,IOUT
      INTEGER*1 FILE(1)
      INTEGER BUFFER(72)
      REWIND 7
      CALL SETFIL(7,FILE)
C LOOP READING FROM FILE
2     DO 1 I=1,N
        READ(7,110) BUFFER
110     FORMAT(72A1)
        WRITE(IOUT,110)BUFFER
1     CONTINUE
      ENDFILE 7
      RETURN
      END
C
C*****************GET AN INTEGER FROM THE SCREEN***************
C
      SUBROUTINE INTGET(I)
C OBTAIN THE NEXT INTEGER FROM SCREEN
      INTEGER*1  LINE(8)
      INTEGER CH0,CH9
      DATA CH0,CH9/48,57/
      CALL TXLINE(LINE,8)
      CALL SPOUT(LINE)
```

```
      I=0
      DO 10 K=1,8
        J=LINE(K)
        IF(J.LT.CH0.OR.J.GT.CH9) GOTO 99
        J=J-CH0
        IF(I-32767)10,11,12
11      IF(J.GE.7) GOTO 12
10      I=I*10+J
12    I=32767
99    RETURN
      END
C
C***********REMOVE SPACES FROM STRING*********************
C
      SUBROUTINE SPOUT(STRING)
C     ITS REMOVES SPACES AND OTHER NON-PRINTING CHARS FROM STRING
C     SUCH AS LINE FEEDS,CARRIAGE RETURNS,AND EOT'S FROM TEXT
      INTEGER*1 STRING(1)
      J=0
      DO 1 I=1,100
2       J=J+1
        K=STRING(J)
        IF(K.LT.0)K=K+128
        IF(K.EQ.32)GOTO 2
        IF(K.EQ.10.OR.K.EQ.13.OR.K.EQ.4)GOTO 2
        STRING(I)=K
1       IF(K.EQ.0) GOTO 3
3     RETURN
      END
C
C ****************OUTPUT CHARACTER STRING AT GIVEN COORDINATES*******
C
      SUBROUTINE DTEXT(X,Y,TEXT,N)
      LOGICAL*1 TEXT(N)
      CALL TXMOVE(X,Y)
      CALL ALPHMD
C OUTPUT THE STRING OF CHARACTERS IN TEXT
      WRITE(6,10)TEXT
10    FORMAT(72A1)
      RETURN
      END
```

```
C                        ***************
C                        * APPENDIX 1.5 *
C                        ***************
C
C CURSOR AND MENU OPERATIONS ROUTINES
C ------------------------------------
C
C
C
C **************ERASE CROSS HAIRS CURSOR*******************X******
C
        SUBROUTINE CURSET(X,Y,ICHAR)
C RETURNS X,Y COORDINATES AND CHARACTER TYPED
        S=CURSON(ICHAR,IHIX1,ILOX1,IHIY1,ILOY1)
        X=32*(IHIX1-32)+ILOX1-32
        Y=32*(IHIY1-32)+ILOY1-32
        RETURN
        END
C
C*************SET UP THE CURSOR AND RETURN A CHARQCTER*************
C
      SUBROUTINE TXCURS(X,Y,ICHAR)
C THIS DISPLAYS THE CROSS-HAIR CURSOR ON THE SCREEN
C THEN RETURNS THE SCALED VALUE OF THE COORDINATES OF
C THE CURSOR IN X,Y AND THE CHARACTER TYPED IN,PUT IN ICHAR
      COMMON/MNGRPH/MODE,LPOS,CHPOS
      INTEGER CHPOS
      COMMON/AREA/SCALE(12)
1     MODE=1
      CALL CURSET (X,Y,ICHAR)
      ICHAR=IREM(ICHAR,128)
C CHECK CURSOR LOCATION ON THE SCREEN,IF OUTSID LIMIT GOBACK
      IF(X.LT.SCALE(1).OR.X.GT.SCALE(3).OR.Y.LT.SCALE(2).OR.Y.GT.
&     SCALE(4)) GOTO 2
C
C COMPUTE SCALED VALUE
      X=SCALE(5)+(X-SCALE(1))*SCALE(9)/SCALE(11)
      Y=SCALE(7)+(Y-SCALE(2))*SCALE(10)/SCALE(12)
      IX=X
      IY=Y
      GOTO 3
2     CALL ALPHMD
      CALL  MESSAG("ILLEGAL CURSOR POSITION^")
      GOTO 1
3     CALL XYVOCH(IX,IY,LPOS,CHPOS,IA,IB)
      RETURN
      END
C
C*************POSITION THE ALPHA CURSOR*************************
C
        SUBROUTINE CURPOS(X,Y)
        CALL TXMOVE(X,Y)
C SET THE DISPLAY TO ALPHA MODE
        CALL ALPHMD
        RETURN
        END
C
C ************** OPEN A NEW MENU****************
C
      SUBROUTINE MNOPEN(X,Y,MNO)
C SET UP ORIGIN FOR MENU,WHICH ANNOUCES THAT A MENU IS
```

```
C TO BE DISPLAYED WHOSE TOP LEFTHAND CORNER IS TO BE AT SCREEN
C COORDINATES (X,Y) AND MENU NUMBER MON
      COMMON/MENUDA/XORIG(3),YORIG(3),STEP,NLINES(3)
      IF(MNO.EQ.1) YORIG(2)=0.
      XORIG(MNO)=X
      YORIG(MNO)=Y
      STEP=22.
      NLINES(MNO)=0
      RETURN
      END
C
C *************OUTPUT LINE OF MENU TEXT****************
C
        SUBROUTINE MNTEXT(TEXT,N,MNO)
C PUT OUT THIS TEXT AS THE NEXT LINE OF A MENU
        COMMON/MENUDA/XORIG(3),YORIG(3),STEP,NLINES(3)
C DEFINED BYTE ARRAY
        LOGICAL*1 TEXT(N)
C MOVE TO THE DESIRED LOCATION
        CALL TXMOVE(XORIG(MNO),YORIG(MNO)-NLINES(MNO)*STEP)
C SET TO ALPHA MODE
        CALL ALPHMD
C
C OUTPUT MENUE ITEM
        WRITE(6,10) TEXT
10      FORMAT(72A1)
        NLINES(MNO)=NLINES(MNO)+1
        RETURN
        END
C
C ************* PICK AN ITEM FROM A MENU****************************
C
      SUBROUTINE MNPICK(I,ICHAR,MNO)
C SETS I TO THE INDEX OF MENU ITEM CHOSEN,ICHAR TO THAT TYPED
      COMMON/MENUDA/XORIG(3),YORIG(3),STEP,NLINES(3)
 1    CALL TXCURS(X1,Y1,ICHAR)
      IF(Y1.GT.YORIG(2)+10) GOTO 5
      MNO=2
      GOTO 3
5     MNO=1
3     RPOS=YORIG(MNO)-Y1+14
      I=(RPOS+STEP)/STEP
      J=MOD(RPOS,STEP)
      IF(J.LT.0)J=J+STEP
      IF(I.LE.0.OR.I.GT.NLINES(MNO).OR.J.GT.14) GOTO 1
C
C MARK THE MENU ITEM WITH ARROW
      CALL TXMOVE(XORIG(MNO)-40.,Y1)
      CALL TXDRAW(XORIG(MNO),Y1)
      CALL TXDRAW(XORIG(MNO)-20.,Y1+10.)
      CALL TXMOVE(XORIG(MNO),Y1)
      CALL TXDRAW(XORIG(MNO)-20.,Y1-10.)
2     RETURN
      END
C
C **********DISPLAY A COMPLETE MENU AT PRE ASSIGNED ORIGIN**********:
C
        SUBROUTINE MNDISP(TEXT,ITEM,LEN,MNO)
        LOGICAL*1 TEXT(1)
        K=0
        DO 77 I=1,ITEM
C OUTPUT AN ITEM OF THE MENU
```

```
        CALL MNTEXT(TEXT(I+K),LEN,MNO)
        K=K+LEN-1
77      CONTINUE
        RETURN
        END
C
C *********** DRAW FRMAE ROUND THE TEXT MENUE**********************
C
        SUBROUTINE FRAME(X1,Y1,NC)
C DRAWS RECTANGLE ROUND THE MENU
        CALL TXMOVE(X1,Y1)
        CALL TXDRAW(X1+145,Y1)
        CALL TXDRAW(X1+145,Y1-22*NC)
        CALL TXDRAW(X1,Y1-22*NC)
        CALL TXDRAW(X1,Y1)
        RETURN
        END
C
C***************** SET LINE AND CHAR POSITION***************
C
      SUBROUTINE XYVOCH(IX,IY,NLINE,NCHAR,IA,IB)
C THIS SETS (NLINE,NCHAR) AND (IA,IB) TO THE CHARACTER
C INDICATED BY (X,Y) AND THE OFFSET RELATIVE TO ITS BLH CORNER
      COMMON/TOPLHC/IXORIG,IYORIG
      IA=MOD(IX-IXORIG,14)
      IF(IA.LT.0)IA=IA+14
      IB=MOD(IY-IYORIG,22)
      IF(IB.LT.0)IB=IB+22
      NCHAR=(IX-IXORIG)/14
      NLINE=(IYORIG-IY+21)/22
      RETURN
      END
```

```
C                       ****************
C                       * APPENDIX 1.6 *
C                       ****************
C
C BASIC TRANSFORMATION ROUTINES
C -----------------------------
C
C
C
C**********SCALE UP OR DOWN THE DATA******************
C
        SUBROUTINE SCALNG(SX,SY,SZ)
        COMMON/MATRIX/CTM(4,4),TM(4,4)
        DIMENSION TT(4,4)
C SETS UP UNIT MATRIX
        CALL UNITY (TT)
C RESET THE APPROPRIATE MATRIX ELEMENTS TO THE SPECIFIED PARAMETERS
        TT(1,1)=SX
        TT(2,2)=SY
        TT(3,3)=SZ
C UPDATE THE TRANSFORMATION MATRIX
        CALL CONCAT(TM,TT,4)
        RETURN
        END
C
C***************TRANSLATION ROUTINE (2D & 3D)*****************
C
        SUBROUTINE TRANSL(TX,TY,TZ)
        COMMON/MATRIX/CTM(4,4),TM(4,4)
        DIMENSION TT(4,4)
C SETS UP UNIT MATRIX
        CALL UNITY (TT)
C RESET THE APPROPRIATE MATRIX ELEMENT TO THE SPECIFIED PARAMETERS
        TT(4,1)=TX
        TT(4,2)=TY
        TT(4,3)=TZ
C UPDATE THE TRANSFORMATION MATRIX
        CALL CONCAT(TM,TT,4)
        RETURN
        END
C
C***********ROTATE A DATA POINT ABOUT-X,Y,Z IN THIS ORDER***********
C
        SUBROUTINE ROTATE(RX,RY,RZ)
        COMMON/MATRIX/CTM(4,4),TM(4,4)
        DIMENSION TT(4,4)
C CONVERT FROM DEGREE TO RADIANS
        DIVS=57.2957795
        RX=RX/DIVS
        RY=RY/DIVS
        RZ=RZ/DIVS
C SET THE ANGLES IN RADIANS
        A1= RY+RZ
        B1= RY-RZ
        A2= RX+RZ
        B2= RX-RZ
        A3=RX+RY
        B3=RX-RY
        C=RX+RZ+RY
        D=RX+RZ-RY
        E=RX-RZ+RY
```

```
      F=RX-RZ-RY
C
C SET THE ELEMENTS OF THE ROTAION MATRIX
      TT(1,1)=0.5*(COS(A1)+COS(B1))
      TT(1,2)=0.5*(SIN(A1)-SIN(B1))
      TT(1,3)=-SIN(RY)
      TT(1,4)=0.
      TT(2,1)=0.25*(COS(C)-COS(D)+COS(E)-COS(F))-0.5*(SIN(A2)-SIN(B2))
      TT(2,2)=-0.25*(SIN(D)-SIN(C)+SIN(E)-SIN(F))+0.5*(COS(A2)+COS(B2))
      TT(2,3)=0.5*(SIN(A3)+SIN(B3))
      TT(2,4)=0.
      TT(3,1)=0.25*(SIN(C)-SIN(D)+SIN(E)-SIN(F))+0.5*(COS(A2)-COS(B2))
      TT(3,2)=0.25*(COS(C)-COS(D)-COS(E)+COS(F))-0.5*(SIN(A2)+SIN(B2))
      TT(3,3)=0.5*(COS(A3)+COS(B3))
      TT(3,4)=0.
      TT(4,1)=0.
      TT(4,2)=0.
      TT(4,3)=0.
      TT(4,4)=1.
      CALL CONCAT(TM,TT,4)
      RETURN
      END
C
C*********CLIPS ALINE TO SPECIFIED WINDOW BOUNDARIES************
C
C                DISPLAY FILE
      SUBROUTINE CLIP(CLINE,X0,Y0,X1,Y1,IREJ)
      COMMON/LIMIT/S1,S2,S3,S4
      COMMON/LINES/SUBLIN(3,2)
      DIMENSION CLINE(2,2),ISEG(2)
      INTEGER ENDPNT,TWOSEG
C
C SET THE LIMIT----INITIALISATION
      ENDPNT=0
      S1=X0
      S2=Y0
      S3=X1
      S4=Y1
C
C TEST FOR FULLY REJECTION/ACCEPTANCE---FIRST TIME
      XX1=CLINE(1,1)
      YY1=CLINE(1,2)
      XX2=CLINE(2,1)
      YY2=CLINE(2,2)
      I=IREJCT(XX1,YY1,XX2,YY2)
      IF(I.NE.1) GOTO 1
C
C FULLY REJECTED
      IREJ=0
      RETURN
C
C TEST FOR ACCEPTANCE
1     J=JACCPT(XX1,YY1,XX2,YY2)
      IF(J.NE.1) GOTO 2
C
C FULLY ACCEPTED
      IREJ=1
      RETURN
C
C NOT FULLY ACCEPTED----SUBDIVIED THE LINE
2     CALL SUBDIV(XX1,YY1,XX2,YY2,ISEG)
      GOTO (11,12,13),ISEG(1)
```

```
C
C TEST FOR EACH CONDITION
C FIRST SEGMENT REJECTED
11      GOTO(104,14,15),ISEG(2)
C
C SECOND SEGMENT ACCEPTED FULLY
14      IF(ENDPNT.EQ.0) GOTO 16
17      CLINE(1,1)=SUBLIN(3,1)
        CLINE(1,2)=SUBLIN(3,2)
177     IF(TWOSEG.EQ.2) GOTO 31
        TWOSEG=0
        IREJ=1
        RETURN
16      CLINE(1,1)=SUBLIN(2,1)
        CLINE(1,2)=SUBLIN(2,2)
        CLINE(2,1)=SUBLIN(3,1)
        CLINE(2,2)=SUBLIN(3,2)
        GOTO 717
        ENDPNT=1
        GOTO 17
C
C NOT FULLY ACCEPTED
15      XX1=SUBLIN(2,1)
        YY1=SUBLIN(2,2)
        XX2=SUBLIN(3,1)
        YY2=SUBLIN(3,2)
        GOTO 2
104     IF(ENDPNT.EQ.11) GOTO 188
        IF(TWOSEG.EQ.1) GOTO 188
        CLINE(1,1)=SUBLIN(3,1)
        CLINE(1,2)=SUBLIN(3,2)
        IF(TWOSEG.EQ.2) GOTO 31
        IF(ENDPNT.NE.0) GOTO 177
        IREJ=0
        RETURN
188     CLINE(2,1)=SUBLIN(1,1)
        CLINE(2,2)=SUBLIN(1,2)
        GOTO 177
707     IF(TWOSEG.EQ.2) GOTO 31
        IREJ=0
        RETURN
C
C TEST FIRST SEGMENT FULLY ACCEPTED
12      IF(ISEG(2).NE.1) GOTO 25
        IF(ENDPNT.NE.0)  GOTO 19
        CLINE(1,1)=SUBLIN(1,1)
        CLINE(1,2)=SUBLIN(1,2)
19      CLINE(2,1)=SUBLIN(2,1)
        CLINE(2,2)=SUBLIN(2,2)
        IF(TWOSEG.EQ.2) GOTO 31
717     IREJ=1
        RETURN
25      IF(ENDPNT.EQ.0) GOTO 35
        CLINE(2,1)=SUBLIN(2,1)
        CLINE(2,2)=SUBLIN(2,2)
        GOTO 15
35      CLINE(1,1)=SUBLIN(1,1)
        CLINE(1,2)=SUBLIN(1,2)
        ENDPNT=11
        GOTO 15
C
C TEST FIRST SEGMENT NOT FULLY ACCEPTED
```

```
13      IF(ISEG(2).NE.1) GOTO 21
18      XX1=SUBLIN(1,1)
        YY1=SUBLIN(1,2)
        XX2=SUBLIN(2,1)
        YY2=SUBLIN(2,2)
        GOTO 2
21      IF(ISEG(2).EQ.2) GOTO 41
        TWOSEG=2
        XX3=SUBLIN(2,1)
        YY3=SUBLIN(2,2)
        XX4=SUBLIN(3,1)
        YY4=SUBLIN(3,2)
        GOTO 18
31      TWOSEG=1
        XX1=XX3
        YY1=YY3
        XX2=XX4
        YY2=YY4
        GOTO 2
41      IF(ENDPNT.EQ.1) GOTO 88
        ENDPNT=1
        CLINE(2,1)=SUBLIN(3,1)
        CLINE(2,2)=SUBLIN(3,2)
        GOTO 18
88      CLINE(1,1)=SUBLIN(3,1)
        CLINE(1,2)=SUBLIN(3,2)
        GOTO 18
        END
C
C***************SUBDIVID THE LINES AT ITS MIDPOINT***************
C
        SUBROUTINE SUBDIV(X1,Y1,X2,Y2,IS)
        COMMON/LINES/SUBLIN(3,2)
        DIMENSION IS(2)
C SET THE SUBDIVIDED ARRAY LINE
        SUBLIN(1,1)=X1
        SUBLIN(1,2)=Y1
        SUBLIN(2,1)=0.5*(X1+X2)
        SUBLIN(2,2)=0.5*(Y1+Y2)
        SUBLIN(3,1)=X2
        SUBLIN(3,2)=Y2
C TEST SEGMENTS FOR REJECTION/ACCEPTANCE
        DO 1 K=1,2
        I1=IREJCT(SUBLIN(K,1),SUBLIN(K,2),SUBLIN(K+1,1),SUBLIN(K+1,2))
        IF(I1.EQ.0) GOTO 7
C COMPLETE REJECTION OF A LINE
        IS(K)=1
        GOTO 1
C TEST LINE FOR ACCEPTANCE
7       J1=JACCPT(SUBLIN(K,1),SUBLIN(K,2),SUBLIN(K+1,1),SUBLIN(K+1,2))
        IF(J1.EQ.0) GOTO 8
C LINE  FULLY ACCEPTED
        IS(K)=2
        GOTO 1
C LINE NOT FULLY ACCEPTED
8       IS(K)=3
C NEXT SEGMENT
1       CONTINUE
        RETURN
        END
C
C***************TEST CONDITION FOR LINE REJECTION****************
```

```
C
        FUNCTION IREJCT(X1,Y1,X2,Y2)
        COMMON/LIMIT/S1,S2,S3,S4
        LOGICAL A1,B1,AA,BB,CC,DD
        N=0
        DIF1=X1-S1
        DIF2=X2-S1
        DIF3=X1-S3
        DIF4=X2-S3
        A=ABS(X2-X1)
1       A1=DIF1.LT.0..AND.DIF2.LT.0
        B1=DIF3.GT.0..AND.DIF4.GT.0.
        AA=DIF1.LT.0..AND.A.LT.0.005
        BB=DIF2.LT.0..AND.A.LT.0.005
        CC=DIF3.GT.0..AND.A.LT.0.005
        DD=DIF4.GT.0..AND.A.LT.0.005
C TEST FOR LINE COMPLETELEY OUTSIDE THE LIMIT/WITHIN 0.005
C TOLERANCE
        IF(A1.OR.B1.OR.AA.OR.BB.OR.CC.OR.DD) GOTO 2
        N=N+1
        IF(N.EQ.2) GOTO 3
C SAME TEST FOR Y
        DIF1=Y1-S2
        DIF2=Y2-S2
        DIF3=Y1-S4
        DIF4=Y2-S4
        A=ABS(Y2-Y1)
        GOTO 1
C NOT REJECTED
3       IREJCT=0
        RETURN
C FULLY REJECTED
 2      IREJCT=1
        RETURN
        END
C
C***************TEST FOR ACCEPTANCE OF A LINE**********************
C
        FUNCTION JACCPT(X1,Y1,X2,Y2)
        COMMON/LIMIT/S1,S2,S3,S4
        LOGICAL A,B
        X11=X1-S1
        X22=X2-S1
        X33=X1-S3
        X44=X2-S3
        A=X11.GE.0..AND.X33.LE.0..AND.X22.GE.0..AND.X44.LE.0.
        Y11=Y1-S2
        Y22=Y2-S2
        Y33=Y1-S4
        Y44=Y2-S4
        B=Y11.GE.0..AND.Y33.LE.0..AND.Y22.GE.0..AND.Y44.LE.0.
        IF(A.AND.B) GOTO 1
C NOT FULLY ACCEPTED
        JACCPT=0
        RETURN
C FULLY ACCEPTED
1       JACCPT=1
        RETURN
        END
C
C**************** PRESPECTIVE TRANSFORMATION**********************
C
```

```
      SUBROUTINE PERSP(PX,PY,PZ)
      COMMON/MATRIX/CTM(4,4),TM(4,4)
      DIMENSION TT(4,4)
C SET MATRIX TT TO UNIT MATRIX
      CALL UNITY(TT)
      TT(1,4)=PX
      TT(2,4)=PY
      TT(3,4)=PZ
      CALL CONCAT(TM,TT,4)
      RETURN
      END
C
C****************** PROJECTION TRANSFORMATION******************
C
      SUBROUTINE PROJCT(NPLANE)
      COMMON/MATRIX/CTM(4,4),TM(4,4)
      DIMENSION TT(4,4)
      CALL UNITY(TT)
C PROJECTION PLANE
      GOTO (1,2,3),NPLANE
1     TT(1,1)=0.
      GOTO 4
2     TT(2,2)=0.
      GOTO 4
3     TT(3,3)=0.
4     CALL CONCAT(TM,TT,4)
      RETURN
      END
C
C**************SAVE A MATRIX 4 BY 4 IN THE STACK****************
C
      SUBROUTINE SAVMAT(A)
      COMMON/STACK/ISPNTR,STACK(64)
      COMMON/IO/IN,IOUT
      DIMENSION A(4,4)
      ISTKSZ=64
C PUSH MATRIX ELEMENT INTO THE STACK
      DO 2 J=1,4
         DO 3 I=1,4
          IF(ISPNTR.EQ.ISTKSZ) GOTO 1
          ISPNTR=ISPNTR+1
3         STACK(ISPNTR)=A(I,J)
2      CONTINUE
      RETURN
1     WRITE(IOUT,10)
10    FORMAT("STACK OVERFLOW")
      RETURN
      END
C
C***************RESTORE 4 BY 4 MATRIX FROM THE STACK**************
C
      SUBROUTINE RESTOR(A)
      COMMON/STACK/ISPNTR,STACK(64)
      COMMON/IO/IN,IOUT
      DIMENSION A(4,4)
C POP MATRIX ELEMENTS FROM THE STACK AND SAVE THEM IN A
      DO 2 J=1,4
      L=4-J+1
         DO 3 I=1,4
           K=4-I+1
           IF(ISPNTR.EQ.0) GOTO 1
           A(K,L)=STACK(ISPNTR)
```

```
3            ISFNTR=ISFNTR-1
2       CONTINUE
        RETURN
1       WRITE(IOUT,10)
10      FORMAT("STACK UNDERFLOW")
        STOP
        RETURN
        END
C
C***************SET AGIVEN MARIX TO UNITY**************
C
        SUBROUTINE UNITY(A)
        DIMENSION A(4,4)
        DO 1 I=1,4
            DO 2 J=1,4
              A(I,J)=0.
              IF(I.EQ.J) A(I,J)=1.
2           CONTINUE
1       CONTINUE
        RETURN
        END
C
C****************SET CTM MATRIX TO AGIVEN MATRIX**********
C
        SUBROUTINE SETMAT(A)
        COMMON/MATRIX/CTM(4,4),TM(4,4)
        DIMENSION A(4,4)
        DO 1 I=1,4
           DO 2 J=1,4
2          CTM(I,J)=A(I,J)
1       CONTINUE
        RETURN.
        END
C
C******************PERFORM MATRIX MULTIPLICATION*****************
C
        SUBROUTINE CONCAT(A,B,N)
        DIMENSION A(N,4),B(4,4),TT(4,4)
C POST-MULTIPLY ARRAY A BY B ,AND RETURN THE RESULT IN A
        DO 1 I=1,N
           DO 2 J=1,4
           TT(I,J)=0
             DO 3 K=1,4
3             TT(I,J)=TT(I,J)+A(I,K)*B(K,J)
2          CONTINUE
1       CONTINUE
        DO 5 J=1,4
           DO 6 I=1,N
           A(I,J)=TT(I,J)
6          CONTINUE
5       CONTINUE
        RETURN
        END
```

```
C                        ****************
C                        * APPENDIX 1.7 *
C                        ****************
C
C MISCELLANEOUS ROUTINES
C ----------------------
C
C
C
C ****************OVERLAYS PROGRAM MODULES***************************
C
        SUBROUTINE OVRLAY(FILENM)
        LOGICAL*1 FILENM(10)
        W=OVLAY(FILENM)
        RETURN
        END
C
C **************** IT REMOVES A NAMED FILE FROM CURRENT DIRECTORY*****
C
        SUBROUTINE RMFILE(NAME)
        LOGICAL*1 NAME(10)
        W=FLRM(NAME)
        RETURN
        END
C
C ************** OUTPUT ERROR MESSAGE*******************
C
      SUBROUTINE WERROR(I)
      WRITE(6,1) I
 1    FORMAT("ERROR MESSAGE",I5)
      RETURN
      END
C
C ************ CALCULATE REMINDAR********************************
C
      INTEGER FUNCTION IREM(I,J)
C GIVE POSITIVE REMANDER OF I/J (J)0)
      IREM=MOD(I,J)
      IF(IREM.LT.0) IREM=IREM+J
      RETURN
      END
%
```

APPENDIX 1.8

INTRODUCTORY NOTES ON USING GT42 GRAPHICS SYSTEM AND THE EMULATOR

1.0 GT42 START UP PROCEDURE (i.e. Rom Bootstrap from PDP 11/10 console on the GT42)

1) Check that the interface line is connected to the PDP 11/40.
2) Determine that the GT42 power cord is connected to the appropriate electrical outlet.
3) Turn the console key switch to POWER position.
4) Turn the front panel ON-OFF/BRIGHTNESS switch fully counter clockwise and then ¾ of the way in clockwise direction, the red power indicator should be on at this time.
5) Press the console ENABLE/HALT switch down to halt the computer.
6) Press the spring loaded START switch (on PDP 11/10 console) twice to reset the computer.
7) Place 166000_8 in the SWITCH REGISTER (SR).
8) Press LOAD-ADDRESS to load this address.
9) Return ENABLE/HALT switch to the up-most position.
10) Press START switch. The Run light indicator should be on at this time.

Once the cursor appears on the display screen, the user can proceed to log-on.

2.0 DOWN LINE LOADER (DOWNLL)

In order to load a program in the GT42 the user must be logged on the terminal.

Loading Procedures of the Emulator

To load a program the user has to type the command as shown below:

downll < filename

where filename = name of the program to be loaded down the line.

/LIB/TK4010 :- Tek 4010

This will load the Tektronix 4010 Emulator.

3.0 SHUTTING DOWN GT42

1) Logout on your terminal.
2) Press ENABLE/HALF switch down to HALT the computer.
3) Turn the ON-OFF/BRIGHTNESS to OFF.
4) Turn the console key switch to OFF position.

APPENDIX 1.9

LIGHT USER GUIDE

APPENDIX 1.9

LIGHT - USER GUIDE - (Loughborough Interactive Graphics System for the Tektronix 4010)

Introduction

Graphics is the pictorial representation of information and has been extensively used as a medium of communications in engineering and other disciplines. In this context, we use the term graphics to mean 'Interactive Computer Graphics'.

LIGHT is basically a set of library subroutines which can be called from a FORTRAN application program on the PDP 11/40 computer, equipped with Tektronix 4010 display unit and operating under UNIX operating system. These subroutines provide the coupling between the application program, the Graphics console, and the user. Thus the programmer can generate graphic displays with cross-hair cursor and keyboard interactions within his Fortran program.

The Storage Tube

The storage tube display (Tektronix 4010) enables the user to have relatively economical graphics access to a computer, compared with refreshable displays such as the GT42.

The 4010 is a storage tube display together with a keyboard and character generator. The screen has 1,024 addressable points in the x-direction and 781 in the y-direction. A straight line segment is generated by hardware on the screen by specifying one end-point. In addition to graphic output of data, it is possible to input data by means of a cross-hair cursor. Once the cursor is positioned and any key is pressed, the co-ordinates of the cursor together with the character represented by the key are sent back to the computer. The ON/OFF switch for 4010 is located underneath the keyboard on the right-hand side of the stand.

It is very important, when drawing on the 4010, to ensure that repeated over-drawing of the same point or line is avoided as this will cause permanent damage to the display by burning holes through the screen phosphor.

For more detail on the Tektronix 4010 see Reference [20].

Refreshable Display

LIGHT may also be used with the GT42 in Emulator Mode as a Tektronix. The package was carefully adapted in order to minimise the amount of flickering that may occur, because when the time taken to draw a complete "picture" exceeds 20 milliseconds then the display will flicker noticeably but the program operation is unaffected. Instead of the cross-hair cursor that appears on the Tektronix 4010, a tracking-cross and the light pen would have exactly the same effect and they can be used to simulate the function of the cross-hair cursor on the Tektronix 4010.

For further details on the 'Emulator' see Reference [24] and for GT42 operation(see Appendix.1.8).

Graphics Library Package

The graphics system consists of the 4010 display, the PDP 11/40 operating under UNIX, and a library of Fortran-callable subroutines. All that is required of the user is that he writes his/her application program in UNIX Fortran (a subset of ANSI) and incorporates the appropriate graphics subroutine calls. The "Back-end" modules, which drive the 4010 display and utilise UNIX, do not directly affect the user and are not described here.

The Graphics library subroutines are listed in the following categories:

1. Initialisation
2. Point and line drawing
3. Tranformation and simple perspective projection
4. Character and text handling
5. Cursor and menu operations
6. Miscellaneous routines

The facilities provided by this package extend far beyond the minimal set (typically 1,2 and a function for text display) needed to use the display. They should prove to be useful over a wide range of applications.

1.0 INITIALISATION

Before any drawing can be made on the display terminal, the problem (or "picture") area and the screen (or display) area need to be defined by the user or by default.

1.1 TXOPEN:- Assigns default values for the display viewport and window (0.,0.,1023.,780.) as a rectangle defined by the two corners (0,0) and (1023,780). This must precede any other graphics calls

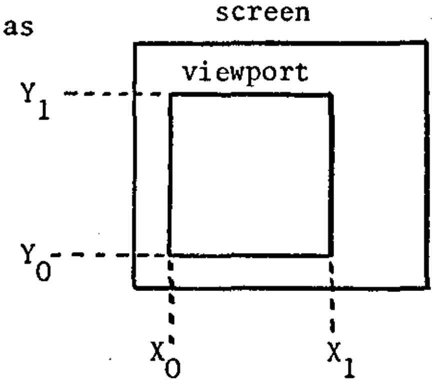
e.g. CALL TXOPEN

1.2 TXCLER:- Clears (erases) the screen ready for the next display picture

e.g. CALL TXCLER

1.3 TXVPRT(XO,YO,X1,Y1):- Sets a display 'Viewport' as the rectangle defined by corners (XO,YO),(X1,Y1) in terms of absolute screen co-ordinates (i.e. $0 \leq X_0 < X_1 \leq 1023$ and $0 \leq Y_0 < Y_1 \leq 780$).

The residual screen area can, of course, be used for other purposes, such as displaying messages and menus.



1.4 TXWIND(X0,Y0,X1,Y1):- Sets a 'window' on the rectangle defined by corners (X_0,Y_0) and (X_1,Y_1) in problem space co-ordinates and maps this area onto the screen viewport.

e.g. CALL TXVPRT(200.,100.,800.,400.)

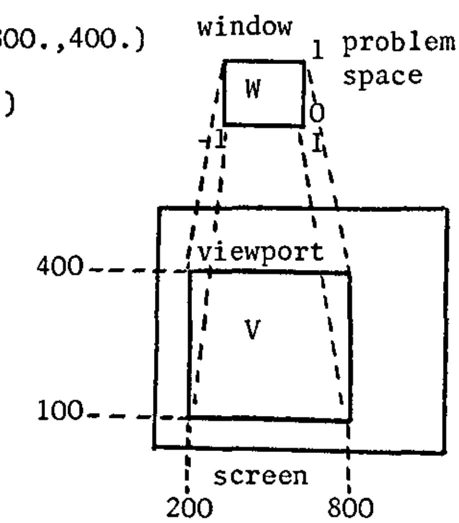
CALL TXWIND(-1,0.,1.,1.)

This maps the window

W $\{-1 \leq X_p \leq 1, 0 \leq Y_p \leq 1\}$

onto the viewport

V $\{200 \leq X_s \leq 800, 100 \leq Y_s \leq 400\}$



1.5 ALPHMD,GRPHMD:-

ALPHMD: sets the display to Alpha mode so that the programmer may display textual information.

GRPHMD: sets the display to Graphic mode. However, as will be seen in section 2.1 a call to TXMOVE will set the display to Graphic mode.

Note: It is important to make sure that before the program is terminated, the display must be set in Alpha mode.

2.0 POINT AND LINE DRAWING

Points and lines form the basic elements of graphic drawing. Separate routines are provided for drawing visible or invisible (Point movement) lines, from the current beam (pen) position to a specified point, or through a given displacement. (These routines may be called the 'graphical primitives', as with aid of these routines the programmer could define his own procedures to draw shapes and symbols that he uses often). The co-ordinates and displacements which are used will normally be specified in the frame of reference of the application problem.

2.1 TXMOVE(X,Y) or TXMOVR(DX,DY):-

TXMOVE(X,Y): Moves the current 'beam' position to the scaled point (X,Y) which is specified as an absolute point.

TXMOVR(DX,DY):Moves the current 'beam' position through the relative displacement DX,DY in x,y direction.

Note: the above two calls will automatically set the terminal to 'Graphic mode', and it is under programmer control to change the mode to 'Alpha mode'. See section 1.5 above.

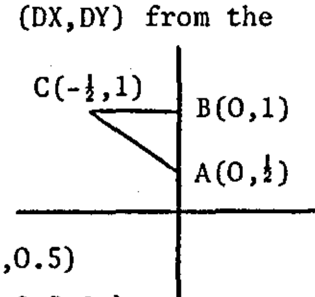
2.2 TXDRAW(X,Y),TXDRWR(DX,DY):-

TXDRAW(X,Y): Draws a visible line from the current 'beam' position to the scaled point (X,Y) in absolute co-ordinates, leaving the beam at X,Y.

TXDRWR(DX,DY):Draws a visible displacement (DX,DY) from the current beam position

e.g. Drawing a Triangle ABC

```
CALL TXMOVE(0.,0.5.)
CALL TXDRAW(0.,1.) or CALL TXDRWR(0.,0.5)
CALL TXDRAW(-0.5,1.) or CALL TXDRWR(-0.5,0.)
CALL TXDRAW(0.,0.5) or CALL TXDRWR(0.5,-0.5)
```



Note: The systematic use of vector increments (i.e. displacement DX,DY) in constructing a picture symbol in the form of a subroutine is useful in displaying a picture with repeated symbols such as logic circuit elements.

3.0 TRANSFORMATION AND SIMPLE PERSPECTIVE PROJECTION

A variety of transformations are provided in this package to make it easy for the programmer to specify and select different views of a picture with different scales and orientations. The set of transformations routines could handle two and three dimension views and are capable of 'scaling', 'translating' and 'rotating' graphical information. They also allow 'clipping' of those parts of a figure which fall outside a previously defined window. Note that for two dimensional transformations the z-parameter must be set to zero.

The start of an object at any display instance is represented by an accumulated transformation matrix TM and a previous reference state is represented by the transformation matrix RTM. Both arrays must be declared by the user in the statement:-

```
COMMON/MATRIX/RTM(4,4),TM(4,4)
```

Perspective views of 3-dimensional objects are useful in certain applications (e.g. architectural drawing). The use of homogeneous co-ordinates (x,y,z,t) to define 3-dimensional objects allows either 'affine' or perspective transformations to be applied with equal ease. The 3-dimensional point (or vector) corresponding to (x,y,z,t) is $(X,Y,Z) = (\frac{x}{t},\frac{y}{t},\frac{z}{t})$. If $t \neq 0$ we usually normalise to $t=1$.

3.1 LINEAR TRANSFORMATION:-

The general linear transformation from point [X, Y, Z] to [X*, Y*, Z*] is represented by

$$[X^* \; Y^* \; Z^* \; 1] = [X \; Y \; Z \; 1] \begin{bmatrix} S_x & C_{12} & C_{13} & 0 \\ C_{21} & S_y & C_{23} & 0 \\ C_{31} & C_{32} & S_z & 0 \\ T_x & T_y & T_z & 1 \end{bmatrix}$$

A user can therefore construct his own transformation(s) to suit his particular application. However, the most common operations which comprise any linear transformation are: scaling, translation, rotation, reflection and shear. These basic operations are available as library subroutines and may be called singly or in a prescribed sequence. Each subroutine call effectively defines a local matrix LTM which performs that particular transformation and is then concatenated in TM.

new TM=TM * LTM

Note: It is under the user's control whether he/she wants to update the RTM by concatenation i.e.

new RTM=RTM * TM

3.1.1 <u>SCALNG(SX,SY,SZ)</u>:- Scales the current point (x,y,z) by the factors SX,SY,SZ.

e.g. CALL SCALNG(0.5,2.0,0.)

causes shrinking in the x-direction and expansion in the y-direction.

In general $$[X^* \; Y^* \; Z^* \; 1] = [X \; Y \; Z \; 1] \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where [X* Y* Z* 1] are the transformed co-ordinates of the point [X Y Z 1].

3.1.2 <u>TRANSL(TX,TY,TZ)</u>:- Translates the current point (X Y Z) through the displacement TX,TY,TZ.

e.g. a Triangle defined by its vertices (20,0), (60,0), (40,100) being translated 100 units to the right and 10 units up,

T_x=100, T_y=10.

```
CALL TRANSL(100.,10.,0.)
```

The resultant transformed points are (120,10), (160,10), (140,110).

3.1.3 ROTATE(RX,RY,RZ):- Rotation is assumed to be positive in a right-hand screw sense as one looks from the origin outward along the axis of rotation. The order in which the rotation is effected is

(1) angle RX about OX

(2) angle RY about OY

(3) angle RZ about OZ

where RX,RY,RZ are specified in degrees.

Note: rotations are not commutative. However, any order may be performed by using separate calls of the routine.

e.g. 45^{o} rotation about OY followed by 30^{o} rotation about OX.

```
CALL ROTATE(0.,45.,0.)
CALL ROTATE(30.,0.,0.)
```

$$[X^* \; Y^* \; Z^* \; 1]=[X \; Y \; Z \; 1] \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\substack{\text{angle } \theta=RX \\ \text{about - OX}}} \underbrace{\begin{bmatrix} \cos\phi & 0 & -\sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ \sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\substack{\text{angle } \phi=RY \\ \text{about -OY}}}$$

$$\times \underbrace{\begin{bmatrix} \cos\psi & \sin\psi & 0 & 0 \\ -\sin\psi & \cos\psi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\substack{\text{angle } \psi=RZ \\ \text{about - OZ}}}$$

3.2 WINDOWING AND VIEWING:-

These basic facilities were described in sections 1.3 and 1.4. However, it is interesting to see that the combined window and viewport definitions constitute a linear (2-dimensional) mapping of the problem area onto the screen and are effectively equivalent to two simple transformations, namely scaling and translation i.e.

$$X_s = S_x X_p + T_x$$
$$Y_s = S_y Y_p + T_y$$

In matrix form (suppressing the z component, which is unchaged)

$$[X_s\ Y_s\ 1]=[X_p\ Y_p\ 1]\begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ T_x & T_y & 1 \end{bmatrix} = [X_p\ Y_p\ 1]\begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ T_x & T_y & 1 \end{bmatrix}$$

where X_s, Y_s are screen co-ordinates

and X_p, Y_p are picture co-ordinates.

e.g. the mapping $W\{-1 \leq X_p \leq 1, 0 \leq Y_p \leq 1\} \rightarrow V\{200 \leq X_s \leq 800, 100 \leq Y_s \leq 400\}$ given in section 1.4 is represented by

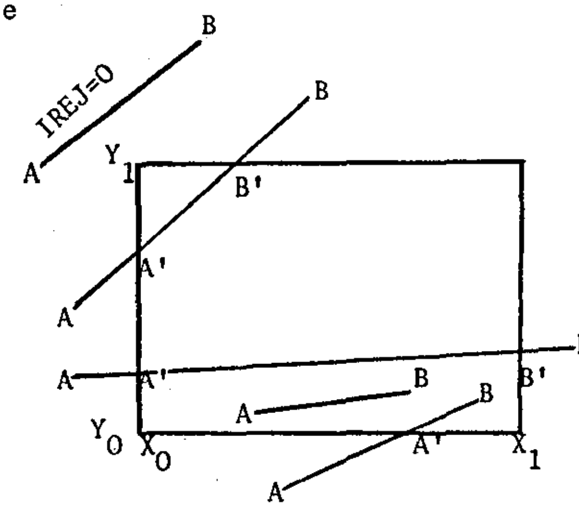
$$X_s = 300\ X_p + 500$$
$$Y_s = 300\ Y_p + 100$$

and the (full) transformation matrix is $\begin{bmatrix} 300 & 0 & 0 & 0 \\ 0 & 300 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 500 & 100 & 0 & 1 \end{bmatrix}$

3.3 CLIP(CLINE,X0,Y0,X1,Y1,IREJ):-

This checks whether the current line segment AB defined by its end points in CLINE (array of 2×2) is cut by the current window boundary defined by X0,Y0,X1,Y1, and if so, returns in CLINE the co-ordinates of the intersection point(s) A'B' as shown.

If AB lies completely outside the window the flag IREJ is set to 0. Otherwise, (AB is completely or partially accepted), IREJ=1. Thus, before using a primitive routine (i.e. MOVE or DRAW) the user normally calls CLIP in order to limit the extent of the picture to a desired window. This routine may be useful in zooming a part of a picture on the viewport chosen on the screen, and consequently enlarging that part.



e.g. CALL CLIP (ALINE,-1.,-1.,1.,1.,IREJ)
where the window limit is given as (-1,-1,1,1).

3.4 SIMPLE PERSPECTIVE PROJECTION:-

In perspective geometry no two lines are parallel. Thus a perspective transformation is frequently associated with a projection onto a plane such as Z=c from a local centre of projection. The combination of perspective transformation with a projective transformation is often called 'perspective projection'. Therefore, a perspective projection represents a transformation from 3-space to 2-space. If the centre of projection is located at infinity, then the perspective projection is called 'Axonometric projection'. This type of projection is commonly used in engineering drawing. For perspective transformation the elements in the last column of the general 4×4 matrix mentioned above are not zero i.e.

$$\begin{bmatrix} 1 & 0 & 0 & P_x \\ 0 & 1 & 0 & P_y \\ 0 & 0 & 1 & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Where P_x,P_y are the reciprocals of the perspective viewing distances from the planes y_z, z_x, x_y respectively.

A perspective projection onto the z=0 "viewing" plane is

$$[x^*\ y^*\ z^*\ t^*]=[x\ y\ z\ 1] \begin{bmatrix} 1 & 0 & 0 & P_x \\ 0 & 1 & 0 & P_y \\ 0 & 0 & 1 & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

projection on z=0 plane.

3.4.1 PERSP(PX,PY,PZ):- A call to this routine would cause the perspective transformation to be performed. However, it can be seen that if the parameters PX,PY, PZ are all set to zero, the transformation matrix becomes a unit matrix, and any subsequent projection would be 'Axonometric projection'.

e.g. CALL PERSP(0.,0.,1.)

i.e. an object is viewed from a position 1 unit on the z-axis. Therefore points on the object at infinity parallel to z-axis are transformed to a finite point on the z-axis.

3.4.2 PROJCT(NPLANE):- This performs the projective transformation from 3-space to 2-space. NPLANE (=1,2 or 3) specifies the co-ordinate plane (x=0, y=0 or z=0) of projection.

e.g. CALL PROJCT(3)

3.5 TRANSFORMATION UTILITY ROUTINES:-

The following routines are provided for manipulating transformation matrices in several different ways:-

(1) Saving and Restoring a 4×4 matrix onto and from a stack respectively.

(2) Initialising a 4×4 matrix to unity or to a previously specified (transformation) matrix.

(3) Concatenating an N×4 matrix with a 4×4 matrix.

3.5.1 SAVMAT(A),RESTOR(A):- A one-dimensional stack is defined internally by the package. These routines can be useful when e.g. a sequence of transformations needs to be interrupted by some new transformations and is subsequently resumed.

e.g. CALL SAVMAT(RTM)

saves the reference transformation matrix (by push-down) on the stack. Subsequently the reference matrix can be restored to its original state by

CALL RESTOR(RTM)

This POPS UP the elements of the matrix from the stack and puts them into the RTM matrix.

3.5.2 UNITY(A),SETMAT(A):-

UNITY(A): Initialises A(4,4) to unit matrix (4×4)

SETMAT(A): Sets RTM (defined in COMMON/MATRIX/...) to A.

3.5.3 CONCAT(A,B,N):- Concatenates matrix (or vector) A(N×4) with matrix B(4×4) by multiplication A*B, and leaves the result in A.

4.0 CHARACTER AND TEXT HANDLING

These routines should help users to incorporate standard keyboard interaction in their Fortran programs.

4.1 SIMPLE INPUT-OUTPUT:-

4.1.1 TXGET(ICHAR):- Inputs the next character from the keyboard into ICHAR. Therefore, ICHAR would contain the ASCII equivalent of the character.

4.1.2 TXPUT(ICHAR):- Outputs ASCII character (ICHAR) to the screen. TAB is interpreted as a suitable number of spaces. RUBOUT is printed as several superimposed characters.

4.2 COMPOSITE INPUT-OUTPUT (TEXT HANDLING):-

4.2.1 TXLINE(STRING,N):- Inputs from the keyboard to the array STRING and echoes to the screen. STRING must be declared as logical*1 array under 'UNIX' Fortran. N is the number of the array elements.
If at any time the user types 'Rubout' the last character not yet deleted is overwritten on the screen and removed from STRING. Input continues until the user types CR, LF or EOT, or until there is no space left in STRING. On return STRING would contain the intended text.

4.2.2 MESSAG(TEXT):- Outputs "TEXT" to the screen, where TEXT is a sequence of hollerith characters terminated by the character '^'. Any occurrences of '/' are replaced by CR/LF (as it is convenient to start a message with '/.).

e.g. CALL MESSAG("HELLO^")

4.2.3 TEXTUP(FILENAME,N):- Displays the whole text of N lines from the named file.

4.2.4 INTGET(I):- Inputs a (base 10 integer) number. (This uses TXLINE to input and echo text). I is then set to the binary equivalent of this integer.

Conversion stops whenever a non-numeric digit is met.

4.2.5 SPOUT(TEXT):- Removes spaces, line feeds, carriage returns, and EOT's from the TEXT parameter.

4.2.6 DTEXT(X,Y,TEXT,N):- Displays the "TEXT" of N characters left-aligned on the scaled point (X,Y) of the previously-defined window.

5.0 CURSOR AND MENU OPERATIONS:-

The following set of routines are for displaying menus and choice of cursor therein:-

5.1 MNOPEN(X,Y,MNO):-

Announces that a menu is to be displayed whose top left-hand corner is to be at screen co-ordinates X,Y. (MNO is the menu number (= 1 or 2) for the current display instance. At most two menus are allowed in the present implementation.

e.g. CALL MNOPEN(800.,200.,1)

5.2 MNTEXT(TEXT,N,MNO):-

Puts up text containing N characters as the next line of the menu MNO.

e.g. CALL MNTEXT("ITEM",4,1)

5.3 MNDISP(MNTXT,ITEMNO,LEN,MNO):- Displays a complete menu whose text is defined by a DATA statement as MNTXT, with number of items given by ITEMNO. LEN specifies the character length of each item. NMTXT must be declared in UNIX Fortran as a 'Logical*1' array.

5.4 MNPICK(I,ICHAR,MNO):- Puts up the cursor and returns in I the index of the line chosen (i.e. Item number) by the user, and the character (ICHAR) typed in. It retries until a valid line is picked up. The item picked up is marked by an arrow.

5.5 TXCURS(X,Y,ICHAR):- Displays the cursor and returns the window co-ordinates of the cursor (X,Y) and the character (ICHAR) typed in. This would enable the user to put graphical information into a program.

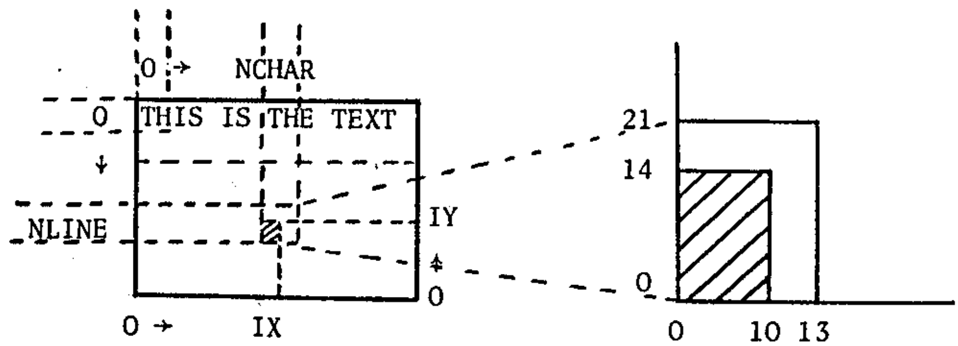
5.6 CURPOS(X,Y):- Positions the 'Alpha' cursor at the specified window co-ordinates (X,Y) at which the user might display some textual information. (Note: this routine would make the Tektronix 4010 compatible with the GT42 operating as an emulated T4010).

5.7 FRAME(X,Y,NC):- Closes the menu with a rectangle drawn round it, in order to distinguish it from other textual information that might appear on the screen. X,Y specifies the top left-hand corner of the rectangle, NC specifies the number of items in the menu.

The following two routines interpret the screen co-ordinates as character positions and vice-versa.

5.8 CHTOXY(NLINE,NCHAR,IX,IY):- Converts the character co-ordinates to the screen co-ordinates of the bottom left-hand corner of the character. The character co-ordinates are represented by line number NLINE, and character number (in the line) NCHAR.

5.9 XYVOCH(IX,IY,NLINE,NCHAR,IA,IB):- Given screen co-ordinates (IX,IY) this sets (NLINE,NCHAR) to indicate the corresponding character co-ordinates (the top left-hand character position 0,0). Also (IA,IB) are set to the position of this point within the indicated character (in 0:13,0:21, the character itself being in 0:10,0:14).



screen   Character zone

6.0 MISCELLANEOUS

6.1 OVRLAY(FILENAME):-

Overlays the calling process with the named file, then transfers control to the beginning of the core image of the file. There can be no return from the file since the calling core image is lost. Thus, application programs larger than the core memory available could be logically segmented into a number of smaller modules and linked by a sequence of overlays.

6.2 RMFILE(FILENAME):- Removes the named file from the current directory.

Note: the above two routines are 'UNIX' dependent.

6.3 WERROR(I):- Displays 'ERROR MESSAGE' and the value of I indicating the error number.

6.4 IREM(I,J):- Returns the positive value of I mod J, where I,J are integers.

7.0 LINKING OF LIGHT LIBRARY SUBROUTINES WITH USER PROGRAM

To incorporate the LIGHT subroutines in his program the user must type the following UNIX command

```
FC USERPROG -LL
```

APPENDIX 2

IDF - PROGRAM MODULES LISTING

APPENDIX 2.1

THE NUMERICAL ALGORITHMS

```
C                         *****************
C                         * APPENDIX 2.11 *
C                         *****************
C
C THIS IS THE MAIN PROGRAM SEGMENT WHICH CALLS THE NUMERICAL ALGORITHMS
C INCORPORATED IN THE EXPLICIT PACKAGE.
C                         ----------------
C
C
C******** MAIN PROGRAM SEGMENT FOR INVOKING THE NUMERICAL ALGORITHMS****
C
      COMMON/IO/IN,IOUT
C I/O COMMON DATA AREA
      COMMON/DATSUP/NPS,NPI(50),IFREES,X(50),Y(50),L(50),IH(5),M(5)
&              ,METHOD,IHELP,IPREV,BOUND(2),SUBSET,INTPNT,IE(2)
      COMMON/CURVEFIT/COEF(50,6),XCORD(200),YCORD(200)
      DATA MODL2,MODL4,DATSP,OUTCRV,JOINFL/"MOD2","MOD4","DATSUPFL"
&           ,"OUTCRV","JONFLE"/
      DATA MODL5/"MOD5"/
      INTEGER SUBSET
      LOGICAL*1 MODL2(10),MODL4(10),DATSP(10),OUTCRV(10),JOINFL(10)
      LOGICAL*1 MODL5(10)
      DIMENSION XO(50),YO(50)
      CALL RDCOM1
      IN=5
      IOUT=6
      CALL REMLNK(XO,YO)
      CALL INTRPT(INTPNT,NPS,NPI,XO,YO)
      CALL JONSUB(IERR,LSUM)
      IF(IERR.EQ.1)GOTO 2
      WRITE(IOUT,10)
10    FORMAT("PLEASE WAIT,FIT IS BEING COMPUTED")
      NC=4
      GOTO(31,32,33,34,35,36,38),METHOD
C
C NEWTON DIVIDED DIFFERENCES METHOD(THE CLASSICAL INTERPOLATION)
31    CALL NEWTON(NPS,NPI,XO,YO,XCORD,YCORD,COEF,METHOD)
      NC=1
      GOTO 3
C
C PIECEWISE QUINTIC INTERPOLATION POLYNOMIAL (MAUD ALGORITHIM)
32    CALL PIECWS(NPS,NPI,XO,YO,XCORD,YCORD,COEF,BOUND,IE,METHOD)
      NC=6
      GOTO 3
C
C CUBIC SPLINE (2ND DERV. BOUNDARY CONDITION)
33    CALL CUBIC1(NPS,NPI,XO,YO,XCORD,YCORD,COEF,BOUND,METHOD)
      GOTO 3
C
C CUBIC SPLINE (2ND DERV. BOUNDARY CONDITION *)
34    CALL CUBIC2(NPS,NPI,XO,YO,XCORD,YCORD,COEF,BOUND,METHOD)
      GOTO 3.
C
C CUBIC SPLINE (1ST DERV. BOUNDARY CONDITION)
35    CALL CUBIC3(NPS,NPI,XO,YO,XCORD,YCORD,COEF,BOUND,METHOD)
      GOTO 3
C
C PERIODIC CUBIC SPLINE "CUBIC P"
36    CALL CUBICP(NPS,NPI,XO,YO,XCORD,YCORD,COEF,BOUND,METHOD)
      GOTO 3
C
```

```
C CONTROL END CONDITION CUBIC SPLINE
38      CALL CSPLEN(NPS,NPI,X0,Y0,XCORD,YCORD,COEF,BOUND,IE,METHOD)
        GOTO 3
30      STOP
C CALL NEXT MODEL FOR CURVE FIT
3       IPREV=0
        CALL WRCOM1
333     CALL WRCOM2(NPS,NPI,NC)
        IF(INTPNT.NE.999) GOTO 99
        CALL OVRLAY(MODL5)
99      CALL OVRLAY(MODL4)
C ERROR MESSAGE DISPLAY
2       IPREV=LSUM
        CALL WRCOM1
        CALL OVRLAY(MODL2)
        STOP
        END
C
C ***********NEWTON DIVIDED DIFFERENCE(GLOBAL)**********
C
        SUBROUTINE NEWTON(N,N1,X1,Y1,XX,YY,C,M)
        DIMENSION N1(1),X1(1),Y1(1),XX(1),YY(1),C(50,6)
C COMPUTE THE POLYNOMIAL COEFFICIENT
        N2=N-1
        DO 1 K=1,N2
         J=N-K
         C(J+1,1)=(Y1(J+1)-Y1(J))/(X1(J+1)-X1(J))
1       CONTINUE
55      C(1,1) =Y1(1)
        N3=N-2
        DO 2 J=1,N3
         K=J+2
         DO 3 L=K,N
            I=N-L+K
            XXX=X1(I)-X1(I-(J+1))
            C(I,1)=(C(I,1)-C(I-1,1))/XXX
3        CONTINUE
2       CONTINUE
C EVALUTE INTERPOLATION FUNCTION
66      IP=1
        DO 4 I=1,N2
         T1=X1(I+1)-X1(I)
         R1=T1/(N1(I)+1)
         IP1=IP+N1(I)+1
         XX(IP)=X1(I)
         Z=X1(I)
         XX(IP1)=X1(I+1)
         YY(IP)=Y1(I)
         YY(IP1)=Y1(I+1)
         N11=N1(I)
         DO 5 K=1,N11
          XX(IP+K)=Z+R1
          Z=Z+R1
          A=C(N,1)
          DO 7 L=1,N2
           J=N-L
           A=C(J,1)+(Z-X1(J))*A
7         CONTINUE
77        YY(IP+K)=A
5        CONTINUE
         IP=IP1
4       CONTINUE
```

```
      RETURN
      END
C
C *********PIECEWISE POLYNOMIAL INTERPOLATION******************
C
      SUBROUTINE PIECWS(N,N1,X1,Y1,XX,YY,C,B,IE,M)
      DIMENSION N1(1),X1(1),Y1(1),XX(1),YY(1),B(1),C(50,6),IE(1)
      CALL MAUDPW(N,X1,Y1,B,C,IE)
C COMPUTE FUNCTION VALUES(I.E INTERPOLATED POINTS)
      CALL COMFIT(N,N1,X1,Y1,XX,YY,C,M)
      RETURN
      END
C
C ***************PIECEWISE POLYNOMIAL INTERPOLATION*****************
C                    (MAUD METHOD)
C
      SUBROUTINE MAUDPW(N,X1,Y1,B,C,IE)
      DIMENSION X1(1),Y1(1),B(1),C(50,6),IE(1)
      IP=1
      N2=N-1
      DO 1 I=1,N2
      R=1.
      S=1.
      F3=1.
      F1=.5
      F2=.5
      IF(IE(1).GT.20) GOTO 20
      IB=1
      IE1=IE(1)-10
      GOTO 30
20    IB=0
      IE1=IE(1)-20
C 1ST INTREVAL ,NOT EQUAL SPACING
30    IF(IB.EQ.0..AND.I.EQ.1.)GOTO 21
C LAST INTERVAL,NOT EQUAL SPACING
      IF(IB.EQ.0..AND.I.EQ.N2) GOTO 28
C 1ST INTERVAL,EQUAL SPACING
      IF(IB.EQ.1..AND.I.EQ.1.)GOTO 21
C LAST INTERVAL,EQUAL SPACING
      IF(IB.EQ.1..AND.I.EQ.N2) GOTO 28
      IF(IB.EQ.1.)GOTO 41
C COMPUTE RATIOS OF X-DIFF,R,S
C IN ANY INTERVAL OTHER THAN 1ST OR LAST
      E1=X1(I)-X1(I-1)
      E2=X1(I+1)-X1(I)
      E3=X1(I+2)-X1(I+1)
      R=E1/E2
      S=E3/E2
      F1=1./(R*(R+1))
      F2=1./(S*(S+1))
      F3=(R+S+2.)/((R+1)*(S+1))
C COMPUTE THE DIFFERENCES
41    D1=Y1(I)-Y1(I-1)
      D2=Y1(I+1)-Y1(I)
      D3=Y1(I+2)-Y1(I+1)
111   T0=F1*D1+F2*D3-F3*D2
      GOTO 51
C COMPUT  DIFFERENCE AT FIRST INTERVAL
21    R=1.
      F1=.5
      E2=X1(I+1)-X1(I)
      E3=X1(I+2)-X1(I+1)
```

```
      S=E3/E2
      F2=1./(S*(S+1))
      F3=(R+S+2)/((R+1)*(S+1))
      D2=Y1(I+1)-Y1(I)
      D3=Y1(I+2)-Y1(I+1)
      GOTO(61,62,63),IE1
C ENCASTRED END CONDITION
61    D1=B(1)
      T1=D2-D1
      GOTO 111
C RELAXED END CONDITION
62    D1=D2
      T1=0.
      GOTO 111
C PARABOLIC END CONDITION
63    T0=0.
      D1=2.*D2-D3
      T1=D3-D2
      GOTO 51
C COMPUTE   DIFFERENCES AT LAST INTERVAL
28    E1=X1(I)-X1(I-1)
      E2=X1(I+1)-X1(I)
      R=E1/E2
      F1=1./(R*(R+1.))
      S=1.
      F2=0.5
      F3=(R+S+2.)/((R+1.)*(S+1.))
31    D1=Y1(I)-Y1(I-1)
      D2=Y1(I+1)-Y1(I)
      GOTO(71,72,73),IE(2)
C ENCASTRED END CONDITION(LAST END)
71    D3=B(2)
      GOTO 111
C RELAXED END CONDITION
72    D3=D2
      GOTO 111
C PARABOLIC END CONDITION
73    T0=0.
      GOTO 51
C COMPUTE THE DESIRED COEFFICIENTS FOR EACH INTERVAL
51    C(I,1)=Y1(I)
      C(I,2)=(D1+D2*R*R)*F1
      IF(I.NE.1)GOTO 55
      C(I,3)=.5*T1
      GOTO 66
55    C(I,3)=(D2*R-D1)*F1
66    C(I,4)=-3.*T0
      C(I,5)=5.*T0
      C(I,6)=-2.*T0
1     CONTINUE
      RETURN
      END
C
C ***********CUBIC FLINE(2ND DERV. BOUNDARY CONDITION)***********
C
      SUBROUTINE CUBIC1(N,N1,X1,Y1,XX,YY,C,B,M)
      DIMENSION N1(1),X1(1),Y1(1),XX(1),YY(1),C(50,6),B(1),Y2(50)
      CALL GAUSS1(N,X1,Y1,Y2,B)
      CALL COEFNT(N,X1,Y1,Y2,C)
      CALL COMFIT(N,N1,X1,Y1,XX,YY,C,M)
      RETURN
      END
```

```
C
C *************** GAUSSIAN ELIMINATION FOR CUBIC 1 *************
C
      SUBROUTINE GAUSS1(N,X1,Y1,Y2,B)
      DIMENSION X1(1)  ,Y1(1),Y2(1),F(50),G(50),B(1)
      Y2(1)=B(1)
      Y2(N)=B(2)
      N1=N-1
      G(1)=0.
      F(1)=0.
      DO 2 K=1,N1
       J2=K+1
       H2=X1(J2)-X1(K)
       R2=(Y1(J2)-Y1(K))/H2
       IF(K.EQ.1) GOTO 1
       Z=1./(2.*(H1+H2)-H1*G(J1))
       G(K)=Z*H2
       H=6.*(R2-R1)
       IF(K.EQ.2)H=H-H1*Y2(1)
       IF(K.EQ.N1)H=H-H2*Y2(N)
       F(K)=Z*(H-H1*F(J1))
1      J1=K
       H1=H2
       R1=R2
2     CONTINUE
      Y2(N1)=F(N1)
      IF(N1.LE.2) RETURN
      N2=N1-1
      DO 3 J1=2,N2
       K=N-J1
       Y2(K)=F(K)-G(K)*Y2(K+1)
3     CONTINUE
      RETURN
      END
C
C *********CUBIC SPLINE(2ND DERV. BOUNDARY CONDITION)*******
C               [Y2(1)=U*Y2(2)..&..VY2(N-1)=Y2(N)]
C

      SUBROUTINE CUBIC2(N,N1,X1,Y1,XX,YY,C,B,M)
      DIMENSION N1(1),X1(1),Y1(1),XX(1),YY(1),C(50,6),B(1),Y2(50)
       CALL GAUSS2(N,X1,Y1,Y2,B)
      CALL COEFNT(N,X1,Y1,Y2,C)
      CALL COMFIT(N,N1,X1,Y1,XX,YY,C,M)
      RETURN
      END
C
C ***************CUBIC2 .....GAUSS2************************
C
      SUBROUTINE GAUSS2(N,X1,Y1,Y2,B)
      DIMENSION X1(1),Y1(1),Y2(1),F(50),G(50),B(1)
      N1=N-1
      G(1)=0.
      F(1)=0.
      F2=2.
      DO 2 K=1,N1
       J2=K+1
       H2=X1(J2)-X1(K)
       R2=(Y1(J2)-Y1(K))/H2
       IF(K.EQ.1) GOTO 1
       F1=2.
       IF(K.EQ.2) F1=2.*B(1)
```

```
          IF(K.EQ.N1)F2=2.*B(2)
          Z=1./((F1-G(J1))*H1+F2*H2)
          G(K)=Z*H2
          F(K)=Z*(6.*(R2-R1)-H1*F(J1))
1         J1=K
          H1=H2
          R1=R2
2        CONTINUE
         Y2(N1)=F(N1)
         IF(N1.LE.2) GOTO 4
         N2=N1-1
         DO 3 J1=2,N2
          K=N-J1
          Y2(K)=F(K)-G(K)*Y2(K+1)
3        CONTINUE
4        Y2(1)=B(1)*Y2(2)
         Y2(N1)=B(2)*Y2(N1)
         RETURN
         END
C
C **********CUBIC SPLINE(1ST DERV. BOUNDARY CONDITION)**********
C
         SUBROUTINE CUBIC3(N,N1,X1,Y1,XX,YY,C,B,M)
         DIMENSION N1(1),X1(1),Y1(1),XX(1),YY(1),C(50,6),B(1),Y2(50)
         CALL GAUSS3(N,X1,Y1,Y2,B)
         CALL COEFNT(N,X1,Y1,Y2,C)
         CALL COMFIT(N,N1,X1,Y1,XX,YY,C,M)
         RETURN
         END
C
C  *************CUBIC 3 .. GAUSS3*******************************
C
         SUBROUTINE GAUSS3(N,X1,Y1,Y2,B)
         DIMENSION X1(1),Y1(1),Y2(1),B(1),F(50),G(50)
         Y2(1)=B(1)
         Y2(N)=B(2)
         N1=N-1
         J1=1
         H1=0.
         F(1)=0.
         G(1)=0.
         R1=Y2(1)
         DO 3 K=1,N
          IF(K.LE.N1) GOTO 1
          H2=0.
          R2=Y2(N)
          GOTO 2
1         J2=K+1
          H2=X1(J2)-X1(K)
          R2=(Y1(J2)-Y1(K))/H2
2         Z=1./(2.*(H1+H2)-H1*G(J1))
          G(K)=Z*H2
          F(K)=Z*(6.*(R2-R1)-H1*F(J1))
          J1=K
          H1=H2
          R1=R2
3        CONTINUE
         Y2(N)=F(N)
         DO 4 J1=1,N1
          K=N-J1
          Y2(K)=F(K)-G(K)*Y2(K+1)
4        CONTINUE
```

```
        RETURN
        END
C
C *********PERIODIC CUBIC SPLINE**********************
C
        SUBROUTINE CUBICP(N,N1,X1,Y1,XX,YY,C,B,M)
        DIMENSION N1(1),X1(1),Y1(1),XX(1),YY(1),C(50,6),B(1),Y2(50)
         CALL GAUSSP(N,X1,Y1,Y2,B)
        CALL COEFNT(N,X1,Y1,Y2,C)
        CALL COMFIT(N,N1,X1,Y1,XX,YY,C,M)
        RETURN
        END
C
C **************GAUSSP************************
C
        SUBROUTINE GAUSSP(N,X1,Y1,Y2,B)
C NORMALISED PERIODIC "TRIDIAGONAL" ALGORITHM FOR THE SOULTION OF
C FACTORISATION OF THE ARRAY A BY NORMALISED ALGORITHM
C A SET OF LINEAR EQUATION (REFC.DR. BENSON PH.D THESIS)
C COMPUTE D(1) TERE ARE N-1 UNKNOWN VALUES IN Y2(N-1) 2ND DERV.
C WHERE A=DT'TD WHICH FACTORISED INTO
        DIMENSION X1(1),Y1(1),Y2(50),B(1),D(50),E(50),F(50)
        Q1=X1(2)-X1(1)
        Q2=X1(N)-X1(N-1)
        B1=2.*(Q1+Q2)
        D(1)=SQRT(B1)
C COMPUTE ALL OTHER D'S UP TO D(N-2)
        N2=N-2
        DO 1 J=2,N2
         C1=X1(J)-X1(J-1)
         Q1=C1
         Q2=X1(J+1)-X1(J)
         B1=2.*(Q1+Q2)
         D(J)=SQRT(B1-C1/D(J-1))
1       CONTINUE
C COMPUTE E'S
        N3=N-3
        DO 2 J=1,N3
         C=X1(J+1)-X1(J)
         E(J)=C/(D(J)*D(J+1))
2       CONTINUE
C COMPUTE THE D(N-1) THE LAST IN THE FACTORISATION AS THERE ARE ONLY
C N-1 UNKNOWN
        S1=1.+E(N-4)*E(N-4)
        N5=N-5
        DO 3 J=N5,1,-1
         S1=1.+E(J)*E(J)*S1
3       CONTINUE
        S2=1.
        DO 4 J=1,N3
4        S2=S2*E(J)
        Q1=X1(N-1)-X1(N-2)
        C1=Q1
        Q2=X1(N)-X1(N-1)
        C2=Q2
        B1=2.*(Q1+Q2)
        D(N-1)=SQRT(B1-(C2/D(1))**2*S1-(C1/D(N-2)+((-1)**(N-1)*S2*C2)/
&       D(1))**2)
        E(N-2)=C1/(D(N-2)*D(N-1))
C COMPUTE F'S
        F(1)=C2/(D(1)*D(N-1))
        DO 5 J=2,N2
```

```
5          F(J)=-E(J-1)*F(J-1)
C COMPUTE THE INTERMEDIATE SOULTION T'H=G,SOLVE FOR H AND STORE IN Y2(J)
C WHERE G=K/D
        R1=Y1(2)-Y1(1)
        R2=Y1(N)-Y1(N-1)
        Q1=X1(2)-X1(1)
        Q2=X1(N)-X1(N-1)
        K1=6.*(R1/Q1-R2/Q2)
        Y2(1)=K1/D(1)
        DO 6 J=2,N2
         R1=Y1(J)-Y1(J-1)
         R2=Y1(J+1)-Y1(J)
         Q1=X1(J)-X1(J-1)
         Q2=X1(J+1)-X1(J)
         K1=6.*(R2/Q2-R1/Q1)
         Y2(J)=K1/D(J)-E(J-1)*Y2(J-1)
6       CONTINUE
        S=0.
C COMPUTE LAST H AS Y2(N-1)
        DO 7 J=1,N2
7        S=S+F(J)*Y2(J)
        R1=Y1(N-1)-Y1(N-2)
        R2=Y1(N)-Y1(N-1)
        Q1=X1(N-1)-X1(N-2)
        Q2=X1(N)-X1(N-1)
        K1=6.*(R2/Q2-R1/Q1)
        Y2(N-1)=K1/D(N-1)-E(N-2)*Y2(N-2)-S
C COMPUTE THE FINAL SOULTION Y2(J) AS 2ND DERV
        DO 8 J=N2,1,-1
8        Y2(J)=Y2(J)-E(J)*Y2(J+1)-F(J)*Y2(N-1)
        N1=N-1
        DO 9 J=1,N1
9        Y2(J)=Y2(J)/D(J)
        Y2(N)=Y2(1)
        RETURN
        END
C
C ********** COMPUTE POLYNOMIAL COEFFICIENTS *******************
C
        SUBROUTINE COEFNT(N,X1,Y1,Y2,C)
        DIMENSION X1(1),Y1(1),Y2(1),C(50,6)
        N2=N-1
C SPLINE COEFFICIENT PER INTERVAL
        DO 1 I=1,N2
         T1=X1(I+1)-X1(I)
         T2=Y1(I+1)-Y1(I)
         D1=Y2(I+1)-Y2(I)
         D2=Y2(I+1)+2.*Y2(I)
         C(I,1)=Y1(I)
         C(I,2)=T2/T1-(T1*D2)/6.
         C(I,3)=Y2(I)/2.
         C(I,4)=D1/(6.*T1)
1       CONTINUE
        RETURN
        END
C
C ********** COMPUTE FUNCTION VALUES ***************************
C
        SUBROUTINE COMFIT(N,N1,X1,Y1,XX,YY,C,M)
        DIMENSION N1(1),X1(1),Y1(1),XX(1),YY(1),C(50,6)
        N2=N-1
        IP=1
```

```
C COMPUTE THE INTERPOLATED ORDINATES
        DO 1 I=1,N2
          CALL INTPLT(I,IP,N1,X1,Y1,XX,YY,C,M)
1       CONTINUE
        RETURN
        END
C
C*************EVALUATE THE INTERPOLANT FUNCTION VALUES**********
C
        SUBROUTINE INTPLT(I,IP,N1,X1,Y1,XX,YY,C,M)
        DIMENSION N1(1),X1(1),Y1(1),XX(1),YY(1),C(50,6)
        INTP=N1(I)
        T1=X1(I+1)-X1(I)
        R1=T1/(INTP+1)
        IP1=IP+INTP+1
        Z=X1(I)
        XX(IP)=X1(I)
        XX(IP1)=X1(I+1)
        YY(IP)=Y1(I)
        YY(IP1)=Y1(I+1)
        DO 2 J=1,INTP
         XX(IP+J)=Z+R1
         Z=Z+R1
         IF(M.EQ.2) GOTO 3
         T=Z-XX(IP)
         YY(IP+J)=C(I,1)+T*(C(I,2)+T*(C(I,3)+T*C(I,4)))
2       CONTINUE
        IP=IP1
        RETURN
3       H=XX(IP1)-XX(IP)
        T=(Z-XX(IP))/H
        YY(IP+J)=C(I,1)+T*(C(I,2)+T*(C(I,3)+T*(C(I,4)+T*(C(I,5)+T*C(I,6)))))
        GOTO 2
        END
C
C **************CUBIC SPLINE WITH VARIABLE END CONDITION**************
C
        SUBROUTINE CSPLEN(N,N1,X1,Y1,XX,YY,C,B,IE,M)
        DIMENSION N1(1),X1(1),Y1(1),XX(1),YY(1),C(50,6),B(1),IE(1)
        DIMENSION BX(50),C1(50),D(50),S(50),DX(50)
        CALL GENXPR(N,X1,Y1,DX)
        S1=DX(2)/DX(3)
        S2=DX(N)/DX(N-1)
C       MAIN PROGRAM
        CALL GENMAT(N,S1,S2,DX,BX,C1,D,IE,EX,A)
        CALL GENRHS(N,S1,S2,X1,Y1,B,DX,S,IE)
        CALL GAUSSC(N,BX,C1,D,S,IE,EX,A)
        CALL ECCOEF(N,X1,Y1,C,S,DX)
        CALL COMFIT(N,N1,X1,Y1,XX,YY,C,M)
        RETURN
        END
C
C *************GENERATE DX PARAMETER**********************
C
        SUBROUTINE GENXPR(N,X1,Y1,DX)
        DIMENSION X1(1),Y1(1),DX(1)
        DX(1)=0.
        N1=N-1
        DO 1 K=1,N1
         DX(K+1)=X1(K+1)-X1(K)
1       CONTINUE
        RETURN
```

```
      END
C
C *************GENERATE COEFFICIENTS MATRIX*********************
C
      SUBROUTINE GENMAT(N,S1,S2,DX,BX,C1,D,IE,EX,A)
      DIMENSION DX(1),BX(1),C1(1),D(1),IE(1)
      N1=N-1
      BX(1)=0.
      D(N)=0.
      C1(1)=1.
      C1(N)=1.
      EX=0.
      A=0.
      GOTO(1,2,3,4),IE(1)
1     D(1)=0.
      GOTO 5
2     D(1)=0.5
      GOTO5
3     D(1)=1.
      GOTO 5
4     D(1)=1.-S1*S1
      EX=-S1*S1
5     GOTO(6,7,8,9),IE(2)
6     BX(N)=0.
      GOTO 10
7     BX(N)=2.
      C1(N)=4.
      GOTO 10
8     BX(N)=1.
      GOTO 10
9     BX(N)=1.-S2*S2
      A=-S2*S2
10    DO 11 J=2,N1
       BX(J)=DX(J+1)
       C1(J)=2.*(DX(J)+DX(J+1))
       D(J)=DX(J)
11    CONTINUE
      RETURN
      END
C
C ****************GENERATE R.H.S **********************************
C
      SUBROUTINE GENRHS(N,S1,S2,X1,Y1,B,DX,S,IE)
      DIMENSION X1(1),Y1(1),B(1),DX(1),S(1),IE(1)
      N1=N-1
      GOTO(1,2,3,4),IE(1)
1     S(1)=B(1)
      S(N)=B(2)
      GOTO 5
2     S(1)=(1.5/DX(2))*(Y1(2)-Y1(1))
      S(N)=(6./DX(N))*(Y1(N)-Y1(N-1))
      GOTO 5
3     S(1)=(2./DX(2))*(Y1(2)-Y1(1))
      S(N)=(2./DX(N))*(Y1(N)-Y1(N-1))
      GOTO 5
4     S(1)=(2./DX(2))*(-Y1(1)+(1.+S1**3)*Y1(2)-(S1**3)*Y1(3))
      S(N)=(2./DX(N))*((S2**3)*Y1(N-2)-(1.+S2**3)*Y1(N-1)+Y1(N))
5     DO 11 I=2,N1
       S(I)=3*(DX(I+1)*DX(I+1)*(Y1(I)-Y1(I-1))+DX(I)*DX(I)*(Y1(I+1)-Y1(I)))
       S(I)=S(I)/(DX(I+1)*DX(I))
11    CONTINUE
      RETURN
```

```
      END
C
C ************GAUSSIAN ELIMINATION FOR VARIABLE END CONDITION**********
C              *ADAPTATION OF QUIN DIAGONAL MATRIX*
C                    **REFC. PROFESSOR EVANS**
C
      SUBROUTINE GAUSSC(N,BX,C1,D,S,IE,EX,A)
      DIMENSION BX(1),C1(1),D(1),S(1),IE(1)
      N1=N-1
      Z=1./C1(1)
      D(1)=D(1)*Z
      S(1)=S(1)*Z
      IF(IE(1).EQ.4)EX=EX*Z
      DO 1 I=2,N1
       I1=I-1
       I2=I+1
       Z=1./(C1(I)-BX(I)*D(I1))
       IF(I.EQ.2)D(I)=(D(I)-BX(I)*EX)*Z
       IF(I.NE.2)D(I)=D(I)*Z
       S(I)=(S(I)-BX(I)*S(I1))*Z
       IF(I2.LT.N) GOTO 1
       S(I2)=S(I2)-A*S(I1)
       BX(I2)=BX(I2)-A*D(I1)
1     CONTINUE
      S(N)=(S(N)-BX(N)*S(N-1))/(C1(N)-BX(N)*D(N-1))
      DO 2 K=1,N1
       I=N-K
       IF(I.NE.1)S(I)=S(I)-D(I)*S(I+1)
       IF(I.EQ.1)S(I)=S(I)-D(I)*S(I+1)-EX*S(I+2)
2     CONTINUE
      RETURN
      END
C
C***************GENERATE COEFFICIENTS********************
C               (WITH VARIABLE END CONDITION)
C
      SUBROUTINE ECCOEF(N,X1,Y1,C,S,DX)
      DIMENSION X1(1),Y1(1),C(50,6),S(1),DX(1)
      N1=N-1
      DO 1 I=1,N1
       C(I,1)=Y1(I)
       C(I,2)=S(I)
       C(I,3)=(3./(DX(I+1)*DX(I+1)))*(Y1(I+1)-Y1(I))-(1./DX(I+1))*
&               (S(I+1)+2.*S(I))
       C(I,4)=-(2./DX(I+1)**3)*(Y1(I+1)-Y1(I))+(1./(DX(I+1)*DX(I+1)))*
&              (S(I+1)+S(I))
1     CONTINUE
      RETURN
      END
C
C *********** COMPUTE INTERMEDIATE POINTS ********************
C
      SUBROUTINE INTRPT(INTPNT,N,N1,X0,Y0)
      DIMENSION N1(1),X0(1),Y0(1)
      N2=N-1
      IF(INTPNT.LT.1) GOTO 101
      IF(INTPNT.EQ.999) RETURN
      DO 1 I=1,N2
1      N1(I)=INTPNT
      RETURN
101   W1=X0(N)-X0(1)
      W2=Y0(N)-Y0(1)
```

```
      W3=SQRT(W1*W1+W2*W2)
      S=W3/100.
      DO 2 I=1,N2
       W1=XO(I+1)-XO(I)
       W2=YO(I+1)-YO(I)
       W3=SQRT(W1*W1+W2*W2)
       N1(I)=W3/S-1.
       N5=(200-N)/(N-1)
        IF(N1(I).GT.N5)N1(I)=N5
2     CONTINUE
      RETURN
      END
C
C*********SETS JOINED CURVE COMMON DATA*****************
C
      SUBROUTINE JONSUB(ERR,LSUM)
C JOIN COMMON DATA AREA AND INPUT COMMON DATA AREA
      COMMON/JOIN/CJ1(500),CJ2(500),J3(12),J4(100),IPNTR(6)
      COMMON/DATSUP/NPS,NPI(50),IFREES,X(50),Y(50),L(50),IH(5),M(5)
&                 ,METHOD,IHELP,IPREV,BOUND(2),SUBSET,INTPNT,IE(2)
      INTEGER ERR,SUBSET
C FIRST CURVE SEGMENT?
      IF(SUBSET.EQ.0.OR.SUBSET.EQ.1) RETURN
      IF(IERROR(103).NE.0) GOTO 3
C READ THE JOIN CURVE COMMON DATA AREA
      CALL RDCOMJ
3     MSUM=0
      N2=NPS-1
      DO 1 I=1,N2
1      MSUM=MSUM+NPI(I)
      MSUM=MSUM+NPS
      LSUM=MSUM+J3(1)
      IP1=J3(1)
      IF(LSUM.GT.500) GOTO 400
C CHECK FIRST POINT AS LAST POINT OF THE PREVIOUS SUBSET
      IF(X(IH(1)).EQ.CJ1(IP1)) RETURN
C ADD ONE POINT TO PRESENT DATA POINTS
      L(IFREES)=IH(1)
      IH(1)=IFREES
      X(IFREES)=CJ1(IP1)
      Y(IFREES)=CJ2(IP1)
      NPS=NPS+1
      IFREES=IFREES+1
      NPI(NPS-1)=NPI(NPS-2)
C SAVE JOIN CURVE COMMON DATA
      CALL WRCOMJ
      RETURN
C ERROR MESSAG
400   ERR=1
      RETURN
      END
```

```
C                       ****************
C                       * APPENDIX 2.12 *
C                       ****************
C
C THIS IS THE MAIN PROGRAM SEGMENT WHICH CALLS THE NUMERICAL ALGORITHMS
C INCORPORATED IN THE PARAMETRIC PACKAGE.
C                      ---------------
C
C
C ******MAIN PROGRAM SEGMENT FOR INVOKING THE NUMERICAL ALGORITHMS ******
C
C
        COMMON/IO/IN,IOUT
C INPUT/OUTPUT COMMON DATA AREA
        COMMON/DATSUP/NPS,NPI(50),IFREES,X(50),Y(50),Z(50),L(50),IH(5),
&                      M(5),METHOD,IHELP,IPREV,BOUND(6),SUBSET,INTPNT,IE(2)
&                 ,ID
        COMMON/CURVEFIT/XCOEF(50,4),YCOEF(50,4),ZCOEF(50,4),XCORD(200),
&                        YCORD(200),ZCORD(200),TCORD(200)
        DATA PMODL2,PMODL4,PDTSP,PMODL5/"PMODL2","PMODL4","PDTSUPFL"
&                ,"PMODL5"/
        INTEGER SUBSET
        LOGICAL*1 PMODL2(10),PMODL4(10),PDTSP(10),PMODL5(10)
        DIMENSION X0(50),Y0(50),Z0(50),T0(50)
        CALL PRDCM1
        IN=5
        IOUT=6
        CALL PRMLNK(X0,Y0,Z0)
        CALL INTRPT(INTPNT,NPS,NPI,X0,Y0,Z0,T0,ID,METHOD)
        WRITE(IOUT,10)
10      FORMAT("PLEASE WAIT,FIT IS BEING COMPUTED")
        NC=4
        GOTO(31,32,33,34,35),METHOD
C
C STANDARD PARAMETRIC CUBIC SPLINE(SECOND DERV.BOUNDARY CONDITION)
31      CALL CUBPAR(NPS,NPI,X0,Y0,Z0,T0,XCORD,YCORD,ZCORD,TCORD,
&XCOEF,YCOEF,ZCOEF,BOUND,ID,METHOD)
        GOTO 3
C
C CYCLIC CUBIC SPLINE
32      CALL CYCLIC(NPS,NPI,X0,Y0,Z0,T0,XCORD,YCORD,ZCORD,TCORD,
&XCOEF,YCOEF,ZCOEF,ID,METHOD)
        GOTO 3
C
C ANTICYCLIC CUBIC SPLINE
33      CALL ANTCYC(NPS,NPI,X0,Y0,Z0,T0,XCORD,YCORD,ZCORD,TCORD,
&XCOEF,YCOEF,ZCOEF,ID,METHOD)
        GOTO 3
C
C CUBIC SPLINE VARIABLE END CONDITION
34      CALL PARCUB(NPS,NPI,X0,Y0,Z0,T0,XCORD,YCORD,ZCORD,TCORD,
&XCOEF,YCOEF,ZCOEF,BOUND,IE,ID,METHOD)
C CALL NEXT MODEL FOR CURVE FIT
3       IPREV=0
        CALL PWRCM1
333     CALL PWRCM2(NPS,NPI,NC,ID)
        IF(INTPNT.EQ.999.AND.ID.EQ.2)CALL OVRLAY(PMODL5)
        CALL OVRLAY(PMODL4)
        CALL PWRCM1
        CALL OVRLAY(PMODL2)
35      STOP
```

```
      END
C
C**********CUBIC SPLINE WITH PARAMETRIC SECOND DERIVATIVES*********
C
      SUBROUTINE CUBPAR(N,N1,XP,YP,ZP,T1,XXP,YYP,ZZP,TTP,XC,YC,ZC,B
&,IDIM,M)
      DIMENSION N1(1),XP(1),YP(1),ZP(1),T1(1),XXP(1),YYP(1),ZZP(1),
&                 TTP(1),XC(50,4),YC(50,4),ZC(50,4),B(1),C(2)
C INTERPOLATE FOR X
      C(1)=B(1)
      C(2)=B(4)
      CALL CUBIC1(N,N1,T1,XP,TTP,XXP,XC,C,M)
C INTERPOLATE FOR Y
      C(1)=B(2)
      C(2)=B(5)
      CALL CUBIC1(N,N1,T1,YP,TTP,YYP,YC,C,M)
C INTERPOLATE FOR Z (3-D)
      C(1)=B(3)
      C(2)=B(6)
      IF(IDIM.EQ.3)CALL CUBIC1(N,N1,T1,ZP,TTP,ZZP,ZC,C,M)
      RETURN
      END

C
C**********CYCLIC CUBIC SPLINE***************************
C
      SUBROUTINE CYCLIC(N,N1,XP,YP,ZP,T1,XXP,YYP,ZZP,TTP,XC,YC,ZC,B
&,IDIM,M)
      DIMENSION N1(1),XP(1),YP(1),ZP(1),T1(1),XXP(1),YYP(1),ZZP(1)
&,TTP(1),XC(50,4),YC(50,4),ZC(50,4),B(1)
      CALL CUBICP(N,N1,T1,XP,TTP,XXP,XC,B,M)
      CALL CUBICP(N,N1,T1,YP,TTP,YYP,YC,B,M)
C THREE DIMENSION
      IF(IDIM.EQ.3)CALL CUBICP(N,N1,T1,ZP,TTP,ZZP,ZC,B,M)
      RETURN
      END
C
C*********GAUSSIAN ELIMINATION FOR CYCLIC CURVES***************
C
      SUBROUTINE GAUSSP(N,X,Y,Y2,B)
      DIMENSION X(1),Y(1),Y2(1),F(50),G(50),H(50),B(1)
      N1=N-1
      N2=N1-1
      J1=1
      G(1)=0.
      F(1)=0.
      H(1)=-1.
      H1=X(N)-X(N1)
      W=H1
      H2=X(N1)-X(N2)
      U=2.*(H1+H2)
      R1=(Y(N)-Y(N1))/H1
      R2=(Y(N1)-Y(N2))/H2
      V=6.*(R1-R2)
      DO 2 K=1,N2
        J2=K+1
        H2=X(J2)-X(K)
        R2=(Y(J2)-Y(K))/H2
        IF(K.EQ.1)GOTO 1
        U=U-W*H(J1)
        V=V-W*F(J1)
        W=-G(J1)*W
```

```
1           Z=1./(2.*(H1+H2)-H1*G(J1))
            G(K)=Z*H2
            H(K)=-Z*H(J1)*H1
            F(K)=Z*(6.*(R2-R1)-H1*F(J1))
            J1=K
            H1=H2
            R1=R2
2         CONTINUE
          H2=W+H1
          H1=(V-H2*F(N2))/(U-H2*(G(N2)+H(N2)))
          Y2(N1)=H1
          DO 3 J1=2,N1
            K=N-J1
            Y2(K)=F(K)-G(K)*Y2(K+1)-H(K)*H1
3         CONTINUE
          Y2(N)=Y2(1)
          RETURN
          END
C
C *************** INTERPOLATION ,EVALUATE FUNCTION VALUES *********
C
          SUBROUTINE INTPLT(I,IP,N1,X1,Y1,XX,YY,C,M)
          DIMENSION N1(1),X1(1),Y1(1),XX(1),YY(1),C(50,4)
          INTP=N1(I)
          IF(M.EQ.4) GOTO 3
          T1=X1(I+1)-X1(I)
          R1=T1/(INTP+1)
          Z=X1(I)
4         IP1=IP+INTP+1
          XX(IP)=X1(I)
          XX(IP1)=X1(I+1)
          YY(IP)=Y1(I)
          YY(IP1)=Y1(I+1)
          DO 2 J=1,INTP
            XX(IP+J)=Z+R1
            IF(M.EQ.4) GOTO 5
            Z=Z+R1
            T=Z-XX(IP)
6           YY(IP+J)=C(I,1)+T*(C(I,2)+T*(C(I,3)+T*C(I,4)))
2         CONTINUE
          IP=IP1
          RETURN
3         Z=0.
          R1=X1(I+1)/(INTP+1)
          GOTO 4
5         T=Z+R1
          Z=Z+R1
          GOTO 6
          END
C
C**********ANTI CYCLIC CUBIC SPLINE**************************
C
          SUBROUTINE ANTCYC(N,N1,XP,YP,ZP,T1,XXP,YYP,ZZP,TTP,XC,YC,ZC
&,IDIM,M)
          DIMENSION N1(1),XP(1),YP(1),ZP(1),T1(1),XXP(1),YYP(1),ZZP(1)
&,TTP(1),XC(50,4),YC(50,4),ZC(50,4)
          CALL CANTCY(N,N1,T1,XP,TTP,XXP,XC,M)
          CALL CANTCY(N,N1,T1,YP,TTP,YYP,YC,M)
          IF(IDIM.EQ.3)CALL CANTCY(N,N1,T1,ZP,TTP,ZZP,ZC,M)
          RETURN
          END
C
```

```
C**********INTERPOLTE FOR EACH X/Y/Z AS IN PARAMETER T*******
C
        SUBROUTINE CANTCY(N,N1,X1,Y1,XX,YY,C,M)
        DIMENSION N1(1),X1(1),Y1(1),XX(1),YY(1),C(50,4),Y2(50)
        CALL ACGAUS(N,X1,Y1,Y2)
        CALL ANTCOF(N,N1,X1,Y1,Y2,XX,YY,C)
        RETURN
        END
C
C*********GAUSSIAN ELIMINATION FOR ANTI CYCLIC CURVES*******
C
        SUBROUTINE ACGAUS(N,X,Y,Y2,B)
        DIMENSION X(1),Y(1),Y2(1),V1(50),W1(50),Y1(50),B(1)
        V1(1)=0.
        W1(1)=0.
        S3=X(N)/X(2)
        Z=1./(2.*(1.+S3))
        V1(2)=Z*S3
        W1(2)=-Z
        Y1(1)=Z*(3./X(2))*(S3*(Y(2)-Y(1))-(1./S3)*(Y(N)-Y(N-1)))
        H=2.*(X(N)+X(N-1))
        F=3.*(X(N)**2*(Y(N-1)-Y(N-2))+X(N-1)**2*(Y(N)-Y(N-1)))
        F=F/(X(N)*X(N-1))
        G=0.
        N2=N-2
        DO 1 I=2,N2
          H=H-G*W1(I)
          F=F-G*Y1(I-1)
          G=-V1(I)*G
          Z=1./(2.*(X(I+1)+X(I))-X(I+1)*V1(I))
          V1(I+1)=0.
          W1(I+1)=-Z*X(I+1)*W1(I)
          B=3.*(X(I+1)**2*(Y(I)-Y(I-1))+X(I)**2*(Y(I+1)-Y(I)))
          B=B/(X(I+1)*X(I))
          Y1(I)=Z*(B-X(I+1)*Y1(I-1))
1       CONTINUE
        H=H-(G+X(N))*(V1(N-1)+W1(N-1))
        Y1(N-1)=F-(G+X(N))*Y1(N-2)
        Y2(N-1)=Y1(N-1)/H
        Y2(N-2)=Y1(N-2)-V1(N-1)*Y2(N-1)
        N3=N-3
        DO 2 I=1,N3
          J=N3+1-I
          Y2(J)=Y1(J)-V1(J+1)*Y2(J+1)-W1(J+1)*Y2(N-1)
2       CONTINUE
        Y2(N)=-Y2(1)
        RETURN
        END
C
C**********ANTICYCLIC COEFFICIENT & FUNCTION*****************
C
        SUBROUTINE ANTCOF(N,N1,X1,Y1,Y2,XX,YY,C)
        DIMENSION N1(1),X1(1),Y1(1),Y2(1),XX(1),YY(1),C(50,4)
        IP=1
        N2=N-1
        DO 1 I=1,N2
          INTP=N1(I)
          V=0.
          R1=X1(I+1)/(INTP+1)
          C(I,1)=Y1(I)
          C(I,2)=Y2(I)
          C(I,3)=(3./X1(I+1)**2)*(Y1(I+1)-Y1(I))-
```

```
&              (1./X1(I+1))*(Y2(I+1)+2.*Y2(I))
          C(I,4)=-(2./X1(I+1)**3)*(Y1(I+1)-Y1(I))+
&              (1./X1(I+1)**2)*(Y2(I+1)+Y2(I))
          IP1=IP+INTP+1
          XX(IP)=X1(I)
          XX(IP1)=X1(I+1)
          YY(IP)=Y1(I)
          YY(IP1)=Y1(I+1)
          DO 2 J=1,INTP
            T=V+R1
            XX(IP+J)=V+R1
            V=V+R1
            YY(IP+J)=C(I,1)+T*(C(I,2)+T*(C(I,3)+T*C(I,4)))
2         CONTINUE
          IP=IP1
1       CONTINUE
        RETURN
        END
C
C********** CONTROL END CONDITON FOR PARAMETRIC CUBIC SPLINE*********
C
        SUBROUTINE PARCUB(N,N1,XP,YP,ZP,T1,XXP,YYP,ZZP,TTP,XC,YC,ZC
&,B,IE,IDIM,M)
        DIMENSION N1(1),XP(1),YP(1),ZP(1),T1(1),XXP(1),YYP(1),ZZP(1)
&,TTP(1),XC(50,4),YC(50,4),ZC(50,4),B(1),IE(1),C(2)
        C(1)=B(1)
        C(2)=B(4)
        CALL CSPLEC(N,N1,T1,XP,TTP,XXP,XC,C,IE,M)
        C(1)=B(2)
        C(2)=B(5)
        CALL CSPLEC(N,N1,T1,YP,TTP,YYP,YC,C,IE,M)
        C(1)=B(3)
        C(2)=B(6)
        IF(IDIM.EQ.3)CALL CSPLEC(N,N1,T1,ZP,TTP,ZZP,ZC,C,IE,M)
        RETURN
        END
C
C **************CUBIC SPLINE CONTROL END CONDITION**************
C
        SUBROUTINE CSPLEC(N,N1,X1,Y1,XX,YY,C,B,IE,M)
        DIMENSION N1(1),X1(1),Y1(1),XX(1),YY(1),C(50,4),B(1),IE(1)
        DIMENSION BX(50),C1(50),D(50),S(50)
        S1=X1(2)/X1(3)
        S2=X1(N)/X1(N-1)
C MAIN PROGRAM CALLS
        CALL GCOFMT(N,S1,S2,X1,BX,C1,D,IE,EX,A)
        CALL GENRHS(N,S1,S2,Y1,B,X1,S,IE)
        CALL GAUSEC(N,BX,C1,D,S,IE,EX,A)
        CALL ECCOEF(N,Y1,C,S,X1)
        CALL COMFIT(N,N1,X1,Y1,XX,YY,C,M)
        RETURN
        END
C
C**********GENERATE PARAMETERS T*****************
C
        SUBROUTINE GENPRT(N,X1,Y1,Z1,T1,IDIM,M)
        DIMENSION X1(1),Y1(1),Z1(1),T1(1)
        T1(1)=0.
        DO 1 K=2,N
          U=X1(K)-X1(K-1)
          V=Y1(K)-Y1(K-1)
          IF(IDIM.EQ.3) Q=Z1(K)-Z1(K-1)
```

```
          D=U*U+V*V
          IF(IDIM.EQ.3)D=D+Q*Q
          D1=SQRT(D)
          IF(M.EQ.3.OR.M.EQ.4) GOTO 2
          T1(K)=T1(K-1)+D1
1       CONTINUE
        RETURN
2       T1(K)=D1
        GOTO 1
        END
C
C ************ COMPUTE INTERMEDIATE POINTS ********************
C
        SUBROUTINE INTRPT(INTPNT,N,N1,X1,Y1,Z1,T1,IDIM,M)
        DIMENSION N1(1),X1(1),Y1(1),Z1(1),T1(1)
        CALL GENPRT(N,X1,Y1,Z1,T1,IDIM,M)
        N2=N-1
        IF(INTPNT.LT.1) GOTO 101
        IF(INTPNT.EQ.999) GOTO  7
55      DO 1 I=1,N2
1        N1(I)=INTPNT
        IF(IFLAG.EQ.999) INTPNT=999
        IFLAG=0
        RETURN
101     W1=X1(N)-X1(1)
        W2=Y1(N)-Y1(1)
        IF(IDIM.EQ.3)W3=Z1(N)-Z1(1)
        D=W1*W1+W2*W2
        IF(IDIM.EQ.3)D=D+W3*W3
        TL=SQRT(D)
        S=TL/100.
        DO 2 I=1,N2
          N1(I)=T1(I+1)/S-1.
          N5=(200-N)/(N-1)
          IF(N1(I).GT.N5)N1(I)=N5
2       CONTINUE
        RETURN
7       INTPNT=5
        IFLAG=999
        GOTO 55
        END
```

APPENDIX 2.2

THE IDF - EXPLICIT PACKAGE

```
C                          *****************
C                          * APPENDIX 2.21 *
C                          *****************
C
C THIS MODULE HANDLES THE FOLLOWING INTERACTIVE DISPLAYS:-
C               1.INTRODUCTORY
C               2.CHOICE OF THE NUMERICAL ALGORITHMS
C               3.DATA ENTRY
C               4.DATA TABULATION
C               5.DATA POINT EDITING
C
C
C
C********* MAIN PROGARM-MODULE 1 **************
C
C INPUT COMMON DATA AREA
        COMMON/DATSUP/NPS,NPI(50),IFREES,X(50),Y(50),L(50),IH(5),M(5),
&                    METHOD,IHELP,IPREV,BOUND(2),SUBSET,INTPNT,IE(2)
        COMMON/IO/IN,IOUT
        DATA MODL2,HELP/"MOD2","HELP"/
        LOGICAL*1 MODL2(10),HELP(10)
        INTEGER SUBSET
        IN=5
        IOUT=6
        CALL IXOPEN
        IF(IERROR(103).NE.0)GOTO 1
        CALL RDCOM1
        IF(IHELP.NE.0) GOTO 55
        IF(IPREV.EQ.1) GOTO 21
C CALL THE DISPLAY SEQUENCE FOR DATA ENTRY ON REQUEST
1       CALL INTROD(IC)
        GOTO(2,3000,1111),IC
2       CALL MENU(METHOD,IC)
        GOTO(10,1,3000,1111,1111),IC
10      CALL DATENT(IC,IA)
        GOTO(20,2,3000,1111,1111),IC
20      CALL DATMAN(IA,IC)
        GOTO(1000,10,40,1111,1111,3000,1111,1111),IC
40      CALL EDIT(IC,IA)
C RETURN TO DATA MANIPULATION DISPLAY
        IF(IC.GT.1) GOTO 20
C NEXT DISPLAY IN THE SEQUENCE
1000    IPREV=0
        IHELP=0
        CALL WRCOM1
        CALL OVRLAY(MODL2)
C HELP DISPLAY
3000    CALL WRCOM1
        CALL OVRLAY(HELP)
C TERMINATE THE PROGRAM
1111    CALL EXIT
        STOP
C LINK LIST UNCHAGE
21      IPREV=0
        IHELP=0
        IA=111
        GOTO 20
C RETURN FROM HELP DISPLAY
55      GOTO(1,2,10,21),IHELP
        END
C
```

```
C ************INTRODUCTORY DISPLAY *****************************
C
      SUBROUTINE INTROD(IC)
      COMMON /IO/ IN,IOUT
C MENU ITEMS
      DATA MNTXT/"+ NEXT    + HELP    + EXIT    "/
      LOGICAL*1 MNTXT(30)
C SET UP THE INTRODUCTORY TEXT AND MENU
      CALL TXCLER
      CALL CURPOS(1.,780.)
      CALL TEXTUP("INTEXT",34)
      CALL MNOPEN(875.,715.,1)
      CALL MNDISP(MNTXT,3,10,1)
      CALL FRAME(870.,733.,3)
2     CALL MNPICK(J,ICHAR,MNO)
22    CALL CONFRM(ICHAR)
      IF(ICHAR.EQ.78) GOTO 2
      IF(ICHAR.NE.89) GOTO 22
      IC=J
      RETURN
      END
C
C *********** CHOICE OF DATA FITTING ALGORITHM ******************
C
      SUBROUTINE MENU(M,IC)
C THIS RETURN THE ALGORITHM INDEX IN THE MENU
      COMMON /IO/IN,IOUT
C MENU ITEM
      DATA MNTXT1 /"+ NEXT    + PREVIOUS+ HELP    + RESTART + EXIT    "/
      LOGICAL*1 MNTXT1(50)
7     CALL TXCLER
      WRITE(IOUT,10)
10    FORMAT("INDICATE YOUR CHOICE OF ALGORITHIM:-")
      CALL MNOPEN(875.,715.,1)
C OUTPUT ALGORITHM LIST
      CALL DTEXT(20.,700.,"***EXPLICIT PACKAGE FOR INTERPOLATORY DATA FITTING***",!
      CALL DTEXT(70.,650.,"NUMERICAL ALGORITHMS:-",22)
      CALL MNDISP(MNTXT1,5,10,1)
      CALL FRAME(870.,733.,5)
      CALL MNOPEN(60.,600.,2)
      CALL MNTEXT("1-GLOBAL POLYNOMIAL INTERPOLATION(NEWTON FORM)",46)
      CALL MNTEXT("2-PIECEWISE QUINTIC POLYNOMIAL INTERPOLATION
&(VARIABLE END CONDITION)",69)
      CALL MNTEXT("3-CUBIC SPLINE(SECOND DERIVATIVES END CONDITION)",48)
      CALL MNTEXT("4-CUBIC SPLINE(SECOND DERIVATIVES END CONDITION(*))",51)
      CALL MNTEXT("5-CUBIC SPLINE(FIRST DERIVATIVES END CONDITION)",47)
      CALL MNTEXT("6-CUBIC SPLINE(PERIODIC END CONDITION)",38)
      CALL MNTEXT("7-CUBIC SPLINE(VARIABLE END CONDITION)",38)
      CALL CURPOS(20.,300.)
      CALL ALPHMD
      WRITE(IOUT,888)
888   FORMAT(47H* WHERE Y"[X1]=U*Y"[X2] & Y"[X(N-1)]=V*Y"[X(N)])
C SET UP CURSOR FOR MENU CHOICE & CONFIRM
1     CALL MNPICK(I,ICHAR,MNO)
      IF(MNO.EQ.1)GOTO 3
      M=I
      GOTO 1
3     CALL CONFRM(ICHAR)
      IF(ICHAR.EQ.78) GOTO 1
      IF(ICHAR.NE.89) GOTO 3
      IF(I.EQ.4)GOTO 7
      IC=I
```

```
      RETURN
      END
C
C *************DATA ENTRY DISPLAY ROUTINE*****************
C
      SUBROUTINE DATENT(IC,IA)
      COMMON /DATSUP/NPS,NPI(50),IFREES,X(50),Y(50),L(50),IH(5),M(5)
&                    ,METHOD,IHELP,IPREV,BOUND(2),SUBSET,INTPNT,IE(2)
      COMMON /IO/IN,IOUT
C MENU ITEMS FOR THIS DISPLAY
      DATA MNTXT1/"+ NEXT    + PREVIOUS+ HELP    + RESTART + EXIT    "/
      DATA MNTXT2/"+ NEW     + OLD     +         + KEYBOARD+DISC FILE"/
      LOGICAL*1 MNTXT1(50),MNTXT2(50)
      INTEGER SUBSET
C SET DISPLAY READY FOR DATA ENTRY
23    CALL TXCLER
      WRITE(IOUT,10)
10    FORMAT("DATA ENTRY:-")
      CALL CURPOS(1.,625.)
      WRITE(IOUT,11)
C OUTPUT INSTRUCTIONS FOR THE USER
11    FORMAT("SELECT THE APPROPRIATE"/
&     "DATA SPECIFICATION(*):-"////"1-STATE OF DATA:-"///"2-DATA",
&" MEDIUM ENTRY:-")
      WRITE(IOUT,20)
20    FORMAT(///////////////"* IF 'OLD' IS SELECTED YOU MAY PROCEED"/
&     "  TO NEXT DISPLAY IMMEDIATELY .OTHERWISE "/
&     "  YOU MUST SELECT THE DESIRED MEDIUM")
      CALL MNOPEN(875.,715.,1)
      CALL MNDISP(MNTXT1,5,10,1)
      CALL FRAME(870.,733.,5)
C DISPLAY MENU,RAISE CURSOR AND WAIT FOR USER ACTION
      CALL MNOPEN(320.,515.,2)
      CALL MNDISP(MNTXT2,5,10,2)
      CALL FRAME(315.,535.,5)
      CALL TXMOVE(315.,475.)
      CALL TXDRAW(460.,475.)
      NFLAG=1
      MD   =0
      IS   =0
5     CALL MNPICK(J,ICHAR,MNO)
C TRANSFER CONTROL TO APPROPRIATE PROGRAM RESPONSE
      IF(MNO.EQ.1) GOTO 2
      IF(IS.EQ.2.OR.NFLAG.EQ.0.OR.J.EQ.3) GOTO 5
      IF(J.LT.3) GOTO 7
      MD=J
      CALL CURPOS(1.,400.)
3     CALL MESSAG("# NUMBER OF DATA POINTS(MAX.50)?^")
      READ(IN,30)N
30    FORMAT(G0.0)
      IF(N.GT.50.OR.N.LT.3) GOTO 3
      NPS=N
      IF(J.EQ.5)GOTO 1010
C INPUT DATA POINTS FROM KEYBOARD
      WRITE(IOUT,40)
40    FORMAT(/,"# X-COORDS.:-")
      IF(IERROR(110).NE.0)GOTO 100
  35  READ(IN,50) (X(I),I=1,N)
50    FORMAT(50G0.0)
      WRITE(IOUT,70)
70    FORMAT(/,"# Y-COORDS.:-")
      IF(IERROR(110).NE.0)GOTO 110
```

```
65      READ(IN,50)(Y(I),I=1,N)
        IHELP=3
        GOTO 5
100     WRITE(IOUT,105)
105     FORMAT("ILLEGAL X-COORDS.,TRY AGAIN")
        ENDFILE 5
        GOTO 35
110     WRITE(IOUT,115)
115     FORMAT("ILLEGAL Y-COORDS.,TRY AGAIN")
        ENDFILE 5
        GOTO 65
2       CALL CONFRM(ICHAR)
        IF(ICHAR.EQ.78) GOTO 5
        IF(ICHAR.NE.89)GOTO 2
        IF(J.EQ.4)GOTO 23
        IF(NFLAG.EQ.1.AND.J.EQ.1.AND.MD.EQ.0) GOTO 5
        IC=J
        RETURN
1010    IFLG=0
C NEW DATA POINTS
        IA=0
C INPUT DATA FROM DISC FILE
        CALL GETFLN(FILE)
        CALL READAT(FILE,X,Y,NPS,IFLG)
        IF(IFLG.EQ.1)GOTO 1010
        IF(IFLG.EQ.2)GOTO 3
        GOTO 5
313     ENDFILE 5
        GOTO 3
C NEW DATA POINTS
7       IA=0
C OLD DATA POINTS
        IF(J.EQ.2)IA=111
        IS=J
        IF(J.EQ.2.AND.MD.GT.0) GOTO 5
        IF(J.EQ.2.AND.MD.EQ.0) GOTO 12
        IF(J.EQ.1) GOTO 5
12      NFLAG=0
        GOTO 5
        END
C
C ***********DATA POINT TABULATION DISPLAY ROUTINE**************
C
        SUBROUTINE DATMAN(IA,IC)
C INPUT COMMON DATA AREA
        COMMON/DATSUP/NPS,NPI(50),IFREES,X(50),Y(50),L(50),IH(5),M(5)
&                          ,METHOD,IHELP,IFREV,BOUND(2),SUBSET,INTPNT,IE(2)
        COMMON/IO/IN,IOUT
C MENU ITEMS
        DATA MNTXT/"+ NEXT     + PREVIOUS+ EDIT     + SORT-X   + SAVE
&+ HELP     + RESTART + EXIT      "/
        DIMENSION XO(50),YO(50)
        LOGICAL*1 MNTXT(80)
        INTEGER SUBSET,S
C OLD DATA POINTS?
22      IF(IA.EQ.111)GOTO7
111     N=NPS-1
C SET LINK LIST
        DO 1 I=1,N
1       L(I)=I+1
        L(NPS)=0
        IFREES=NPS+1
```

```
      DO 2 I=1,5
2     M(I)=10*I
      S=0
      DO 3 I=1,5
      IH(I)=S*10+1
3     S=S+1
C SET DISPLAY MENU,AND DATA POINT TABLE
7     CALL TXCLER
      WRITE(IOUT,10)
10    FORMAT("TABULATION OF DATA:-")
      CALL MNOPEN(875.,715.,1)
      CALL MNDISP(MNTXT,8,10,1)
      CALL FRAME(870.,733.,8)
      CALL TABLE
      IHELP=4
C PROMPT USER FOR CURSOR PICKING ACTION AND CONFIRM
4     CALL MNPICK(J,ICHAR,MNO)
77    CALL CONFRM(ICHAR)
      IF(ICHAR.EQ.78)GOTO 4
      IF(ICHAR.NE.89) GOTO 77
      IF(J.EQ.4) GOTO 44
      IF(J.EQ.5) GOTO 444
      IF(J.EQ.7) GOTO 22
      IC=J
      RETURN
C SORT DATA POINTS IN X
44    CALL REMLNK(X0,Y0)
      CALL SORTX(X0,Y0,NPS)
      DO 101 I=1,NPS
      X(I)=X0(I)
      Y(I)=Y0(I)
101   CONTINUE
      GOTO 111
C SAVE DATA POINT IN DISC FILE
444   CALL REMLNK(X0,Y0)
      CALL SAVE(X0,Y0,NPS)
      GOTO 4
      END
C
C ***********EDITOR DISPLAY******************************
C
      SUBROUTINE EDIT(IC,IA)
      COMMON/DATSUP/NPS,NPI(50),IFREES,X(50),Y(50),L(50),IH(5),M(5)
&                    ,METHOD,IHELP,IPREV,BOUND(2),SUBSET,INTPNT,IE(2)
      COMMON/IO/IN,IOUT
C MENU ITEMS
      DATA MNTXT/"+ NEXT    + PREVIOUS+ CORRECT + DELETE  + INSERT
&+ RESTART + EXIT    "/
      DIMENSION A(31)
      INTEGER SUBSET
      LOGICAL*1 MNTXT(70)
1     CALL TXCLER
      WRITE(IOUT,10)
10    FORMAT("DATA POINTS EDITING:-")
C SET UP DISPLAY MENU
      CALL MNOPEN(875.,715.,1)
      CALL MNDISP(MNTXT,7,10,1)
      CALL FRAME(870.,733.,7)
C RAISE CURSOR READY FOR USER INTERACTION AND CONFIRM ACTION
77    CALL MNPICK(J,ICHAR,MNO)
88    CALL CONFRM(ICHAR)
      IF(ICHAR.EQ.78) GOTO 77
```

```
      IF(ICHAR.NE.89) GOTO 88
C ACTIVATE APPROPRIATE ROUTINES OR OPTIONS
      GOTO(30,30,40,50,60,1,70),J
C NEXT/PREVIOUS DISPLAY
30    CALL UPDATE
      IC =J
      GOTO 114
C CORRECT DATA POINTS
40    WRITE(IOUT,20)
20    FORMAT(////"CORRECTION:-")
C PROMT USER FOR INPUTING EDIT INFORMATION
5     CALL MESSAG("# NUMBER OF DATA POINTS(MAX.10)?^")
      READ(IN,45) M2
45    FORMAT(G0.0)
      IF(M2.GT.10)GOTO 5
221   WRITE(IOUT,80)
80    FORMAT("# ENTER I , X , Y:")
      LAST=3*M2+1
      IF(IERROR(110).NE.0)GOTO 222
      READ(IN,90)(A(I),I=1,LAST)
90    FORMAT(30G0.0)
      A(LAST)=99
      CALL CORECT(A)
      CALL UPDATE
      GOTO 100
C DELETE DATA POINTS
50    WRITE(IOUT,110)
110   FORMAT(////"DELETION:-")
C USER SUPPLIES EDIT-DELETE INPUT
7     CALL MESSAG("# NUMBER OF DATA POINT(MAX.30)?^")
      READ(IN,45)M2
      IF(M2.GT.30)GOTO 7
125   WRITE(IOUT,130)
130   FORMAT("# ENTER I IN DESCENDING ORDER:")
      LAST=M2+1
      IF(IERROR(110).NE.0)GOTO 333
      READ(IN,90)(A(I),I=1,LAST)
      A(LAST)=99
      IF(M2.EQ.1)GOTO 11
      M1=M2-1
      DO 9 I=1,M1
      IF(A(I).LT.A(I+1))GOTO 125
9     CONTINUE
11    CALL DELETE(A)
      CALL UPDATE
      GOTO 100
C ADDITION DATA POINTS
60    WRITE(IOUT,150)
150   FORMAT(////"INSERTION:-")
155   CALL MESSAG("# NUMBER OF DATA POINTS(MAX.1 PER INTERVAL,TOTAL 10)?^")
      READ(IN,45)M2
      IF(M2.GT.10) GOTO 155
165   WRITE(IOUT,170)
170   FORMAT("# ENTER I , X , Y  IN DESCENDING ORDER:")
      LAST=3*M2+1
      IF(IERROR(110).NE.0)GOTO 444
      READ(IN,90)(A(I),I=1,LAST)
      A(LAST)=99
      IF(M2.EQ.1)GOTO 190
      LAST1=3*M2-5
      DO 18 I=1,LAST1,3
      IF(A(I).LT.A(I+3)) GOTO 165
```

```
18      CONTINUE
190     CALL ADD(A)
        CALL UPDATE
100     IC=3
114     IA=111
        RETURN
222     LFLAG=1
224     WRITE(IOUT,223)
223     FORMAT("WRONG INPUT !,TRY AGAIN")
        ENDFILE 5
        GOTO(221,125,165),LFLAG
333     LFLAG=2
        GOTO 224
444     LFLAG=3
        GOTO 224
70      STOP
        END
C
C********** H E L P  D I S P L A Y ******************
C
        COMMON/DATSUP/NPS,NPI(50),IFREES,X(50),Y(50),L(50),IH(5),M(5)
&                ,METHOD,IHELP,IPREV,BOUND(2),SUBSET,INTPNT,IE(2)
        COMMON/IO/IN,IOUT
C MENU ITEMS ,EXECUTABLE PROGRAM MODULES
        DATA MNTXT1/"+ PREVIOUS+ EXIT     "/
        DATA MODL1,MODL2,MODL3,MODL4,MODL5,MODL6,MODL7/"EXPLICIT","MOD2"
&,"MOD3","MOD4","MOD5","MOD6","MOD7"/
        LOGICAL*1 MNTXT1(20),MODL1(10),MODL2(10),MODL3(10),MODL4(10)
        LOGICAL*1 MODL5(10),MODL6(10),MODL7(10)
        IN=5
        IOUT=6
        CALL TXOPEN
        CALL RDCOM1
C OUTPUT DISPLAY SEQUENCE NAMES IN MENU FORMAT
1       CALL TXCLER
        WRITE(IOUT,10)
10      FORMAT(10X,"**************** H E L P ********************"/
&       //5X,"THE FOLLOWING DISPLAY SEQUENCE CONSTITUTE THE COMPLETE"/
& 5X,"DATA FITTING PROCESS."//
&       5X,"YOU MAY ENTER ANY OF THESE DISPLAY BY USING THE CROSS-HAIR"/
& 5X,"CURSOR ON THE T4010 OR TRACKING CROSS WITH LIGHT-PEN ON THE GT42 :-"/)
        CALL MNOPEN(50.,540.,1)
        CALL MNTEXT("+ INTRODUCTION:- BRIEFLY GIVING THE USE OF THE SYST
&EM.",54)
        CALL MNTEXT("+ ALGORITHMS:- LIST OF AVAILABLE INTERPOLATORY METH
&ODS.",55)
        CALL MNTEXT("+ DATA ENTRY:- ENTER DATA POINTS INTO THE SYSTEM FR
&OM DISC FL/KEYBD.",68)
        CALL MNTEXT("+ TABULATION OF DATA POINTS:- INCLUDES EDIT,SORT &
&SAVE DATA POINTS.",68)
        CALL MNTEXT("+ POLYGONAL PLOT:- DATA POINTS JOINED BY STRAIGHT L
&INE SEGMENTS.",64)
        CALL MNTEXT("+ PARAMETER ENTRY:- PARAMETERS REQUIRED BY PARTICUL
&AR ALGORITHM.",65)
        CALL MNTEXT("+ CURVE FIT:- DISPLAY OF THE SMOOTH CURVE INCLUDES
&ZOOM OPTION..ETC.",69)
        CALL MNTEXT("+ CURVE DESIGN:- INTERMEDIATE POINTS SPECIFIED
& BY CURSOR POSITION.",66)
        CALL MNTEXT("+ TABLE OF INTERPOLATED POINTS:- INCLUDES OPTION FO
&R COEFF.&HARDCOPY.",70)
        CALL MNTEXT("+ SUPERIMPOSED CURVE:- SIMULTANEOUS DISPLY OF SEVER
&AL CURVES.",63)
```

```
        CALL MNTEXT("+ ERROR REFFERENCE:- ERROR CURVE W.R.T. GLOBAL POLY
&NOMIAL INTERP..",67)
        CALL MNTEXT("+ JOINED CURVES:- CONCATENATION OF SEVERAL SEGMENT
&OF CURVES.",60)
        CALL MNTEXT("+ USAGE OF CONTROL COMMANDS:- LIST OF ALL COMMAND U
&SED HERE.",59)
        CALL MNTEXT("+ TERMINATE THE PROCESS:- EXIT FROM THE SYSTEM.",47)
2       CALL MNPICK(I,ICHAR,MNO)
3       CALL CONFRM(ICHAR)
        IF(ICHAR.EQ.78) GOTO 2
        IF(ICHAR.NE.89) GOTO 3
        IF(I.GT.4) GOTO 5
        IHELP=I
C READ THE APPROPRIATE MODULE INTO THE CORE READY FOR EXECUTION
        CALL WRCOM1
        CALL OVRLAY(MODL1)
5       IF(I.GT.6) GOTO 6
        IHELP=I-4
        CALL WRCOM1
        CALL OVRLAY(MODL2)
6       IF(I.LT.9.OR.I.GT.11) GOTO 7
        IHELP=I-8
        CALL WRCOM1
        CALL OVRLAY(MODL6)
7       IF(I.EQ.7) CALL OVRLAY(MODL4)
        IF(I.EQ.8) CALL OVRLAY(MODL5)
        IF(I.EQ.12) CALL OVRLAY(MODL7)
        IF(I.NE.14) GOTO 133
        CALL EXIT
        STOP
C GOTO NEXT DISPLAY,GIVING COMMAND USAGE & OPTIONS
133     CALL TXCLER
        CALL MNOPEN(875.,715.,1)
        CALL MNDISP(MNTXT1,2,10,1)
        CALL FRAME(870.,733.,2)
        CALL ALPHMD
        CALL CURPOS(1.,770.)
        CALL TEXTUP("HELPTEXT",28)
22      CALL MNPICK(J,ICHAR,MNO)
222     CALL CONFRM(ICHAR)
        IF(ICHAR.EQ.78) GOTO 22
        IF(ICHAR.NE.89) GOTO 222
        GOTO (1,111),J
111     CALL EXIT
        STOP
        END
C
C ***********EDIT-INSERT FUNCTION*****************
C
        SUBROUTINE ADD(C)
        COMMON/DATSUP/NPS,NPI(50),IFREES,X(50),Y(50),L(50),IH(5),M(5),
&                     METHOD,IHELP,IPREV,BOUND(2),SUBSET,INTPNT,IE(2)
        DIMENSION C(31)
        INTEGER SUBSET
C ADDS DATA POINT TO LINK LIST DATA STRUCTURE
        IP = IFREES
        DO 3 I=1,31,3
C CHECKS DATA POINT TABLE INDEX
          IF(C(I).EQ.99) GOTO 2
          IF(C(I).GT.NPS)GOTO 3
          IF(C(I).EQ.0)GOTO 4
          IC=C(I)
```

```
C GET LINK LIST LOCATION OF THE DATA POINT AND SET LINK APPROPRIATELY
          IS=INDEX(IC)
          L(IP)=L(IS)
          L(IS)=IP
          GOTO 5
4         L(IP)=IH(1)
          IH(1)=IP
C ADD DATA POINTS TO FREE LOCATION IN THE LINK LIST
5         X(IP)=C(I+1)
          Y(IP)=C(I+2)
          IP=IP+1
          NPS=NPS+1
3       CONTINUE
C SET FREE LINK LIST POINTER
2       IFREES=IP
        RETURN
        END
C
C ************EDIT - CORRECT FUNCTION***********************
C
        SUBROUTINE CORECT(C)
        COMMON/DATSUP/NPS,NPI(50),IFREES,X(50),Y(50),L(50),IH(5),M(5)
&                     ,METHOD,IHELP,IPREV,BOUND(2),SUBSET,INTPNT,IE(2)
        DIMENSION C(31)
        INTEGER SUBSET
        DO 2 I=1,31,3
C CHECKS DATA POINTS TABLE INDEX
          IF(C(I).EQ.99)GOTO3
          IF(C(I).GT.NPS) GOTO 2
          IC=C(I)
C GET LINK LIST LOCATION AND REPLACE DATA POINT
          K=INDEX(IC)
          X(K)=C(I+1)
          Y(K)=C(I+2)
2       CONTINUE
3       RETURN
        END
C
C *************EDIT-DELETE FUNCTION*********************
C
        SUBROUTINE DELETE(C)
        COMMON/DATSUP/NPS,NPI(50),IFREES,X(50),Y(50),L(50),IH(5),M(5)
&                     ,METHOD,IHELP,IPREV,BOUND(2),SUBSET,INTPNT,IE(2)
        DIMENSION C(1),X0(50),Y0(50),L0(50)
        INTEGER SUBSET
        DO 1 I=1,31
C CHECKS DATA POINTS TABLE INDEX
          IF(C(I).EQ.99)GOTO 3
          IF(C(I).GT.NPS)GOTO 5
          IF(C(I).EQ.1) GOTO 4
          IC=C(I)-1
          IS=INDEX(IC)
C GET LINK LIST LOCATION AND DELETE DATA POINT
          L(IS)=L(L(IS))
          GOTO 1
5         NPS=NPS+1
          GOTO 1
4         IH(1)=L(IH(1))
1       CONTINUE
3       NPS=NPS-I+1
        IFREES=IFREES-I+1
C GARBAGE COLLECTION
```

```
      IP=IH(1)
      DO 6 K=1,NPS
        XO(K)=X(IP)
        YO(K)=Y(IP)
        LO(K)=K+1
        IP=L(IP)
6     CONTINUE
      DO 7 J=1,NPS
        X(J)=XO(J)
        Y(J)=YO(J)
        L(J)=LO(J)
7     CONTINUE
      L(NPS)=0
      IS=0
      DO 9 K=1,5
        IH(K)=IS*10+1
        IS=IS+1
9     CONTINUE
      RETURN
      END
C
C **************KEYBOARD ENTRY OF FILE NAME*****************
C
      SUBROUTINE GETFLN(NAME)
C GET FILE NAME & SAVE IT IN ARRAY NAME
      INTEGER NAME(3)
      CALL MESSAG("÷ DATA FILE NAME(MAX.10 CHARACTERS)?^")
      READ(5,30)NAME
30    FORMAT(2A4,A2)
      RETURN
      END
C
C ***********FINDS DATA POINT LOCATION IN THE LINK LIST************
C
      INTEGER FUNCTION INDEX(INX )
C INPUT COMMON DATA AREA
      COMMON/DATSUP/NPS,NPI(50),IFREES,X(50),Y(50),L(50),IH(5),M(5)
&                 ,METHOD,IHELP,IPREV,BOUND(2),SUBSET,INTPNT,IE(2)
      INTEGER SUBSET
C FIND LINK LIST PORTION?
      DO 1 I=1,5
      IF(M(I).GE.INX)GOTO 2
1     CONTINUE
2     IS=IH(I)
C COMPUTE LOCATION
      INX=INX-(I-1)*10
      INX1=INX-1
      IF(INX1.EQ.0) GOTO 55
      DO 3 I=1,INX1
3     IS=L(IS)
55    INDEX=IS
      RETURN
      END
C
C **********DATA ENTRY FROM DISC FILE******************
C
      SUBROUTINE READAT(FLNAME,A,B,N,IF)
      DIMENSION A(1),B(1)
      INTEGER FLNAME(3)
      REWIND 9
C OPEN INPUT FILE
      CALL SETFIL(9,FLNAME)
```

```
      IF(IERROR(103).NE.0)GOTO 99
C NUMBER OF POINT SUPPLIED
      READ (9,20)N1
20    FORMAT(I2)
      IF(N.GT.N1)GOTO 100
C INPUT THE DATA POINTS
      READ(9,10)(A(I),B(I),I=1,N )
10    FORMAT(F12.4)
      ENDFILE 9
      RETURN
99    ENDFILE 5
101   IF=1
      RETURN
100   IF=2
      RETURN
      END
C
C *************SAVE DATA POINTS ON DISC FILE*****************
C
      SUBROUTINE SAVE(A,B,N)
      COMMON/IO/IN,IOUT
      INTEGER FILE(3)
      DIMENSION A(1),B(1)
C GET FILE NAME FROM THE USER THROUGH KEYBOARD
      WRITE(IOUT,10)
10    FORMAT(//////////60X,"# FILE NAME?")
      CALL MESSAG("
&         ^")
      READ(IN,20)FILE
20    FORMAT( 2A4,A2)
C WRITE OUT DATA POINTS
      CALL WRTDAT(FILE,A,B,N)
      RETURN
      END
C
C ************** SORT IN X COORDINATES**********************************
C
      SUBROUTINE SORTX(X1,Y1,N)
      DIMENSION X1(1),Y1(1)
      N1=N-1
C PERFORM THE SORT
      DO 3 I=1,N1
         DO 2 J=I,N1
           IF(X1(I).LE.X1(J+1))GOTO 2
           A1=X1(I)
           B1=Y1(I)
           X1(I)=X1(J+1)
           Y1(I)=Y1(J+1)
           X1(J+1)=A1
           Y1(J+1)=B1
2       CONTINUE
3     CONTINUE
      RETURN
      END
C
C ***********TABULATION OF DATA POINTS**********************
C
      SUBROUTINE TABLE
C INPUT COMMON DATA AREA
      COMMON /DATSUP/NPS,NPI(50),IFREES,X(50),Y(50),L(50),IH(5),M(5)
&                 ,METHOD,IHELP,IPREV,BOUND(2),SUBSET,INTPNT,IE(2)
      COMMON /IO/IN,IOUT
```

```
      INTEGER S1,S2,SUBSET
      CALL CURPOS(2.,710.)
C OUTPUT DATA COLUMN HEADINGS
      WRITE(IOUT,10)
10    FORMAT(" I",7X,"X(I)",8X ,"Y(I)",7X," I",5X,"X(I)",10X,"Y(I)"/)
      S2=IH(1)
      S1=L(S2)
      IF(S1.NE.0)GOTO12
      WRITE(IOUT,11) X(S2),Y(S2)
11    FORMAT(" 1",2X,2(E11.4,3X))
      RETURN
C OUTPUT TABLE ITEMS FROM LINK LIST
12    DO 7 I=1,50,2
        J=I+1
        WRITE(IOUT,20) I,X(S2),Y(S2),J,X(S1),Y(S1)
20      FORMAT(I2,2X,2(E11.4,3X),I2,2X,2(E11.4,3X))
        S2=L(S1)
        IF(S2.EQ.0)GOTO 15
        S1=L(S2)
        IF(S1.NE.0)GOTO 7
        I=I+2
        WRITE(IOUT,30) I,X(S2),Y(S2)
30      FORMAT(I2,2X,2(E11.4,3X))
        GOTO 15
7     CONTINUE
15    RETURN
      END
C
C ************UPDATE LINK LIST AFTER AN EDITING OPERATION**********
C
      SUBROUTINE UPDATE
C INPUT COMMON AREA
      COMMON/DATSUP/NPS,NPI(50),IFREES,X(50),Y(50),L(50),IH(5),M(5)
&                   ,METHOD,IHELP,IPREV,BOUND(2),SUBSET,INTPNT,IE(2)
      INTEGER SUBSET
      IR=IH(1)
      DO 1 J=2,5
         DO 2 K=1,10
            IR=L(IR)
              IF(IR.EQ.0)GOTO3
2          CONTINUE
        IH(J)=IR
1     CONTINUE
3     RETURN
      END
C
C *************OUTPUT DATA POINT ON DISC FILE**************
C
      SUBROUTINE WRTDAT(FLNAME,A,B,N)
      DIMENSION A(1),B(1)
      INTEGER FLNAME(3)
      REWIND 9
C OPEN OUTPUT FILE
      CALL SETFIL(9,FLNAME)
C STORE DATA POINT ON THE FILE
      WRITE(9,20)N
20    FORMAT(I2)
      WRITE(9,10)(A(I),B(I),I=1,N)
10    FORMAT(F12.4)
      ENDFILE 9
      RETURN
      END
```

```
C                          *****************
C                          * APPENDIX 2.22 *
C                          *****************
C
C THIS MODULE HANDLES THE FOLLOWING INTERACTIVE DISPLAYS:
C                 1.POLYGONAL PLOT
C                 2.PARAMETER ENTRY
C
C
C
C*********** MAIN PROGRAM-MODULE 2 **********************
C
C INPUT COMMON DATA AREA
        COMMON/DATSUP/NPS,NPI(50),IFREES,X(50),Y(50),L(50),IH(5),M(5)
&                    ,METHOD,IHELP,IPREV,BOUND(2),SUBSET,INTPNT,IE(2)
        COMMON/IO/IN,IOUT
C EXECUTABLE PROGRAM MODULE NAMES
        DATA MODL3,HELP,MODL1/"MOD3","HELP","EXPLICIT"/
        LOGICAL*1 MODL1(10),MODL3(10),HELP(10)
        INTEGER SUBSET
C READ IN COMMON DATA FILE
        CALL RDCOM1
C INITIALISE DISPLAY TERMINAL
        CALL TXOPEN
        IN=5
        IOUT=6
        IF(IHELP.NE.0) GOTO 6
        IF(IPREV.GT.1) GOTO 5
        IF(IPREV.EQ.1)GOTO 4
C POLYGONAL DISPLAY
3       CALL POLYGL(X,Y,NPS,IC)
C PASS CONTROL TO APPROPRIATE PART OF THE PROGRAM
        GOTO(1,2,20,20,20,20,25,20,30),IC
C PREVIOUS DISPLAY
2       IPREV=1
        IHELP=0
C SAVE INPUT COMMON DATA AREA
        CALL WRCOM1
        CALL OVRLAY(MODL1)
C BRINGS THE PARAMETER ENTRY DISPLAY
 1      CALL PARMET(METHOD,SUBSET,INTPNT,BOUND,NPS,IE,IC)
        CALL CURPOS(410.,780.)
        GOTO(10,3,25,1,30),IC
C NEXT DISPLAY
10      IPREV=0
        IHELP=0
        CALL [illegible](MODL3)
C TERMINATE PROGRAM
30      CALL EXIT
20      STOP
4       IPREV=0
        IHELP=0
        GOTO 1
C HELP DISPLAY
25      CALL WRCOM1
        CALL OVRLAY(HELP)
        STOP
5       CALL ERRMES(IC)
        GOTO (1,20),IC
C RETURN FROM HELP
6       GOTO(3,1),IHELP
```

```
      END
C
C **************** POLYGONAL PLOT OF DATA POINTS*************
C
      SUBROUTINE POLYGL(A,B,N,IC)
C PLOT THE DATA POINTS SUPPLIED AND JOINED WITH STRAIGHT LINES
      COMMON/IO/IN,IOUT
C MENU ITEMS
      DATA MNTXT/"+ NEXT    + PREVIOUS+ GRAPH   + COORDS. + DISP.ORG+
&AX. MARK+ HELP    + RESTART + EXIT    "/
      LOGICAL*1 MNTXT(90)
      DIMENSION A(1),B(1),XO(50),YO(50)
      EXTERNAL PPLOT
C PREPARE DISPLAY
1     CALL TXCLER
      ICORD=0
      WRITE(IOUT,10)
10    FORMAT("POLYGONAL PLOT :-")
C OUTPUT MENU ITEMS
      CALL MNOPEN(875.,715.,1)
      CALL MNDISP(MNTXT,9,10,1)
      CALL FRAME(870.,733.,9)
      CALL REMLNK(XO,YO)
C FIND MAXIMUM & MINIMUM OF DATA POINTS
      CALL MINMAX(S1,S2,S3,S4,XO,YO,N)
C SET UP CURSOR FOR USER ACTION
2     CALL LMTARA
      CALL MNPICK(J,ICHAR,MNO)
      IF(J.EQ.4.AND.ICORD.EQ.2)GOTO2
C CONFIRM USER ACTION
7     CALL CONFRM(ICHAR)
      IF(ICHAR.EQ.78)GOTO 2
      IF(ICHAR.NE.89) GOTO 7
      GOTO(20,20,3,4,5,6,20,1,20),J
C BACK TO CALLING PROGRAM
20    IC=J
      RETURN
C GRAPH(PLOT THE CURVE)
3     CALL DRAW(PPLOT,R,S1,S2,S3,S4,N)
      GOTO 2
C DISPLAY CURSOR COORDINATE INPUT
 4    CALL DISCOR(ICORD,S1,S2,S3,S4)
      GOTO 2
C DISPLAY ORIGIN
 5    CALL DISORG(S1,S2,S3,S4)
      GOTO 2
C AXES MARKING
6     CALL AXSMRK(S1,S2,S3,S4)
      GOTO 2
      END
C
C ************** PARAMETER DISPLAY ENTRY ************************
C
      SUBROUTINE PARMET(M,SUBT,INTPNT,b,N,IE,IC)
      COMMON/IO/IN,IOUT
C MENU ITEMS
      DATA MNTXT1/"+ NEXT    + PREVIOUS+ HELP    + RESTART + EXIT    "/
      DATA MNTXT2/"+ COMPLETE+ SUBSET  + SUBSET
&+ SPECIFY + DEFAULT + CURSOR  "/
      DATA MNTXT3/"+ CLAMPED + RELAXED + PARABOLA+ Q-SPLINE"/
      DATA MNTXT4/"+ EQ.SPACE+ NE.SPACE          "/
      DATA MNTXT5/"+ BOUNDARY"/
```

```
      LOGICAL*1 MNTXT1(50),MNTXT2(80),MNTXT3(40),MNTXT4(30),
&                MNTXT5(10)
      INTEGER SUBT
      DIMENSION B(2),IE(2)
1     CALL TXCLER
C SET DEFAULT VALUES
      SUBT=0.
      IQ=0
      INTPNT=0.
      B(1)=0.
      B(2)=0.
      IF(M.NE.2) GOTO 112
      IE(1)=21
      IE(2)=1
      GOTO 1011
112   IE(1)=1.
      IE(2)=1.
C SET DISPLAY READY
1011  WRITE(IOUT,10)
10    FORMAT("PARAMETER ENTRY:-"/)
      GOTO (101,102,103,104,105,106,108),M
20    CALL CURPOS(1.,660.)
      WRITE(IOUT,21)
C DISPLAY INSTRUCTIONS TO GUIDE THE USR
21    FORMAT("SELECT THE APPROPRIATE OPTION FROM THE "/
&     "FOLLOWING PARAMETER SPECIFICATION(*) :-"///
&     6X,"1-DATA TYPE:-"/8X,"(FOR JOINING",22X,"FIRST"/
&     9X,"CURVES)",22X,"SUBSEQUENT"//6X,"2-CHOICE OF INTERMEDIATE"/
&     7X," POINTS REQUIRED FOR SMOOTH DRAWING:-"//)
      IF(M.NE.7) GOTO 1220
      WRITE(IOUT,15)
15    FORMAT(//6X,"3-SELECT THE CONDITION")
1433  WRITE(IOUT,1333)
1333  FORMAT(7X," AT EACH END OF THE CURVE:-"/
&              8X,"N.B:-LEFT-HAND END TYPE '1'"/
&              11X," RIGHT-HAND END TYPE '2'")
      GOTO 143
1220  IF(M.NE.2) GOTO 1221
      WRITE(IOUT,14)
14    FORMAT(/6X,"3- X-SPACING OF DATA POINTS SUPPLIED:-"/)
      WRITE(IOUT,16)
 16   FORMAT(/6X,"4-INDICATE YOUR CHOICE OF END")
      GOTO 1433
1221  IF(M.EQ.1.OR.M.EQ.6) GOTO 142
      WRITE(IOUT,22)
22    FORMAT(/6X,"3-BOUNDARY CONDITION:-"//)
143   WRITE(IOUT,7)
7     FORMAT(////////"* IF NO APPROPRIATE PARAMETER IS SELECTED "
&     /"WHEN REQUIRED THEN  DEFAULT VALUES ARE ASSUMED")
C DISPLAY CONTROL COMMAND
      CALL MNOPEN(875.,715.,1)
      CALL MNDISP(MNTXT1,5,10,1)
      CALL FRAME(870.,733.,5)
      CALL MNOPEN(670.,565.,2)
      CALL MNDISP(MNTXT2,8,10,2)
      CALL DTEXT(810.,565.,'*',1)
      CALL DTEXT(810.,455.,'*',1)
      IF(M.NE.7) GOTO 919
      CALL MNDISP(MNTXT3,4,10,2)
      CALL DTEXT(810.,365.,'*',1)
      MF=12
      GOTO 88
```

```
919     IF(M.NE.2) GOTO 99
        CALL MNDISP(MNTXT4,3,10,2)
        CALL DTEXT(810.,365.,'*',1)
        CALL MNDISP(MNTXT3,3,10,2)
        CALL DTEXT(810.,300.,'*',1)
        MF=14
        GOTO 88
142     WRITE(IOUT,1420)
1420    FORMAT(///)
        GOTO 143
99      MF=7
        IF(M.EQ.1.OR.M.EQ.6) GOTO 88
        CALL MNDISP(MNTXT5,1,10,2)
        MF=9
88      CALL FRAME(663.,582.,MF)
        CALL TXMOVE(665.,505.)
        CALL TXDRAW(805.,505.)
        IF(M.EQ.1.OR.M.EQ.6) GOTO 2
        CALL TXMOVE(665.,420.)
        CALL TXDRAW(805.,420.)
        IF(M.NE.2) GOTO 2
        CALL TXMOVE(665.,350.)
        CALL TXDRAW(805.,350.)
2       CALL MNPICK(J,ICHAR,MNO)
C WHICH MENU ?
        IF(MNO.EQ.2) GOTO 130
C CONFIRM USER REQUEST
5       CALL CONFRM(ICHAR)
        IF(ICHAR.EQ.78)GOTO 2
        IF(ICHAR.NE.89)GOTO 5
        IF(J.EQ.4) GOTO 1
        IF(M.NE.2) GOTO 707
        IF(IQ.EQ.10)IE(1)=IE(1)+10
        IF(IQ.EQ.20.OR.IQ.EQ.0)IE(1)=IE(1)+20
707     IC=J
        RETURN
C PARAMETER SPECIFICATION CONTROL
130     GOTO(31,32,33,34,35,36,37,38,39,40,41,42,43,43),J
C COMPLETE CURVE
31      SUBT=0
        GOTO 2
C CURVE SEGMENT
32      SUBT=1
        GOTO 2
C CURVE SEGMENT & SUBSEQUENT SUBT
33      SUBT=2
34      GOTO 2
C USER SPECIFIED NUMBER OF INTERMEDIATE POINTS PER INTREVAL
35      CALL CURPOS(1.,220.)
66      CALL MESSAG("# NUMBER OF INTERPOLATED POINTS PER INTERVAL?^")
        IF(IERROR(110).NE.0) GOTO 171
        READ(IN,77) INTPNT
77      FORMAT(G0.0)
        ISUM=INTPNT*(N-1)+N
        IF(ISUM.GT.200) GOTO 187
        GOTO 2
171     ENDFILE 5
        GOTO 66
187     WRITE(IOUT,403)
403     FORMAT("TOTAL NUMBER OF POINT EXCEEDING LIMIT,TRY AGAIN")
        GOTO 66
C DEFAULT OPTIONS
```

```
36      INTFNT=0
        GOTO2
C CURSOR
37      INTFNT=999
38      GOTO2
C BOUNDARY CONDITION/X-SPACING
39      GOTO(2,399,139,139,139,2,391),M
        GOTO 2
391     IF(ICHAR.EQ.49.OR.ICHAR.EQ.50) GOTO 392
        GOTO 2
392     ICHAR=ICHAR-48
        GOTO(272,202),ICHAR
        GOTO 2
272     IE(1)=1
C USER INPUT BOUNDARY CONDITION
2721    CALL CURPOS(1.,195.)
2120    CALL MESSAG("# SLOPE BOUNDARY VALUE(1ST END)?^")
        IF(IERROR(110).NE.0) GOTO 2120
        READ(IN,1110) B(1)
1110    FORMAT(G0.0)
        GOTO 2
202     IE(2)=1
2021    CALL CURPOS(1.,170.)
2110    CALL MESSAG("# SLOPE BOUNDARY VALUE(2ND END)?^")
        IF(IERROR(110).NE.0) GOTO 2110
        READ(IN,1110)B(2)
        GOTO 2
399     IQ=10
        GOTO 2
139     CALL CURPOS(1.,195.)
404     CALL MESSAG("# BOUNDARY VALUE OF END POINT CONDITION?^")
        IF(IERROR(110).NE.0)GOTO 404
        READ(IN,11)B(1),B(2)
11      FORMAT(2G0.0)
        GOTO 2
C NOT EQUAL SPACING
40      GOTO(2,444,2,2,2,2,411),M
444     IQ=20
        GOTO 2
41      GOTO(2,2,2,2,2,2,411),M
411     IF(ICHAR.EQ.49)IE(1)=J-8
        IF(ICHAR.EQ.50)IE(2)=J-8
        GOTO 2
42      GOTO(2,421,2,2,2,2,411),M
421     IF(ICHAR.EQ.49.OR.ICHAR.EQ.50)GOTO 422
        GOTO 2
422     ICHAR=ICHAR-48
        GOTO(4211,4222),ICHAR
        GOTO 2
4211    IE(1)=J-11
        GOTO 2721
4222    IE(2)=J-11
        GOTO 2021
43      GOTO(2,431,2,2,2,2,2),M
431     IF(ICHAR.EQ.49)IE(1)=J-11
        IF(ICHAR.EQ.50)IE(2)=J-11
        GOTO 2
101     WRITE(IOUT,111)
C OUTPUT ALGORITHM TITLES
111     FORMAT("*** GLOBAL POLYNOMIAL INTERPOLATION(NEWTON FORM)***")
        GOTO 20
102     WRITE(IOUT,122)
```

```
      GOTO 20
122   FORMAT("***PIECEWISE QUINTIC POLYNOMIAL INTERPOLATION***"/
&     12X,"(VARIABLE END CONDITION)")
103   WRITE(IOUT,133)
133   FORMAT(" ** CUBIC SPLINE(SECOND DERIVATIVES END CONDITION) **")
      GOTO 20
104   WRITE(IOUT,144)
144   FORMAT("*** CUBIC SPLINE(SECOND DERIVATIVES END CONDITION(*)) ***")
      WRITE(IOUT,1404)
1404  FORMAT(60HWHERE Y"[X1]=U*Y"[X2] & Y"[X(N-1)]=V*Y"[X(N)],SPECIFY U & V  )
      GOTO 20
105   WRITE(IOUT,155)
      GOTO 20
155   FORMAT("*** CUBIC SPLINE (FIRST DERIVATIVES END CONDITION)***")
106   WRITE(IOUT,166)
166   FORMAT("*** CUBIC SPLINE(PERIODIC END CONDITION) ***")
      GOTO 20
108   WRITE(IOUT,188)
188   FORMAT("****CUBIC SPLINE (VARIABLE END CONDITION)****")
      GOTO 20
      END
C
C **************ERROR MESSAGE DISPLAY***************************
C
      SUBROUTINE ERRMES(IC)
      COMMON/IO/IN,IOUT
C MENU ITEMS
      DATA MNTXT/"+ PREVIOUS+ HELP    + RESTART + EXIT    "/
      DATA DATSP/"DATSUPFL"/
      LOGICAL*1 MNTXT(40),DATSP(10)
1     CALL TXCLER
      WRITE(IOUT,20)
20    FORMAT("ERROR MESSAG:-"/"TOO MANY POINTS FOR JOINNING CURVES"/
&          "WHICH EXCEED CORE LIMIT"/
&          "YOU MAY PROCEED BY TAKING THE FOLLOWING ACTION:-"/
&          "EITHER 1- USE PREVIOUS COMMAND TO GO BACK TO PARAMETER"/
&          "DISPLAY,SO THAT TO ALTER NO.OF INTERMEDIATE POINTS."/
&          "OR     2- USE HELP COMMAND IN ORDER TO BRANCH TO ANY"/
&          "DISPLAY ,E.G DATA ENTRY OR DATA TABULATION DISPLAYS..ETC")
      CALL MNOPEN(875.,760.,1)
      CALL MNDISP(MNTXT,4,10,1)
      CALL FRAME(870.,778.,4)
120   CALL MNPICK(J,ICHAR,MNO)
110   IF(ICHAR.EQ.78) GOTO 110
      IF(ICHAR.NE.89)GOTO 120
      GOTO (40,40,1,50),J
C PREVIOUS DISPLAY
40    IC=J
      RETURN
50    CALL RMFILE(DATSP)
      STOP
      END
C
C ******** PLOT THE POLYGONAL OF DATA POINTS********************
C
      SUBROUTINE PPLOT(R,SCL1,SCL2,SCL3,SCL4,N)
C INPUT COMMON DATA
      COMMON/DATSUP/NPS,NPI(50),IFREES,X(50),Y(50),L(50),IH(5),M(5)
&                       ,METHOD,IHELP,IPREV,BOUND(2),SUBSET,INTPNT,IE(2)
      DIMENSION X0(50),Y0(50)
C REMOVES LINKS FROM INPUT DATA
      CALL REMLNK(X0,Y0)
```

```
      DO 1 I=1,NPS
      IF(I.EQ.1) GOTO 2
C JOIN DATA POINT WITH LINE SEGMENTS
      CALL TXDRAW(XO(I),YO(I))
2     CALL PLUSGN(SCL1,SCL2,SCL3,SCL4,XO(I),YO(I))
1     CALL TXMOVE(XO(I),YO(I))
      RETURN
      END
```

```
C                           *****************
C                           * APPENDIX 2.23 *
C                           *****************
C
C THIS MODULE HANDLES THE FOLLOWING INTERACTIVE DISPLAYS:-
C              1.CURVE FIT
C              2.CURVE ZOOM
C
C
C
C******** MAIN PROGRAM-MODULE 3 ***********************
C
C I/O COMMON DATA AREA
        COMMON/DATSUP/NPS,NPI(50),IFREES,X(50),Y(50),L(50),IH(5),M(5)
&                    ,METHOD,IHELP,IPREV,BOUND(2),SUBSET,INTPNT,IE(2)
        COMMON/CURVEFIT/COEF(50,6),XCORD(200),YCORD(200)
        COMMON/IO/IN,IOUT
C OVERLAY EXECUTABLE  PROGRAM NAME
        DATA MODL1,MODL2,MODL5,MODL6,MODL7/"EXPLICIT","MOD2","MOD5","MOD6",
&"MOD7"/
        DATA HELP/"HELP"/
        INTEGER SUBSET
        LOGICAL*1 MODL1(10),MODL2(10),MODL5(10),MODL6(10),MODL7(10)
        LOGICAL*1 HELP(10)
        CALL TXOPEN
C READ I/O COMMON DATA FILES
        CALL RDCOM1
        IF(METHOD.NE.1)GOTO 1110
        NC=1
        GOTO 21
1110    IF(METHOD.NE.2)GOTO 11
        NC=6
        GOTO 21
11      NC=4
21      IN=5
        IOUT=6
        CALL RDCOM2(NPS,NPI,NC)
C CALL CURVE DESIGN DISPLAY
        IF(INTPNT.EQ.999) GOTO 111
C CURVE FIT DISPLAY
        CALL CRVFIT(NPS,NPI,XCORD,YCORD,COEF,SUBSET,IC)
        GOTO(2,3,2,2,2,6,7,8,2,10,111,2,2,14,2,15,2,22),IC
C PROGRAM TERMINATION
22      CALL EXIT
2       STOP
C PREVIOUS DISPLAY
3       IPREV=1
        IHELP=0
        CALL WRCOM1
        CALL OVRLAY(MODL2)
C COMPLETE TABLE OF INTERPOLATED DATA POINTS
6       IPREV=1
100     IHELP=0
        CALL WRCOM1
        CALL OVRLAY(MODL6)
C  DISPLAY JOINED CURVE SEGMENTS
7       IPREV=0
        IHELP=0
        CALL WRCOM1
        CALL OVRLAY(MODL7)
C SUPERIMPOSED CURVES DISPLAY
```

```
8       IPREV=3
        GOTO 100
C ERROR REFERENCE
10      IPREV=2
        GOTO 100
C CURVE DESIGN DISPLAY
111     IPREV=0
        IHELP=0
        CALL WRCOM1
        CALL OVRLAY(MODL5)
C ALGORITHM DISPLAY
14      IPREV=0
        IHELP=0
        CALL WRCOM1
        CALL OVRLAY(MODL1)
C HELP DISPLAY
15      CALL WRCOM1
        CALL OVRLAY(HELP)
        END
C
C **************** CURVE FIT DISPLAY ROUTINE ************************
C
        SUBROUTINE CRVFIT(N,N1,XX,YY,C,SUBT,IC)
        COMMON/CURVES/NCRV(10),XYSCL(4)
        COMMON/IO/IN,IOUT
C MENU ITEMS
        DATA MNTXT1/"+ NEXT    + PREVIOUS+ GRAPH   + COORDS. + DISP.ORG
&+ TABLES  + JOIN CRV+ NGRAPH  "/
        DATA MNTXT2/"+ ZOOM    + ERRO.REF+ CRV.DES.+ SAVE    + REDRAW
&+ METHODS + AX. MARK+ HELP    + RESTART + EXIT    "/
        DIMENSION N1(1),XX(1),YY(1),C(50,6)
        LOGICAL*1 MNTXT1(80),MNTXT2(100)
        EXTERNAL CPLOT
        IF(IERROR(103).NE.0) GOTO 99
        CALL RDCRVS
99      MSUM=0
        N2=N-1
C COMPUTE TOTAL NUMBER OF INTERPOLATED POINTS
        DO 6 I=1,N2
6       MSUM=MSUM+N1(I)
        MSUM=MSUM+N
1       CALL MINMAX(S1,S2,S3,S4,XX,YY,MSUM)
        CALL TXCLER
        ICORD=0
        JON=0
        NG=1
C SET UP CURVE FIT DISPLAY MENU
        WRITE(IOUT,10)
10      FORMAT("CURVE FIT:-")
        CALL MNOPEN(875.,715.,1)
        CALL MNDISP(MNTXT1,8,10,1)
        CALL MNDISP(MNTXT2,10,10,1)
        CALL FRAME(870.,733.,18)
C SET UP CURSOR FOR USER CHOICE FROM THE MENU
2       CALL LMTARA
        CALL MNPICK(J,ICHAR,MNO)
        IF(J.EQ.4.AND.ICORD.EQ.2) GOTO 2
        IF(J.EQ.8.AND.NG.EQ.2) GOTO 30
17      CALL CONFRM(ICHAR)
        IF(ICHAR.EQ.78) GOTO 2
        IF(ICHAR.NE.89) GOTO 17
C TRANSFER CONTROL ACCORDING TO USER CHOICE OF THE MENU
```

```
      GOTO (30,30,3,4,5,30,7,8,9,30,30,11,12,30,15,30,1,30),J
30    IC=J
      RETURN
C DRAW THE CURVE
3     CALL DRAW(CPLOT,R,S1,S2,S3,S4,MSUM)
      GOTO 2
C CURSOR COORDINATE INPUT
4     CALL DISCOR(ICORD,S1,S2,S3,S4)
      GOTO 2
C DISPLAY AXES ORIGIN
5     CALL DISORG(S1,S2,S3,S4)
      GOTO 2
C SAVE CURRENT CURVE ON DISC FILE
11    CALL CRVSAV(XX,YY,N,N1,MSUM,S1,S2,S3,S4,J)
      GOTO 2
C PLOT AURVE FROM NAMED FILE
12    CALL REDRAW(XX,YY,N,N1,MSUM,S1,S2,S3,S4)
      GOTO 2
C AXES MARKING
15    CALL AXSMRK(S1,S2,S3,S4)
      GOTO 2
C CURVE JOIN
7     IF(SUBT.EQ.0)GOTO 2
      JON=JON+1
      IF(JON.EQ.2)GOTO 77
      CALL DTEXT(730.,585.,"JOIN DISP.",10)
      CALL SETJON(XX,YY,MSUM,N,N1,SUBT)
      GOTO 2
C BACK TO CALLING PROGRAM
77    IC=7
      RETURN
C NGRAPH COMMAND FOR THE FIRST TIME (I.E SAVE CURVE FOR LATER USE)
 8    CALL NGRAPH(N,N1,MSUM,XX,YY,S1,S2,S3,S4,J)
      CALL WRCRVS
      NG=NG+1
      GOTO 2
C ZOOMING
9     CALL ZOOM(XX,YY,C,S1,S2,S3,S4,IC)
      GOTO (1,31,90,90,90,90,90),IC
90    IF(IC.EQ.3) IC=14
      IF(IC.EQ.5)IC=16
      IF(IC.EQ.4.OR.IC.EQ.6) STOP
      IF(IC.EQ.7)IC=18
31    RETURN
      END
C
C ************** PLOTTING CURVE FIT ****************************
C
      FUNCTION CPLOT(R,X0,Y0,X1,Y1,N)
C I/O COMMON DATA AREA
      COMMON/DATSUP/NPS,NPI(50),IFREES,X(50),Y(50),L(50),IH(5),M(5)
&                   ,METHOD,IHELP,IPREV,BOUND(2),SUBSET,INTPNT,IE(2)
      COMMON/CURVEFIT/COEF(50,6),XCORD(200),YCORD(200)
      INTEGER SUBSET,R
      IP=1
      IF=1
      I=1
C PLOT SUPPLIED DATA POINTS
3     CALL PLUSGN(X0,Y0,X1,Y1,XCORD(I),YCORD(I))
      CALL TXMOVE(XCORD(I),YCORD(I))
      IF(I.EQ.N)RETURN
      IP1=IP+1
```

```
      IP2=IP+NPI(IF)+1
C PLOT INTERPOLATED POINTS BETWEEN THE INTERVALS
      DO 1 J=IP1,IP2
1     CALL TXDRAW(XCORD(J),YCORD(J))
      IP=IP2
      IF=IF+1
      I=I+NPI(IF-1)+1
      GOTO 3
      END
C
C***********SAVES THE CURVE ON FILE FOR LATER USE*****************
C
      SUBROUTINE CRVSAV(XX,YY,N,N1,MSUM,SCL1,SCL2,SCL3,SCL4,J)
      DIMENSION XX(1),YY(1),N1(1)
      INTEGER FILE(3)
      N2=N-1
      IF(J.EQ.8) GOTO 88
C USER FILE NAME ENTRY
      CALL CURPOS(10.,730.)
      CALL MESSAG("FILE NAME?^")
      CALL GETFLN(FILE )
      REWIND 9
C OPEN AN OUTPUT FILE WITH USER SUPLIED NAME
      CALL SETFIL(9,FILE)
88    WRITE(9,20)MSUM,N
      WRITE(9,30)(XX(I),YY(I),I=1,MSUM)
      WRITE(9,25)(N1(I),I=1,N2)
      WRITE(9,35)SCL1,SCL2,SCL3,SCL4
25    FORMAT(I3)
20    FORMAT(2I3)
30    FORMAT(F11.4)
35    FORMAT(4F11.4)
      ENDFILE 9
      RETURN
      END
C
C**************INPUT FILE NAME************************
C
      SUBROUTINE GETFLN(NAME)
      COMMON/IO/IN,IOUT
      INTEGER NAME(3)
      READ(IN,10)NAME
10    FORMAT(2A4,A2)
      RETURN
      END
C
C ******SAVES CURVES FOR SUPERIMPOSED DISPLAY*************
C
      SUBROUTINE NGRAPH(N,N1,MSUM,XX,YY,SCL1,SCL2,SCL3,SCL4,J)
C COMMON DATA AREA FOR SUPERIMPOSED DISPLAY
      COMMON/CURVES/NCRV(10),XYSCL(4)
C DISPLAY MENU ITEMS
      DATA SUPFLS/"CURVE1    CURVE2    CURVE3    CURVE4    CURVE5
&CURVE6    CURVE7    CURVE8    CURVE9    CURVE10   "/
      LOGICAL*1 SUPFLS(100)
      K=0
C FIND FREE ENTRY
      DO 1 I=1,10
        IF(NCRV(I).NE.99) GOTO 2
        K=K+9
1     CONTINUE
C MARK LAST FREE ENTRY
```

```
2       NCRV(I)=99
        K=K+I
        REWIND 9
C OPEN INPUT FILE & SAVE THE CURVE
        CALL SETFIL(9,SUFFLS(K))
        CALL CRVSAV(XX,YY,N,N1,MSUM,SCL1,SCL2,SCL3,SCL4,J)
C FIRST CURVE
        IF(NCRV(1).EQ.99) GOTO 3
C SET CURVE SCALLING
        XYSCL(1)=SCL1
        XYSCL(2)=SCL2
        XYSCL(3)=SCL3
        XYSCL(4)=SCL4
        GOTO 4
C SUBSEQUENT CURVES ES SET THE DISPLAY SCALE
3       IF(XYSCL(1).GT.SCL1)XYSCL(1)=SCL1
        IF(XYSCL(2).GT.SCL2)XYSCL(2)=SCL2
        IF(XYSCL(3).LT.SCL3)XYSCL(3)=SCL3
        IF(XYSCL(4).LT.SCL4)XYSCL(4)=SCL4
C CALL SUPERIMPOSED DISPLAY COMAND
4       CALL DTEXT(725.,555.,"+ GRAPH",7)
        ENDFILE 9
        RETURN
        END
C
C *********SAVE INPUT COMMON DATA AREA FOR SUPERIMPOSED DISPLAY******
C
        SUBROUTINE RDCRVS
        COMMON/CURVES/NCRV(10),XYSCL(4)
        REWIND 7
        CALL SETFIL(7,"SUPCRVES")
        READ(7,10)(NCRV(I),I=1,10)
        READ(7,20)(XYSCL(I),I=1,4)
10      FORMAT(I2)
20      FORMAT(F11.4)
        ENDFILE 7
        RETURN
        END
C
C*********PLOT CURVE FROM DATA FILE *************************
C
        SUBROUTINE REDRAW(XX,YY,N,N1,MSUM,SCL1,SCL2,SCL3,SCL4)
        COMMON/IO/IN,IOUT
        DIMENSION XX(1),YY(1),N1(1)
        INTEGER FILE(3)
        CALL MESSAG("                                        FILE NAME?^")
        CALL GETFLN(FILE)
        REWIND 9
        CALL SETFIL(9,FILE)
        READ(9,25)MSUM,N
        N2=N-1
        READ(9,30)(XX(I),YY(I),I=1,MSUM)
        READ(9,20)(N1(I),I=1,N2)
        READ(9,35)SCL1,SCL2,SCL3,SCL4
20      FORMAT(I3)
25      FORMAT(2I3)
30      FORMAT(F11.4)
35      FORMAT(4F11.4)
        CALL LMTSCL(SCL1,SCL2,SCL3,SCL4)
        CALL CPLOT(R,SCL1,SCL2,SCL3,SCL4,MSUM)
        ENDFILE 9
        RETURN
```

```
      END
C
C *************SET JOIN CURVE INFORMATION*************
C
      SUBROUTINE SETJON(XX,YY,MSUM,N,N1,SUBT)
C JOIN COMMON DATA AREA
      COMMON/JOIN/CJ1(500),CJ2(500),J3(12),J4(100),IPNTR(6)
      DIMENSION XX(1),YY(1),N1(1)
      INTEGER SUBT
      IF(IERROR(103).NE.0) GOTO 7
C READ THE JOIN COMMON DATA AREA
      CALL RDCOMJ
7     IF(SUBT.EQ.2) GOTO 1
C SET ARRAY JOIN POINTERS
      J3(1)=1
      J3(2)=5
      J3(3)=N
      J3(4)=2
      J4(1)=N
      DO 2 I=2,N
2     J4(I)=N1(I-1)
      IPNTR(1)=3
      IPNTR(2)=1
C STORE THE JOIN CURVE FOR JOIN DISPLAY
4     IP1=J3(1)
      DO 3 I=1,MSUM
        CJ1(IP1)=XX(I)
        CJ2(IP1)=YY(I)
        IP1=IP1+1
3     CONTINUE
      J3(1)=IP1-1
C WRITE OUT THE JOIN COMMON DATA AREA IN OUTPUT FILE
      CALL WRCOMJ
      RETURN
C SET POINTER AND SAVE DATA POINT OF THE SEGMENT
1     J3(J3(2))=N
      J3(J3(2)+1)=J4(1)+1
      DO 6 I=2,N
6     J4(J4(1)-1+I)=N1(I-1)
      J4(1)=J4(1)+N-1
      J3(2)=J3(2)+2
      IPNTR(IPNTR(1))=J3(1)
      IPNTR(1)=IPNTR(1)+1
      GOTO 4
      END
C
C********SAVE COMMON DATA FOR THE SUPERIMPOSED DISPLAY*********
C
      SUBROUTINE WRCRVS
      COMMON/CURVES/NCRV(10),XYSCL(4)
      REWIND 8
C OPEN AN OUTPUT FILE
      CALL SETFIL(8,"SUPCRVES")
      WRITE(8,10)(NCRV(I),I=1,10)
      WRITE(8,20)(XYSCL(I),I=1,4)
10    FORMAT(I2)
20    FORMAT(F11.4)
      ENDFILE 8
      RETURN
      END
C
C ********ZOOM DISPLAY ROUTINE***************************
```

```
C
        SUBROUTINE ZOOM(XX,YY,C,SCL1,SCL2,SCL3,SCL4,IC)
C INPUT COMMON DATA AREA
        COMMON/DATSUP/NPS,NPI(50),IFREES,X(50),Y(50),L(50),IH(5),M(5)
&                        ,METHOD,IHELP,IPREV,BOUND(2),SUBSET,INTPNT,IE(2)
        COMMON/IO/IN,IOUT
C MENU ITEMS
        DATA MNTXT/"+ NEXT    + PREVIOUS+ METHODS + AX. MARK+ HELP
&+ RESTART + EXIT    "/
        DIMENSION XX(1),YY(1),C(50,6),X0(50),Y0(50),CX(2)
        LOGICAL*1 MNTXT(70)
C REMOVES LINKS & SET WINDOW
        CALL REMLNK(X0,Y0)
        CALL LMTSCL(SCL1,SCL2,SCL3,SCL4)
        DY=(SCL4-SCL2)/20
C SET ZOOMING WINDOW
        DO 1 I=1,2
          CALL TXCURS(CXX,CYY,ICHAR)
          CX(I)=CXX
118       CDY1=CYY+DY
          CDY2=CYY-DY
          IF(CDY1.GT.SCL4.OR.CDY2.LT.SCL2) GOTO 117
          CALL TXMOVE(CXX,CDY1)
          CALL TXDRAW(CXX,CDY2)
          GOTO 1
117       DY=DY/2.
          GOTO 118
1       CONTINUE
C SORT SMALLER CXX
        IF(CX(1).LT.CXX) GOTO 2
        CX(2)=CX(1)
        CX(1)=CXX
C DETERMINE BETWEEN WHICH INTERVAL THE ZOOMED CURVE IS?
2       IF1=0
        I=1
3       IF(CX(1).LE.XX(I)) GOTO 4
        IF1=IF1+1
        I=I+NPI(IF1)+1
        GOTO 3
4       K1=I-NPI(IF1)-1
        IF2=0
        I=1
5       IF(CX(2).LE.XX(I)) GOTO 6
        IF2=IF2+1
        I=I+NPI(IF2)+1
        GOTO 5
6       K2=I
        L2=K1+NPI(IF1)+1
        DO 7 I=K1,L2
        IF(CX(1).LE.XX(I)) GOTO 8
7       CONTINUE
8       K3=I-1
        L2=NPI(IF2)+1
        DO 9 I=1,L2
          L1=K2-I+1
          IF(CX(2).GT.XX(L1)) GOTO 11
9       CONTINUE
11      K4=L1+1
C SET THE ZOOMED DISPLAY & MENU
        CALL TXCLER
        CALL ALFHMD
        WRITE(IOUT,10)
```

```
10      FORMAT("ZOOMING:-")
        CALL LMTARA
        CALL MNOPEN(875.,715.,1)
        CALL MNDISP(MNTXT,7,10,1)
        CALL FRAME(870.,733.,7)
        S1=XX(K3)
        S3=XX(K4)
        S2=YY(K3)
        S4=S2
        DO 50 I=K3,K4
          IF(YY(I).GT.S4) S4=YY(I)
          IF(YY(I).LT.S2) S2=YY(I)
50      CONTINUE
        CALL LMTSCL(S1,S2,S3,S4)
        CALL PFRAME(S1,S2,S3,S4)
C DRAW THE ZOOMED CURVE
        L1=K1+NPI(IF1)+1
        L2=K2-NPI(IF2)-1
        IF(L2.LT.L1) GOTO 77
        I=L1
C PLOT THE SUPPLIED POINTS IN THE ZOOMED PORTION
66      CALL PLUSGN(S1,S2,S3,S4,XX(I),YY(I))
        IF1=IF1+1
        IF(I.EQ.L2) GOTO 77
        I=I+NPI(IF1)+1
        GOTO 66
 77     CALL TXMOVE(XX(K3),YY(K3))
        L1=K4-1
        DO 88 I=K3,L1
            XXM=(XX(I)+XX(I+1))/2.
            DO 99 J=1,NPS
             IF(XXM.LT.X0(J)) GOTO 111
99          CONTINUE
111         J1=J-1
            IF(METHOD.EQ.1) GOTO 888
            T=XXM-X0(J1)
            IF(METHOD.EQ.2 )GOTO 222
            YYM=C(J1,1)+T*(C(J1,2)+T*(C(J1,3)+T*C(J1,4)))
            GOTO 121
222         T=T/(X0(J)-X0(J1))
        YYM=C(J1,1)+T*(C(J1,2)+T*(C(J1,3)+T*(C(J1,4)+T*(C(J1,5)+T*C(J1,6)))))
121         W1=YYM-S4
            W2=YYM-S2
            IF(W1.GT.0.) YYM=S4
            IF(W2.LT.0.) YYM=S2
            CALL TXDRAW(XXM,YYM)
            CALL TXDRAW(XX(I+1),YY(I+1))
88      CONTINUE
C DISPLAY MENU
202     CALL LMTARA
        CALL MNPICK(J,ICHAR,MNO)
17      CALL CONFRM(ICHAR)
        IF(ICHAR.EQ.78) GOTO 202
        IF(ICHAR.NE.89) GOTO 17
C TRANSFER CONTROL ACCORDING TO USER CHOICE
        GOTO(414,414,414,404,414,2,414),J
414     IC=J
        RETURN
C AXES MARKING
404     CALL AXSMRK(S1,S2,S3,S4)
        GOTO 202
888     N2=NPS-1
```

```
      A=C(NPS,1)
      DO 808 LL=1,N2
      JJ=NPS-LL
      A=C(JJ,1)+(XXM-X0(JJ))*A
808   CONTINUE
      YYM=A
      GOTO 121
      END
```

```
C                          ****************
C                          * APPENDIX 2.24 *
C                          ****************
C
C THIS MODULE HANDLES THE CURVE DESIGN INTERACTIVE DISPLAY.
C
C
C
C ****** MAIN PROGRAM - MODULE 4 ***************
C
C INPUT/OUTPUT COMMON DATA AREA
        COMMON/DATSUP/NPS,NPI(50),IFREES,X(50),Y(50),L(50),IH(5),M(5)
&                    ,METHOD,IHELP,IPREV,BOUND(2),SUBSET,INTPNT,IE(2)
        COMMON/CURVEFIT/COEF(50,6),XCORD(200),YCORD(200)
        COMMON/IO/IN,IOUT
C OVERLAY EXECUTABLE PROGRAM NAMES
        DATA MODL2,MODL4,HELP/"MOD2","MOD4","HELP"/
        DIMENSION X0(50),Y0(50)
        LOGICAL*1 MODL2(10),MODL4(10),HELP(10)
        INTEGER SUBSET
        CALL TXOPEN
C READS  THE COMMON DATA FILES
11      CALL RDCOM1
        GOTO(1,2,3,3,3,3,3,3),METHOD
1       NC=1
        GOTO 21
2       NC=6
        GOTO 21
3       NC=4
21      IN=5
        IOUT=6
        CALL RDCOM2(NPS,NPI,NC)
        CALL REMLNK(X0,Y0)
C SET LINK LIST FOR THE INTERPOLATED POINTS
        CALL SETLNK(NPS,NPI,INTPNT)
C CURVE DESIGN DISPLAY
        CALL CURDES(NPS,NPI,X0,Y0,XCORD,YCORD,INTPNT,METHOD,COEF,IC)
        GOTO(4,5,6,6,6,6,6,6,6,9,11,8),IC
6       STOP
C NEXT DISPLAY
4       IPREV=0
        IHELP=0
        INTPNT=0
        CALL WRCOM1
        CALL SWRCOM(NPS,NPI,NC,X0,Y0)
        CALL OVRLAY(MODL4)
C PREVIOUS DISPLAY
5       IPREV=1
        IHELP=0
        CALL WRCOM1
        CALL OVRLAY(MODL2)
C PROGRAM TERMINATION
8       CALL EXIT
        STOP
C HELP DISPLAY
9       CALL WRCOM1
        CALL SWRCOM(NPS,NPI,NC,X0,Y0)
        CALL OVRLAY(HELP)
        END
C
C*************CURVE DESIGN DISPLAY*************************
```

```
C
        SUBROUTINE CURDES(N,N1,X0,Y0,XX,YY,INTPNT,IA,C,IC)
C LINK LIST COMMON AREA
        COMMON/LNKLST/LINK(200),INTVAL(50),IFREE,IFCNT
        COMMON/IO/IN,IOUT
C MENU ITEMS
        DATA MNTXT/"+ NEXT    + PREVIOUS+ DISP.ORG+ GRAPH    + COORDS.
&+ ADD PNT*+ DEL INTV+ REFRESH + AX. MARK+ HELP    + RESTART + EXIT    "/
        DIMENSION N1(1),X0(1),Y0(1),XX(1),YY(1),C(50,6)
        LOGICAL*1 MNTXT(120)
        EXTERNAL XPLOT
        CALL SUM(N,N1,MSUM)
        CALL MINMAX(S1,S2,S3,S4,XX,YY,MSUM)
        IZERO=1
11      IF(IZERO.EQ.0)INTPNT=999
C SET UP THE DISPLAY
        CALL TXCLER
        ICORD=0
        WRITE(IOUT,10)
10      FORMAT("CURVE DESIGN:-")
C OUTPUT  MENU
        CALL MNOPEN(875.,715.,1)
        CALL MNDISP(MNTXT,12,10,1)
        CALL FRAME(870.,733.,12)
C DISPLAY INSTRUCTION FOR KEYBOARD COMMAND
        CALL DTEXT(830.,447.,"*TYPE CAPT.:",12)
        CALL DTEXT(830.,428.,"E-JOIN INTV.",12)
        CALL DTEXT(830.,406.,"F-FINISH ADD",12)
2       CALL LMTARA
C CHECK FOR REFRESH COMMAND
        IF(J.EQ.8) GOTO 34
        CALL MNPICK(J,ICHAR,MNO)
        IF(J.EQ.5.AND.ICORD.EQ.2)GOTO 2
17      CALL CONFRM(ICHAR)
        IF(ICHAR.EQ.78)GOTO 12
        IF(ICHAR.NE.89) GOTO 17
        GOTO(41,41,33,34,35,36,36,11,110,41,66,41),J
12       J=0
        GOTO 2
41      IC=J
        RETURN
C DISPLAY ORIGIN
33      CALL DISORG(S1,S2,S3,S4)
        GOTO 2
C PLOT THE GRAPH
34      CALL DRAW(XPLOT,R,S1,S2,S3,S4,MSUM)
        J=0
        GOTO 2
C AXES MARKING
110     CALL AXSMRK(S1,S2,S3,S4)
        GOTO2
C CURSOR INPUT COORDINATES
35      CALL DISCOR(ICORD,S1,S2,S3,S4)
        GOTO 2
C ADD INTERMEDIATE POINTS INTERACTIVELLY USING THE CURSOR
C DETERMINE WHICH INTERVAL OF THE CURVE
36      CALL LMTSCL(S1,S2,S3,S4)
22      CALL TXCURS(CX,CY,ICHAR)
        IF(ICHAR.EQ.70) GOTO 2
C GIVES THE INTERVAL NUMBER
        INTVNO=INO(CX,N,X0)
C IGNORE IF INTERVAL NUMBER IS ZERO
```

```
      IF(INTVNO.EQ.0) GOTO 22
      IF(J.EQ.7) GOTO 37
      IF(INTPNT.EQ.999.AND.IZERO.EQ.1)GOTO 66
61    INTPNT=0
C ADDS INTERMEDIATE POINT TO THE LINK LIST AND UPDATE DATA POINTS ARRAY
      CALL ADDPNT(N,N1,X0,Y0,XX,YY,INTVNO,IA,CX,ICHAR,C)
      GOTO 22
66    N2=N-1
      DO 16 I=1,N2
      INTVAL(I)=0
16    N1(I)=0
      IF(J.EQ.11) GOTO 41
      IZERO=0
      GOTO 61
C MARK DELETED INTERVAL
37    CALL DTEXT(CX,CY,"D",1)
      CALL DELINT(INTVNO,N,N1)
      GOTO 2
      RETURN
      END
C
C********** ADD INTERMEDIATE POINT ******************
C
      SUBROUTINE ADDPNT(N,N1,X0,Y0,XX,YY,INTVNO,IA,CX,ICHAR,C)
C LINK LIST COMMON DATA AREA
      COMMON/LNKLST/LINK(200),INTVAL(50),IFREE,IFCNT
      DIMENSION N1(1),X0(1),Y0(1),XX(1),YY(1),C(50,6)
C CHECK FIRST TIME IN THE INTERVAL
      IF(INTVAL(INTVNO).NE.0) GOTO 2
C ENTER THE START POINT AND THE INTERMEDIATE POINT
      INTVAL(INTVNO)=IFREE
      XX(IFREE)=X0(INTVNO)
      YY(IFREE)=Y0(INTVNO)
C CHECK END OF INTERVAL
      IF(ICHAR.EQ.69) GOTO 7
      IF(IFCNT.GT.1) GOTO 11
      LINK(IFREE)=IFREE+1
      CALL TXMOVE(XX(IFREE),YY(IFREE))
      IFREE=LINK(IFREE)
5     LINK(IFREE)=0
C COMPUTE INTERPOLATED POINT FROM THE COEFFICIENTS
6     CALL CXXYY(N,X0,Y0,XX,YY,INTVNO,IA,CX,C,IFREE)
      N1(INTVNO)=N1(INTVNO)+1
      IFREE=IFREE+1
      RETURN
C COMPARE THE CURRENT CX WITH THE ALERADY IN THE TABLE
2     IF(ICHAR.EQ.69) GOTO 1
      NEXT=INTVAL(INTVNO)
4     IF(CX.LT.XX(NEXT)) GOTO 3
      K=NEXT
      NEXT=LINK(NEXT)
      IF(NEXT.EQ.0) GOTO 41
      GOTO 4
41    LINK(K)=IFREE
      CALL TXMOVE(XX(K),YY(K))
      IF(IFCNT.GT.0) GOTO 12
      GOTO 5
C ADD POINT IN THE INTERVAL WHICH HAS GREATER VALUE POINT
3     IF(IFCNT.GT.0)GOTO 16
      LINK(K)=IFREE
      LINK(IFREE)=NEXT
      CALL TXMOVE(XX(K),YY(K))
```

```
        GOTO 6
C ADDITION INTO PREVIOUSELY DELETED ITEM (CARBIGE COLLECTION)
11      IFCNT=IFCNT-1
        CALL TXMOVE(XX(IFREE),YY(IFREE))
        K=LINK(LINK(IFREE))
        LINK(LINK(IFREE))=0
        IFREE=LINK(IFREE)
        CALL CXXYY(N,X0,Y0,XX,YY,INTVNO,IA,CX,C,IFREE)
        IFREE=K
14      IFCNT=IFCNT-1
        N1(INTVNO)=N1(INTVNO)+1
        RETURN
12      CALL CXXYY(N,X0,Y0,XX,YY,INTVNO,IA,CX,C,IFREE)
        K=IFREE
        IFREE=LINK(IFREE)
        LINK(K)=0
        GOTO 14
16      CALL TXMOVE(XX(K),YY(K))
        CALL CXXYY(N,X0,Y0,XX,YY,INTVNO,IA,CX,C,IFREE)
        K1=LINK(IFREE)
        LINK(K)=IFREE
        LINK(IFREE)=NEXT
        IFREE=K1
        GOTO 14
C END OF INTERVAL WITH ONE OR MANY ENTERIES
C FIND THE LAST ENTRY IN THIS INTERVAL(LINK=0)
1       NEXT=INTVAL(INTVNO)
        J=N1(INTVNO)+1
        DO 9 L=1,J
          IP=NEXT
          NEXT=LINK(NEXT)
          IF(NEXT.EQ.0) GOTO10
9       CONTINUE
10      CALL TXMOVE(XX(IP),YY(IP))
8       CALL TXDRAW(X0(INTVNO+1),Y0(INTVNO+1))
        RETURN
C NO INTERMEDIATE POINTS
7       LINK(IFREE)=0
        IFREE=IFREE+1
        CALL TXMOVE(X0(INTVNO),Y0(INTVNO))
        GOTO 8
        END
C
C***********COMPUTE INTERPOLATED POINT****************
C
        SUBROUTINE CXXYY(N,X0,Y0,XX,YY,INTVNO,IA,CX,C,IFREE)
        DIMENSION X0(1),Y0(1),XX(1),YY(1),C(50,6)
        IF(IA.EQ.1) GOTO 3
        T=CX-X0(INTVNO)
        XX(IFREE)=CX
        I=INTVNO
        IF(IA.EQ.2) GOTO 1
C SPLINE METHOD
        YY(IFREE)=C(I,1)+T*(C(I,2)+T*(C(I,3)+T*C(I,4)))
        GOTO 2
C NEWTON DIVIDED DIFFERENCE
3       N2=N-1
        A=C(N,1)
        DO 7 L=1,N2
         J=N-L
7        A=C(J,1)+(CX-X0(J))*A
        YY(IFREE)=A
```

```
      GOTO 2
C PIECEWISE POLYNOMIAL INTERPOLATION (MAUD METHOD)
1     T=T/(XO(INTVNO+1)-XO(INTVNO))
      YY(IFREE)=C(I,1)+T*(C(I,2)+T*(C(I,3)+T*(C(I,4)+T*(C(I,5)+T*C(I,6)))))
2     CALL TXDRAW(XX(IFREE),YY(IFREE))
      RETURN
      END
C
C********** DELETE UNWANTED INTERVAL *****************
      SUBROUTINE DELINT(INTVNO,N,N1)
      COMMON/LNKLST/LINK(200),INTVAL(50),IFREE,IFCNT
      DIMENSION N1(1)
C EMPTY LIST?
      IF(INTVAL(INTVNO).EQ.0)RETURN
3     IF(IFCNT.NE.0)GOTO 1
C FIRST DELETION
      NEXT=INTVAL(INTVNO)
      K=NEXT
      IF(LINK(NEXT).EQ.0)GOTO 5
      N2=N1(INTVNO)
      K=IZRLNK(N2,LINK,NEXT)
5     LINK(K)=IFREE
      IFREE=INTVAL(INTVNO)
2     IFCNT=IFCNT+N1(INTVNO)+1
      INTVAL(INTVNO)=0
      N1(INTVNO)=0
      RETURN
C DELETE OF MORE INTREVAL
1     NEXT1=INTVAL(INTVNO)
      K1=NEXT1
      IF(LINK(NEXT1).EQ.0) GOTO 6
      N2=N1(INTVNO)
      K1=IZRLNK(N2,LINK,K1)
6     NEXT2=IFREE
      K2=NEXT2
      N2=IFCNT-1
      IF(N2.EQ.0) GOTO7
      K2=IZRLNK(N2,LINK,K2)
7     K3=LINK(K2)
      LINK(K2)=NEXT1
      LINK(K1)=K3
      GOTO 2
      END
C
C ***********FINDS THE INTERVAL IN WHICH X CHOSEN*********
C
      FUNCTION INO(CX,N,XO)
      DIMENSION XO(1)
      DO 1 I=1,N
        IF(CX.LT.XO(I)) GOTO 2
1     CONTINUE
      I=1
2     INO=I-1
      RETURN
      END
C ***********FIND END OF THE INTREVAL  FOR DELETION******************
      FUNCTION IZRLNK(N,LK,NXT)
      DIMENSION LK(1)
      DO 1 I=1,N
1     NXT=LK(NXT)
      IZRLNK=NXT
      RETURN
```

```
      END
C
C ***********SET UP THE LINK LIST*****************
C
      SUBROUTINE SETLNK(N,N1,INTPNT)
      COMMON/LNKLST/LINK(200),INTVAL(50),IFREE,IFCNT
      DIMENSION N1(1)
C INTERMEDIATE POINTS SPECIFIED BY CURSOR?
      IF(INTPNT.EQ.999)GOTO 9
      INTVAL(1)=1
      N2=N-1
C SET POINTER TO THE START OF EACH INTERVAL
      DO 1 I=2,N
1     INTVAL(I)=INTVAL(I-1)+N1(I-1)+1
      K=1
C SET LINKS
      DO 2 I=1,N2
        IF(N1(I).NE.0) GOTO 4
        LINK(K)=0
        K=K+1
        GOTO 2
4       N11=N1(I)+K-1
        DO 3 J=K,N11
3         LINK(J)=J+1
        LINK(J)=0
        K=J+1
2     CONTINUE
      CALL SUM(N,N1,MSUM)
C SET FREE POINTER
      IFREE=MSUM+1
      GOTO 10
C NO INTERMEDIATE POINT IS SPECIFIED
9     IFREE=1
      IFCNT=0
10    RETURN
      END
C
C ********* FIND TOTAL NUMBER OF INTERPOLATED POINTS*X**********
C
      SUBROUTINE SUM(N,N1,MSUM)
      DIMENSION N1(1)
      MSUM=0
      N2=N-1
      DO 1 I=1,N2
1     MSUM=MSUM+N1(I)
      MSUM=MSUM+N
      RETURN
      END
C
C ***********SAVE CURVE ON OUTPUT FILE*************
C
C                    *****A SIMPLE LIST******
      SUBROUTINE SWRCOM(N,N1,NC,X0,Y0)
C OUTPUT COMMON DATA AREA
      COMMON/CURVEFIT/COEF(50,6),XCORD(200),YCORD(200)
      COMMON/LNKLST/LINK(200),INTVAL(50)
      DIMENSION N1(1),X0(1),Y0(1)
      REWIND 8
C OPEN OUTPUT FILE
      CALL SETFIL(8,"OUTFIT")
      N2=N-1
      WRITE(8,20)((COEF(I,J),J=1,NC),I=1,N2)
```

```
      DO 1 I=1,N2
       NEXT=INTVAL(I)
       N3=N1(I)+1
         DO 2 J=1,N3
            WRITE(8,10)XCORD(NEXT),YCORD(NEXT)
            NEXT=LINK(NEXT)
       2 CONTINUE
1     CONTINUE
      WRITE(8,10)X0(N),Y0(N)
10    FORMAT(2F12.4)
20    FORMAT(F12.4)
      ENDFILE 8
      RETURN
      END
C
C************PLOT THE DESIGNED CURVE**************
C
      SUBROUTINE XPLOT(R,SCL1,SCL2,SCL3,SCL4,NSUM)
C I/O COMMON DATA AREAS
      COMMON/DATSUP/NPS,NPI(50),IFREES,X(50),Y(50),L(50),IH(5),M(5)
&                    ,METHOD,IHELP,IPREV,BOUND(2),SUBSET(2),INTPNT,IE(2)
      COMMON/CURVEFIT/COEF(50,6),XCORD(200),YCORD(200)
      COMMON/LNKLST/LINK(200),INTVAL(50),IFREE,IFCNT
      INTEGER SUBSET
      DIMENSION X0(50),Y0(50)
      CALL REMLNK(X0,Y0)
      N2=NPS-1
C OUTPUT SEGMENTS OF THE CURVE FOR EACH INTERVAL
5     DO 1 I=1,N2
        NEXT=INTVAL(I)
        IF(NEXT.NE.0)GOTO 4
        K=I
        GOTO 11
4       CALL PLUSGN(SCL1,SCL2,SCL3,SCL4,XCORD(NEXT),YCORD(NEXT))
        CALL TXMOVE(XCORD(NEXT),YCORD(NEXT))
        N11=NPI(I)
        IF(N11.NE.0)GOTO 3
        K=INTVAL(I+1)
        IF(K.NE.0) GOTO 6
        IF(NEXT.NE.0) GOTO7
        K=I+1
        GOTO 11
6       CALL TXDRAW(XCORD(K),YCORD(K))
        GOTO 1
3       DO 2 J=1,N11
          NEXT=LINK(NEXT)
          CALL TXDRAW(XCORD(NEXT),YCORD(NEXT))
2       CONTINUE
7       K= I+1
        CALL TXDRAW(X0(  K),Y0(K))
11      CALL PLUSGN(SCL1,SCL2,SCL3,SCL4,X0(K),Y0(K))
        CALL TXMOVE(X0(K),Y0(K))
1     CONTINUE
      CALL PLUSGN(SCL1,SCL2,SCL3,SCL4,X0(I),Y0(I))
8     RETURN
      END
```

```
C                         *****************
C                         * APPENDIX 2.25 *
C                         *****************
C
C THIS MODULE HANDLES THE FOLLOWING INTERACTIVE DISPLAYS:-
C                1.TABLE OF THE INTERPOLATED POINTS
C                2.TABLE OF THE POLYNOMIAL COEFFICIENTS
C                3.ERROR REFERRENCE
C                4.CURVE SUPERIMPOSE
C
C
C
C ***** MAIN PROGRAM - MODULE 5 ******************
C
C
C I/O COMMON DATA AREA
      COMMON/DATSUP/NPS,NPI(50),IFREES,X(50),Y(50),L(50),IH(5),M(5)
&                 ,METHOD,IHELP,IPREV,BOUND(2),SUBSET,INTPNT,IE(2)
      COMMON/CURVEFIT/COEF(50,6),XCORD(200),YCORD(200)
      COMMON/IO/IN,IOUT
C OVERLAY EXECUTABLE PROGRAM NAMES
      DATA MODL2,MODL4,MODL1,HELP/"MOD2","MOD4","EXPLICIT","HELP"/
      LOGICAL*1 MODL2(10),MODL4(10),MODL1(10),HELP(10)
      INTEGER SUBSET
      CALL TXOPEN
C READ I/O FILES
11    CALL RDCOM1
      GOTO(1,2,3,3,3,3,3,3),METHOD
1     NC=1
      GOTO 21
2     NC=6
      GOTO 21
3     NC=4
21    IN=5
      IOUT=6
      CALL RDCOM2(NPS,NPI,NC)
      IF(IHELP.NE.0) GOTO 66
      GOTO(33,31,34),IPREV
C DISPLAY TABLE OF INTERPOLATED POINTS
33    CALL TABINT(NPS,NPI,XCORD,YCORD,IOUT,IC)
6     GOTO(51,52,59,54,59,59,57,33,58),IC
C RETURN TO CURVE FIT DISPLAY
51    IPREV=0
      IHELP=0
      CALL WRCOM1
      CALL OVRLAY(MODL4)
C RETURN TO THE PARAMETER ENTRH DISPLAY
52    IPREV=1
      IHELP=0
      CALL WRCOM1
      CALL OVRLAY(MODL2)
C OUTPUT COEFFICIENTS
54    CALL TABCOF(NPS,COEF,METHOD,IOUT,IC)
      GOTO(51,33,59,59,59,57,54,58),IC
C ERROR REFERENCE DISPLAY
31    CALL ERRREF(NPS,NPI,XCORD,YCORD,X,Y,COEF,IC)
      GOTO(51,52,59,59,59,59,59,57,59,58),IC
C SUPERIMPOSED CURVES DISPLAY
34    CALL SUPIMP(NPS,NPI,XCORD,YCORD,IC)
      GOTO(51,52,59,59,59,59,61,59,57,59,58),IC
C HELP DISPLAY
```

```
57      CALL WRCOM1
        CALL OVRLAY(HELP)
C ALGORITHM DISPLAY
61      IPREV=0
        IHELP=0
        CALL WRCOM1
        CALL OVRLAY(MODL1)
C EXIT
58      CALL EXIT
59      STOP
66      GOTO(33,34,31),IHELP
        END
C
C **********TABULATION OF INTERPOLATED POINTS*****************
C
        SUBROUTINE TABINT(N,N1,XX,YY,IDEV,IC)
        DIMENSION N1(1),XX(1),YY(1),IP(4)
11      CALL OUTTIL(1,9,IDEV)
C FIND THE TABLE SIZE
        CALL SUM(N,N1,MSUM)
        IF(MSUM.GT.50) GOTO 1
C TABLE SIZE ONE OR LESS THAN A PAGE
        IROLL=0
        CALL OUTPGE(MSUM,1,XX,YY,IDEV)
        GOTO 12
C TABLE SIZE MORE THAN ONE PAGE
1       IP(1)=1
        IROLL=1
        DO 2 I=2,4
2       IP(I)=IP(I-1)+50
        IS=1
C FIND NUMBER OF PAGES & THE REMAINDER
        MREM=IREM(MSUM,50)
        NPAGE=(MSUM-MREM)/50
14      IFIFTY=50
15      IPNTR=IP(IS)
        CALL OUTPGE(IFIFTY,IPNTR,XX,YY,IDEV)
        WRITE(IDEV,20)
20      FORMAT(/"*'TO DISPLAY THE NEXT/PREVIOUS PAGE OF THE TABLE"/
&               "USE THE FORWARD/BACKWARD AS APPROPRIATE'")
C SET UP CURSOR FOR USER SELECTION
12      CALL MNPICK(J,ICHAR,MNO)
17      CALL CONFRM(ICHAR)
        IF(ICHAR.EQ.78) GOTO 12
        IF(ICHAR.NE.89) GOTO 17
        GOTO(21,21,23,21,25,26,21,11,21),J
C BACK TO THE MAIN PROGRAM TO PROCESS OTHER COMMAND
21      IC=J
        RETURN
C HARDCOPY
23      REWIND 7
        CALL SETFIL(7,"/DEV/TTYM")
        WRITE(7,30)
30      FORMAT("COMPLETE TABLE OF THE INTERPOLATED POINTS:-")
        IFIFTY=MSUM
        IPNTR=IP(1)
        CALL OUTPGE(IFIFTY,IPNTR,XX,YY,7)
        GOTO 12
C TO ROLL THE TABLE(FORWARD)
25      IF(IROLL.EQ.0)GOTO 12
        IF(IS.EQ.NPAGE.AND.MREM.GT.0)GOTO 31
        IF(IS.EQ.NPAGE.OR.IS.GT.NPAGE)GOTO 12
```

```
      IS=IS+1
      CALL OUTTIL(1,9,IDEV)
      GOTO 14
C OUTPUT THE REMAINDER
31    IFIFTY=MREM
      IS=IS+1
      CALL OUTTIL(1,9,IDEV)
      GOTO 15
C BACKWARD
26    IF(IROLL .EQ.0) GOTO 12
      IF(IS.EQ.1) GOTO 12
      IS=IS-1
      CALL OUTTIL(1,9,IDEV)
      GOTO 14
      END
C
C************** TABULATE COEFFICIENTS************************
C
      SUBROUTINE TABCOF(N,C,METHOD,IDEV,IC)
      DIMENSION C(50,6),IP(4)
C OUTPUT PAGE HEADER
11    CALL OUTTIL(2,8,IDEV)
      NC=N-1
      IF(NC.GT.20) GOTO 1
C LESS THAN 20 ,ONE PAGE OF TABLE
      IROLL=0
      CALL OUTCOF(NC,1,METHOD,C,IDEV)
      GOTO 2
C GREATER THAN 20 ,MORE THAN ONE PAGE OF TABLE
1     IROLL=1
      IP(1)=1
      DO 7 I=2,4
7     IP(I)=IP(I-1)+20
      IS=1
C FIND NUBER OF PAGES AND THE REMINDAR
      NREM=IREM(NC,20)
      NCPGE=(NC-NREM)/20
14    N3=20
15    IPNTR=IP(IS)
12    CALL OUTCOF(N3,IPNTR,METHOD,C,IDEV)
      WRITE(IDEV,20)
20    FORMAT(/"*'TO DISPLAY THE NEXT/PREVIOUS PAGE OF THE TABLE"/
&                "  USE THE FORWARD/BACKWARD AS APPROPRIATE'")
C USE CURSOR TO PICK UP MENU OPTIONS
2     CALL MNPICK(J,ICHAR,MNO)
17    CALL CONFRM(ICHAR)
      IF(ICHAR.EQ.78) GOTO 2
      IF(ICHAR.NE.89) GOTO 17
      GOTO(21,21,23,24,25,21,11,21),J
C OTHER COMMAND
21    IC=J
      RETURN
C HARDCOPY
23    REWIND 7
      CALL SETFIL(7,"/DEV/TTYM")
      WRITE(7,30)
30    FORMAT("POLNOMIAL COEFFICIENTS:-")
      N3=N-1
      IPNTR=IP(1)
      CALL OUTCOF(N3,IPNTR,METHOD,C,7)
      GOTO 2
C FORWARD COMMAND (TABLE PAGE ROLLING)
```

```
24      IF(IROLL.EQ.0)GOTO 2
        IF(IS.EQ.NCPGE.AND.NREM.GT.0) GOTO 31
        IF(IS.EQ.NCPGE.OR.IS.GT.NCPGE) GOTO 2
        IS=IS+1
        CALL OUTTIL(2,8,IDEV)
        GOTO14
31      N3=NREM
        IS=IS+1
        CALL OUTTIL(2,8,IDEV)
        GOTO 15
C BACKWARD COMMAND
25      IF(IROLL.EQ.0) GOTO 2
        IF(IS.EQ.1) GOTO 2
        IS=IS-1
        CALL OUTTIL(2,8,IDEV)
        GOTO 14
        END
C
C***********SUPERIMPOSED CURVE DISPLAY***********************
C
        SUBROUTINE SUPIMP(N,N1,XX,YY,IC)
        COMMON/CURVES/NCRV(10),XYSCL(4)
        COMMON/IO/IN,IOUT
C NENU ITEMS
        DATA MNTXT/"+ NEXT    + PREVIOUS+ GRAPH   + DISP.ORG+ DELETE*
&+ REFRESH + METHOD  + AX. MARK+ HELP    + RESTART + EXIT    "/
        DATA SUPFLS/"CURVE1    CURVE2    CURVE3    CURVE4    CURVE5
&CURVE6    CURVE7    CURVE8    CURVE9    CURVE10   "/
        DIMENSION N1(1),XX(1),YY(1)
        LOGICAL*1 MNTXT(110),SUPFLS(100)
        CALL RDCRVS
11      CALL TXCLER
        CALL SUM(N,N1,MSUM)
C SET UP PLOTTING SCALE
        S1=XYSCL(1)
        S2=XYSCL(2)
        S3=XYSCL(3)
        S4=XYSCL(4)
C PREPARE THE DISPLAY
        WRITE(IOUT,10)
10      FORMAT("SUPERIMPOSED CURVES:-")
        CALL MNOPEN(875.,715.,1)
        CALL MNDISP(MNTXT,11,10,1)
        CALL FRAME(870.,733.,11)
        CALL ALPHMD
        WRITE(IOUT,20)
20      FORMAT(/////////////62X,"* TYPE IN"/62X,"CURVE NO.")
2       CALL LMTARA
C CHECK FOR REFRESH
        IF(J.EQ.6) GOTO 23
3       CALL MNPICK(J,ICHAR,MNO)
        NFDEL=ICHAR-48
        IF(J.EQ.5.AND.NFDEL.GT.10.OR.J.EQ.5.AND.NFDEL.LT.1) GOTO 3
17      CALL CONFRM(ICHAR)
        IF(ICHAR.EQ.78) GOTO 12
        IF(ICHAR.NE.89) GOTO 17
        GOTO(21,21,23,24,25,11,21,27,21,11,21),J
12      J=0
        GOTO 2
21      IC=J
        RETURN
C GRAPH/REFRESH OPTION
```

```
23      K=0
        CALL LMTSCL(S1,S2,S3,S4)
        CALL PFRAME(S1,S2,S3,S4)
        DO 1 I=1,10
        REWIND 9
        IF(NCRV(I).NE.99) GOTO 22
        CALL SETFIL(9,SUPFLS(I+K))
77      CALL SREDRW(N,N1,I,XX,YY,MSUM,S1,S2,S3,S4)
22      K=K+9
1       CONTINUE
        ENDFILE 9
        J=0
        GOTO 2
C DISPLAY CURVE ORIGIN
24      CALL DISORG(S1,S2,S3,S4)
        GOTO 2
C DELETE CURVE OPTION
25      NCRV(NFDEL)=0
        CALL WRCRVS
        GOTO 2
C AXES MARKING
27      CALL AXSMRK(S1,S2,S3,S4)
        GOTO 2
        END
C
C**********ERROR REFERRENCE DISPLAY*******************
C
        SUBROUTINE ERRREF(N,N1,XX,YY,X1,Y1,C,IC)
        COMMON/IO/IN,IOUT
C MENU ITEMS
        DATA MNTXT/"+ NEXT    + PREVIOUS+ GRAPH   + COORDS. + DISP.ORG
&+ ZOOM    + AX. MARK+ HELP    + RESTART + EXIT    "/
        DATA MODL1/"EXPLICIT"/
        LOGICAL*1 MNTXT(100),MODL1(10)
        DIMENSION N1(1),XX(1),YY(1),X1(1),Y1(1),C(50,6),X0(50),Y0(50)
        EXTERNAL EPLOT
        CALL REMLNK(X0,Y0)
C FIND THE DIFFERENCE OF THE ORDINATES W.R.T.NEWTON DIVIDED DIFF.METHOD
        CALL NEWTRF(N,N1,X0,Y0,XX,YY,C)
        CALL SUM(N,N1,MSUM)
C SET UP THE ERROR REFERRENCE DISPLAY
        CALL MINMAX(S1,S2,S3,S4,XX,YY,MSUM)
1       CALL TXCLER
        ICORD=0
        WRITE(IOUT,10)
10      FORMAT("ERROR REFRENCE:-")
        CALL MNOPEN(875.,715.,1)
        CALL MNDISP(MNTXT,10,10,1)
        CALL FRAME(870.,733.,10)
2       CALL LMTARA
        CALL MNPICK(J,ICHAR,MNO)
        IF(J.EQ.4.AND.ICORD.EQ.2)GOTO 2
17      CALL CONFRM(ICHAR)
        IF(ICHAR.EQ.78) GOTO 2
        IF(ICHAR.NE.89) GOTO 17
        GOTO(31,31,33,34,35,36,37,31,1,31),J
31      IC=J
44      RETURN
C PLOT THE ERROR REFERENCE CURVE
33      NC=0
        CALL DRAW(EPLOT,NC,S1,S2,S3,S4,MSUM)
        NC=0
```

```
        GOTO 2
C INPUT CURSOR COORDINATES
34      CALL DISCOR(ICORD,S1,S2,S3,S4)
        GOTO 2
C DISPLAY AXES ORIGIN
35      CALL DISORG(S1,S2,S3,S4)
        GOTO2
C CURVE ZOOMING
36      CALL EZOOM(XX,YY,C,S1,S2,S3,S4,IC)
        GOTO (44,1,45,99,99,99,99),IC
45      IPREV=0
        CALL OVRLAY(MODL1)
99      IF(IC.EQ.5) IC=7
        IF(IC.EQ.4.OR.IC.EQ.6)STOP
        IF(IC.EQ.7)IC=9
        GOTO 44
C AXES MARKING
37      CALL AXSMRK(S1,S2,S3,S4)
        GOTO 2
        END
C
C ***********CURVE PLOTTING ROUTINE*******************
C
        FUNCTION EPLOT(NC,SCL1,SCL2,SCL3,SCL4,NSUM)
C I/O COMMON DATA AREA
        COMMON/DATSUP/NPS,NPI(50),IFREES,X(50),Y(50),L(50),IH(5),M(5)
&                       ,METHOD,IHELP,IPREV,BOUND(2),SUBSET,INTPNT,IE(2)
        COMMON/CURVEFIT/COEF(50,6),XCORD(200),YCORD(200)
        DATA IDC/"123456789"/
        LOGICAL*1 IDC(9)
        IP=1
        IF=1
        I=1
C PLOT SUPPLIED POINTS
3       CALL PLUSGN(SCL1,SCL2,SCL3,SCL4,XCORD(I),YCORD(I))
2       CALL TXMOVE(XCORD(I),YCORD(I))
        IF(I.EQ.NSUM)RETURN
        IP1=IP+1
        IP2=IP+NPI(IF)+1
        DO 1 J=IP1,IP2
1       CALL TXDRAW(XCORD(J),YCORD(J))
        IF(IF.NE.1.OR.NC.EQ.0) GOTO 4
C CURVE NUMBERING FOR SUPERIMPOSED CURVES DISPLAY
        CALL DTEXT(XCORD(J-3),YCORD(J-3),IDC(NC),1)
4       IP=IP2
        IF=IF+1
        I=I+NPI(IF-1)+1
        GOTO 3
        END
C
C***************ZOOMIN FOR SINGLE CURVE*********************
C
        SUBROUTINE EZOOM(XX,YY,C,SCL1,SCL2,SCL3,SCL4,IC)
C INPUT COMMON DATA AREA
        COMMON/DATSUP/NPS,NPI(50),IFREES,X(50),Y(50),L(50),IH(5),M(5)
&                       ,METHOD,IHELP,IPREV,BOUND(2),SUBSET,INTPNT,IE(2)
        COMMON/IO/IN,IOUT
        DATA MNTXT/"+ NEXT    + PREVIOUS+ METHODS + AX. MARK+ HELP
&+ RESTART + EXIT    "/
        DIMENSION XX(1),YY(1),C(50,6),XO(50),YO(50),CX(2)
        LOGICAL*1 MNTXT(70)
        CALL REMLNK(XO,YO)
```

```
C SET WINDOW
        CALL LMTSCL(SCL1,SCL2,SCL3,SCL4)
        DY=(SCL4-SCL2)/20
C SELECT WITH CURSOR PORTION OF CURVE TO BE ZOOMED
        DO 1 I=1,2
          CALL TXCURS(CXX,CYY,ICHAR)
          CX(I)=CXX
118       CDY1=CYY+DY
          CDY2=CYY-DY
          IF(CDY1.GT.SCL4.OR.CDY2.LT.SCL2) GOTO 117
          CALL TXMOVE(CXX,CDY1)
          CALL TXDRAW(CXX,CDY2)
          GOTO 1
117       DY=DY/2.
          GOTO 118
1       CONTINUE
C SORT SMALLER CXX
        IF(CX(1).LT.CXX) GOTO 2
        CX(2)=CX(1)
        CX(1)=CXX
C DETERMINE WHICH INTERVAL
2       IF1=0
        I=1
3       IF(CX(1).LE.XX(I)) GOTO 4
        IF1=IF1+1
        I=I+NPI(IF1)+1
        GOTO 3
4       K1=I-NPI(IF1)-1
        IF2=0
        I=1
5       IF(CX(2).LE.XX(I)) GOTO 6
        IF2=IF2+1
        I=I+NPI(IF2)+1
        GOTO 5
6       K2=I
        L2=K1+NPI(IF1)+1
        DO 7 I=K1,L2
        IF(CX(1).LE.XX(I)) GOTO 8
7       CONTINUE
8       K3=I-1
        L2=NPI(IF2)+1
        DO 9 I=1,L2
          L1=K2-I+1
          IF(CX(2).GT.XX(L1)) GOTO 11
9       CONTINUE
11      K4=L1+1
C SET UP THE ZOOMED DISPLAY
        CALL TXCLER
        CALL ALPHMD
        WRITE(IOUT,10)
10      FORMAT("ZOOMING:-")
        CALL LMTARA
        CALL MNOPEN(875.,715.,1)
        CALL MNDISP(MNTXT,7,10,1)
        CALL FRAME(870.,733.,7)
        S1=XX(K3)
        S3=XX(K4)
        S2=YY(K3)
        S4=S2
        DO 50 I=K3,K4
        IF(YY(I).GT.S4) S4=YY(I)
        IF(YY(I).LT.S2) S2=YY(I)
```

```
50      CONTINUE
        CALL LMTSCL(S1,S2,S3,S4)
        CALL PFRAME(S1,S2,S3,S4)
C DRAW THE ZOOMED CURVE
        L1=K1+NPI(IF1)+1
        L2=K2-NPI(IF2)-1
        IF(L2.LT.L1) GOTO 77
        I=L1
66      CALL PLUSGN(S1,S2,S3,S4,XX(I),YY(I))
        IF1=IF1+1
        IF(I.EQ.L2) GOTO 77
        I=I+NPI(IF1)+1
        GOTO 66
 77     CALL TXMOVE(XX(K3),YY(K3))
        L1=K3+1
        DO 88 I=L1,K4
 121    CALL TXDRAW(XX(I),YY(I))
88      CONTINUE
202     CALL LMTARA
        CALL MNPICK(J,ICHAR,MNO)
17      CALL CONFRM(ICHAR)
        IF(ICHAR.EQ.78) GOTO 202
        IF(ICHAR.NE.89) GOTO 17
        GOTO(414,414,414,404,414,2,414),J
C RETURING TO CALLING PROGRAM
414     IC=J
        RETURN
C AXES MARKING
404     CALL AXSMRK(S1,S2,S3,S4)
        GOTO 202
        END
C
C **************** NEWTON DIVIDED DIFFERENCE (GLOBLE)***************
C
        SUBROUTINE NEWTRF(N,N1,X1,Y1,XX,YY,C)
        DIMENSION N1(1),X1(1),Y1(1),XX(1),YY(1),C(50,6)
C COMPUTE THE POLYNOMIAL COEFFICIENT
        N2=N-1
        DO 1 K=1,N2
          J=N-K
          C(J+1,1)=(Y1(J+1)-Y1(J))/(X1(J+1)-X1(J))
1       CONTINUE
55      C(1,1) =Y1(1)
        N3=N-2
        DO 2 J=1,N3
          K=J+2
          DO 3 L=K,N
              I=N-L+K
              XXX=X1(I)-X1(I-(J+1))
              C(I,1)=(C(I,1)-C(I-1,1))/XXX
3        CONTINUE
2       CONTINUE
C EVALUTE INTERPOLATION FUNCTION
66      IP=1
        DO 4 I=1,N2
          T1=X1(I+1)-X1(I)
          R1=T1/(N1(I)+1)
          IP1=IP+N1(I)+1
          XX(IP)=X1(I)
          Z=X1(I)
          XX(IP1)=X1(I+1)
          YY(IP)=0
```

```
          YY(IP1)=0
          N11=N1(I)
          DO 5 K=1,N11
            XX(IP+K)=Z+R1
            Z=Z+R1
            A=C(N,1)
            DO 7 L=1,N2
              J=N-L
              A=C(J,1)+(Z-X1(J))*A
7           CONTINUE
77          YY(IP+K)=A-YY(IP+K)
5         CONTINUE
          IP=IP1
4       CONTINUE
        RETURN
        END
C
C *************** OUTPUT COEFFICIENTS**********************
C
        SUBROUTINE OUTCOF(NCOEF,IPNTR,METHOD,C,IDEV)
        DIMENSION C(50,6)
        I=IPNTR
        CALL CURPOS(1.,580.)
        IF(METHOD.EQ.2) GOTO 1
C OUTPUT TABLE HEADER
        WRITE(IDEV,10)
10      FORMAT(" I",7X,"C1",11X,"C2",11X,"C3",11X,"C4"/)
C OUPUT SPLINE COEFFICIENTS SOFT/HARD COPY
3       DO 5 J=1,NCOEF
            WRITE(IDEV,40)I,C(I,1),C(I,2),C(I,3),C(I,4)
            I=I+1
40          FORMAT(I2,4(2X,E11.4))
5       CONTINUE
        GOTO 7
C OUTPUT PIECEWISE POLYNOMIAL COEFFICIENTS
1       WRITE(IDEV,20)
20      FORMAT(" I",8X,"C1",8X,"C2",8X,"C3",10X,"C4",10X,"C5",9X,"C6"/)
        DO 6 J=1,NCOEF
          WRITE(IDEV,50)I,C(I,1),C(I,2),C(I,3),C(I,4),C(I,5),C(I,6)
          I=I+1
50        FORMAT(I2,E11.4,4(X,E11.4),E11.4)
6       CONTINUE
7       RETURN
        END
C
C**************OUTPUT A PAGE OF THE TABLE************************
C
        SUBROUTINE OUTPGE(NPOINT,IPNT,XX,YY,IDEV)
        DIMENSION XX(1),YY(1)
        CALL CURPOS(1.,710.)
C OUPUT A PAGE OF TABLE OF THE INTERPOLATED POINTS
        WRITE(IDEV,10)
10      FORMAT(//" I",7X,"X(I)",8X,"Y(I)",7X," I",5X,"X(I)",8X,"Y(I)"/)
        IF(NPOINT/2*2.LT.NPOINT) GOTO 4
        IPNT1=IPNT+NPOINT/2
        GOTO 2
4       IPNT1=IPNT+NPOINT/2+1
2       DO 3 I=2,NPOINT,2
          WRITE(IDEV,20)IPNT,XX(IPNT),YY(IPNT),IPNT1,XX(IPNT1),YY(IPNT1)
          IPNT=IPNT+1
          IPNT1=IPNT1+1
          IF(I+1.NE.NPOINT) GOTO 3
```

```
          WRITE(IDEV,30)IPNT,XX(IPNT),YY(IPNT)
3         CONTINUE
20        FORMAT(2(I2,2X,2(E11.4,3X)))
30        FORMAT(I2,2X,2(E11.4,3X))
          RETURN
          END
C
C****************** OUTPUT THE TITLE OF THE DISPLAY********
C                   *******AND THE COMMAND MENU***************
C
          SUBROUTINE OUTTIL(IC,ITEM,IDEV)
C MENU ITEMS
          DATA MNTXT1/"+ NEXT    + PREVIOUS+ HARDCOPY+ COEFF'NT+ FORWARD
&+ BACKWARD+ HELP     + RESTART + EXIT     "/
          DATA MNTXT2/"+ NEXT    + PREVIOUS+ HARDCOPY+ FORWARD + BACKWARD
&+ HELP     + RESTART + EXIT     "/
          LOGICAL*1 MNTXT1(90),MNTXT2(80)
          CALL TXCLER
          IF(IC.EQ.2) GOTO 2
C OUTPUT THE INTERPOLATED POINTS
          WRITE(IDEV,10)
10        FORMAT("COMPLETE TABLE OF THE INTERPOLATED POINTS:-")
          GOTO 3
2         WRITE(IDEV,20)
20        FORMAT(///"              POLYNOMIAL COEFFICIENTS:-"////)
3         CALL MNOPEN(875.,715.,1)
          IF(IC.EQ.2) GOTO 22
          CALL MNDISP(MNTXT1,ITEM,10,1)
          GOTO 4
22        CALL MNDISP(MNTXT2,ITEM,10,1)
4         CALL FRAME(870.,733.,ITEM)
          RETURN
          END
C
C ***********INPUT COMMON DATA FOR SUPERIMPOSED DISPLAY**********
C
          SUBROUTINE RDCRVS
          COMMON/CURVES/NCRV(10),XYSCL(4).
          REWIND 7
          CALL SETFIL(7,"SUPCRVES")
          READ(7,10) (NCRV(I),I=1,10)
          READ(7,20) (XYSCL(I),I=1,4)
10        FORMAT(I2)
20        FORMAT(F11.4)
          ENDFILE 7
          RETURN
          END
C
C********** DISPLAY THE SUPERIMPOSED CURVES*********************
C
          SUBROUTINE SREDRW(N,N1,NC,XX,YY,MSUM,SCL1,SCL2,SCL3,SCL4)
          DIMENSION N1(1),XX(1),YY(1)
          READ(9,25)MSUM,N
          N2=N-1
          READ(9,30) (XX(I),YY(I),I=1,MSUM)
          READ(9,20) (N1(I),I=1,N2)
30        FORMAT(F11.4)
20        FORMAT(I3)
25        FORMAT(2I3)
          PC= EPLOT(NC,SCL1,SCL2,SCL3,SCL4,MSUM)
          RETURN
          END
```

```
C ********* FIND TOTAL NUMBER OF INTERPOLATED POINTS*X**********
        SUBROUTINE SUM(N,N1,MSUM)
        DIMENSION N1(1)
        MSUM=0
        N2=N-1
        DO 1 I=1,N2
1       MSUM=MSUM+N1(I)
        MSUM=MSUM+N
        RETURN
        END
C
C***********WRITE COMMOM BLOCK CURVES*************
C
        SUBROUTINE WRCRVS
        COMMON/CURVES/NCRV(10),XYSCL(4)
        REWIND 8
        CALL SETFIL(8,"SUPCRVES")
        WRITE(8,10)(NCRV(I),I=1,10)
        WRITE(8,20)(XYSCL(I),I=1,4)
10      FORMAT(I2)
20      FORMAT(F11.4)
        ENDFILE 8
        RETURN
        END
```

```
C                          ****************
C                          * APPENDIX 2.26 *
C                          ****************
C
C THIS MODULE HANDLES THE JOIN DISPLAY AND ITS SUBSEQUENT ZOOMING
C
C
C
C ********* MAIN PROGRAM - MODULE 6 ***************
C
        COMMON/JOIN/CJ1(500),CJ2(500),J3(12),J4(100),IPNTR(6)
        COMMON/IO/IN,IOUT
        DATA MODL4,MODL1,HELP/"MOD4","EXPLICIT","HELP"/
        LOGICAL*1 MODL4(10),MODL1(10),HELP(10)
        IN=5
        IOUT=6
        CALL TXOPEN
        CALL RDCOMJ
        CALL JOIN(CJ1,CJ2,J3,J4,IPNTR,IC)
        GOTO(2,3,1,1,1,1,7,1,8,1,1),IC
C PROGRAM TERMINATES
1       CALL EXIT
C NEXT DISPLAY
2       STOP
C PREVIOUS DISPLAY
3       CALL OVRLAY(MODL4)
C ALGORITHM DISPLAY
7       IPREV=0
        IHELP=0
        CALL WRCOM1
        CALL OVRLAY(MOD1)
C HELP DISPLAY
8       CALL WRCOMJ
        CALL OVRLAY(HELP)
        STOP
        END
C
C**************JOIN DISPLAY*************************
C
        SUBROUTINE JOIN(XJ,YJ,JJ3,JJ4,IPNTR,IC)
        COMMON/IO/IN,IOUT
C MENU ITEMS
        DATA MNTXT/"+ NEXT    + PREVIOUS+ GRAPH   + COORDS. + DISP.ORG
&+ ZOOM    + METHOD  + AX. MARK+ HELP    + RESTART + EXIT    "/
        DIMENSION XJ(1),YJ(1),JJ3(1),JJ4(1),IPNTR(1),JM1(11)
        LOGICAL*1 MNTXT(110)
        EXTERNAL JPLOT
C NUMBER OF CURVE SEGMENTS TO BE JOINED
        JM1(1)=IPNTR(1)-2
        IT=0
        K=2*JM1(1)+1
C COMPUTE TOTAL NUMBER OF POINTS
        DO 1 I=3,K,2
          JM1(I-1)=0
          K1=JJ3(I+1)
          K2=JJ3(I+1)+JJ3(I)-2
          DO 12 J=K1,K2
            JM1(I-1)=JM1(I-1)+JJ4(J)
12        CONTINUE
          JM1(I-1)=JM1(I-1)+JJ3(I)
          JM1(I)=JJ3(I+1)
```

```
          IT=IT+JM1(I-1)
1        CONTINUE
         MSUM=IT-JM1(1)+1
C SCALES FOR PLOTTING THE JOINED CURVES
         CALL MINMAX(S1,S2,S3,S4,XJ,YJ,MSUM)
11       CALL TXCLER
         ICORD=0
         WRITE(IOUT,10)
10       FORMAT("JOINED CURVES:-")
C OUTPUT MENU
         CALL MNOPEN(875.,715.,1)
         CALL MNDISP(MNTXT,11,10,1)
         CALL FRAME(870.,733.,11)
2        CALL LMTARA
C CURSOR CHOICE
         CALL MNPICK(J,ICHAR,MNO)
         IF(J.EQ.4.AND.ICORD.EQ.2)GOTO 2
17       CALL CONFRM(ICHAR)
         IF(ICHAR.EQ.78) GOTO 2
         IF(ICHAR.NE.89) GOTO 17
         GOTO(21,21,23,24,25,26,21,28,21,11,21),J
21       IC=J
         RETURN
C PLOT THE CURVES
23       CALL DRAW(JPLOT,R,S1,S2,S3,S4,JM1)
         GOTO 2
C CURSOR INPUT COORDINATES
24       CALL DISCOR(ICORD,S1,S2,S3,S4)
         GOTO 2
C DISPLAY ORIGIN
25       CALL DISORG(S1,S2,S3,S4)
         GOTO 2
C ZOOMING
26       CALL JZOOM(JM1,XJ,YJ,JJ4,IPNTR,S1,S2,S3,S4,IC)
         GOTO(27,11,27,29,27,26,29),IC
         GOTO2
C AXES MARKING
28       CALL AXSMRK(S1,S2,S3,S4)
         GOTO 2
29       STOP
27       IF(IC.EQ.3)J=7
         IF(IC.EQ.5)J=9
         GOTO 21
         END
C
C***************ZOOMING IN JOINED CYRVES********************
C
         SUBROUTINE JZOOM(JM1,XJ,YJ,J4,IPNTR,SCL1,SCL2,SCL3,SCL4,IC)
         COMMON/IO/IN,IOUT
         DATA MNTXT/"+ NEXT     + PREVIOUS+ METHODS + AX. MARK+ HELP
&+ RESTART + EXIT     "/
         DIMENSION JM1(1),XJ(1),YJ(1),J4(1),IPNTR(1),CX(2),LB(2),N11(2),M1(2)
         LOGICAL*1 MNTXT(70)
         CALL LMTSCL(SCL1,SCL2,SCL3,SCL4)
         DY=(SCL4-SCL2)/20
         DO 1 I=1,2
         CALL TXCURS(CXX,CYY,ICHAR)
         CX(I)=CXX
         CALL TXMOVE(CXX,CYY+DY)
         CALL TXDRAW(CXX,CYY-DY)
1        CONTINUE
C SORT FOR SMALLER CXX
```

```
        IF(CX(1).LT.CXX) GOTO 2
        CX(2)=CX(1)
        CX(1)=CXX
C DETERMINE WHICH SUBSET
2       DO 11 J=1,2
        K=JM1(1)
3       DO 4 I=1,K
        IF(I.EQ.K) GOTO 5
        IF(CX(J).LE.XJ(IPNTR(I+2))) GOTO 5
4       CONTINUE
5       LB(J)=I
11      CONTINUE
        DO 9 J=1,2
        M1(J)=JM1(2*LB(J))
        N11(J)=JM1(2*LB(J)+1)
9       CONTINUE
        I=IPNTR(LB(1)+1)
        IF1=N11(1)-1
71      IF(CX(1).GT.XJ(I)) GOTO 7
        K11=I
        K1=I-J4(IF1)-1
        GOTO 8
7       IF1=IF1+1
        I=I+J4(IF1)+1
        GOTO 71
8       I=IPNTR(LB(2)+1)
        IF2=N11(2)-1
81      IF(CX(2).GT. XJ(I)) GOTO 91
        K2=I
        GOTO 101
91      IF2=IF2+1
        I=I+J4(IF2)+1
        GOTO 81
101     L2=K1+J4(IF1)+1
        DO 22 I=K1,L2
        IF(CX(1).LE.XJ(I)) GOTO 23
22      CONTINUE
23      K3=I-1
        L2=J4(IF2)+1
        DO 24 I=1,L2
        L1=K2-I+1
        IF(CX(2).GT.XJ(L1)) GOTO 25
24      CONTINUE
25      K4=L1+1
12      CALL TXCLER
        CALL ALPHMD
        WRITE(IOUT,20)
20      FORMAT("ZOOMING:-")
        CALL LMTARA
        CALL MNOPEN(875.,715.,1)
        CALL MNDISP(MNTXT,7,10,1)
        CALL FRAME(870.,733.,7)
        S1=XJ(K3)
        S3=XJ(K4)
        S2=YJ(K3)
        S4=S2
        DO 50 I=K3,K4
        IF(YJ(I).GT.S4) S4=YJ(I)
        IF(YJ(I).LT.S2) S2=YJ(I)
50      CONTINUE
C DRAW THE ZOOMED CURVE
        CALL LMTSCL(S1,S2,S3,S4)
```

```
        CALL PFRAME(S1,S2,S3,S4)
        IF(K11.EQ.K2) GOTO 45
        IF3=1
        IF(LB(1).EQ.LB(2).AND. IF3.EQ.1) GOTO 31
33      IF(LB(1).EQ.LB(2)) GOTO 32
        N3=J4(IF1)+1
        L1=K1+N3
        L2=IPNTR(LB(1)+2)
        I=L1
35      CALL PLUSGN(S1,S2,S3,S4,XJ(I),YJ(I))
        IF1=IF1+1
        IF(I.EQ.L2) GOTO 66
        I=I+J4(IF1)+1
        GOTO 35
66      LB(1)=LB(1)+1
        N11(1)=JM1(2*LB(1)+1)
        K1=L2
        GOTO 33
32      N3=J4(IF1)+1
        L1=L2+N3
        N4=J4(IF2)+1
        L2=K2-N4
40      I=L1
37      CALL PLUSGN(S1,S2,S3,S4,XJ(I),YJ(I))
        IF1=IF1+1
        IF(I.EQ.L2) GOTO 45
        I=I+J4(IF1)+1
        GOTO 37
31      N3=J4(IF1)+1
        L1=K1+N3
        N4=J4(IF2)+1
        L2=K2-N4
        GOTO 40
45      CALL TXMOVE(XJ(K3),YJ(K3))
        L1=K3+1
        DO 55 I=L1,K4
55      CALL TXDRAW(XJ(I),YJ(I))
  202   CALL LMTARA
        CALL MNPICK(J,ICHAR,MNO)
17      CALL CONFRM(ICHAR)
        IF(ICHAR.EQ.78) GOTO 202
        IF(ICHAR.NE.89) GOTO 17
        GOTO(41,41,41,404,41,12,41),J
41      IC=J
        RETURN
C AXES MARKING
404     CALL AXSMRK(S1,S2,S3,S4)
        GOTO 202
        END
C
C*************** PLOT JOINED CURVES********************
C
        SUBROUTINE JPLOT(R,SCL1,SCL2,SCL3,SCL4,JM1)
C JOIN COMMON DATA AREA
        COMMON/JOIN/CJ1(500),CJ2(500),J3(12),J4(100),IPNTR(6)
          DIMENSION JM1(1)
        J=3
        M=0
        K=JM1(1)
        DO 1 I=1,K
          M=M+JM1(J-1)-I+1
          N1=JM1(J)
```

```
          IP=IPNTR(I+1)
          IP1=IP
          K1=IP
          IF1=N1
C OUTPUT SUPPLIED POINTS
4         CALL PLUSGN(SCL1,SCL2,SCL3,SCL4,CJ1(K1),CJ2(K1))
          CALL TXMOVE(CJ1(K1),CJ2(K1))
          IF(K1.EQ.M) GOTO 3
          L11=IP1+1
          L22=IP1+J4(IF1)+1
C OUTPUT INTERPOLATED POINTS
          DO 2 L1=L11,L22
            CALL TXDRAW(CJ1(L1),CJ2(L1))
2         CONTINUE
          IP1=L22
          IF1=IF1+1
          K1=K1+J4(IF1-1)+1
          GOTO 4
3         J=J+2
1       CONTINUE
        RETURN
        END
```

APPENDIX 2.3

THE IDF - PARAMETRIC PACKAGE

```
C                         ****************
C                         * APPENDIX 2.31 *
C                         ****************
C
C THIS MODULE HANDLES THE FOLLOWING INTERACTIVE DISPLAYS:-
C                1.INTRODUCTORY
C                2.CHOICE OF THE NUMERICAL ALGORITHM
C                3.DATA ENTRY
C                4.DATA TABULATION
C                5.DATA POINT EDITING
C
C
C
C ******** MAIN PROGRAM - MODULE 1 ************
C
C
C INPUT COMMON DATA AREA
        COMMON/DATSUP/NPS,NPI(50),IFREES,X(50),Y(50),Z(50),L(50),IH(5),
&                   M(5),METHOD,IHELP,IPREV,BOUND(6),SUBSET,INTPNT,IE(2)
&                       ,ID.
        COMMON/IO/IN,IOUT
        DATA PMODL2,PHELP/"PMODL2","HELP"/
        LOGICAL*1 PMODL2(10),PHELP(10)
        INTEGER SUBSET
        IN=5
        IOUT=6
        CALL TXOPEN
C READ INPUT DATA IF NOT FOR THE FIRST TIME
        IF(IERROR(103).NE.0)GOTO 1
        CALL PRDCM1
        IF(IHELP.NE.0)GOTO 55
        IF(IPREV.EQ.1) GOTO 21
C CALL DISPLAYS SEQUENCE FOR DATA ENTRY AND EDITING
1       CALL INTROD(IC)
        GOTO(2,3000,1111),IC
2       CALL MENU(METHOD,IC)
        GOTO(10,1,3000,1111,1111),IC
10      CALL DATENT(IC,IA)
        GOTO(20,2,3000,1111,1111),IC
20      CALL DATMAN(IA,IC)
        GOTO(1000,10,40,1111,1111,3000,1111,1111),IC
40      CALL EDIT(IC,IA)
C CALL BACK THE DATA MANIPILATION DISPLAY
        IF(IC.GT.1) GOTO 20
C NEXT DISPLAY IN SEQUENCE
1000    IPREV=0
        IHELP=0
        CALL PWRCM1
        CALL OVRLAY(PMODL2)
C HELP DISPLAY
3000    CALL PWRCM1
        CALL OVRLAY(PHELP)
C TERMINATE PROGRAM
1111    CALL PEXIT
        STOP
C LINK LIST UNCHANGED
21      IPREV=0
        IHELP=0
        IA=111
        GOTO 20
C RETURN FROM HELP DISPLAY
```

```
55      GOTO(1,2,10,21),IHELP
        END
C
C ************INTRODUCTORY DISPLAY *****************************
C
        SUBROUTINE INTROD(IC)
        COMMON /IO/ IN,IOUT
C MENU ITEMS
        DATA MNTXT/"+ NEXT    + HELP    + EXIT    "/
        LOGICAL*1 MNTXT(30)
C SET UP THE INTRODUCTORY DISPLAY
        CALL TXCLER
        CALL CURPOS(1.,780.)
        CALL TEXTUP("PRINTEXT",34)
        CALL MNOPEN(875.,715.,1)
        CALL MNDISP(MNTXT,3,10,1)
        CALL FRAME(870.,733.,3)
2       CALL MNPICK(J,ICHAR,MNO)
22      CALL CONFRM(ICHAR)
        IF(ICHAR.EQ.78) GOTO 2
        IF(ICHAR.NE.89) GOTO 22
        IC=J
        RETURN
        END
C
C**********DATA FITTING ALGORITHM*******************
C
        SUBROUTINE MENU(M,IC)
C RETURNS ALGRITHM INDEX
        COMMON /IO/IN,IOUT
C MENU ITEMS
        DATA MNTXT1 /"+ NEXT    + PREVIOUS+ HELP    + RESTART + EXIT    "/
        LOGICAL*1 MNTXT1(50)
7       CALL TXCLER

C OUTPUT THE ALGORITHM LIST
        WRITE(IOUT,10)
10      FORMAT("INDICATE YOUR CHOICE OF ALGORITHM:-")
        CALL MNOPEN(875.,715.,1)
        CALL DTEXT(20.,700.,"***PARAMETRIC PACKAGE FOR INTERPOLATORY
& DATA FITTING***",55)
        CALL DTEXT(70.,650.,"NUMERICAL ALGORITHMS:-",22)
        CALL MNDISP(MNTXT1,5,10,1)
        CALL FRAME(870.,733.,5)
        CALL MNOPEN(60.,600.,2)
        CALL MNTEXT("1-CUBIC SPLINE(PARAMETRIC SECOND DERV. END CONDITION)",53)
        CALL MNTEXT("2-CUBIC SPLINE(CYCLIC END CONDITION)",36)
        CALL MNTEXT("3-CUBIC SPLINE(ANTICYCLIC END CONDITION)",41)
        CALL MNTEXT("4-CUBIC SPLINE(VARIABLE END CONDITION)",39)
C SET UP CURSOR FOR MENU CHOICE
1       CALL MNPICK(I,ICHAR,MNO)
        IF(MNO.EQ.1)GOTO 3
        M=I
        GOTO 1
3       CALL CONFRM(ICHAR)
        IF(ICHAR.EQ.78) GOTO 1
        IF(ICHAR.NE.89) GOTO 3
        IF(I.EQ.4)GOTO 7
        IC=I
        RETURN
        END
C
```

```
C***X********DATA ENTRY DHSPLAY ROUTINE********************
C
      SUBROUTINE DATENT(IC,IA)
C DATA ENTRY DISPLAY
C INPUT COMMON DATA AREA
      COMMON/DATSUP/NPS,NPI(50),IFREES,X(50),Y(50),Z(50),L(50),IH(5)
&                ,M(5),METHOD,IHELP,IPREV,BOUND(6),SUBSET,INTPNT,IE(2)
&                ,ID
      COMMON /IO/IN,IOUT
C MENU ITEMS
      DATA MNTXT1/"+ NEXT    + PREVIOUS+ HELP    + RESTART + EXIT    "/
      DATA MNTXT2/"+ NEW     + OLD                + 2-DIMEN.+ 3-DIMEN.
&        + KEYBOARD+DISC FILE"/
      LOGICAL*1 MNTXT1(50),MNTXT2(80)
      INTEGER SUBSET
C SET UP DATA ENTRY DISPLAY
23    CALL TXCLER
      WRITE(IOUT,10)
10    FORMAT("DATA ENTRY:-")
C OUTPUT INSTRUCTION TO THE USER
      CALL CURPOS(1.,700.)
      WRITE(IOUT,11)
11    FORMAT("SELECT THE APPROPRIATE"/"DATA SPECIFICATION(*):-")
      WRITE(IOUT,1222)
1222  FORMAT(///2X,"1-STATE OF DATA:-"///2X,"2-DIMENSIONALITY:-"///
&          2X,"3-DATA MEDIUM ENTRY:-")
      WRITE(IOUT,20)
20    FORMAT(///////////////"* IF 'OLD' IS SELECTED YOU MAY PROCEED"/
&     "  TO NEXT DISPLAY IMMEDIATELY .OTHERWISE "/
&     "  YOU MUST SELECT THE DESIRED MEDIUM")
C DISPLAY MENU,RAISE CURSOR & WAIT FOR USER ACTION
      CALL MNOPEN(875.,715.,1)
      CALL MNDISP(MNTXT1,5,10,1)
      CALL FRAME(870.,733.,5)
      CALL MNOPEN(320.,600.,2)
      CALL MNDISP(MNTXT2,8,10,2)
      CALL FRAME(315.,620.,8)
      CALL TXMOVE(315.,570.)
      CALL TXDRAW(460.,570.)
      CALL TXMOVE(315.,500.)
      CALL TXDRAW(460.,500.)
      NFLAG=1
      MD   =0
      IS   =0
5     CALL MNPICK(J,ICHAR,MNO)
C FIRST OR SECOND MENU
      IF(MNO.EQ.1) GOTO 2
      IF(IS.EQ.2.OR.NFLAG.EQ.0.OR.J.EQ.3.OR.J.EQ.6) GOTO 5
      IF(J.LT.3) GOTO 7
      IF(J.LT.6) GOTO 8
      MD=J
      CALL CURPOS(1.,400.)
3     CALL MESSAG("# NUMBER OF DATA POINTS(MAX.50)?^")
      READ(IN,30)N
30    FORMAT(G0.0)
      IF(N.GT.50.OR.N.LT.3) GOTO 3
      NPS=N
      IF(J.EQ.8)GOTO 1010
      WRITE(IOUT,40)
40    FORMAT(/,"# X-COORDS.:-")
      IF(IERROR(110).NE.0)GOTO 100
 35   READ(IN,50)(X(I),I=1,N)
```

```
50      FORMAT(50G0.0)
        WRITE(IOUT,70)
70      FORMAT(/,"# Y-COORDS.:-")
        IF(IERROR(110).NE.0)GOTO 110
65      READ(IN,50)(Y(I),I=1,N)
        IF(ID.NE.3) GOTO 333
        WRITE(IOUT,80)
80      FORMAT(/,"# Z-COORDS.:-")
        IF(IERROR(110).NE.0) GOTO 120
75      READ(IN,50)(Z(I),I=1,N)
333     IHELP=3
        GOTO 5
8       IF(J.EQ.4) ID=2
        IF(J.EQ.5) ID=3
        GOTO 5
100     WRITE(IOUT,105)
105     FORMAT("ILLEGAL X-COORDS.,TRY AGAIN")
        ENDFILE 5
        GOTO 35
110     WRITE(IOUT,115)
115     FORMAT("ILLEGAL Y-COORDS.,TRY AGAIN")
        ENDFILE 5
        GOTO 65
120     WRITE(IOUT,118)
118     FORMAT("ILLEGAL Z-COORDS.,TRY AGAIN")
        ENDFILE 5
        GOTO 75
2       CALL CONFRM(ICHAR)
        IF(ICHAR.EQ.78) GOTO 5
        IF(ICHAR.NE.89)GOTO 2
        IF(J.EQ.4)GOTO 23
        IF(NFLAG.EQ.1.AND.J.EQ.1.AND.MD.EQ.0) GOTO 5
        IC=J
        IF(J.EQ.1.AND.ID.EQ.0) GOTO 5
        RETURN
1010    IFLG=0
C NEW DATA POINTS
        IA=0
        CALL GETFLN(FILE)
        CALL READAT(FILE,X,Y,Z,NPS,IFLG,ID)
        IF(IFLG.EQ.1)GOTO 1010
        IF(IFLG.EQ.2)GOTO 3
        GOTO 5
313     ENDFILE 5
        GOTO 3
C NEW DATA POINTS
7       IA=0
C OLD DATA POINTS
        IF(J.EQ.2) IA=111
        IS=J
        IF(J.EQ.2.AND.MD.GT.0) GOTO 5
        IF(J.EQ.2.AND.MD.EQ.0) GOTO 12
        IF(J.EQ.1) GOTO 5
12      NFLAG=0
        GOTO 5
        END
C
C *************DATA POINTS TABULATION DISPLAY***************
C
        SUBROUTINE DATMAN(IA,IC)
C INPUT COMMON DATA AREA
        COMMON/DATSUP/NPS,NPI(50),IFREES,X(50),Y(50),Z(50),L(50),IH(5)
```

```
&                   ,M(5),METHOD,IHELP,IFREV,BOUND(6),SUBSET,INTPNT,IE(2)
&                          ,ID
        COMMON/IO/IN,IOUT
C MENU ITEMS
        DATA MNTXT/"+ NEXT    + PREVIOUS+ EDIT    + SORT-X  + SAVE
&+ HELP    + RESTART + EXIT    "/
        DIMENSION X0(50),Y0(50),Z0(50)
        LOGICAL*1 MNTXT(80)
        INTEGER SUBSET,S
C NEW/OLD?
22      IF(IA.EQ.111)GOTO7
111     N=NPS-1
C SET LINK LIST
        DO 1 I=1,N
1       L(I)=I+1
        L(NPS)=0
        IFREES=NPS+1
        DO 2 I=1,5
2       M(I)=10*I
        S=0
        DO 3 I=1,5
          IH(I)=S*10+1
3         S=S+1
C SET DISPLAY MENU AND DATA POINTS TABLE
7       CALL TXCLER
        WRITE(IOUT,10)
10      FORMAT("TABULATION OF DATA:-")
        CALL MNOPEN(875.,715.,1)
        CALL MNDISP(MNTXT,8,10,1)
        CALL FRAME(870.,733.,8)
        CALL TABLE
        IHELP=4
C SET UP CURSOR MENU PICKING
4       CALL MNPICK(J,ICHAR,MNO)
77      CALL CONFRM(ICHAR)
        IF(ICHAR.EQ.78)GOTO 4
        IF(ICHAR.NE.89) GOTO 77
        IF(J.EQ.4) GOTO 44
        IF(J.EQ.5) GOTO 444
        IF(J.EQ.7) GOTQ 22
        IC=J
        RETURN
C SORT DATA POINTS IN X
44      CALL PRMLNK(X0,Y0,Z0)
        CALL SORTX(X0,Y0,Z0,NPS,ID)
        DO 101 I=1,NPS
          X(I)=X0(I)
          Y(I)=Y0(I)
C THREE-DIMENSIONS
          IF(ID.EQ.3) Z(I)=Z0(I)
101     CONTINUE
        GOTO 111
C SAVE DATA POINTS ON DISC FILE AS USER REQUEST
444     CALL PRMLNK(X0,Y0,Z0)
        CALL SAVE(X0,Y0,Z0,NPS,ID)
        GOTO 4
        END
C
C***********EDIT DISPLAY*************************
C
        SUBROUTINE EDIT(IC,IA)
C INPUT COMMON DATA AREA
```

```
      COMMON/DATSUP/NPS,NPI(50),IFREES,X(50),Y(50),Z(50),L(50),IH(5),
&                 M(5),METHOD,IHELP,IPREV,BOUND(6),SUBSET,INTPNT,IE(2)
&                 ,ID
      COMMON/IO/IN,IOUT
C MENU ITEMS
      DATA MNTXT/"+ NEXT    + PREVIOUS+ CORRECT + DELETE  + INSERT
&+ RESTART + EXIT    "/
      DIMENSION A(41)
      INTEGER SUBSET
      LOGICAL*1 MNTXT(70)
1     CALL TXCLER
C SET UP DISPLAY
      WRITE(IOUT,10)
10    FORMAT("DATA POINTS EDITING:-")
      CALL MNOPEN(875.,715.,1)
      CALL MNDISP(MNTXT,7,10,1)
      CALL FRAME(870.,733.,7)
C RAISE CURSOR READY FOR USER INTERACTION
77    CALL MNPICK(J,ICHAR,MNO)
88    CALL CONFRM(ICHAR)
      IF(ICHAR.EQ.78) GOTO 77
      IF(ICHAR.NE.89) GOTO 88
C TRANSFER CONTROL TO APPROPRIATE CODE IN THE PROGRAM
      GOTO(30,30,40,50,60,1,70),J
30    CALL UPDATE
C RETURN TO MAIN PROGRAM
      IC =J
      GOTO 114
C CORRECT DATA POINTS
40    WRITE(IOUT,20)
20    FORMAT(////"CORRECTION:-")
C PROMPT USER FOR INPUTING EDITTING INFORMATION
5     CALL MESSAG("$ NUMBER OF DATA POINTS(MAX.10)?^")
      READ(IN,45) M2
45    FORMAT(G0.0)
      IF(M2.GT.10)GOTO 5
221   IF(ID.EQ.3) WRITE(IOUT,97)
      IF(ID.EQ.2) WRITE(IOUT,80)
80    FORMAT("$ ENTER I , X , Y:")
97    FORMAT("$ ENTER I , X , Y , Z :")
      LAST=(ID+1)*M2+1
      IF(IERROR(110).NE.0)GOTO 222
      READ(IN,90)(A(I),I=1,LAST)
90    FORMAT(40G0.0)
      A(LAST)=99
      CALL PCORCT(A)
      CALL UPDATE
      GOTO 100
C DELETE DATA POINTS
50    WRITE(IOUT,110)
110   FORMAT(////"DELETION:-")
C USER INPUT DELETE INFORMATION
7     CALL MESSAG("$ NUMBER OF DATA POINT(MAX.30)?^")
      READ(IN,45)M2
      IF(M2.GT.30)GOTO 7
125   WRITE(IOUT,130)
130   FORMAT("$ ENTER I IN DESCENDING ORDER:")
      LAST=M2+1
      IF(IERROR(110).NE.0)GOTO 333
      READ(IN,90)(A(I),I=1,LAST)
      A(LAST)=99
      IF(M2.EQ.1)GOTO 11
```

```
        M1=M2-1
        DO 9 I=1,M1
          IF(A(I).LT.A(I+1))GOTO 125
9       CONTINUE
11      CALL PDELET(A)
        CALL UPDATE
        GOTO 100
C DATA POINTS INSERTION
60      WRITE(IOUT,150)
150     FORMAT(////"INSERTION:-")
155     CALL MESSAG("# NUMBER OF DATA POINTS (MAX. 1 PER INTERVAL,TOTAL 10?^")
        READ(IN,45)M2
        IF(M2.GT.10) GOTO 155
165     IF(ID.EQ.3)WRITE(IOUT,180)
        IF(ID.EQ.2)WRITE(IOUT,170)
170     FORMAT("# ENTER I , X , Y  IN DESCENDING ORDER:")
180     FORMAT("# ENTER I , X , Y , Z IN DESCENDING ORDER:")
        LAST=(ID+1)*M2+1
        IF(IERROR(110).NE.0)GOTO 444
        READ(IN,90)(A(I),I=1,LAST)
        A(LAST)=99
        IF(M2.EQ.1)GOTO 190
        IF(ID.EQ.3)LAST1=4*M2-7
        IF(ID.EQ.2)LAST1=3*M2-5
        ID1=ID+1
        DO 18 I=1,LAST1,ID1
          IF(A(I).LT.A(I+ID+1)) GOTO 165
18      CONTINUE
190     CALL PADD(A)
        CALL UPDATE
100     IC=3
114     IA=111
        RETURN
222     LFLAG=1
224     WRITE(IOUT,223)
223     FORMAT("WRONG INPUT !,TRY AGAIN")
        ENDFILE 5
        GOTO(221,125,165),LFLAG
333     LFLAG=2
        GOTO 224
444     LFLAG=3
        GOTO 224
70      STOP
        END
C
C********** H E L P   D I S P L A Y ******************
C
C INPUT COMMON DATA AREA
        COMMON/DATSUP/NPS,NPI(50),IFREES,X(50),Y(50),Z(50),L(50),IH(5),
&                M(5),METHOD,IHELP,IPREV,BOUND(6),SUBSET,INTPNT,IE(2)
&                          ,ID
        COMMON/IO/IN,IOUT
C MENU ITEMS
        DATA MNTXT1/"+ PREVIOUS+ EXIT      "/
C OVERLAY EXECUTABLE PROGRAM NALES
        DATA PMODL1,PMODL2,PMODL3,PMODL4,PMODL5,PMODL6/"PARAMETRIC"
&,"PMODL2","PMODL3","PMODL4","PMODL5","PMODL6"/
        LOGICAL*1 MNTXT1(20),PMODL1(20),PMODL2(10),PMODL3(10)
        LOGICAL*1 PMODL4(10),PMODL5(10),PMODL6(10)
        IN=5
        IOUT=6
        CALL TXOPEN
```

```
C GET INPUT FILE
        CALL PRDCM1
1       CALL TXCLER
C OUTPUT DISPLAYS TITLES
        WRITE(IOUT,10)
10      FORMAT(10X,"*************** H E L P *******************"/
&       //5X,"THE FOLLOWING DISPLAY SEQUENCE CONSTITUTE THE COMPLETE"/
& 5X,"DATA FITTING PROCESS."//
&       5X,"YOU MAY ENTER ANY OF THESE DISPLAYS BY USING THE CROSS-HAIR"/
& 5X,"CURSOR ON THE T4010 OR TRACKING CROSS ON LIGHT PEN ON THE GT42 :-"/)
        CALL MNOPEN(50.,540.,1)
        CALL MNTEXT("+ INTRODUCTION:- BRIEFLY GIVING THE USE OF THE SYST
&EM.",54)
        CALL MNTEXT("+ ALGORITHMS:- LIST OF AVAILABLE INTERPOLATORY METH
&ODS.",55)
        CALL MNTEXT("+ DATA ENTRY:- ENTER DATA POINTS INTO THE SYSTEM FR
&OM DISC FL/KEYBD.",68)
        CALL MNTEXT("+ TABULATION OF DATA POINTS:- INCLUDES EDIT,SORT &
&SAVE DATA POINTS.",68)
        CALL MNTEXT("+ POLYGONAL PLOT:- DATA POINTS JOINED BY STRAIGHT L
&INE SEGMENTS.",64)
        CALL MNTEXT("+ PARAMETER ENTRY:- PARAMETERS REQUIRED BY PARTICUL
&AR ALGORITHM.",65)
        CALL MNTEXT("+ CURVE FIT:- DISPLAY OF THE SMOOTH CURVE INCLUDES
&ZOOM OPTION..ETC.",69)
        CALL MNTEXT("+ CURVE DESIGN:- INTERMEDIATE POINTS SPECIFIED
& BY CURSOR POSITION.",66)
        CALL MNTEXT("+ TABLE OF INTERPOLATED POINTS:- INCLUDES OPTION FO
&R COEFF.&HARDCOPY.",70)
        CALL MNTEXT("+ SUPERIMPOSED CURVE:- SIMULTANEOUS DISPLY OF SEVER
&AL CURVES",63)
        CALL MNTEXT("+ USAGE OF CONTROL COMMANDS:- LIST OF ALL COMMAND U
&SED HERE.",59)
        CALL MNTEXT("+ TERMINATE THE PROCESS:- EXIT FROM THE SYSTEM.",47)
2       CALL MNPICK(I,ICHAR,MNO)
3       CALL CONFRM(ICHAR)
        IF(ICHAR.EQ.78) GOTO 2
        IF(ICHAR.NE.89) GOTO 3
        IF(I.GT.4) GOTO 5
C OVERLAY THE APPROPRIATE MODULES
        IHELP=I
        CALL PWRCM1
        CALL OVRLAY(PMODL1)
5       IF(I.GT.6) GOTO 6
        IHELP=I-4
        CALL PWRCM1
        CALL OVRLAY(PMODL2)
6       IF(I.LT.9.OR.I.GT.10) GOTO 7
        IHELP=I-8
        CALL PWRCM1
        CALL OVRLAY(PMODL6)
7       IF(I.EQ.7) CALL OVRLAY(PMODL4)
        IF(I.EQ.8) CALL OVRLAY(PMODL5)
        IF(I.NE.12) GOTO 133
C PROGRAM TERMINATE
        CALL PEXIT
        STOP
C DISPLAY COMMAND USAGE DISPLAY
133     CALL TXCLER
        CALL MNOPEN(875.,715.,1)
        CALL MNDISP(MNTXT1,2,10,1)
        CALL FRAME(870.,733.,2)
```

```
      CALL ALPHMD
      CALL CURPOS(1.,770.)
      CALL TEXTUP("PHELPTXT",26)
22    CALL MNPICK(J,ICHAR,MNO)
222   CALL CONFRM(ICHAR)
      IF(ICHAR.EQ.78) GOTO 22
      IF(ICHAR.NE.89) GOTO 222
      GOTO (1,111),J
111   CALL PEXIT
      STOP
      END
C
C **********EDIT-INSERT FUNCTION**X*************************
C
      SUBROUTINE PADD(C)
C INPUT COMMON DATA AREA
      COMMON/DATSUP/NPS,NPI(50),IFREES,X(50),Y(50),Z(50),L(50),IH(5),
&                 M(5),METHOD,IHELP,IPREV,BOUND(6),SUBSET,INTPNT,
&                   IE(2),ID
      DIMENSION C(41)
      INTEGER SUBSET
C ADD DATA POINTS AND UPDATE LINK LIST
      IP = IFREES
      ID1=ID+1
      DO 3 I=1,41,ID1
C CHECK DATA POINTS TABLE INDEX
        IF(C(I).EQ.99) GOTO 2
        IF(C(I).GT.NPS)GOTO 3
        IF(C(I).EQ.0)GOTO 4
        IC=C(I)
C GET LINK LIST LOCATION OF THE DATA POINTS AND SET LINKS
        IS=INDEX(IC)
        L(IP)=L(IS)
        L(IS)=IP
        GOTO 5
4       L(IP)=IH(1)
        IH(1)=IP
C NOW ADD POINT TO FREE LOCATION
5       X(IP)=C(I+1)
        Y(IP)=C(I+2)
C 3-DIMENSIONAL CURVE
        IF(ID.EQ.3)Z(IP)=C(I+3)
        IP=IP+1
        NPS=NPS+1
3     CONTINUE
C SET FREE LINK LIST POINTER
2     IFREES=IP
      RETURN
      END
C
C***********EDIT - CORRECT***********X***************
C
      SUBROUTINE PCORCT(C)
C INPUT COMMON DATA AREA
      COMMON/DATSUP/NPS,NPI(50),IFREES,X(50),Y(50),Z(50),L(50),IH(5),
&                M(5),METHOD,IHELP,IPREV,BOUND(6),SUBSET,INTPNT,
&                IE(2),ID
      DIMENSION C(41)
      INTEGER SUBSET
      ID1=ID+1
      DO 2 I=1,41,ID1
C CHECKS ENTRY OF CORRECTION OF DATA POINTS
```

```
          IF(C(I).EQ.99)GOTO3
          IF(C(I).GT.NPS) GOTO 2
          IC=C(I)
C GET LINK LIST LOCATION AND REPLACE POINT
          K=INDEX(IC)
          X(K)=C(I+1)
          Y(K)=C(I+2)
C 3-D CURVE
          IF(ID.EQ.3)Z(K)=C(I+3)
2        CONTINUE
3        RETURN
         END
C
C***********EDIT-DELETE**************************
C
         SUBROUTINE PDELET(C)
C INPUT COMMON DATA AREA
         COMMON/DATSUP/NPS,NPI(50),IFREES,X(50),Y(50),Z(50),L(50),IH(5),
&                    M(5),METHOD,IHELP,IPREV,BOUND(2),SUBSET,INTPNT,
&                       IE(2),ID
         DIMENSION C(41),X0(50),Y0(50),Z0(50),L0(50)
         INTEGER SUBSET
         DO 1 I=1,41
C CHECKS DATA POINTS TABLE INDEX
          IF(C(I).EQ.99)GOTO 3
          IF(C(I).GT.NPS)GOTO 5
          IF(C(I).EQ.1) GOTO 4
          IC=C(I)-1
C GET LINK LIST LOCATION AND DELETE DATA POINTS
          IS=INDEX(IC)
          L(IS)=L(L(IS))
          GOTO 1
5         NPS=NPS+1
          GOTO 1
4         IH(1)=L(IH(1))
1        CONTINUE
3        NPS=NPS-I+1
         IFREES=IFREES-I+1
C GARBAGE COLLECTION
         IP=IH(1)
         DO 6 K=1,NPS
          X0(K)=X(IP)
          Y0(K)=Y(IP)
          IF(ID.EQ.3)Z0(K)=Z(IP)
          L0(K)=K+1
          IP=L(IP)
6        CONTINUE
         DO 7 J=1,NPS
          X(J)=X0(J)
          Y(J)=Y0(J)
          IF(ID.EQ.3)Z(J)=Z0(J)
          L(J)=L0(J)
7        CONTINUE
         L(NPS)=0
         IS=0
         DO 9 K=1,5
          IH(K)=IS*10+1
          IS=IS+1
9        CONTINUE
         RETURN
         END
C
```

```
C ******************KEYBOARD ENTRY FILE NAME******************
C
      SUBROUTINE GETFLN(NAME)
      INTEGER NAME(3)
C GET FILE NAME & SAVE IT IN AN INTEGER ARRAY
      CALL MESSAG("# DATA FILE NAME(MAX.10 CHARACTERS)?^")
      READ(5,30)NAME
30    FORMAT(2A4,A2)
      RETURN
      END
C
C **************FINDS DATA POINT LOCATION IN THE LINK LIST**********
C
      INTEGER FUNCTION INDEX(INX )
C INPUT COMMON DATA AREA
      COMMON/DATSUF/NPS,NPI(50),IFREES,X(50),Y(50),Z(50),L(50),IH(5),
&                 M(5),METHOD,IHELP,IPREV,BOUND(6),SUBSET,INTPNT,IE(2)
&                    ,ID
      INTEGER SUBSET
C FINDS WHICH PORTION OF THE LINK LIST?
      DO 1 I=1,5
        IF(M(I).GE.INX)GOTO 2
1     CONTINUE
2     IS=IH(I)
C COMPUTE LOCATION
      INX=INX-(I-1)*10
      INX1=INX-1
      IF(INX1.EQ.0) GOTO 55
      DO 3 I=1,INX1
3     IS=L(IS)
55    INDEX=IS
      RETURN
      END
C
C ***********DATA ENTRY FROM DISC FILE********************
C
      SUBROUTINE READAT(FLNAME,A,B,C,N,IF,IDIM)
      DIMENSION A(1),B(1),C(1)
      INTEGER FLNAME(3)
      REWIND 9
C OPEN INPUT FILE
      CALL SETFIL(9,FLNAME)
      IF(IERROR(103).NE.0)GOTO 99
C NUMBER OD DATA POINTS
      READ (9,20)N1
20    FORMAT(I3)
      IF(N.GT.N1)GOTO 100
C THREE - DIMENSIONS
      IF(IDIM.EQ.3) GOTO333
      READ(9,10)(A(I),B(I),I=1,N)
      ENDFILE 9
      RETURN
333   READ(9,10)(A(I),B(I),C(I),I=1,N)
      ENDFILE 9
      RETURN
10    FORMAT(F12.4)
99    ENDFILE 5
101   IF=1
      RETURN
100   IF=2
      RETURN
      END
```

```
C
C ************SAVE DATA POINTS ON DISC FILE**********************
C
      SUBROUTINE SAVE(A,B,C,N,IDIM)
      COMMON/IO/IN,IOUT
      INTEGER FILE(3)
      DIMENSION A(1),B(1),C(1)
C GET FILE NAME FROM THE USER THROUGH THE KEYBOAD
      WRITE(IOUT,10)
10    FORMAT(//////////61X,"$FILE NAME?")
      CALL MESSAG("
&         ^")
      READ(IN,20)FILE
20    FORMAT( 2A4,A2)
C OUTPUT DATA POINTS ON DISC FILE
      CALL WRTDAT(FILE,A,B,C,N,IDIM)
      RETURN
      END
C
C*********SORT IN X COORDINATE***********************
C
      SUBROUTINE SORTX(X1,Y1,Z1,N,IDIM)
      DIMENSION X1(1),Y1(1),Z1(1)
      N1=N-1
C PERFORM QUICK SORT
      DO 3 I=1,N1
         DO 2 J=I,N1
           IF(X1(I).LE.X1(J+1))GOTO 2
           A1=X1(I)
           B1=Y1(I)
           IF(IDIM.EQ.3)C1=Z1(I)
           X1(I)=X1(J+1)
           Y1(I)=Y1(J+1)
           IF(IDIM.EQ.3) Z1(I)=Z1(J+1)
           X1(J+1)=A1
           Y1(J+1)=B1
         IF(IDIM.EQ.3)Z1(J+1)=C1
2       CONTINUE
3     CONTINUE
      RETURN
      END
C
C************TABULATION OF DATA POINTS ROUTINE*************
C
      SUBROUTINE TABLE
C INPUT COMMON DATA AREA
      COMMON /DATSUP/NPS,NPI(50),IFREES,X(50),Y(50),Z(50),L(50),IH(5),
&                M(5),METHOD,IHELP,IPREV,BOUND(6),SUBSET,INTPNT,IE(2)
&                ,ID
      COMMON /IO/IN,IOUT
      INTEGER S1,S2,SUBSET
      CALL CURPOS(2.,710.)
C 3-DIMENSION
      IF(ID.EQ.3)GOTO 33
C OUTPUT TITLE COLUMN
      WRITE(IOUT,10)
10    FORMAT(" I",7X,"X(I)",8X ,"Y(I)",7X," I",5X,"X(I)",10X,"Y(I)"/)
      GOTO 34
33    WRITE(IOUT,40)
40    FORMAT(" I",3X,"X(1)",6X,"Y(I)",5X,"Z(I)",3X," I",3X,"X(I)",6X,
&          "Y(I)",5X,"Z(I)")
C GET STARTING POINTER
```

```
34      S2=IH(1)
        S1=L(S2)
        IF(S1.NE.0)GOTO12
C 3-D
        IF(ID.EQ.3) GOTO 44
        WRITE(IOUT,11) X(S2),Y(S2)
11      FORMAT(" 1",2X,2(E11.4,3X))
        RETURN
44      WRITE(IOUT,31)X(S2),Y(S2),Z(S2)
31      FORMAT(" 1",2(E9.2,X),E9.2))
        RETURN
C OUTPUT TABLE ITEMS FROM THE LINK LIST
12      DO 7 I=1,50,2
          J=I+1
          IF(ID.EQ.3) GOTO 55
          WRITE(IOUT,20) I,X(S2),Y(S2),J,X(S1),Y(S1)
20        FORMAT(I2,2X,2(E11.4,3X),I2,2X,2(E11.4,3X))
155       S2=L(S1)
          IF(S2.EQ.0)GOTO 15
          S1=L(S2)
          IF(S1.NE.0)GOTO 7
          I=I+2
          IF(ID.EQ.3) GOTO 333
          WRITE(IOUT,30) I,X(S2),Y(S2)
30        FORMAT(I2,2X,2(E11.4,3X))
          GOTO 15
7       CONTINUE
15      RETURN
55      WRITE(IOUT,120)I,X(S2),Y(S2),Z(S2),J,X(S1),Y(S1),Z(S1)
120     FORMAT(I2,2(E9.2,X),E9.2,I2,2(E9.2,X),E9.2)
        GOTO 155
333     WRITE(IOUT,303)I,X(S2),Y(S2),Z(S2)
303     FORMAT(I2,2(E9.2,X),E9.2)
        GOTO 15
        END
C
C ********************UPDATE LINK LIST AFTER EDITING********************
C
        SUBROUTINE UPDATE
C INPUT COMMON DATA AREA
        COMMON/DATSUP/NPS,NPI(50),IFREES,X(50),Y(50),Z(50),L(50),IH(5),
&                     M(5),METHOD,IHELP,IPREV,BOUND(6),SUBSET,INTPNT,
&                   IE(2),ID
        INTEGER SUBSET
        IR=IH(1)
        DO 1 J=2,5
           DO 2 K=1,10
              IR=L(IR)
                IF(IR.EQ.0)GOTO3
2            CONTINUE
          IH(J)=IR
1       CONTINUE
3       RETURN
        END
C
C ************** SAVE DATA ON DISC FILE ********************************
C
        SUBROUTINE WRTDAT(FLNAME,A,B,C,N,IDIM)
        DIMENSION A(1),B(1),C(1)
        INTEGER FLNAME(3)
        REWIND 9
C OPEN OUTPUT FILE
```

```
      CALL SETFIL(9,FLNAME)
      WRITE(9,20)N
C 3-DIMENSION
      IF(IDIM.EQ.3) GOTO 333
      WRITE(9,10)(A(I),B(I),I=1,N)
      ENDFILE 9
      RETURN
333   WRITE(9,10)(A(I),B(I),C(I),I=1,N)
      ENDFILE 9
      RETURN
10    FORMAT(F12.4)
20    FORMAT(I3)
      END
```

```
C                         *****************
C                         * APPENDIX 2.32 *
C                         *****************
C
C THIS MODULE HANDLES THE FOLLOWING INTERACTIVE DISPLAYS:-
C                1.POLYGONAL PLOT (2-D ONLY)
C                2.PARAMETER ENTRY
C
C
C
C ********* MAIN PROGRAM - MODULE 2 *************
C
C
C INPUT COMMON DATA AREA
        COMMON/DATSUP/NPS,NPI(50),IFREES,X(50),Y(50),Z(50),L(50),IH(5),
&                  M(5),METHOD,IHELP,IPREV,BOUND(6),SUBSET,INTPNT,IE(2)
&                       ,ID
        COMMON/IO/IN,IOUT
C OVERLAY EXECUTABLE PROGRAM NAMES
        DATA PMODL3,PMODL1,PHELP/"PMODL3","PARAMETRIC","HELP"/
        LOGICAL*1 PMODL3(10),PMODL1(20),PHELP(10)
        INTEGER SUBSET
C INPUT COMMON DATA
        CALL PRDCM1
        CALL TXOPEN
C TERMINAL INPUT/OUTPUT CHANNEL
        IN=5
        IOUT=6
        IF(IHELP.GT.1) GOTO 6
        IF(IPREV.GT.1) GOTO 5
        IF(IPREV.EQ.1.OR.ID.EQ.3) GOTO 4
C POLYGONAL DISPLAY
3       CALL POLYGL(X,Y,NPS,IC)
        GOTO(1,2,20,20,20,20,25,20,30),IC
C PREVIOUS DISPLAY
2       IPREV=1
        IHELP=0
        CALL PWRCM1
        CALL OVRLAY(PMODL1)
C PARAMETER DISPLAY
1       CALL PARMET(METHOD,SUBSET,INTPNT,BOUND,NPS,IE,ID,IC)
        CALL CURPOS(410.,780.)
        GOTO(10,31,25,1,30),IC
31      IF(ID.EQ.2) GOTO 3
        GOTO 2
C NEXT DISPLAY
10      IPREV=0
        IHELP=0
C SAVE INPUT COMMON DATA
        CALL PWRCM1
        CALL OVRLAY(PMODL3)
C PROGRAM TERMINATE
30      CALL PEXIT
20      STOP
4       IPREV=0
        IHELP=0
        GOTO 1
C HELP DISPLAY
25      CALL PWRCM1
        CALL OVRLAY(PHELP)
5       CALL ERRMES(IC)
```

```
      GOTO(1,20),IC
C RETURN FROM HELP DISPLAY
6     GOTO(3,1),IHELP
      END
C
C **************** POLYGONAL PLOT OF DATA POINTS*************
C
      SUBROUTINE POLYGL(A,B,N,IC)
C PLOT THE DATA POINTS SUPPLIED AND JOINED THEM WITH STRAIGHT LINES
      COMMON/IO/IN,IOUT
C MDNU ITEMS
      DATA MNTXT/"+ NEXT    + PREVIOUS+ GRAPH   + COORDS. + DISP.ORG+
&AX. MARK+ HELP    + RESTART + EXIT    "/
      LOGICAL*1 MNTXT(90)
      DIMENSION A(1),B(1),X0(50),Y0(50)
      EXTERNAL PPLOT
C SET UP DISPLAY
1     CALL TXCLER
      ICORD=0
      WRITE(IOUT,10)
10    FORMAT("POLYGONAL PLOT :-")
C OUTPUT MENU
      CALL MNOPEN(875.,715.,1)
      CALL MNDISP(MNTXT,9,10,1)
      CALL FRAME(870.,733.,9)
C REMOVE LINKS & FIND SCALE VALUES
      CALL PRMLNK(X0,Y0,Z0)
      CALL MINMAX(S1,S2,S3,S4,X0,Y0,N)
C SET UP CURSOR FOR USER MENU
2     CALL LMTARA
      CALL MNPICK(J,ICHAR,MNO)
      IF(J.EQ.4.AND.ICORD.EQ.2)GOTO2
7     CALL CONFRM(ICHAR)
      IF(ICHAR.EQ.78)GOTO 2
      IF(ICHAR.NE.89) GOTO 7
      GOTO(20,20,3,4,5,6,20,1,20),J
C RETURN TO CALLING PROGRAM
20    IC=J
      RETURN
C PLOT THE POLYGONAL OF THE DATA POINTS
3     CALL DRAW(PPLOT,R,S1,S2,S3,S4,N)
      GOTO 2
C CURSOR INPUT COORDINATES
 4    CALL DISCOR(ICORD,S1,S2,S3,S4)
      GOTO 2
C DISPLAY ORIGIN
 5    CALL DISORG(S1,S2,S3,S4)
      GOTO 2
C AXES MARKING
6     CALL AXSMRK(S1,S2,S3,S4)
      GOTO 2
      END
C
C *************** PARAMETER DISPLAY ENTRY ************************
C
      SUBROUTINE PARMET(M,SUBT,INTPNT,B,N,IE,IDIM,IC)
      COMMON/IO/IN,IOUT
C MENU ITEMS
      DATA MNTXT1/"+ NEXT    + PREVIOUS+ HELP    + RESTART + EXIT    "/
      DATA MNTXT2/"+ SPECIFY + DEFAULT + CURSOR  "/
      DATA MNTXT3/"         + CLAMPED + RELAXED + PARABOLA+ Q-SPLINE"/
      DATA MNTXT4/"         + X-BOND'Y+ Y-BOND'Y+ Z-BOND'Y"/
```

```
      LOGICAL*1 MNTXT1(50),MNTXT2(30),MNTXT3(50),MNTXT4(40)
      INTEGER SUBT
      DIMENSION B(1),IE(1)
1     CALL TXCLER
C DEFAULT VALUE SETTING
      INTPNT=0.
      DO 555 I=1,6
555   B(I)=0.
112   IE(1)=1.
      IE(2)=1.
C SET UP THE DISPLAY
1011  WRITE(IOUT,10)
10    FORMAT("PARAMETER ENTRY:-"/)
      GOTO (101,102,103,104),M
20    CALL CURPOS(1.,660.)
C OUTPUT INSTRUCTION FOR USER TO TAKE APPROPRIATE ACTIONS
      WRITE(IOUT,21)
21    FORMAT("SELECT THE APPROPRIATE OPTION FROM THE "/
&     "FOLLOWING PARAMETER SPECIFICATION(*) :-"///
&     6X,"1-CHOICE OF INTERMEDIATE"/
&     8X,"POINTS REQUIRED FOR SMOOTH DRAWING:-")
      IF(M.NE.4) GOTO 1221
      WRITE(IOUT,15)
15    FORMAT(//6X,"2-SELECT THE CONDITION")
1433  WRITE(IOUT,1333)
1333  FORMAT(7X," AT EACH END OF THE CURVE:-"/
&              8X,"N.B:- FIRST END , TYPE '1'"/
&              13X,"SECOND END , TYPE '2'")
      GOTO 143
1221  IF(M.EQ.2.OR.M.EQ.3) GOTO 142
      WRITE(IOUT,22)
22    FORMAT(/6X,"2-BOUNDARY CONDITION:-"//)
143   WRITE(IOUT,7)
7     FORMAT(/////////////"* IF NO APPROPRIATE PARAMETER IS SELECTED "
&     /"WHEN REQUIRED THEN  DEFAULT VALUES ARE ASSUMED")
C DISPLAY CONTROL COMMAND MENU
      CALL MNOPEN(875.,715.,1)
      CALL MNDISP(MNTXT1,5,10,1)
      CALL FRAME(870.,733.,5)
      CALL MNOPEN(670.,565.,2)
      CALL MNDISP(MNTXT2,3,10,2)
      CALL DTEXT(810.,540.,'*',1)
      IF(M.NE.4) GOTO 99
      CALL MNDISP(MNTXT3,5,10,2)
      CALL DTEXT(810.,455.,'*',1)
      MF=8
      GOTO 88
142   WRITE(IOUT,1420)
1420  FORMAT(///)
      GOTO 143
99    MF=3
      IF(M.EQ.2.OR.M.EQ.3) GOTO 88
      CALL MNDISP(MNTXT4,4,10,2)
      MF=7
88    CALL FRAME(670.,582.,MF)
      IF(M.EQ.2.OR.M.EQ.3) GOTO 2
      CALL TXMOVE(670.,512.)
      CALL TXDRAW(810.,512.)
C CURSOR PICKING
2     CALL MNPICK(J,ICHAR,MNO)
      IF(MNO.EQ.2) GOTO 130
5     CALL CONFRM(ICHAR)
```

```
        IF(ICHAR.EQ.78)GOTO 2
        IF(ICHAR.NE.89)GOTO 5
        IF(J.EQ.4) GOTO 1
707     IC=J
        RETURN
C PARAMETER SPECIFICATION CONTROL
130     GOTO(35,36,37,38,39,40,41,42),J
C USER SPECIFIED NUMBER OF INTERMEDIATE POINTS PER INTREVAL
35      CALL CURPOS(1.,360.)
66      CALL MESSAG("# NUMBER OF INTERPOLATED POINTS PER INTERVAL?^")
        IF(IERROR(110).NE.0) GOTO 171
        READ(IN,77) INTPNT
77      FORMAT(G0.0)
        ISUM=INTPNT*(N-1)+N
        IF(ISUM.GT.200) GOTO 187
        GOTO 2
171     ENDFILE 5
        GOTO 66
187     WRITE(IOUT,403)
403     FORMAT("TOTAL NUMBER OF POINT EXCEEDING LIMIT,TRY AGAIN")
        GOTO 66
C DEFAULT
36      INTPNT=0
        GOTO2
C       CURSOR
37      INTPNT=999
38      GOTO2
C BOUNDARY CONDITION
39      GOTO(139,2,2,391,2),M
        GOTO 2
391     IF(ICHAR.EQ.49.OR.ICHAR.EQ.50)GOTO 392
        GOTO2
392     ICHAR=ICHAR-48
        GOTO(272,202),ICHAR
        GOTO 2
272     IE(1)=1
2721    CALL CURPOS(1.,300.)
2120    IF(IDIM.EQ.3)CALL MESSAG("# SLOPE OF X,Y,Z W.R.T. T AS BOUNDARY
& VALUE(1ST END)?^")
        IF(IDIM.EQ.2)CALL MESSAG("# SLOPE OF X,Y W.R.T. T AS BOUNDARY
& VALUE(1ST END)?^")
        IF(IERROR(110).NE.0) GOTO 2120
        READ(IN,1110) B(1),B(2),B(3)
1110    FORMAT(3G0.0)
        GOTO 2
202     IE(2)=1
2021    CALL CURPOS(1.,270.)
2110    IF(IDIM.EQ.3)CALL MESSAG("# SLOP OF X,Y,Z W.R.T. T AS BOUNDARY
& VALUE(2ND END)?^")
        IF(IDIM.EQ.2)CALL MESSAG("# SLOP OF X,Y W.R.T T AS BOUNDARY
& VALUE(2ND END)?^")
        IF(IERROR(110).NE.0) GOTO 2110
        READ(IN,1110)B(4),B(5),B(6)
        GOTO 2
139     CALL CURPOS(1.,300.)
404     CALL MESSAG("# PARAMETRIC BOUNDARY VALUES AT BOTH END(D2X/DT2)?^")
        IF(IERROR(110).NE.0)GOTO 404
        READ(IN,11)B(1),B(4)
11      FORMAT(2G0.0)
        GOTO 2
40      GOTO(444,2,2,411,2),M
444     CALL CURPOS(1.,270.)
```

```
4444    CALL MESSAG("# PARAMETRIC BOUNDARY VALUES AT BOTH END(D2Y/DT2)?^")
        IF(IERROR(110).NE.0) GOTO 4444
        READ(IN,11)B(2),B(5)
        GOTO2
411     IF(ICHAR.EQ.49)IE(1)=J-4
        IF(ICHAR.EQ.50)IE(2)=J-4
        GOTO 2
41      GOTO(421,2,2,411,2),M
421     CALL CURPOS(1.,240.)
4141    IF(IDIM.EQ.2)GOTO 2
        CALL MESSAG("# PARAMETRIC BOUNDARY VALUES AT BOTH END(D2Z/DT2)?^")
        IF(IERROR(110).NE.0) GOTO 4141
        READ(IN,11)B(3),B(6)
        GOTO 2
42      GOTO(2,2,2,411,2),M
        GOTO 2
101     WRITE(IOUT,111)
111     FORMAT("*** STANDARD PARAMETRIC CUBIC SPLINE***")
        GOTO 20
102     WRITE(IOUT,122)
        GOTO 20
122     FORMAT("*** CYCLIC CUBIC SPLINE***")
103     WRITE(IOUT,133)
133     FORMAT("*** ANTICYCLIC CUBIC SPLINE ***")
        GOTO 20
104     WRITE(IOUT,144)
144     FORMAT("*** CUBIC SPLINE WITH VARIABLE END CONDITION ***")
        GOTO 20
        END
C
C **************ERROR MESSAGE DISPLAY****************************
C
        SUBROUTINE ERRMES(IC)
        COMMON/IO/IN,IOUT
C MENU ITEMS
        DATA MNTXT/"+ PREVIOUS+ HELP    + RESTART + EXIT    "/
        DATA DATSP/"DATSUPFL"/
        LOGICAL*1 MNTXT(40),DATSP(10)
1       CALL TXCLER
        WRITE(IOUT,20)
20      FORMAT("ERROR MESSAG:-"/"TOO MANY POINTS FOR JOINNING CURVES"/
&             "WHICH EXCEED CORE LIMIT"/
&             "YOU MAY PROCEED BY TAKING THE FOLLOWING ACTION:-"/
&             "EITHER 1- USE PREVIOUS COMMAND TO GO BACK TO PARAMETER"/
&             "DISPLAY,SO THAT TO ALTER NO.OF INTERMEDIATE POINTS."/
&             "OR     2- USE HELP COMMAND IN ORDER TO BRANCH TO ANY"/
&             "DISPLAY ,E.G DATA ENTRY OR DATA TABULATION DISPLAYS..ETC")
        CALL MNOPEN(875.,760.,1)
        CALL MNDISP(MNTXT,4,10,1)
        CALL FRAME(870.,778.,4)
120     CALL MNPICK(J,ICHAR,MNO)
110     IF(ICHAR.EQ.78) GOTO 110
        IF(ICHAR.NE.89)GOTO 120
        GOTO (40,40,1,50),J
C PREVIOUS DISPLAY
40      IC=J
        RETURN
50      CALL RMFILE(DATSP)
        STOP
        END
C
C*********PLOT THE POLYGONAL OF THE DATA POINTS************
```

```
C
      SUBROUTINE PPLOT(R,SCL1,SCL2,SCL3,SCL4,N)
C INPUT COMMON DATA POINTS
      COMMON/DATSUP/NPS,NPI(50),IFREES,X(50),Y(50),Z(50),L(50),IH(5),
&             M(5),METHOD,IHELP,IPREV,BOUND(2),SUBSET,INTPNT,IE(2)
&             ,ID
      DIMENSION X0(50),Y0(50)
C REMOVES LINKS
      CALL PRMLNK(X0,Y0)
      DO 1 I=1,NPS
        IF(I.EQ.1) GOTO 2
        CALL TXDRAW(X0(I),Y0(I))
2       CALL PLUSGN(SCL1,SCL2,SCL3,SCL4,X0(I),Y0(I))
1       CALL TXMOVE(X0(I),Y0(I))
      RETURN
      END
```

```
C                       ****************
C                       * APPENDIX 2.33 *
C                       ****************
C
C THIS MODULE HANDLES THE FOLLOWING INTERACTIVE DISPLAYS:-
C              1.CURVE FIT
C              2.CURVE ZOOM
C
C
C
C ******* MAIN PROGRAM - MODULE 3 ***************
C
C
C I/O COMMON DATA AREA
        COMMON/DATSUP/NPS,NPI(50),IFREES,X(50),Y(50),Z(50),L(50),IH(5),
&                  M(5),METHOD,IHELP,IPREV,BOUND(6),SUBSET,INTPNT,IE(2)
&              ,ID
        COMMON/CURVEFIT/XCOEF(50,4),YCOEF(50,4),ZCOEF(50,4),XCORD(200),
&                       YCORD(200),ZCORD(200),TCORD(200)
        COMMON/IO/IN,IOUT
C OVERLAY EXECUTABLE PROGRAM MODULES
        DATA PMODL1,PMODL2,PMODL5,PMODL6/"PARAMETRIC","PMODL2","PMODL5"
&,"PMODL6"/
        DATA PHELP/"HELP"/
        INTEGER SUBSET
        LOGICAL*1 PMODL1(20),PMODL2(10),PMODL5(10),PMODL6(10),PHELP(10)
        CALL TXOPEN
C READ I/O FILES
        CALL PRDCM1
11      NC=4
21      IN=5
        IOUT=6
        CALL PRDCM2(NPS,NPI,NC,ID)
C CURVE DESIGN DISPLAY ?
        IF(INTPNT.EQ.999) GOTO 111
C CURVE FIT DISPLAY
        CALL CRVFIT(NPS,NPI,XCORD,YCORD,ZCORD,ID,IC)
        GOTO(2,3,2,2,2,2,6,8,2,111,2,2,14,2,15,2,22),IC
C PROGRAM TERMINATE
22      CALL PEXIT
2       STOP
C PREVIOUS DISPLAY
3       IPREV=1
        IHELP=0
        CALL PWRCM1
        CALL OVRLAY(PMODL2)
C TABULATION OF INTERPLOATED DATA POINTS DISPLAY
6       IPREV=1
100     IHELP=0
        CALL PWRCM1
        CALL OVRLAY(PMODL6)
C SUPERIMPOSED CURVES DISPLAY
8       IPREV=2
        GOTO 100
C CURVE DESIGN DISPLAY
111     IPREV=0
        IHELP=0
        CALL PWRCM1
        CALL OVRLAY(PMODL5)
14      IPREV=0
        IHELP=0
```

```
        CALL PWRCM1
        CALL OVRLAY(PMODL1)
C HELP DISPLAY
15      CALL PWRCM1
        CALL OVRLAY(PHELP)
        END
C
C **************** CURVE FIT DISPLAY ************************
C
        SUBROUTINE CRVFIT(N,N1,XX,YY,ZZ,IDIM,IC)
        COMMON/CURVES/NCRV(10),XYSCL(4)
        COMMON/IO/IN,IOUT
C MENU ITEMS
        DATA MNTXT1/"+ NEXT    + PREVIOUS+ YX-GRAPH+ XZ-GRAPH+ ZY-GRAPH
1+ COORDS. + TABLES  + NGRAPH  + ZOOM    "/
        DATA MNTXT2/"+ CRV.DES.+ SAVE    + REDRAW  + METHOD  + AX. MARK
1+ HELP    + RESTART + EXIT    "/
        DIMENSION N1(1),XX(1),YY(1),ZZ(1)
        LOGICAL*1 MNTXT1(90),MNTXT2(80)
        INTEGER R
        EXTERNAL CPLOT
        IF(IERROR(103).NE.0) GOTO 99
        CALL RDCRVS
99      MSUM=0
        N2=N-1
C COMPUTE TOTAL NUMBER OF POINTS
        DO 6 I=1,N2
6         MSUM=MSUM+N1(I)
          MSUM=MSUM+N
1       CALL MINMAX(S1,S2,S4,S5,XX,YY,MSUM)
        IF(IDIM.EQ.3)CALL MINMAX(S1,S3,S4,S6,XX,ZZ,MSUM)
        R=0
 101    CALL TXCLER
        ICORD=0
        NG=1
C SET UP DISPLAY
        WRITE(IOUT,10)
10      FORMAT("CURVE FIT:-")
        CALL MNOPEN(875.,715.,1)
        CALL MNDISP(MNTXT1,9,10,1)
        CALL MNDISP(MNTXT2,8,10,1)
        CALL FRAME(870.,733.,17)
        IF(R.EQ.0) GOTO 2
        GOTO(31,331,3331),R
C SET UP CURSOR PICKING
2       CALL LMTARA
        CALL MNPICK(J,ICHAR,MNO)
        IF(J.EQ.6.AND.ICORD.EQ.2) GOTO 2
        IF(J.EQ.8.AND.NG.EQ.2) GOTO 30
17      CALL CONFRM(ICHAR)
        IF(ICHAR.EQ.78) GOTO 2
        IF(ICHAR.NE.89) GOTO 17
        GOTO(30,30,3,33,333,4,30,8,9,30,11,12,30,15,30,1,30),J
C NO CURSOR DESIGN FOR 3-D
        IF(J.EQ.10.AND.IDIM.EQ.3) GOTO 2
30      IC=J
        RETURN
C YX-GRAPH PLANE PROJECTION
3       IF(R.EQ.0) GOTO 31
        R=1
        GOTO 101
31      R=1
```

```
      SC1=S1
      SC2=S2
      SC3=S4
      SC4=S5
      GOTO 303
C XZ-GRAPH PLANE PROJECTION
33    IF(IDIM.EQ.2) GOTO 2
      IF(R.EQ.0) GOTO 331
      R=2
      GOTO 101
331   R=2
      SC1=S3
      SC2=S1
      SC3=S6
      SC4=S4
      GOTO 303
C ZY-GRAPH PLANE PROJECTION
333   IF(IDIM.EQ.2) GOTO 2
      IF(R.EQ.0) GOTO 3331
      R=3
      GOTO 101
3331  R=3
      SC1=S2
      SC2=S3
      SC3=S5
      SC4=S6
      GOTO 303
C DISPLAY ORIGIN
303   CALL DISORG(SC1,SC2,SC3,SC4)
      CALL DRAW(CPLOT,R,SC1,SC2,SC3,SC4,MSUM)
      GOTO 2
C CURSOR COORDINATES
4     IF(R.EQ.0) GOTO 2
      CALL DISCOR(ICORD,SC1,SC2,SC3,SC4)
      GOTO 2
C SAVE CURVE
11    IF(R.EQ.0) GOTO 2
      GOTO(51,52,53),R
51    CALL CRVSAV(XX,YY,N,N1,MSUM,SC1,SC2,SC3,SC4,J)
      GOTO 2
52    CALL CRVSAV(ZZ,XX,N,N1,MSUM,SC1,SC2,SC3,SC4,J)
      GOTO 2
53    CALL CRVSAV(YY,ZZ,N,N1,MSUM,SC1,SC2,SC3,SC4,J)
      GOTO 2
C REDRAW A CURVE
12    CALL REDRAW(XX,YY,N,N1,MSUM,SC1,SC2,SC3,SC4)
      GOTO 2
C AXES MARKING
15    IF(IDIM.EQ.2.AND.R.EQ.0)CALL AXSMRK(S1,S2,S4,S5)
      IF(R.EQ.0) GOTO 2
      CALL AXSMRK(SC1,SC2,SC3,SC4)
      GOTO 2
C NGRAPH COMMAND FOR THE FIRST TIME (I.E SAVE CURVE FOR LATER USE)
8     IF(IDIM.EQ.2.AND.R.EQ.0)CALL NGRAPH(N,N1,MSUM,XX,YY,S1,S2,S4,S5,J)
      IF(R.EQ.0) GOTO 808
      GOTO(81,82,83),R
81    CALL NGRAPH(N,N1,MSUM,XX,YY,SC1,SC2,SC3,SC4,J)
      GOTO 808
82    CALL NGRAPH(N,N1,MSUM,ZZ,XX,SC1,SC2,SC3,SC4,J)
      GOTO 808
83    CALL NGRAPH(N,N1,MSUM,YY,ZZ,SC1,SC2,SC3,SC4,J)
      GOTO 808
```

```
108      CALL WRCRVS
         NG=NG+1
         GOTO 2
C ZOOMING
9        CALL LMTSCL(SC1,SC2,SC3,SC4)
         CALL TXCURS(XZ1,YZ1,ICHAR)
19       CALL TXCURS(XZ2,YZ2,ICHAR)
         IF(XZ1.EQ.XZ2.OR.YZ1.EQ.YZ2) GOTO 19
         CALL TXMOVE(XZ1,YZ1)
         CALL TXDRAW(XZ2,YZ1)
         CALL TXDRAW(XZ2,YZ2)
         CALL TXDRAW(XZ1,YZ2)
         CALL TXDRAW(XZ1,YZ1)
         CALL LMTARA
119      CALL CONFRM(ICHAR)
         IF(ICHAR.EQ.78) GOTO 9
         IF(ICHAR.NE.89) GOTO 119
C CALL ZOOM ROUTINE
         XMIN=AMIN1(XZ1,XZ2)
         XMAX=AMAX1(XZ1,XZ2)
         YMIN=AMIN1(YZ1,YZ2)
         YMAX=AMAX1(YZ1,YZ2)
555      CALL PZOOM(R,MSUM,XMIN,YMIN,XMAX,YMAX,IC)
         GOTO(1,30,109,109,109,109,109),IC
109      IF(IC.EQ.3) IC=13
         IF(IC.EQ.4.OR.IC.EQ.6)STOP
         IF(IC.EQ.5)IC=15
         IF(IC.EQ.7)IC=17
         RETURN
         END
C
C************ZOOMING INTO PARAMETRIC CURVES****************
C
         SUBROUTINE PZOOM(IR,NPNT,XMIN,YMIN,XMAX,YMAX,IC)
C OUTPUT COMMON DATA AREA
         COMMON/CURVEFIT/XCOEF(50,4),YCOEF(50,4),ZCOEF(50,4),XCORD(200)
&                 ,YCORD(200),ZCORD(200),TCORD(200)
         COMMON/IO/IN,IOUT
C MENU ITEMS
         DATA MNTXT/"+ NEXT    + PREVIOUS+ METHOD  + AX.MARK + HELP
&+ RESTART + EXIT    "/
         DIMENSION ZLINE(2,2)
         LOGICAL*1 MNTXT(70)
C SET UP THE DISPLAY
1        CALL TXCLER
         CALL ALPHMD
         WRITE(IOUT,10)
10       FORMAT("ZOOMING:-")
         CALL MNOPEN(875.,715.,1)
         CALL MNDISP(MNTXT,7,10,1)
         CALL FRAME(870.,733.,7)
C SET UP WINDOW & VIEWPORT
         CALL LMTSCL(XMIN,YMIN,XMAX,YMAX)
         CALL PFRAME(XMIN,YMIN,XMAX,YMAX)
C DRAW THE ZOOMED CURVE
         I=1
15       GOTO(20,30,40),IR
20       ZLINE(1,1)=XCORD(I)
         ZLINE(1,2)=YCORD(I)
         ZLINE(2,1)=XCORD(I+1)
         ZLINE(2,2)=YCORD(I+1)
         GOTO 25
```

```
30      ZLINE(1,1)=ZCORD(I)
        ZLINE(1,2)=XCORD(I)
        ZLINE(2,1)=ZCORD(I+1)
        ZLINE(2,2)=XCORD(I+1)
        GOTO 25
40      ZLINE(1,1)=YCORD(I)
        ZLINE(1,2)=ZCORD(I)
        ZLINE(2,1)=YCORD(I+1)
        ZLINE(2,2)=ZCORD(I+1)
C CLIP THE PORTION OF THE CURVE
25      CALL CLIP(ZLINE,XMIN,YMIN,XMAX,YMAX,IREJ)
        IF(IREJ.EQ.0) GOTO 35
        CALL TXMOVE(ZLINE(1,1),ZLINE(1,2))
        CALL TXDRAW(ZLINE(2,1),ZLINE(2,2))
35      I=I+1
        IF(I.EQ.NPNT) GOTO 202
        GOTO 15
C SET UP CURSOR
202     CALL LMTARA
        CALL MNPICK(J,ICHAR,MNO)
17      CALL CONFRM(ICHAR)
        IF(ICHAR.EQ.78) GOTO 202
        IF(ICHAR.NE.89) GOTO 17
        GOTO(414,414,414,404,414,1,414),J
414     IC=J
        RETURN
C AXES MARKING
404     CALL AXSMRK(XMIN,YMIN,XMAX,YMAX)
        GOTO 202
        END
C
C ************** PLOTTING CURVE FIT ***************************
C
        FUNCTION CPLOT(R,X0,Y0,X1,Y1,N)
C I/O COMMON DATA AREA
        COMMON/DATSUP/NPS,NPI(50),IFREES,X(50),Y(50),Z(50),L(50),IH(5)
&               ,M(5),METHOD,IHELP,IPREV,BOUND(6),SUBSET,INTPNT,IE(2)
&               ,ID
        COMMON/CURVEFIT/XCOEF(50,4),YCOEF(50,4),ZCOEF(50,4),XCORD(200),
&                YCORD(200),ZCORD(200),TCORD(200)
        INTEGER SUBSET,R
        IP=1
        IF=1
        I=1
3       GOTO(10,20,30),R
C PLOT SUPPLIED POINTS
10      CALL PLUSGN(X0,Y0,X1,Y1,XCORD(I),YCORD(I))
        CALL TXMOVE(XCORD(I),YCORD(I))
        GOTO 6
20      CALL PLUSGN(X0,Y0,X1,Y1,ZCORD(I),XCORD(I))
        CALL TXMOVE(ZCORD(I),XCORD(I))
        GOTO 6
30      CALL PLUSGN(X0,Y0,X1,Y1,YCORD(I),ZCORD(I))
        CALL TXMOVE(YCORD(I),ZCORD(I))
6       IF(I.EQ.N)RETURN
        IP1=IP+1
        IP2=IP+NPI(IF)+1
        DO 1 J=IP1,IP2
C DRAW INTERPOLATED LINE SEGMENTS ACCORDING TO PROJECTION PLANE
          GOTO(40,50,60),R
40        CALL TXDRAW(XCORD(J),YCORD(J))
          GOTO 1
```

```
50          CALL TXDRAW(ZCORD(J),XCORD(J))
            GOTO 1
60          CALL TXDRAW(YCORD(J),ZCORD(J))
1         CONTINUE
          IP=IP2
          IF=IF+1
          I=I+NPI(IF-1)+1
          GOTO 3
          END
C
C***********SAVE THE CURVE ON FILE FOR LATER USE*****************
C
          SUBROUTINE CRVSAV(XX,YY,N,N1,MSUM,SCL1,SCL2,SCL3,SCL4,J)
          DIMENSION XX(1),YY(1),N1(1)
          INTEGER FILE(3)
          N2=N-1
          IF(J.EQ.8) GOTO 88
C GET FILE NAME
          CALL CURPOS(10.,730.)
          CALL MESSAG("FILE NAME?^")
          CALL GETFLN(FILE )
          REWIND 9
C OPEN OUTPUT FILE
          CALL SETFIL(9,FILE)
88        WRITE(9,20)MSUM,N
          WRITE(9,30)(XX(I),YY(I),I=1,MSUM)
          WRITE(9,25)(N1(I),I=1,N2)
          WRITE(9,35)SCL1,SCL2,SCL3,SCL4
25        FORMAT(I3)
20        FORMAT(2I3)
30        FORMAT(F11.4)
35        FORMAT(4F11.4)
          ENDFILE 9
          RETURN
          END
C
C**************INPUT FILE NAME***********************
C
          SUBROUTINE GETFLN(NAME)
          COMMON/IO/IN,IOUT
          INTEGER NAME(3)
          READ(IN,10)NAME
10        FORMAT(2A4,A2)
          RETURN
          END
C
C************SUOERIMPOSE OPTION 'NGRAPH'*****************
C
          SUBROUTINE NGRAPH(N,N1,MSUM,XX,YY,SCL1,SCL2,SCL3,SCL4,J)
          COMMON/CURVES/NCRV(10),XYSCL(4)
C DATA FILE NAMES FOR SUPERIMPOSED CURVES
          DATA SUPFLS/"CURVE1    CURVE2    CURVE3    CURVE4    CURVE5
&CURVE6    CURVE7    CURVE8    CURVE9    CURVE10   "/
          LOGICAL*1 SUPFLS(100)
          K=0
C FIND FREE ENTRY
          DO 1 I=1,10
            IF(NCRV(I).NE.99) GOTO 2
            K=K+9
1         CONTINUE
2         NCRV(I)=99
          K=K+I
```

```
      REWIND 9
C OPEN OUTPUT FILE FOR SAVING CURVES
      CALL SETFIL(9,SUPFLS(K))
      CALL CRVSAV(XX,YY,N,N1,MSUM,SCL1,SCL2,SCL3,SCL4,J)
C FIRST CURVE
      IF(NCRV(1).EQ.99) GOTO 3
      XYSCL(1)=SCL1
      XYSCL(2)=SCL2
      XYSCL(3)=SCL3
      XYSCL(4)=SCL4
      GOTO 4
C SUBSEQUENT CURVES
C SET THE DISPLAY SCALES
3     IF(XYSCL(1).GT.SCL1)XYSCL(1)=SCL1
      IF(XYSCL(2).GT.SCL2)XYSCL(2)=SCL2
      IF(XYSCL(3).LT.SCL3)XYSCL(3)=SCL3
      IF(XYSCL(4).LT.SCL4)XYSCL(4)=SCL4
4     CALL DTEXT(725.,555.,"+ GRAPH",7)
      ENDFILE 9
      RETURN
      END
C
C **********SAVE THE DATA ON A FILE**************
C
      SUBROUTINE RDCRVS
      COMMON/CURVES/NCRV(10),XYSCL(4)
      REWIND 7
      CALL SETFIL(7,"SUFCRVES")
      READ(7,10) (NCRV(I),I=1,10)
      READ(7,20) (XYSCL(I),I=1,4)
10    FORMAT(I2)
20    FORMAT(F11.4)
      RETURN
      END
C
C ***********PLOT CURVE FROM NAMED FILE*******************
C
      SUBROUTINE REDRAW(XX,YY,N,N1,MSUM,SCL1,SCL2,SCL3,SCL4)
      COMMON/IO/IN,IOUT
      DIMENSION XX(1),YY(1),N1(1)
      INTEGER FILE(3),R
C INPUT FILE NAME
      CALL MESSAG("                                        FILE NAME?^")
      CALL GETFLN(FILE)
      REWIND 9
C OPEN INPUT FILE
      CALL SETFIL(9,FILE)
      READ(9,25)MSUM,N
      N2=N-1
      READ(9,30) (XX(I),YY(I),I=1,MSUM)
      READ(9,20) (N1(I),I=1,N2)
      READ(9,35)SCL1,SCL2,SCL3,SCL4
20    FORMAT(I3)
25    FORMAT(2I3)
30    FORMAT(F11.4)
35    FORMAT(4F11.4)
C SET UP WINDOW SCALE
      CALL LMTSCL(SCL1,SCL2,SCL3,SCL4)
      R=1
C PLOT THE CURVE
      CALL CPLOT(R,SCL1,SCL2,SCL3,SCL4,MSUM)
      ENDFILE 9
```

```
      RETURN
      END
C
C ****** SAVE SUPERIMPOSE COMMON DATA AREA ***************
C
      SUBROUTINE WRCRVS
      COMMON/CURVES/NCRV(10),XYSCL(4)
      REWIND 7
C OPEN OUTPUT FILE
      CALL SETFIL(7,"SUPCRVES")
      WRITE(7,10)(NCRV(I),I=1,10)
      WRITE(7,20)(XYSCL(I),I=1,4)
10    FORMAT(I2)
20    FORMAT(F11.4)
      ENDFILE 7
      RETURN
      END
```

```
C                         ****************
C                         * APPENDIX 2.34 *
C                         ****************
C
C THIS MODULE HANDLES THE CURVE DESIGN INTERACTIVE DISPLAY.
C
C
C
C ******** PROGRAM MODULE - MODULE 4 **************
C
C
C I/O COMMON DATA AREA
      COMMON/DATSUP/NPS,NPI(50),IFREES,X(50),Y(50),Z(50),L(50),IH(5)
&               ,M(5),METHOD,IHELP,IPREV,BOUND(6),SUBSET,INTPNT,IE(2)
&             ,ID
      COMMON/CURVEFIT/XCOEF(50,4),YCOEF(50,4),ZCOEF(50,4)
&                   ,XCORD(200),YCORD(200),ZCORD(200),TCORD(200)
      COMMON/IO/IN,IOUT
C OVERLAY EXECUTABLE PROGRAM NAME
      DATA PMODL2,PMODL4,PHELP/"PMODL2","PMODL4","HELP"/
      DIMENSION XO(50),YO(50),ZO(50),TO(50)
      LOGICAL*1 PMODL2(10),PMODL4(10),PHELP(10)
      INTEGER SUBSET
      CALL TXOPEN
C READ I/O FILES FOR THE COMMON DATA AREA
11    CALL PRDCM1
3     NC=4
21    IN=5
      IOUT=6
      CALL PRDCM2(NPS,NPI,NC,ID)
      CALL PRMLNK(XO,YO,ZO)
C THE CURVE IS DISPLAYED WITH DEFAULT SETTING
      INTPNT=0
      CALL GENPRT(NPS,XO,YO,ZO,TO,ID,METHOD)
      CALL SETLNK(NPS,NPI,INTPNT)
C DESIGN CURVE WITH CURSOR
      CALL CURDES(NPS,NPI,XO,YO,TO,XCORD,YCORD,TCORD,INTPNT,METHOD
&              ,XCOEF,YCOEF,IC)
      GOTO(4,5,6,6,6,6,6,6,6,9,11,8),IC
6     STOP
C NEXT DISPLAY
4     IPREV=0
      IHELP=0
      INTPNT=0
      CALL PWRCM1
      CALL SWRCOM(NPS,NPI,NC,XO,YO)
      CALL OVRLAY(PMODL4)
C PREVIOUS DISPLAY
5     IPREV=1
      IHELP=0
      CALL PWRCM1
      CALL OVRLAY(PMODL2)
C PROGRAM TERMINATION
8     CALL PEXIT
      STOP
C HELP DISPLAY
9     CALL PWRCM1
      CALL SWRCOM(NPS,NPI,NC,XO,YO)
      CALL OVRLAY(PHELP)
      END
C
```

```
C**********CURSOR DESIGN DISPLAY*******************
C
        SUBROUTINE CURDES(N,N1,X0,Y0,T0,XX,YY,TT,INTPNT,IA,XC,YC,IC)
C LINK LIST
        COMMON/LNKLST/LINK(200),INTVAL(50),IFREE,IFCNT
        COMMON/IO/IN,IOUT
C MENU ITEMS
        DATA MNTXT/"+ NEXT    + PREVIOUS+ DISP.ORG+ GRAPH   + COORDS.
&+ ADD PNT*+ DEL INTV+ REFRESH + AX. MARK+ HELP    + RESTART + EXIT    "/
        DIMENSION N1(1),X0(1),Y0(1),T0(1),XX(1),YY(1),TT(1),XC(50,4),
&                YC(50,4)
        LOGICAL*1 MNTXT(120)
        EXTERNAL XPLOT
        CALL SUM(N,N1,MSUM)
        CALL MINMAX(S1,S2,S3,S4,XX,YY,MSUM)
        S1=S1-(S3-S1)/20
        S2=S2-(S4-S2)/20
        IZERO=1
11      IF(IZERO.EQ.0)INTPNT=999
        INTVNO=1
C SET UP THE DISPLAY
        CALL TXCLER
        ICORD=0
        WRITE(IOUT,10)
10      FORMAT("CURVE DESIGN:-")
C OUTPUT MENU ITEMS
        CALL MNOPEN(875.,715.,1)
        CALL MNDISP(MNTXT,12,10,1)
        CALL FRAME(870.,733.,12)
C USER KEYBOARD COMMAND
        CALL DTEXT(828.,447.,"* TYPE CAPT.:",13)
        CALL DTEXT(828.,428.,"E-JOIN INTV.",12)
        CALL DTEXT(828.,406.,"F-FINISH INTV",13)
        CALL DTEXT(828.,384.,"N-NEXT INTV.",12)
        CALL DTEXT(20.,30.,"$ USE INTERVAL SEQUENCE MARKED BY '$'",37)
2       CALL LMTARA
C CHECK FOR REFRESH COMMAND
        IF(J.EQ.8) GOTO 34
C CURSOR MENU PICKING
        CALL MNPICK(J,ICHAR,MNO)
        IF(J.EQ.5.AND.ICORD.EQ.2)GOTO 2
17      CALL CONFRM(ICHAR)
        IF(ICHAR.EQ.78)GOTO 12
        IF(ICHAR.NE.89) GOTO 17
        GOTO(41,41,33,34,35,36,36,11,110,41,66,41),J
12       J=0
        GOTO 2
41      IC=J
        RETURN
C DISPLAY ORIGIN
33      CALL DISORG(S1,S2,S3,S4)
        GOTO 2
C PLOT THE GRAPH
34      CALL DRAW(XPLOT,R,S1,S2,S3,S4,MSUM)
        J=0
        GOTO 2
C AXES MARKING
110     CALL AXSMRK(S1,S2,S3,S4)
        GOTO2
C CURSOR INPUT COORDINATES
35      CALL DISCOR(ICORD,S1,S2,S3,S4)
        GOTO 2
```

```
C ADD INTERMEDIATE POINTS INTERACTIVELLY USING THE CURSOR
36      CALL LMTSCL(S1,S2,S3,S4)
22      CALL TXCURS(CX,CY,ICHAR)
        IF(ICHAR.EQ.70) GOTO 2
C MOVE TO NEXT INTERVAL
        IF(ICHAR.EQ.78) GOTO 88
        IF(J.EQ.7) GOTO 37
        IF(INTPNT.EQ.999.AND.IZERO.EQ.1)GOTO 66
61      INTPNT=0
        CALL ADDPNT(N,N1,XO,YO,TO,XX,YY,TT,INTVNO,IA,CX,CY,ICHAR,XC,YC)
        GOTO 22
66      N2=N-1
        DO 16 I=1,N2
          INTVAL(I)=0
16      N1(I)=0
        IF(J.EQ.11) GOTO 41
        IZERO=0
        GOTO 61
C MARK DELETED INTERVAL
37      CALL DTEXT(CX,CY,"D",1)
        CALL DELINT(INTVNO,N,N1)
        GOTO 22
88      INTVNO=INTVNO+1
        CALL DTEXT(XO(INTVNO),YO(INTVNO),"$",1)
        GOTO 22
        END
C
C**********GENERATE PARAMETERS T*****************
C
        SUBROUTINE GENPRT(N,X1,Y1,Z1,T1,IDIM,M)
        DIMENSION X1(1),Y1(1),Z1(1),T1(1)
        T1(1)=0.
C GENERATE T PARAMETER FOR COMPUTING THE INTERPOLATED POINTS
        DO 1 K=2,N
          U=X1(K)-X1(K-1)
          V=Y1(K)-Y1(K-1)
          IF(IDIM.EQ.3) Q=Z1(K)-Z1(K-1)
          D=U*U+V*V
          IF(IDIM.EQ.3)D=D+Q*Q
          D1=SQRT(D)
          IF(M.EQ.3.OR.M.EQ.4) GOTO 2
          T1(K)=T1(K-1)+D1
1       CONTINUE
        RETURN
2       T1(K)=D1
        GOTO 1
        END
C
C *************ADD INTERMEDITE POINT ***********************************
C
        SUBROUTINE ADDPNT(N,N1,XO,YO,TO,XX,YY,TT,INTVNO,IA,CX,CY,ICHAR,
&                              XC,YC)
C COMMON DATA LINK LIST
        COMMON/LNKLST/LINK(200),INTVAL(50),IFREE,IFCNT
        DIMENSION N1(1),XO(1),YO(1),TO(1),XX(1),YY(1),TT(1),XC(50,4)
&             ,YC(50,4)
C CHECK FIRST TIME IN THE INTERVAL
        IF(INTVAL(INTVNO).NE.0) GOTO 2
C ENTER THE START POINT AND THE INTERMEDIATE POINT
        INTVAL(INTVNO)=IFREE
        XX(IFREE)=XO(INTVNO)
        YY(IFREE)=YO(INTVNO)
```

```
C CHECK END OF INTERVAL
        IF(ICHAR.EQ.69) GOTO 7
        IF(IFCNT.GT.1) GOTO 11
        LINK(IFREE)=IFREE+1
        CALL TXMOVE(XX(IFREE),YY(IFREE))
        IFREE=LINK(IFREE)
5       LINK(IFREE)=0
C COMPUTE THE INTERPLOATED POINTS FROM THE CURSOR POSITONING
6       CALL CXXYY(N,XO,YO,TO,XX,YY,TT,INTVNO,IA,CX,CY,XC,YC,IFREE,IERC)
        IF(IERC.EQ.1) GOTO 66
        N1(INTVNO)=N1(INTVNO)+1
        IFREE=IFREE+1
        RETURN
C COMPARE THE CURRENT CX WITH THE ALERADY IN THE TABLE
2       IF(ICHAR.EQ.69) GOTO 1
        NEXT=INTVAL(INTVNO)
C CHECK FOR RUNING BACK CURVE
        DXX=XX(NEXT)-XX(LINK(NEXT))
        IF(DXX.GT.0.) GOTO 888
        DXO=XO(INTVNO)-XO(INTVNO+1)
        IF(DXO.GE.0.) GOTO 818
4       DCX=CX-XX(NEXT)
        IF(DCX.LT.0.) GOTO 3
        K=NEXT
        NEXT=LINK(NEXT)
        IF(NEXT.EQ.0) GOTO 41
        GOTO 4
41      LINK(K)=IFREE
        CALL TXMOVE(XX(K),YY(K))
        IF(IFCNT.GT.0) GOTO 12
        GOTO 5
C ADD POINT IN THE INTERVAL WHICH HAS GREATER VALUE POINT
3       IF(IFCNT.GT.0)GOTO 16
        LINK(K)=IFREE
        LINK(IFREE)=NEXT
        CALL TXMOVE(XX(K),YY(K))
        GOTO 6
C ADDITION INTO PREVIOUSLY DELETED ITEM (CARBIGE COLLECTION)
11      IFCNT=IFCNT-1
        CALL TXMOVE(XX(IFREE),YY(IFREE))
        K=LINK(LINK(IFREE))
        LINK(LINK(IFREE))=0
        IFREE=LINK(IFREE)
        CALL CXXYY(N,XO,YO,TO,XX,YY,TT,INTVNO,IA,CX,CY,XC,YC,IFREE,IERC)
        IF(IERC.EQ.1) GOTO 66
        IFREE=K
14      IFCNT=IFCNT-1
        N1(INTVNO)=N1(INTVNO)+1
        RETURN
12      CALL CXXYY(N,XO,YO,TO,XX,YY,TT,INTVNO,IA,CX,CY,XC,YC,IFREE,IERC)
        IF(IERC.EQ.1) GOTO 66
        K=IFREE
        IFREE=LINK(IFREE)
        LINK(K)=0
        GOTO 14
16      CALL TXMOVE(XX(K),YY(K))
        CALL CXXYY(N,XO,YO,TO,XX,YY,TT,INTVNO,IA,CX,CY,XC,YC,IFREE,IERC)
        IF(IERC.EQ.1) GOTO 66
        K1=LINK(IFREE)
        LINK(K)=IFREE
        LINK(IFREE)=NEXT
        IFREE=K1
```

```
      GOTO 14
C END OF INTERVAL WITH ONE OR MANY ENTERIES
C FIND THE LAST ENTRY IN THIS INTERVAL(LINK=0)
1     NEXT=INTVAL(INTVNO)
      J=N1(INTVNO)+1
      DO 9 L=1,J
      IP=NEXT
      NEXT=LINK(NEXT)
      IF(NEXT.EQ.0) GOTO10
9     CONTINUE
10    CALL TXMOVE(XX(IP),YY(IP))
8     CALL TXDRAW(XO(INTVNO+1),YO(INTVNO+1))
      RETURN
C NO INTERMEDITE POINTS
7     LINK(IFREE)=0
      IFREE=IFREE+1
      CALL TXMOVE(XO(INTVNO),YO(INTVNO))
      GOTO 8
66    IERC=0
      RETURN
C RUNING BACK CURVE ....CHECK FOR POINT FALL IN BETWEEN TWO
C EXISTING POINTS OR LOOP ON IT SELF
888   DCX=CX-XX(NEXT)
      IF(DCX.GT.O.) GOTO 808
818   K=NEXT
      NEXT=LINK(NEXT)
      IF(NEXT.EQ.O.AND.IFLAG.EQ.1) GOTO 333
      IF(NEXT.EQ.O) GOTO 41
      GOTO 888
333   IFLAG=0
      GOTO 3
C CHECK IF POINT STILL FALL IN OPPOSITE THE SUB INTERVAL
808   IF(IFLAG.EQ.1) GOTO 818
828   DXO=XO(INTVNO+1)-XO(INTVNO)
      IF(DXO.LE.O.) GOTO 3
      DY1=ABS(YO(INTVNO)-CY)
      DY2=ABS(YO(INTVNO+1)-CY)
      DYY=DY1-DY2
      IF(DYY.LT.0) GOTO 3
      IF(DYY.EQ.O.) RETURN
C NOT THE FIRST FOUND SUB INTVAL
      IFLAG=1
      GOTO 818
      END
C
C***********COMPUTE INTERPOLATED POINT***************
C
      SUBROUTINE CXXYY(N,XO,YO,TO,XX,YY,TT,INTVNO,IA,CX,CY,XC,YC,IFREE,IERC)
      DIMENSION XO(1),YO(1),TO(1),XX(1),YY(1),TT(1),XC(50,4),YC(50,4)
C DETERMINE PARAMETER T FROM CX,CY
      I=INTVNO
      DX=CX-XO(I)
      DY=CY-YO(I)
      DD=DX*DX+DY*DY
      T1=SQRT(DD)
      IF(IA.GT.2) GOTO 1
      T=TO(I)+T1
1     T=T1
      IF(T.GT.TO(I+1)) GOTO 3
      XX(IFREE)=XC(I,1)+T*(XC(I,2)+T*(XC(I,3)+T*XC(I,4)))
      YY(IFREE)=YC(I,1)+T*(YC(I,2)+T*(YC(I,3)+T*YC(I,4)))
      TT(IFREE)=T
```

```
2       CALL TXDRAW(XX(IFREE),YY(IFREE))
        RETURN
3       IERC=1
        RETURN
        END
C
C ****************DELETE AN INTERVAL *************:*************
C
        SUBROUTINE DELINT(INTVNO,N,N1)
        COMMON/LNKLST/LINK(200),INTVAL(50),IFREE,IFCNT
        DIMENSION N1(1)
C       EMPTY LIST?
        IF(INTVAL(INTVNO).EQ.0)RETURN
3       IF(IFCNT.NE.0)GOTO 1
C FIRST DELETION
        NEXT=INTVAL(INTVNO)
        K=NEXT
        IF(LINK(NEXT).EQ.0)GOTO 5
        N2=N1(INTVNO)
        K=IZRLNK(N2,LINK,NEXT)
5       LINK(K)=IFREE
        IFREE=INTVAL(INTVNO)
2       IFCNT=IFCNT+N1(INTVNO)+1
        INTVAL(INTVNO)=0
        N1(INTVNO)=0
        RETURN
C DELETE OF MORE INTREVAL
1       NEXT1=INTVAL(INTVNO)
        K1=NEXT1
        IF(LINK(NEXT1).EQ.0) GOTO 6
        N2=N1(INTVNO)
        K1=IZRLNK(N2,LINK,K1)
6       NEXT2=IFREE
        K2=NEXT2
        N2=IFCNT-1
        IF(N2.EQ.0) GOTO7
        K2=IZRLNK(N2,LINK,K2)
7       K3=LINK(K2)
        LINK(K2)=NEXT1
        LINK(K1)=K3
        GOTO 2
        END
C
C ***********FIND END OF THE INTREVAL  FOR DELETION******************
C
        FUNCTION IZRLNK(N,LK,NXT)
        DIMENSION LK(1)
        DO 1 I=1,N
1       NXT=LK(NXT)
        IZRLNK=NXT
        RETURN
        END
C
C***************SET DATA STRUCTURE LINKS****************
C
        SUBROUTINE SETLNK(N,N1,INTPNT)
        COMMON/LNKLST/LINK(200),INTVAL(50),IFREE,IFCNT
        DIMENSION N1(1)
        IF(INTPNT.EQ.999)GOTO 9
        INTVAL(1)=1
        N2=N-1
        DO 1 I=2,N
```

```
1         INTVAL(I)=INTVAL(I-1)+N1(I-1)+1
          K=1
          DO 2 I=1,N2
            IF(N1(I).NE.0) GOTO 4
            LINK(K)=0
            K=K+1
            GOTO 2
4           N11=N1(I)+K-1
C SET LINK LIST
            DO 3 J=K,N11
3             LINK(J)=J+1
            LINK(J)=0
            K=J+1
2         CONTINUE
            CALL SUM(N,N1,MSUM)
            IFREE=MSUM+1
            GOTO 10
9           IFREE=1
            IFCNT=0
10        RETURN
          END
C
C********OUTPUT THE DESIGNED CURVE****************
C
C                       *****A SIMPLE LIST******
          SUBROUTINE SWRCOM(N,N1,NC,X0,Y0)
C OUTPUT COMMON DATA AREA
          COMMON/CURVEFIT/XCOEF(50,4),YCOEF(50,4),ZCOEF(50,4)
&                         ,XCORD(200),YCORD(200),ZCORD(200),TCORD(200)
C LINK LIST USED IN THE DESIGNED PROCESS
          COMMON/LNKLST/LINK(200),INTVAL(50)
          DIMENSION N1(1),X0(1),Y0(1)
          REWIND 8
C OPEN OUTPUT FILE
          CALL SETFIL(8,"POUTFITC")
          N2=N-1
          WRITE(8,20)((XCOEF(I,J),YCOEF(I,J),J=1,NC),I=1,N2)
          DO 1 I=1,N2
            NEXT=INTVAL(I)
            N3=N1(I)+1
             DO 2 J=1,N3
                WRITE(8,10)XCORD(NEXT),YCORD(NEXT),TCORD(NEXT)
                NEXT=LINK(NEXT)
2            CONTINUE
1         CONTINUE
          WRITE(8,10)X0(N),Y0(N),TCORD(NEXT+1)
10        FORMAT(F12.4)
20        FORMAT(F12.4)
          ENDFILE 8
          RETURN
          END
C
C**********PLOT THE DESIGNED CURVE*****************
C
          SUBROUTINE XPLOT(R,SCL1,SCL2,SCL3,SCL4,NSUM)
C I/O COMMON DATA AREA
          COMMON/DATSUP/NPS,NPI(50),IFREES,X(50),Y(50),Z(50),L(50),IH(5)
&                  ,M(5),METHOD,IHELP,IPREV,BOUND(6),SUBSET(2),INTPNT,IE(2)
&                  ,ID
          COMMON/CURVEFIT/XCOEF(50,4),YCOEF(50,4),ZCOEF(50,4)
&                         ,XCORD(200),YCORD(200),ZCORD(200),TCORD(200)
C LINK LIST
```

```
      COMMON/LNKLST/LINK(200),INTVAL(50),IFREE,IFCNT
      INTEGER SUBSET
      DIMENSION X0(50),Y0(50),Z0(50)
      CALL FRMLNK(X0,Y0,Z0)
      N2=NPS-1
      CALL DTEXT(X0(1),Y0(1),"$",1)
5     DO 1 I=1,N2
        NEXT=INTVAL(I)
        IF(NEXT.NE.0)GOTO 4
        K=I
        GOTO 11
4       CALL PLUSGN(SCL1,SCL2,SCL3,SCL4,XCORD(NEXT),YCORD(NEXT))
        CALL TXMOVE(XCORD(NEXT),YCORD(NEXT))
        N11=NPI(I)
        IF(N11.NE.0)GOTO 3
        K=INTVAL(I+1)
        IF(K.NE.0) GOTO 6
        IF(NEXT.NE.0) GOTO7
        K=I+1
        GOTO 11
6       CALL TXDRAW(XCORD(K),YCORD(K))
        GOTO 1
3       DO 2 J=1,N11
          NEXT=LINK(NEXT)
          CALL TXDRAW(XCORD(NEXT),YCORD(NEXT))
2       CONTINUE
7       K= I+1
        CALL TXDRAW(X0(K),Y0(K))
11      CALL PLUSGN(SCL1,SCL2,SCL3,SCL4,X0(K),Y0(K))
        CALL TXMOVE(X0(K),Y0(K))
1     CONTINUE
      CALL PLUSGN(SCL1,SCL2,SCL3,SCL4,X0(I),Y0(I))
8     RETURN
      END
C ********* FIND TOTAL NUMBER OF INTERPOLATED POINTS*X**********
      SUBROUTINE SUM(N,N1,MSUM)
      DIMENSION N1(1)
      MSUM=0
      N2=N-1
      DO 1 I=1,N2
1     MSUM=MSUM+N1(I)
      MSUM=MSUM+N
      RETURN
      END
```

```
C                           ****************
C                           * APPENDIX 2.35 *
C                           ****************
C
C THIS MODULE HANDLES THE FOLLOWING INTERACTIVE DISPLAYS:-
C                 1.TABLE OF THE INTERPOLATED POINTS
C                 2.TABLE OF THE POLYNOMIAL COEFFICIENTS
C                 3.CURVE SUPERIMPOSE
C
C
C
C ****** MAIN PROGRAM - MODULE 5 **************
C
C
C I/O COMMON DATA AREA
        COMMON/DATSUP/NPS,NPI(50),IFREES,X(50),Y(50),Z(50),L(50),IH(5)
&                         ,M(5),METHOD,IHELP,IPREV,BOUND(6),SUBSET,INTPNT,IE(2)
&                  ,ID
        COMMON/CURVEFIT/XCOEF(50,4),YCOEF(50,4),ZCOEF(50,4),XCORD(200)
&                         ,YCORD(200),ZCORD(200),TCORD(200)
        COMMON/IO/IN,IOUT
C OVERLAY EXECUTABLE PROGRAM NAMES
        DATA PMODL2,PMODL4,PMODL1,PHELP/"PMOLD2","PMODL4",
&"PARAMETRIC","HELP"/
        LOGICAL*1 PMODL2(10),PMODL4(10),PMODL1(20),PHELP(10)
        INTEGER SUBSET
        CALL TXOPEN
C READ I/O DATA FILE
11      CALL PRDCM1
3       NC=4
21      IN=5
        IOUT=6
        CALL PRDCM2(NPS,NPI,NC,ID)
        IF(IHELP.NE.0)GOTO 66
        GOTO(33,34),IPREV
C DISPLAY INTERPOLATED POINT AS TABLE
33      CALL TABINT(NPS,NPI,XCORD,YCORD,ZCORD,TCORD,IOUT,IC,ID)
6       GOTO(51,52,59,54,59,59,57,33,58),IC
C NEXT DISPLAY
51      IPREV=0
        IHELP=0
        CALL PWRCM1
        CALL OVRLAY(PMODL4)
C PREVIOUS DISPLAY
52      IPREV=1
        IHELP=0
        CALL PWRCM1
        CALL OVRLAY(PMODL2)
C DISPLAY TABLE OF THE  COEFFICIENTS
54      CALL TABCOF(NPS,XCOEF,YCOEF,ZCOEF,METHOD,IOUT,IC,ID)
        GOTO(51,33,59,59,59,59,59,59,57,54,58),IC
C SUPERIMPOSED CURVES
34      CALL SUPIMP(NPS,NPI,XCORD,YCORD,IC)
        GOTO(51,52,59,59,59,59,61,59,57,59,58),IC
C HELP DISPLAY
57      CALL PWRCM1
        CALL OVRLAY(PHELP)
C ALGORITHM DISPLAY
61      IPREV=0
        IHELP=0
        CALL PWRCM1
```

```
        CALL OVRLAY(PMODL1)
C PROGRAM TERMINATION
58      CALL PEXIT
59      STOP
C RETURN FROM HELP
66      GOTO(33,34),IHELP
        END
C
C ***********TABULATION OF INTERPOLATED POINTS*****************
C
        SUBROUTINE TABINT(N,N1,XX,YY,ZZ,TT,IDEV,IC,IDIM)
        DIMENSION N1(1),XX(1),YY(1),ZZ(1),TT(1),IP(8)
11      CALL OUTTIL(1,9,IDEV,ICOEF)
C FIND THE TABLE SIZE
        CALL SUM(N,N1,MSUM)
        IF(MSUM.GT.25) GOTO 1
C TABLE SIZE ONE OR LESS THAN A PAGE
        IROLL=0
        CALL OUTPGE(MSUM,1,XX,YY,ZZ,TT,IDEV,IDIM)
        GOTO 12
C TABLE SIZE MORE THAN ONE PAGE
1       IP(1)=1
        IROLL=1
        DO 2 I=2,8
2       IP(I)=IP(I-1)+25
        IS=1
C FIND NUMBER OF PAGES & THE REMAINDER
        MREM=IREM(MSUM,25)
        NPAGE=(MSUM-MREM)/25
14      ITWNFIF=25
15      IPNTR=IP(IS)
        CALL OUTPGE(ITWNFIF,IPNTR,XX,YY,ZZ,TT,IDEV,IDIM)
        WRITE(IDEV,20)
20      FORMAT(/"*'TO DISPLAY THE NEXT/PREVIOUS PAGE OF THE TABLE"/
&              "USE THE FORWARD/BACKWARD AS APPROPRIATE'")
C SET UP CURSOR FOR USER SELECTION
12      CALL MNPICK(J,ICHAR,MNO)
17      CALL CONFRM(ICHAR)
        IF(ICHAR.EQ.78) GOTO 12
        IF(ICHAR.NE.89) GOTO 17
        GOTO(21,21,23,21,25,26,21,11,21),J
C BACK TO THE MAIN PROGRAM TO PROCESS OTHER COMMAND
21      IC=J
        RETURN
C HARDCOPY
23      REWIND 7
        CALL SETFIL(7,"/DEV/TTYM")
        WRITE(7,30)
30      FORMAT("COMPLETE TABLE OF THE INTERPOLATED POINTS:-")
        ITWNFIF=MSUM
        IPNTR=IP(1)
        CALL OUTPGE(ITWNFIF,IPNTR,XX,YY,ZZ,TT,7,IDIM)
        GOTO 12
C TO ROLL THE TABLE(FORWARD)
25      IF(IROLL.EQ.0)GOTO 12
        IF(IS.EQ.NPAGE.AND.MREM.GT.0)GOTO 31
        IF(IS.EQ.NPAGE.OR.IS.GT.NPAGE)GOTO 12
        IS=IS+1
        CALL OUTTIL(1,9,IDEV,ICOEF)
        GOTO 14
C OUTPUT THE REMAINDER
31      ITWNFIF=MREM
```

```
      IS=IS+1
      CALL OUTTIL(1,9,IDEV,ICOEF)
      GOTO 15
C BACKWARD
26    IF(IROLL .EQ.0) GOTO 12
      IF(IS.EQ.1) GOTO 12
      IS=IS-1
      CALL OUTTIL(1,9,IDEV,ICOEF)
      GOTO 14
      END
C
C************** TABULATE COEFFICIENTS***********************
C
      SUBROUTINE TABCOF(N,XC,YC,ZC,METHOD,IDEV,IC,IDIM)
      DIMENSION XC(50,4),YC(50,4),ZC(50,4),IP(4)
      ICOEF=1
C OUFUT TABLE TITLES
11    CALL OUTTIL(2,11,IDEV,ICOEF)
      NC=N-1
      IF(NC.GT.20) GOTO 1
C LESS THAN 20,ONE PAGE OF TABLE
      IROLL=0
      IF(ICOEF.EQ.1)CALL OUTCOF(NC,1,METHOD,XC,IDEV)
      IF(ICOEF.EQ.2)CALL OUTCOF(NC,1,METHOD,YC,IDEV)
      IF(ICOEF.EQ.3)CALL OUTCOF(NC,1,METHOD,ZC,IDEV)
      GOTO 2
C GREATER THAN 20 ,MORE THAN ONE PAGE
1     IROLL=1
      IP(1)=1
      DO 7 I=2,4
7     IP(I)=IP(I-1)+20
      IS=1
C FIND NUBER OF PAGES AND THE REMAINDER
      NREM=IREM(NC,20)
      NCPGE=(NC-NREM)/20
14    N3=20
15    IPNTR=IP(IS)
12    IF(ICOEF.EQ.1)CALL OUTCOF(N3,IPNTR,METHOD,XC,IDEV)
      IF(ICOEF.EQ.2)CALL OUTCOF(N3,IPNTR,METHOD,YC,IDEV)
      IF(ICOEF.EQ.3)CALL OUTCOF(N3,IPNTR,METHOD,ZC,IDEV)
      WRITE(IDEV,20)
20    FORMAT(/"*'TO DISPLAY THE NEXT/PREVIOUS PAGE OF THE TABLE"/
&               "  USE THE FORWARD/BACKWARD AS APPROPRIATE'")
C RAISE CURSOR
2     CALL MNPICK(J,ICHAR,MNO)
17    CALL CONFRM(ICHAR)
      IF(ICHAR.EQ.78) GOTO 2
      IF(ICHAR.NE.89) GOTO 17
      GOTO(21,21,32,33,34,23,24,25,21,11,21),J
32    ICOEF=1
      GOTO 11
33    ICOEF=2
      GOTO 11
34    IF(IDIM.EQ.2) GOTO 2
      ICOEF=3
      GOTO 11
C OTHER COMMAND
21    IC=J
      RETURN
C HARDCOPY
23    REWIND 7
      CALL SETFIL(7,"/DEV/TTYM")
```

```
        WRITE(7,77)
77      FORMAT(/"PARAMETRIC COEFFICIENTS OF X:-")
        CALL OUTCOF(NC,1,METHOD,XC,7)
        WRITE(7,88)
88      FORMAT(/"PARAMETRIC COEFFICIENTS OF Y:-")
        CALL OUTCOF(NC,1,METHOD,YC,7)
        IF(IDIM.EQ.2) GOTO 2
        WRITE(7,99)
99      FORMAT(/"PARAMETRIC COEFFICIENTS OF Z:-")
        CALL OUTCOF(NC,1,METHOD,ZC,7)
        GOTO 2
C FORWARD COMMAND(TABLE ROLLING)
24      IF(IROLL.EQ.0)GOTO 2
        IF(IS.EQ.NCPGE.AND.NREM.GT.0) GOTO 31
        IF(IS.EQ.NCPGE.OR.IS.GT.NCPGE) GOTO 2
        IS=IS+1
        CALL OUTTIL(2,11,IDEV,ICOEF)
        GOTO14
31      N3=NREM
        IS=IS+1
        CALL OUTTIL(2,11,IDEV,ICOEF)
        GOTO 15
C BACKWARD COMMAND
25      IF(IROLL.EQ.0) GOTO 2
        IF(IS.EQ.1) GOTO 2
        IS=IS-1
        CALL OUTTIL(2,11,IDEV,ICOEF)
        GOTO 14
        END
C
C**********SUPERIMPOSED CURVE DISPLAY***************
C
        SUBROUTINE SUPIMP(N,N1,XX,YY,IC)
        COMMON/CURVES/NCRV(10),XYSCL(4)
        COMMON/IO/IN,IOUT
C MENU ITEMS
        DATA MNTXT/"+ NEXT    + PREVIOUS+ GRAPH   + DISP.ORG+ DELETE*
&+ REFRESH + METHOD  + AX. MARK+ HELP    + RESTART + EXIT     "/
        DATA SUPFLS/"CURVE1    CURVE2    CURVE3    CURVE4    CURVE5
&CURVE6    CURVE7    CURVE8    CURVE9    CURVE10   "/
        DIMENSION N1(1),XX(1),YY(1)
        LOGICAL*1 MNTXT(110),SUPFLS(100)
        CALL RDCRVS
11      CALL TXCLER
        CALL SUM(N,N1,MSUM)
C SET UP PLOTTING SCALES
        S1=XYSCL(1)
        S2=XYSCL(2)
        S3=XYSCL(3)
        S4=XYSCL(4)
C PREPARE THE DISPLAY
        WRITE(IOUT,10)
10      FORMAT("SUPERIMPOSED CURVES:-")
        CALL MNOPEN(875.,715.,1)
        CALL MNDISP(MNTXT,11,10,1)
        CALL FRAME(870.,733.,11)
        CALL ALPHMD
        WRITE(IOUT,20)
20      FORMAT(/////////////62X,"* TYPE IN"/62X,"CURVE NO.")
2       CALL LMTARA
C CHECK FOR REFRESH
        IF(J.EQ.6) GOTO 23
```

```
3       CALL MNPICK(J,ICHAR,MNO)
        NFDEL=ICHAR-48
        IF(J.EQ.5.AND.NFDEL.GT.9.OR.J.EQ.5.AND.NFDEL.LT.1) GOTO 3
17      CALL CONFRM(ICHAR)
        IF(ICHAR.EQ.78) GOTO 12
        IF(ICHAR.NE.89) GOTO 17
        GOTO(21,21,23,24,25,11,21,27,21,11,21),J
12      J=0
        GOTO 2
21      IC=J
        RETURN
C GRAPH/REFRESH
23      K=0
        CALL LMTSCL(S1,S2,S3,S4)
        CALL PFRAME(S1,S2,S3,S4)
        DO 1 I=1,10
        REWIND 9
        IF(NCRV(I).NE.99) GOTO 22
        CALL SETFIL(9,SUPFLS(I+K))
77      CALL SREDRW(N,N1,I,XX,YY,MSUM,S1,S2,S3,S4)
22      K=K+9
1       CONTINUE
        J=0
        GOTO 2
C DISPLAY CURVE ORIGIN
24      CALL DISORG(S1,S2,S3,S4)
        GOTO 2
C DELETE CURVE OPTION
25      NCRV(NFDEL)=0
        CALL WRCRVS
        GOTO 2
C AXES MARKING
27      CALL AXSMRK(S1,S2,S3,S4)
        GOTO 2
        END
C
C ************** PLOTTING CURVE FIT ****************************
C
        FUNCTION EPLOT(NC,SCL1,SCL2,SCL3,SCL4,NSUM)
C I/O COMMON DATA AREA
        COMMON/DATSUP/NPS,NPI(50),IFREES,X(50),Y(50),Z(50),L(50),IH(5)
&               ,M(5),METHOD,IHELP,IPREV,BOUND(6),SUBSET,INTPNT,IE(2)
&               ,ID
        COMMON/CURVEFIT/XCOEF(50,4),YCOEF(50,4),ZCOEF(50,4),XCORD(200)
&                       ,YCORD(200),ZCORD(200),TCORD(200)
        DATA IDC/"123456789"/
        LOGICAL*1 IDC(9)
        IP=1
        IF=1
        I=1
C PLOT SUPPLIED POINTS
3       CALL PLUSGN(SCL1,SCL2,SCL3,SCL4,XCORD(I),YCORD(I))
2       CALL TXMOVE(XCORD(I),YCORD(I))
        IF(I.EQ.NSUM)RETURN
        IP1=IP+1
        IP2=IP+NPI(IF)+1
        DO 1 J=IP1,IP2
1       CALL TXDRAW(XCORD(J),YCORD(J))
        IF(IF.NE.1.OR.NC.EQ.0) GOTO 4
C NUMBER THE CURVES
        CALL DTEXT(XCORD(J-3),YCORD(J-3),IDC(NC),1)
4       IP=IP2
```

```
      IF=IF+1
      I=I+NPI(IF-1)+1
      GOTO 3
      END
C
C *************** OUTPUT COEFFICIENTS***********************
C
      SUBROUTINE OUTCOF(NCOEF,IPNTR,METHOD,C,IDEV)
      DIMENSION C(50,4)
      I=IPNTR
      CALL CURPOS(1.,580.)
      WRITE(IDEV,10)
C DISPLAY COLUMN TITLES
10    FORMAT(" I",7X,"C1",11X,"C2",11X,"C3",11X,"C4"/)
3     DO 5 J=1,NCOEF
        WRITE(IDEV,40)I,C(I,1),C(I,2),C(I,3),C(I,4)
          I=I+1
5     CONTINUE
40      FORMAT(I2,4(2X,E11.4))
7     RETURN
      END
C
C**************OUTPUT A PAGE OF THE TABLE************************
C
      SUBROUTINE OUTPGE(NPOINT,IPNT,XX,YY,ZZ,TT,IDEV,IDIM)
      DIMENSION XX(1),YY(1),ZZ(1),TT(1)
      CALL CURPOS(1.,710.)
      IF(IDIM.EQ.3)WRITE(IDEV,10)
      IF(IDIM.EQ.2)WRITE(IDEV,12)
10    FORMAT(//" I",7X,"T(I)",10X,"X(I)",9X,"Y(I)",9X,"Z(I)"/)
12    FORMAT(//" I",7X,"T(1)",10X,"X(I)",9X,"Y(I)"/)
1     DO 3 I=1,NPOINT
      IF(IDIM.EQ.3)WRITE(IDEV,30)IPNT,TT(IPNT),XX(IPNT),YY(IPNT),ZZ(IPNT)
      IF(IDIM.EQ.2)WRITE(IDEV,20)IPNT,TT(IPNT),XX(IPNT),YY(IPNT)
      IPNT=IPNT+1
3     CONTINUE
20    FORMAT(I2,3X,3(E11.4,3X))
30    FORMAT(I2,3X,4(E11.4,3X))
      RETURN
      END
C
C***************** OUTPUT THE TITLE OF THE DISPLAY********
C
C                  *******AND THE COMMAND MENU***************
      SUBROUTINE OUTTIL(IC,ITEM,IDEV,ICOEF)
C MENU ITEMS
      DATA MNTXT1/"+ NEXT    + PREVIOUS+ HARDCOPY+ COEFF'NT+ FORWARD
&+ BACKWARD+ HELP    + RESTART + EXIT     "/
      DATA MNTXT2/"+ NEXT    + PREVIOUS+ X-COEF'N+ Y-COEF'N+ Z-COEF'N
&+ HARDCOPY+ FORWARD + BACKWARD+ HELP    + RESTART + EXIT     "/
      LOGICAL*1 MNTXT1(90),MNTXT2(110)
      CALL TXCLER
      IF(IC.EQ.2) GOTO 2
      WRITE(IDEV,10)
10    FORMAT("COMPLETE TABLE OF THE INTERPOLATED POINTS:-")
      GOTO 3
2     IF(ICOEF.EQ.1)WRITE(IDEV,20)
      IF(ICOEF.EQ.2)WRITE(IDEV,30)
      IF(ICOEF.EQ.3)WRITE(IDEV,40)
20    FORMAT(///"        PARAMETRIC X-COEFFICIENTS:-"///)
30    FORMAT(///"        PARAMETRIC Y-COEFFICIENTS:-"///)
40    FORMAT(///"        PARAMETRIC Z-COEFFICIENTS:-"///)
```

```
C OUTPUT MENU
3       CALL MNOPEN(875.,715.,1)
        IF(IC.EQ.2) GOTO 22
        CALL MNDISP(MNTXT1,ITEM,10,1)
        GOTO 4
22      CALL MNDISP(MNTXT2,ITEM,10,1)
4       CALL FRAME(870.,733.,ITEM)
        RETURN
        END
C
C**********SAVE COMMON DATA AREA**************
C
        SUBROUTINE RDCRVS
        COMMON/CURVES/NCRV(10),XYSCL(4)
        REWIND 7
C OPEN OUTPUT FILE
        CALL SETFIL(7,"SUPCRVES")
        READ(7,10)(NCRV(I),I=1,10)
        READ(7,20)(XYSCL(I),I=1,4)
10      FORMAT(I2)
20      FORMAT(F11.4)
        ENDFILE 7
        RETURN
        END
C
C********** DISPLAY THE SUPERIMPOSED CURVES*********************
C
        SUBROUTINE SREDRW(N,N1,NC,XX,YY,MSUM,SCL1,SCL2,SCL3,SCL4)
        DIMENSION N1(1),XX(1),YY(1)
        READ(9,25)MSUM,N
        N2=N-1
        READ(9,30)(XX(I),YY(I),I=1,MSUM)
        READ(9,20)(N1(I),I=1,N2)
30      FORMAT(F11.4)
20      FORMAT(I3)
25      FORMAT(2I3)
        PC= EPLOT(NC,SCL1,SCL2,SCL3,SCL4,MSUM)
        RETURN
        END
C
C ********* FIND TOTAL NUMBER OF INTERPOLATED POINTS*X**********
C
        SUBROUTINE SUM(N,N1,MSUM)
        DIMENSION N1(1)
        MSUM=0
        N2=N-1
        DO 1 I=1,N2
1       MSUM=MSUM+N1(I)
        MSUM=MSUM+N
        RETURN
        END
C
C**********SAVE COMMON DATA FOR SUPERIMPOSED CURVE DISPLAY*****
C
        SUBROUTINE WRCRVS
        COMMON/CURVES/NCRV(10),XYSCL(4)
        REWIND 8
        CALL SETFIL(8,"SUPCRVES")
        WRITE(8,10)(NCRV(I),I=1,10)
        WRITE(8,20)(XYSCL(I),I=1,4)
10      FORMAT(I2)
20      FORMAT(F11.4)
```

```
ENDFILE 8
RETURN
END
```

APPENDIX 2.4

THE COMMON LIBRARY SUBROUTINES 'EPLIB'

```
C                        ****************
C                        * APPENDIX 2.40 *
C                        ****************
C
C THIS IS AN ARCHIVE LIBRARY SUBROUTINES USED BY BOTH PACKAGES.
C THEY PROVIDES THE FOLLOWING FUNCTIONS:-
C                 1.HANDLES INPUT/OUTPUT FOR READING AND WRITING COMMON DATA AREA
C                 2. UTILITY ROUTINES FOR DISPLAY IMAGES
C
C
C
C
C***********DRAWING AND MARKING THE AXES*******************
C
C SETS SCREEN AREA , SCALE AND CHECHS AXES POSITIONING
        SUBROUTINE AXSMRK(SCL1,SCL2,SCL3,SCL4)
        CALL LMTSCL(SCL1,SCL2,SCL3,SCL4)
        P1=SCL1*SCL3
        P2=SCL2*SCL4
        IF(P1)11,12,12
11      P1=0
        GOTO 22
12      P1=SCL1
22      IF(P2)111,112,112
111     P2=0
        GOTO 33
112     P2=SCL2
33      IF(P1.EQ.SCL1.AND.P2.EQ.SCL2)GOTO 1
C DRAW AXES
        CALL TXMOVE(P1,SCL4)
        CALL TXDRAW(P1,P2)
        CALL TXDRAW(SCL3,P2)
        IF(P1.NE.0) GOTO2
        CALL TXMOVE(P1,P2)
        CALL TXDRAW(SCL1,P2)
2       IF(P2.NE.0) GOTO 1
        CALL TXMOVE(P1,P2)
        CALL TXDRAW(P1,SCL2)
C MARKS THE AXES(10-DIVISION)
1       CALL XYMARK(SCL1,SCL2,SCL3,SCL4)
        CALL LMTARA
C OUTPUT SCALE FACTORS
        CALL XYVALU(SCL1,SCL2,SCL3,SCL4)
        RETURN
        END
C
C ********** OUTPUT A CONFIRMATION MESSAGE ************
C
        SUBROUTINE CONFRM(ICHAR)
        COMMON/IO/IN,IOUT
C DISPLAY THE MESSAGE ,RAISE THE CURSOR AND WAIT FOR ANSWER
        CALL CURPOS(800.,750.)
        WRITE(IOUT,10)
10      FORMAT("CONTINUE(Y/N)?")
        CALL TXCURS(X1,Y1,ICHAR)
        RETURN
        END
C
C **************READS CURSOR INPUT COORDINATES********************
C
        SUBROUTINE DISCOR(ICD,SCL1,SCL2,SCL3,SCL4)
```

```
      COMMON/IO/IN,IOUT
C ACTIVATE CURSOR OVER SPECIFIED SCREEN AREA
      CALL LMTSCL(SCL1,SCL2,SCL3,SCL4)
      CALL TXCURS(X1,Y1,ICHAR)
      ICD=ICD+1
      CALL LMTARA
      IF(ICD.EQ.1)GOTO 15
C OUTPUT CURSOR COORDINATES
      CALL CURPOS(1.,270.)
11    WRITE(IOUT,10)X1,Y1
10    FORMAT(59X,"X=",E11.4/59X,"Y=",E11.4)
2     RETURN
15    CALL CURPOS(1.,310.)
      GOTO 11
      END
C
C *************OUTPUTS DISPLAY ORIGIN *********************
C
      SUBROUTINE DISORG(SCL1,SCL2,SCL3,SCL4)
      COMMON/IO/IN,IOUT
C OUTPUT DISPLAY COORDINATS OF THE ORIGIN
      CALL CURPOS(1.,750.)
      WRITE(IOUT,20)SCL1,SCL2,SCL3,SCL4
20    FORMAT(18X,"MIN(",E11.4,",",E11.4,")"/18X,"MAX(",E11.4,",",E11.4,
&            ")")
      X0=SCL1
      Y0=SCL2
      X1=SCL3
      Y1=SCL4
C PROMPT USER TO ENTER ALTERNATIVE ORIGIN COORDINATE
25    CALL MESSAG("$ DISPLAY AXES(MIN.& MAX.)?^")
      IF(IERROR(110).NE.0) GOTO 333
      READ(IN,30)SCL1,SCL2,SCL3,SCL4
30    FORMAT(4G0.0)
C CHECK USER COORDINATES
      IF(SCL1.GT.SCL3.OR.SCL2.GT.SCL4) GOTO 25
      IF(SCL1.GT.X0.OR.SCL2.GT.Y0) GOTO 25
      IF(SCL3.LT.X1.OR.SCL4.LT.Y1) GOTO 25
      RETURN
333   ENDFILE 5
      GOTO 25
      END
C
C ****************DRAWS A SPECIFIED PICTURE/CURVE***********
C
      SUBROUTINE DRAW(PICTUR,R,X0,Y0,X1,Y1,N)
C OUTPUT ROUTINE DEFINED AS EXTERNAL TO DRAW THE PICTURE/CURVE
      EXTERNAL PICTUR
      INTEGER R
C SETS VIEWPORT ,WINDOW AND DRAWS RECTANGULAR FRAME
      CALL LMTSCL(X0,Y0,X1,Y1)
      CALL PFRAME(X0,Y0,X1,Y1)
      PC=PICTUR(R,X0,Y0,X1,Y1,N)
      RETURN
      END
C
C *****************EXIT AND DELETE ALL INTERMEDIATE FILES************
C
      SUBROUTINE EXIT
C  EXIT ROUTINE FOR THE EXPLICIT ROUTINE
      COMMON/CURVES/NCRV(10),XYSCL(4)
C FILES CONTIANING COMMON DATA AREAS
```

```
        DATA SUPCRV,DATSP,OUTCRV,COMJ/"SUPCRVES","DATSUFFL","OUTFIT",
&"COMJON"/
C FILES CONTIANING SUPREIMPOSED CURVES
        DATA C1,C2,C3,C4,C5,C6,C7,C8,C9/"CURVE1","CURVE2","CURVE3"
&,"CURVE4","CURVE5","CURVE6","CURVE7","CURVE8","CURVE9"/
        LOGICAL*1 SUPCRV(10),DATSP(10),OUTCRV(10),COMJ(10)
        LOGICAL*1 C1(10),C2(10),C3(10),C4(10),C5(10),C6(10),C7(10)
&,C8(10),C9(10)
C REMOVES UNWANTED FILES BEFORE TERMINATION
        CALL RMFILE(DATSP)
        CALL RMFILE(OUTCRV)
        CALL RMFILE(COMJ)
C       REMOVE SUPERIMPOSE CURVES
        CALL RMFILE(C1)
        CALL RMFILE(C2)
        CALL RMFILE(C3)
        CALL RMFILE(C4)
        CALL RMFILE(C5)
        CALL RMFILE(C6)
        CALL RMFILE(C7)
        CALL RMFILE(C8)
        CALL RMFILE(C9)
        CALL RMFILE(SUPCRV)
        RETURN
        END
C
C ***********RESET VIEWPORT & WINDOW DEFAULT VALUES**************
C
        SUBROUTINE LMTARA
        CALL TXVPRT(0.,0.,1023.,780.)
        CALL TXWIND(0.,0.,1023.,780.)
        RETURN
        END
C
C *************SETS UP SCREEN WINDOW ****************************
C
        SUBROUTINE LMTSCL(SCL1,SCL2,SCL3,SCL4)
        CALL TXVPRT(0.,0.,855.,700.)
        CALL TXWIND(SCL1,SCL2,SCL3,SCL4)
        RETURN
        END
C
C ********** SEARCH FOR MINIMUM & MAXIMUM VALUES OF X,Y***********
C
        SUBROUTINE MINMAX(SCL1,SCL2,SCL3,SCL4,X1,Y1,N)
        DIMENSION X1(1),Y1(1)
        SCL1=X1(1)
        SCL3=X1(1)
        SCL2=Y1(1)
        SCL4=Y1(1)
        DO 2 I=2,N
C TEST FOR MAXIMUM AND MINIMUM
        IF(X1(I).GT.SCL3) SCL3=X1(I)
        IF(X1(I).LT.SCL1) SCL1=X1(I)
        IF(Y1(I).GT.SCL4) SCL4=Y1(I)
        IF(Y1(I).LT.SCL2) SCL2=Y1(I)
2       CONTINUE
        RETURN
        END
C
C *************EXIT AND DELETE ALL INTERMEDIATE FILES**************
C
```

```
      SUBROUTINE PEXIT
C  EXIT ROUTINE FOR THE PARAMETRIC PACKAGE
      COMMON/CURVES/NCRV(10),XYSCL(4)
C FILES CONTAINING I/O COMMON DATA AREAS
    DATA SUPCRV,PDATSP,OUTCRV/"SUPCRVES","PDTSUPFL","POUTFITC"/
C FILES CONTAINING SUPERIMPOSED CURVES
      DATA C1,C2,C3,C4,C5,C6,C7,C8,C9/"CURVE1","CURVE2","CURVE3"
&,"CURVE4","CURVE5","CURVE6","CURVE7","CURVE8","CURVE9"/
      LOGICAL*1 SUPCRV(10),PDATSP(10),OUTCRV(10)
      LOGICAL*1 C1(10),C2(10),C3(10),C4(10),C5(10),C6(10),C7(10),C8(10)
&     ,C9(10)
C REMOVES UNWANTED FILES BEFORE TERMINATION
      CALL RMFILE(PDATSP)
      CALL RMFILE(OUTCRV)
C     REMOVE SUPERIMPOSE CURVES
      CALL RMFILE(C1)
      CALL RMFILE(C2)
      CALL RMFILE(C3)
      CALL RMFILE(C4)
      CALL RMFILE(C5)
      CALL RMFILE(C6)
      CALL RMFILE(C7)
      CALL RMFILE(C8)
      CALL RMFILE(C9)
      CALL RMFILE(SUPCRV)
      RETURN
      END
C
C***********DRAWS A RRECTANGULAR FRAME ROUND THE PICTURE*********
C
      SUBROUTINE PFRAME(X0,Y0,X1,Y1)
      CALL TXMOVE(X0,Y0)
      CALL TXDRAW(X0,Y1)
      CALL TXDRAW(X1,Y1)
      CALL TXDRAW(X1,Y0)
      CALL TXDRAW(X0,Y0)
      RETURN
      END
C
C ********DRAWS A PLUS SIGN*****************************************
C
      SUBROUTINE PLUSGN(SCL1,SCL2,SCL3,SCL4,X1,Y1)
C SETTING THE SIZE OF THE PLUS SIGN
      SCL5=(SCL3-SCL1)*5/855
      SCL6=(SCL4-SCL2)*5/700
C CLIP THE SIGN IF DRAWN ON THE EDGES OF THE VIEWPORT
      IF(X1.NE.SCL1) GOTO 1
      Z1=X1
      Z2=X1+SCL5
      GOTO 3
1     IF(X1.NE.SCL3) GOTO 2
      Z1=X1
      Z2=X1-SCL5
      GOTO 3
2     Z1=X1-SCL5
      Z2=X1+SCL5
      IF(Z1.LT.SCL1)Z1=X1
      IF(Z2.GT.SCL3)Z2=X1
C DRAWS THE PLUS SIGN
3     CALL TXMOVE(Z1,Y1)
      CALL TXDRAW(Z2,Y1)
      IF(Y1.NE.SCL2) GOTO 4
```

```
      Z3=Y1
      Z4=Y1+SCL6
      GOTO 6
4     IF(Y1.NE.SCL4)GOTO 5
      Z3=Y1
      Z4=Y1-SCL6
      GOTO 6
5     Z3=Y1+SCL6
      Z4=Y1-SCL6
      IF(Z4.LT.SCL2)Z4=Y1
      IF(Z3.GT.SCL4)Z3=Y1
6     CALL TXMOVE(X1,Z3)
      CALL TXDRAW(X1,Z4)
      RETURN
      END
C
C ***********RESTORES INPUT COMMON DATA AREA*****************
C
      SUBROUTINE PRDCM1
C INPUT COMMON DATA AREA
      COMMON/DATSUP/NPS,NPI(50),IFREES,X(50),Y(50),Z(50),L(50),IH(5),
&                  M(5),METHOD,IHELP,IPREV,BOUND(6),SUBSET,INTPNT,IE(2)
&                  ,ID
      INTEGER SUBSET
      REWIND 7
C OPEN INPUT FILE
      CALL SETFIL(7,"PDTSUPFL")
      READ(7,10)NPS,IFREES,METHOD,IHELP,IPREV,SUBSET,INTPNT,ID
10    FORMAT(8I3)
      IF(ID.EQ.3)READ(7,20)(NPI(I),X(I),Y(I),Z(I),L(I),I=1,NPS)
      IF(ID.EQ.2)READ(7,20)(NPI(I),X(I),Y(I),L(I),I=1,NPS)
20    FORMAT(F10.4)
      READ(7,30)(IH(I),M(I),I=1,5)
      READ(7,40)(BOUND(I),I=1,6)
      READ(7,30)IE(1),IE(2)
30    FORMAT(I3)
40    FORMAT(6F6.2)
      ENDFILE 7
      RETURN
      END
C
C ***********RESTORE OUTPUT COMMON DATA AREA**************
C
      SUBROUTINE PRDCM2(N,N1,NC,IDIM)
C OUTPUT COMMON DATA AREA
      COMMON/CURVEFIT/XCOEF(50,4),YCOEF(50,4),ZCOEF(50,4),XCORD(200),
&     YCORD(200),ZCORD(200),TCORD(200)
      DIMENSION N1(1)
      REWIND 7
C OPEN OUTPUT FILE
      CALL SETFIL(7,"POUTFITC")
      MSUM=0
      N2=N-1
      DO 1 I=1,N2
1     MSUM=MSUM+N1(I)
      MSUM=MSUM+N
      IF(IDIM.EQ.3)READ(7,10)((XCOEF(I,J),YCOEF(I,J),ZCOEF(I,J),J=1,NC),I=1,N2)
      IF(IDIM.EQ.2)READ(7,10)((XCOEF(I,J),YCOEF(I,J),J=1,NC),I=1,N2)
10    FORMAT(F12.4)
      IF(IDIM.EQ.3)READ(7,20)(XCORD(I),YCORD(I),ZCORD(I),TCORD(I)
&,I=1,MSUM)
      IF(IDIM.EQ.2)READ(7,20)(XCORD(I),YCORD(I),TCORD(I),I=1,MSUM)
```

```
20      FORMAT(F12.4)
        ENDFILE 7
        RETURN
        END
C
C ******** REMOVES ALL LINKS AND PUT DATA INTO ARRAY LIST*************
C
        SUBROUTINE PRMLNK(X1,Y1,Z1)
C INPUT COMMON DATA AREA
        COMMON/DATSUP/NPS,NPI(50),IFREES,X(50),Y(50),Z(50),L(50),IH(5),
&                   M(5),METHOD,IHELP,IPREV,BOUND(6),SUBSET,INTPNT,IE(2)
&                   ,ID
        DIMENSION X1(1),Y1(1),Z1(1)
        INTEGER SUBSET
C THREADS THROUGH THE LINK LIST AND COPY TO ARRAY LIST
        IP=IH(1)
        X1(1)=X(IP)
        Y1(1)=Y(IP)
        IF(ID.EQ.3)Z1(1)=Z(IP)
        DO 1 I=2,NPS
        IP=L(IP)
        X1(I)=X(IP)
        Y1(I)=Y(IP)
        IF(ID.EQ.3)Z1(I)=Z(IP)
1       CONTINUE
        RETURN
        END
C
C **********SAVE INPUT COMMON DATA AREA*************
C
        SUBROUTINE PWRCM1
C INPUT COMMON DATA AREA
        COMMON/DATSUP/NPS,NPI(50),IFREES,X(50),Y(50),Z(50),L(50),IH(5),
&                   M(5),METHOD,IHELP,IPREV,BOUND(6),SUBSET,INTPNT,IE(2)
&                   ,ID
        INTEGER SUBSET
        REWIND 8
C OPEN OUTPUT FILE
        CALL SETFIL(8,"PDTSUPFL")
        WRITE(8,10)NPS,IFREES,METHOD,IHELP,IPREV,SUBSET,INTPNT,ID
10      FORMAT(8I3)
        IF(ID.EQ.3)WRITE(8,20)(NPI(I),X(I),Y(I),Z(I),L(I),I=1,NPS)
        IF(ID.EQ.2)WRITE(8,20)(NPI(I),X(I),Y(I),L(I),I=1,NPS)
20      FORMAT(F10.4)
        WRITE(8,30)(IH(I),M(I),I=1,5)
        WRITE(8,40)(BOUND(I),I=1,6)
        WRITE(8,30)IE(1),IE(2)
30      FORMAT(I3)
40      FORMAT(6F6.2)
        ENDFILE 8
        RETURN
        END
C
C ***********SAVE OUTPUT COMMON DATA AREA**************
C
        SUBROUTINE PWRCM2(N,N1,NC,IDIM)
C OUTPUT COMMON DATA AREA
        COMMON/CURVEFIT/XCOEF(50,4),YCOEF(50,4),ZCOEF(50,4),XCORD(200),
&       YCORD(200),ZCORD(200),TCCFD(200)
        DIMENSION N1(1)
        REWIND 8
C OPEN OUTPUT FILE
```

```
      CALL SETFIL(8,"FOUTFITC")
      MSUM=0
      N2=N-1
      DO 1 I=1,N2
1     MSUM=MSUM+N1(I)
      MSUM=MSUM+N
      IF(IDIM.EQ.3)WRITE(8,10)((XCOEF(I,J),YCOEF(I,J),ZCOEF(I,J),J=1,NC),I=1,
      IF(IDIM.EQ.2)WRITE(8,10)((XCOEF(I,J),YCOEF(I,J),J=1,NC),I=1,N2)
10    FORMAT(F12.4)
      IF(IDIM.EQ.3)WRITE(8,20)(XCORD(I),YCORD(I),ZCORD(I),TCORD(I)
&,I=1,MSUM)
      IF(IDIM.EQ.2)WRITE(8,20)(XCORD(I),YCORD(I),TCORD(I),I=1,MSUM)
20    FORMAT(F12.4)
      ENDFILE 8
      RETURN
      END
C
C *********RESTORES INPUT COMMON DATA AREA******************
C
      SUBROUTINE RDCOM1
C INPUT COMMON DATA AREA
      COMMON/DATSUP/NPS,NPI(50),IFREES,X(50),Y(50),L(50),IH(5),M(5),
&                 METHOD,IHELP,IPREV,BOUND(2),SUBSET,INTPNT,IE(2)
      INTEGER SUBSET
      REWIND 7
C OPEN AN INTERMEDIATE INPUT DATA FILE
      CALL SETFIL(7,"DATSUPFL")
      READ(7,10)NPS,IFREES,METHOD,IHELP,IPREV,SUBSET,INTPNT
10    FORMAT(7I3)
      READ(7,20)(NPI(I),X(I),Y(I),L(I),I=1,NPS)
20    FORMAT(F10.4)
      READ(7,30)(IH(I),M(I),I=1,5)
      READ(7,40) BOUND(1),BOUND(2)
      READ(7,30)IE(1),IE(2)
30    FORMAT(I3)
40    FORMAT(2F6.2)
      ENDFILE 7
      RETURN
      END
C
C ***********RESTORES OUTPUT COMMON DATA AREA**************
C
      SUBROUTINE RDCOM2(N,N1,NC)
C OUTPUT COMMON DATA AREA
      COMMON/CURVEFIT/COEF(50,6),XCORD(200),YCORD(200)
      DIMENSION N1(1)
      REWIND 7
C OPEN AN INTERMEDIATE OUTPUT FILE
      CALL SETFIL(7,"OUTFIT")
      MSUM=0
      N2=N-1
      DO 1 I=1,N2
1     MSUM=MSUM+N1(I)
      MSUM=MSUM+N
      READ(7,10)((COEF(I,J),J=1,NC),I=1,N2)
10    FORMAT(F12.4)
      READ(7,20)(XCORD(I),YCORD(I),I=1,MSUM)
20    FORMAT(2F12.4)
      ENDFILE 7
      RETURN
      END
C
```

```
C**************READ COMMON AREA FOR JOIN***********************
C
        SUBROUTINE RDCOMJ
C COMMON JOIN DATA AREA
        COMMON/JOIN/CJ1(500),CJ2(500),J3(12),J4(100),IPNTR(6)
        REWIND 7
C OPEN JOIN DATA FILE
        CALL SETFIL(7,"COMJON")
        READ(7,10)(J3(I),I=1,12)
        READ(7,10)(IPNTR(I),I=1,6)
        READ(7,10)L
        L=L+1
        READ(7,10)(J4(I),I=1,L)
        K2=J3(1)
        READ(7,20)(CJ1(I),CJ2(I),I=1,K2)
10      FORMAT(I4)
20      FORMAT(F11.4)
        ENDFILE 7
        RETURN
        END
C
C ******** REMOVES ALL LINKS AND PUT DATA INTO ARRAY LIST*************
C
        SUBROUTINE REMLNK(X1,Y1)
C INPUT COMMON DATA AREA
        COMMON/DATSUP/NPS,NPI(50),IFREES,X(50),Y(50),L(50),IH(5),M(5)
&                         ,METHOD,IHELP,IPREV,BOUND(2),SUBSET,INTPNT,IE(2)
        DIMENSION X1(1),Y1(1)
        INTEGER SUBSET
C THREADS THROUGH THE LINK LIST AND COPY TO SIMPLE LIST ARRAY
        IP=IH(1)
        X1(1)=X(IP)
        Y1(1)=Y(IP)
        DO 1 I=2,NPS
          IP=L(IP)
          X1(I)=X(IP)
          Y1(I)=Y(IP)
1       CONTINUE
        RETURN
        END
C
C *********AXES MARKING & SCALLING*************************
C
        SUBROUTINE SCALE(A1,A2,B1,B2,IFLG,SCL1,SCL2,SCL3,SCL4)
        DO 1 I=1,9
          CALL PLUSGN(SCL1,SCL2,SCL3,SCL4,B1,B2)
          IF(IFLG.EQ.1)GOTO 2
          B1=B1+A1
1       CONTINUE
        RETURN
2       B2=B2+A2
        GOTO 1
        END
C
C ************SAVE INPUT COMMON DATA AREA******************
C
        SUBROUTINE WRCOM1
C INPUT COMMON DATA AREA
        COMMON/DATSUP/NPS,NPI(50),IFREES,X(50),Y(50),L(50),IH(5),M(5),
&                        METHOD,IHELP,IPREV,BOUND(2),SUBSET,INTPNT,IE(2)
        INTEGER SUBSET
        REWIND 8
```

```
C OPEN AN INTERMEDIATE INPUT FILE
      CALL SETFIL(8,"DATSUFFL")
      WRITE(8,10)NPS,IFREES,METHOD,IHELP,IPREV,SUBSET,INTPNT
10    FORMAT(7I3)
      WRITE(8,20)(NPI(I),X(I),Y(I),L(I),I=1,NPS)
20    FORMAT(F10.4)
      WRITE(8,30)(IH(I),M(I),I=1,5)
      WRITE(8,40) BOUND(1),BOUND(2)
      WRITE(8,30)IE(1),IE(2)
30    FORMAT(I3)
40    FORMAT(2F6.2)
      ENDFILE 8
      RETURN
      END
C
C *************SAVES OUTPUT COMMON DATA AREA**************
C
      SUBROUTINE WRCOM2(N,N1,NC)
C OUTPUT COMMON DATA AREA
      COMMON/CURVEFIT/COEF(50,6),XCORD(200),YCORD(200)
      DIMENSION N1(1)
      REWIND 8
C OPEN AN INTERMEDIATE OUTPUT FILE
      CALL SETFIL(8,"OUTFIT")
      MSUM=0
      N2=N-1
      DO 1 I=1,N2
1     MSUM=MSUM+N1(I)
      MSUM=MSUM+N
      WRITE(8,10)((COEF(I,J),J=1,NC),I=1,N2)
10    FORMAT(F12.4)
      WRITE(8,20)(XCORD(I),YCORD(I),I=1,MSUM)
20    FORMAT(2F12.4)
      ENDFILE 8
      RETURN
      END
C
C*****************WRITE COMMON AREA FOR JOIN*********************
C
      SUBROUTINE WRCOMJ
C COMMON JOIN DATA AREA
      COMMON/JOIN/CJ1(500),CJ2(500),J3(12),J4(100),IPNTR(6)
      REWIND 8
C OPEN JOIN DATA FILE
      CALL SETFIL(8,"COMJON")
      WRITE(8,10)(J3(I),I=1,12)
      WRITE(8,10)(IPNTR(I),I=1,6)
      L=0
      K1=IPNTR(1)-2
      J=3
      DO 1 I=1,K1
      L=L+J3(J)-1
      J=J+2
1     CONTINUE
      WRITE(8,10)L
      L=L+1
      WRITE(8,10)(J4(I),I=1,L)
      K2=J3(1)
      WRITE(8,20)(CJ1(I),CJ2(I),I=1,K2)
10    FORMAT(I4)
20    FORMAT(F11.4)
      ENDFILE 8
```

```
      RETURN
      END
C
C ************MARKS THE GRAPH AXES**********************
C
      SUBROUTINE XYMARK(SCL1,SCL2,SCL3,SCL4)
      A1=(SCL3-SCL1)/10
      A2=(SCL4-SCL2)/10
      IFLG=0
      B1=SCL1+A1
      B2=SCL2
C MARK THE AXES
      CALL SCALE(A1,A2,B1,B2,IFLG,SCL1,SCL2,SCL3,SCL4)
      B1=SCL1+A1
      B2=SCL4
      CALL SCALE(A1,A2,B1,B2,IFLG,SCL1,SCL2,SCL3,SCL4)
      IFLG=1
      B1=SCL1
      B2=SCL2+A2
      CALL SCALE(A1,A2,B1,B2,IFLG,SCL1,SCL2,SCL3,SCL4)
      B1=SCL3
      B2=SCL2+A2
      CALL SCALE(A1,A2,B1,B2,IFLG,SCL1,SCL2,SCL3,SCL4)
      RETURN
      END
C ********* OUTPUT MAX & MIN OF X ,Y AXES**********************************
      SUBROUTINE XYVALU(SCL1,SCL2,SCL3,SCL4)
      COMMON/IO/IN,IOUT
      CALL CURPOS(0.,185.)
      WRITE(IOUT,20) SCL1,SCL3,SCL2,SCL4
20    FORMAT(64X,"X-MIN."/60X,E11.4,65X,"X-MAX."/60X,E11.4,65X,"Y-MIN."/
&     60X,E11.4,65X,"Y-MAX."/60X,E11.4)
      RETURN
      END
X
```

APPENDIX 3

ICT - PROGRAM LISTING

APPENDIX 3.1

THE CONTOURING ALGORITHM SUBROUTINES

```
C                         ****************
C                         * APPENDIX 3.1 *
C                         ****************
C
C THIS IS THE CONTOUR TRACING ALGORITHM WHICH USES THE
C TRIANGULAR CELL DISCRETIZATION.IT TRACES THE CONTOUR LINE
C WITHIN AGIVEN RECTANGLEF THE REGION CONSIDERED.
C THE ALGORITHM CONSISTS MAINLY OF FOUR SUBROUTINES TOGETHER
C WITH OTHER TWO WHICH HANDLES THE SPECIAL CASE WHEN CONTOUR
C CROSSES THE RECTANGLE EDGE TWICE.THE MAIN SUBROUTINE ARE:
C  "RECANG".....TRACES THE CONTOUR FOR EACH SIDE OF RECTANGLE
C  "DCRVSD".....FOLLOWS THE CONTOUR LINE WITHIN THE RECTANGLE
C  "DLINE ".....CONSTRUCTS THE TRIANGULAR CELL AND DRAWS THE LINE
C  "ROOT  ".....CHECK FOR THE EXISTENCE OF A ROOT ALONG AGIVEN SIDE
C
C
C
C***********TRACE THE CONTOUR IN ONE RECTANGLE***************
       SUBROUTINE RECANG(XMIN,XMAX,YMIN,YMAX,XSTEP,YSTEP,CONTOUR,II,JJ
       COMMON/LIMIT/XX1,YY1,XX2,YY2,XEXIT(4),YEXIT(4),IESID(4),CONT,
&                               XXL,YYL,ITRACE
C SET UP THE COORDINATES LIMIT OF THE RECTANGLE IN QUESTION AND
C INITIALISE EXIT LIST....
       XX1=XMIN
       YY1=YMIN
       XX2=XMAX
       YY2=YMAX
       CONT=CONTOUR
       MNSTEP=4.0*(XMAX+YMAX-XMIN-YMIN)/(XSTEP+YSTEP)
       DO 1 I=1,4
        XEXIT(I)=0.
        YEXIT(I)=0.
        IESID(I)=0
1      CONTINUE
       NEXITS=0
C CHECK FOR SPECIAL RECTANGLE
       CALL SPCREC(ISP,II,JJ,CONT,MNSTEP,XSTEP,YSTEP,NEXITS)
       IF(ISP.EQ.1) GOTO 2
C SIDE 1
       CALL DCRVSD(XMIN,YMIN,XMAX,YMIN,XSTEP,YSTEP,MNSTEP,NEXITS,1)
C SIDE 2
       CALL DCRVSD(XMIN,YMAX,XMAX,YMAX,XSTEP,YSTEP,MNSTEP,NEXITS,2)
C SIDE 3
       CALL DCRVSD(XMIN,YMIN,XMIN,YMAX,XSTEP,YSTEP,MNSTEP,NEXITS,3)
C SIDE 4
       CALL DCRVSD(XMAX,YMIN,XMAX,YMAX,XSTEP,YSTEP,MNSTEP,NEXITS,4)
C TEST SPECIAL CASE WITH TWO EXITS,AND SAVE EXIT COORDS.
2      CALL TWOEXT(II,JJ,XSTEP,YSTEP,MNSTEP)
       RETURN
       END
C
C***************DRAWS THE CURVE FROM AGIVEN RECTANGLE SIDE**********
C
       SUBROUTINE DCRVSD(X1,Y1,X2,Y2,XS,YS,M,N,ISIDE)
       COMMON/LIMIT/XX1,YY1,XX2,YY2,XEXIT(4),YEXIT(4),IESID(4),CONT,
&                       XXL,YYL,ITRACE
       COMMON/CORNERS/CNRLT(2),CNRRT(2),CNRLB(2),ISIGNS(3),FUN(3)
       COMMON/STATCS/NS,NEF1,NEF2
       DIMENSION CLD(2),CRD(2),CLU(2)
       LOGICAL RFOUND
```

```
C CHECK FOR SPECIAL CASE(TWO EXIT)
        IT=1
        IF(ITRACE.EQ.2.OR.ITRACE.EQ.4) GOTO 33
C FIND SINGLE ROOT ALONG THE GIVEN SIDE
        CALL ROOT(X1,Y1,X2,Y2,XS,YS,ISIDE,RFOUND)
        IF(RFOUND.EQ..FALSE.) RETURN
C CHECK FOR EXIT POINTS
33      DO 2 K=1,N
         XDIF=ABS(CNRLT(1)-XEXIT(K))
         YDIF=ABS(CNRLT(2)-YEXIT(K))
         XYSTEP=XS+YS
         XYDIF=XDIF+YDIF-XYSTEP
         IF(XYDIF.LT.O.) RETURN
2       CONTINUE
C MOVE BEAM TO THE LINEAR INTERPOLATED POINT OF THE FIRST BASE LINE
C CHECK FOR SPECIAL CASE
        IF(ITRACE.EQ.2.OR.ITRACE.EQ.4) GOTO 333
        XLT=CNRLT(1)
        YLT=CNRLT(2)
        FLT=FUN(1)
        XRT=CNRRT(1)
        YRT=CNRRT(2)
        FRT=FUN(2)
        X=(XRT*FLT-XLT*FRT)/(FLT-FRT)
        Y=(YRT*FLT-YLT*FRT)/(FLT-FRT)
212     CALL TXMOVE(X,Y)
        XXL=X
        YYL=Y
C TRACE THE CURVE FURTHER IN THE CURRENT RECRANGLE
333     DO 3 NSTEP=1,M
         X3=CNRLB(1)
         Y3=CNRLB(2)
         F3=F(X3,Y3)-CONT
         FUN(3)=F3
C RUN TIME STATISTICS
        NEF1=NEF1+1
        NS=NS+1
         ISIGNS(3)=1
         IF(F3.LT.O.)ISIGNS(3)=-1
         CLD(1)=CNRLT(1)
         CLD(2)=CNRLT(2)
         CRD(1)=CNRRT(1)
         CRD(2)=CNRRT(2)
         CLU(1)=CNRLB(1)
         CLU(2)=CNRLB(2)
         ICASE=O
C TEST FOR WHICH SIDE CURVE CROSSES AND TEST FOR DEGENERATE CELL
         IF(ISIGNS(3).NE.ISIGNS(1)) ICASE=ICASE+1
         IF(ISIGNS(2).NE.ISIGNS(3)) ICASE=ICASE+2
         GOTO(20,30),ICASE
C CURVE PASSES OUT LEFT HAND SIDE OF CELL
20       FL=FUN(1)
         FR=FUN(3)
         CALL DLINE(CLD,CLU,CRD,XL,YL,FL,FR)
         ISIGNS(2)=ISIGNS(3)
         FUN(2)=FUN(3)
         GOTO 7
C CURVE PASSES OUT OF RIGHT HAND SIDE OF CELL
30       FL=FUN(3)
         FR=FUN(2)
         CALL DLINE(CLU,CRD,CLD,XL,YL,FL,FR)
         ISIGNS(1)=ISIGNS(3)
         FUN(1)=FUN(3)
7        XLD1=XL-XX1-0.001
         XLD2=XX2-XL-0.001
         YLD1=YL-YY1-0.001
```

```
          YLD2=YY2-YL-0.001
          IF(YLD1.LT.0..OR.YLD2.LT.0..OR.XLD1.LT.0..OR.XLD2.LT.0.) GOTO
3        CONTINUE
         RETURN
C SET EXIT POINTS
9        IF(IT.EQ.1) GOTO 31
         N=N+1
C MARK EXIT POINT SIDE FOR SPECIAL CASE
         WRITE(9,7777)NSTEP
7777     FORMAT(I3)
         IF(YLD1.LT.0.) IESID(N)=1
         IF(YLD2.LT.0.) IESID(N)=2
         IF(XLD1.LT.0.) IESID(N)=3
         IF(XLD2.LT.0.) IESID(N)=4
         XEXIT(N)=XL
         YEXIT(N)=YL
         RETURN
31       IT=0
         GOTO 3
         END
C
C************DRAW STAIGHT LINE IN A CELL*******************
C
         SUBROUTINE DLINE(CNRL,CNRR,CNROL,X,Y,F1,F2)
         COMMON/CORNERS/CNRLT(2),CNRRT(2),CNRLB(2),ISIGNS(3),FUN(3)
         COMMON/LIMIT/XX1,YY1,XX2,YY2,XEXIT(4),YEXIT(4),IESID(4),CONT,
&                        XXL,YYL,ITRACE
         DIMENSION CNRL(1),CNRR(1),CNROL(1)
C COMPUTE THE CONTOUR INTERSECTION POINT BY LINEAR INTERPOLATION
         XLT=CNRL(1)
         YLT=CNRL(2)
         XRT=CNRR(1)
         YRT=CNRR(2)
         X=(XRT*F1-XLT*F2)/(F1-F2)
         Y=(YRT*F1-YLT*F2)/(F1-F2)
C CLIP THE LINE IF NECESSARY
22       CALL CCLIP(X,Y)
         XXL=X
         YYL=Y
         CALL TXDRAW(X,Y)
C SET UP THE COORDINATE OF THE THIRD VERTEX OF NEXT CELL
         DO 1 I=1,2
          CNRLB(I)=CNRR(I)+CNRL(I)-CNROL(I)
          CNRRT(I)=CNRR(I)
          CNRLT(I)=CNRL(I)
1        CONTINUE
         RETURN
         END
C
C*********FINDS ASINGLE ROOT ALONG THE SIDE OF RECTANGLE********
C
         SUBROUTINE ROOT(X1,Y1,X2,Y2,XS,YS,ISIDE,RFOUND)
         COMMON/LIMIT/XX1,YY1,XX2,YY2,XEXIT(4),YEXIT(4),IESID(4),CONT,
&                        XXL,YYL,ITRACE
         COMMON/CORNERS/CNRLT(2),CNRRT(2),CNRLB(2),ISIGNS(3),FUN(3)
         COMMON/STATCS/NS,NEF1,NEF2
         LOGICAL RFOUND
C COMPUTE FUNCTION VALUES AT BOTH ENDS OF THE RECTANGLE SIDE &
C ADJUST THE SIGNS
         F1=F(X1,Y1)-CONT
         FUN(1)=F1
         ISIGNS(1)=1
         IF(F1.LT.0.) ISIGNS(1)=-1
         F2=F(X2,Y2)-CONT
         FUN(2)=F2
C RUN TIME STATISTICS
```

```
      NEF2=NEF2+2
      ISIGNS(2)=1
      IF(F2.LT.0.)ISIGNS(2)=-1
C IF EQUAL SIGN ,NO ROOT IS FOUND RETURN
      IF(ISIGNS(1).EQ.ISIGNS(2)) GOTO 1
      RFOUND=.TRUE.
C BY MEANS OF REPEATED BISECTION OF THE SIDE ,DETERMINE
C THE ENDS OF THE INTERVAL EITHER SIDE OF THE ROOT.
      STEP=(XS+YS)/2.
      IF(ISIDE.GT.2) GOTO 2
C X-VARIES
      NINTS=IFIX((X2-X1)/STEP+.5)
      WINT=(X2-X1)/NINTS
      GOTO 3
C Y-VARIES
2     NINTS=IFIX((Y2-Y1)/STEP+.5)
      WINT=(Y2-Y1)/NINTS
3     INTLR=0
      INTRR=NINTS
7     INTC=IFIX((INTLR+INTRR)/2+.5)
      IF(ISIDE.GT.2) GOTO 8
      W=X1+INTC*WINT
      FW=F(W,Y1)-CONT
      GOTO 9
8     W=Y1+INTC*WINT
      FW=F(X1,W)-CONT
C ADJUST THE SIGN OF THE APPROPRIATE ENDS
9     ISIGNC=1
      IF(FW.LT.0.) ISIGNC=-1
      IF(ISIGNS(1).EQ.ISIGNC)GOTO 11
      INTRR=INTC
      FUN(2)=FW
C RUN TIME STATISTICS
111   NEF2=NEF2+1
      IF((INTRR-INTLR).GT.1) GOTO 7
C SET CORNER COORDINATES OF THE SIDE
      GOTO (10,10,20,20),ISIDE
11    INTLR=INTC
      FUN(1)=FW
      GOTO 111
C SIDE 1 OR 2,SETTING THE TRIANGLE CELL COORDINATES ON
C THE HORIZONTAL SIDES
10    IF(ISIDE.EQ.1)XYS=0.866025*STEP
      IF(ISIDE.EQ.2)XYS=-0.866025*STEP
      CNRLT(1)=X1+INTLR*WINT
      CNRLT(2)=Y1
      CNRRT(1)=CNRLT(1)+WINT
      CNRRT(2)=Y1
      CNRLB(1)=(CNRLT(1)+CNRRT(1))/2.
      CNRLB(2)=Y1+XYS
      RETURN
C SIDE 3 OR 4,SETTING UP THE TRIANGLE CELL COORDINATES ON
C THE VERTICAL SIDES
20    IF(ISIDE.EQ.3) XYS=0.866025*STEP
      IF(ISIDE.EQ.4) XYS=-0.866025*STEP
      CNRLT(1)=X1
      CNRLT(2)=Y1+INTLR*WINT
      CNRRT(1)=X1
      CNRRT(2)=CNRLT(2)+WINT
      CNRLB(1)=X1+XYS
      CNRLB(2)=(CNRLT(2)+CNRRT(2))/2.
      RETURN
C ROOT NOT FOUND
1     RFOUND=.FALSE.
      RETURN
      END
```

```
C
C************* CHECK SPECIAL CASE & P@OCESSES******************
C
        SUBROUTINE SPCREC(ISP,II,JJ,CONTOUR,MNSTEP,XSTEP,YSTEP,NEXITS)
        COMMON/LIMIT/XX1,YY1,XX2,YY2,XEXIT(4),YEXIT(4),IESID(4),CONT,
&                       XXL,YYL,ITRACE
        COMMON/SPCSD2/XE2(10),YE2(10),IROW2(10),JCOL2(10),ISD2(10),IPNT
        COMMON/SPCSD4/XE4(2),YE4(2),ISD4
        COMMON/CORNERS/CNRLT(2),CNRRT(2),CNRLB(2),ISIGNS(3),
&                       FUN(3)
        COMMON/STATCS/NS,NEF1,NEF2
        STEP=(XSTEP+YSTEP)/2.
        XYS=0.866025*STEP
C CHECK THE EXISTENCE OF TWO EXIT AND ON WHICH SIDE
        DO 1 K=1,10
         IF(ISD2(K).NE.2) GOTO 1
         IF(IROW2(K).EQ.II.AND.JCOL2(K).EQ.JJ) GOTO 5
1       CONTINUE
        IF(ISD4.EQ.4.OR.ISD4.EQ.1.OR.ISD4.EQ.3) GOTO 4
        ISP=0
        RETURN
C TOW EXITS ON SIDE 2 ,ESTABLISH A CELL ALD TRACE CURVE
5       DO 8 L=1,2
         CNRLT(1)=XE2(K)-STEP/2.
         CNRLT(2)=YE2(K)
         CNRRT(1)=XE2(K)+STEP/2.
         CNRRT(2)=YE2(K)
         CNRLB(1)=(CNRLT(1)+CNRRT(1))/2.
         CNRLB(2)=YE2(K)+XYS
         XL1=CNRLT(1)
         YL1=CNRLT(2)
         FL1=F(XL1,YL1)-CONT
         FUN(1)=FL1
         ISIGNS(1)=1
         IF(FL1.LT.0.) ISIGNS(1)=-1
         XR1=CNRRT(1)
         YR1=CNRRT(2)
         FR1=F(XR1,YR1)-CONT
         FUN(2)=FR1
C RUN TIME STATISTICS
        NEF2=NEF2+2
         ISIGNS(2)=1
         IF(FR1.LT.0.) ISIGNS(2)=-1
         IF(ISIGNS(1).NE.ISIGNS(2)) GOTO 117
         FDIF=ABS(FL1)-ABS(FR1)
         IF(FDIF.LT.0.)ISIGNS(1)=-ISIGNS(1)
         IF(FDIF.GE.0.)ISIGNS(2)=-ISIGNS(2)
117      CALL TXMOVE(XE2(K),YE2(K))
         ITRACE=2
         CALL DCRVSD(XX1,YY1,XX2,YY2,XSTEP,YSTEP,MNSTEP,NEXITS,2)
         ISD2(K)=0
         K=K+1
8       CONTINUE
        ITRACE=0
        IF(ISD4.EQ.4)GOTO 4
        GOTO 10
C TWO EXIT ON SIDE 4
4       DO 9 L=1,2
         IF(ISD4.EQ.1) GOTO 44
         CNRLT(1)=XE4(L)
         CNRLT(2)=YE4(L)-STEP/2.
         CNRRT(1)=XE4(L)
         CNRRT(2)=YE4(L)+STEP/2.
         IF(ISD4.EQ.4)CNRLB(1)=CNRLT(1)+XYS
         IF(ISD4.EQ.3)CNRLB(1)=CNRLT(1)-XYS
         CNRLB(2)=(CNRLT(2)+CNRRT(2))/2.
```

```
          GOTO 77
44        CNRLT(1)=XE4(L)-STEP/2.
          CNRLT(2)=YE4(L)
          CNRRT(1)=XE4(L)+STEP/2.
          CNRRT(2)=YE4(L)
          CNRLB(1)=(CNRLT(1)+CNRRT(1))/2.
          CNRLB(2)=YE4(L)-XYS
77        XL1=CNRLT(1)
          YL1=CNRLT(2)
          FL1=F(XL1,YL1)-CONT
          FUN(1)=FL1
          ISIGNS(1)=1
          IF(FL1.LT.0.) ISIGNS(1)=-1
          XR1=CNRRT(1)
          YR1=CNRRT(2)
          FR1=F(XR1,YR1)-CONT
          FUN(2)=FR1
C RUN TIME STATISTICS
         NEF2=NEF2+2
          ISIGNS(2)=1
          IF(FR1.LT.0.) ISIGNS(2)=-1
          IF(ISIGNS(1).NE.ISIGNS(2)) GOTO 17
          FDIF=ABS(FL1)-ABS(FR1)
          IF(FDIF.LT.0.)ISIGNS(1)=-ISIGNS(1)
          IF(FDIF.GE.0.)ISIGNS(2)=-ISIGNS(2)
17        CALL TXMOVE(XE4(L),YE4(L))
          ITRACE=4
          CALL DCRVSD(XX1,YY1,XX2,YY2,XSTEP,YSTEP,MNSTEP,NEXITS,ISD4)
9        CONTINUE
         ISD4=0
         ITRACE=0
10       ISP=1
         RETURN
         END
C
C************** CHECK FOR TWO EXITS****************
C
         SUBROUTINE TWOEXT(II,JJ,XSTEP,YSTEP,MNSTEP)
         COMMON/LIMIT/XX1,YY1,XX2,YY2,XEXIT(4),YEXIT(4),IESID(4),CONT,
&                        XXL,YYL,ITRACE
         COMMON/SPCSD2/XE2(10),YE2(10),IROW2(10),JCOL2(10),ISD2(10),IPN1
         COMMON/SPCSD4/XE4(2),YE4(2),ISD4
         COMMON/REGION/S1,S2,S3,S4
         COMMON/STATCS/NS,NEF1,NEF2
C CHECK FOR SPECIAL CASE I.E CONTOUR LINE CROSSES SIDE TWICE
         IS1=0
         IS2=0
         IS3=0
         IS4=0
         DO 2 I1=1,4
          IF(IESID(I1).EQ.1) IS1=IS1+1
          IF(IESID(I1).EQ.2) IS2=IS2+1
          IF(IESID(I1).EQ.3) IS3=IS3+1
          IF(IESID(I1).EQ.4) IS4=IS4+1
2        CONTINUE
         IF(IS1.EQ.2) GOTO 1
         IF(IS2.EQ.2) GOTO 3
         IF(IS3.EQ.2) GOTO 4
         IF(IS4.EQ.2) GOTO 5
6        RETURN
C TWO EXIT ON SIDE 1 EXAMINE LIMIT
1        IF(YY1.EQ.S2) RETURN
         GOTO 11
C TWO EXIT ON SIDE 2 EXAMINE LIMIT
3        IF(Y/2.EQ.S4) RETURN
         GOTO 17
```

```
C TWO EXIT ON SIDE 3 EXAMINE LIMIT
4        IF(XX1.EQ.S1) RETURN
         GOTO 11
C TWO EXIT ON SIDE 4 EXAMINE LIMIT
5        IF(XX2.EQ.S3) RETURN
C CHECK FOR ROOT DETECTION FOR TOP & ADJACENT RECTANGLE
17       IF(IS2.EQ.2) F1=F(XX1,YY2)-CONT
         IF(IS4.EQ.2) F1=F(XX2,YY1)-CONT
         ISN1=1
         IF(F1.LT.0.)ISN1=-1
         F2=F(XX2,YY2)-CONT
         ISN2=1
         IF(F2.LT.0.)ISN2=-1
         IF(IS2.EQ.2)YMAX1=2*YY2-YY1
         IF(IS4.EQ.2)XMAX1=2*XX2-XX1
         IF(IS2.EQ.2)F3=F(XX1,YMAX1)-CONT
         IF(IS4.EQ.2)F3=F(XMAX1,YY1)-CONT
         ISN3=1
         IF(F3.LT.0.)ISN3=-1
         IF(IS2.EQ.2) F4=F(XX2,YMAX1)-CONT
         IF(IS4.EQ.2) F4=F(XMAX1,YY2)-CONT
         ISN4=1
         NEF2=NEF2+4
         IF(F4.LT.0.)ISN4=-1
         IF(ISN1.NE.ISN2.OR.ISN3.NE.ISN4.OR.ISN1.NE.ISN3.OR.ISN2.NE.ISN4)
&        RETURN
C SAVE THE EXIT POINTS SIDE 2
         IF(IS4.EQ.2) GOTO 8
         DO 7 I2=1,10
          IF(IROW2(I2).NE.2) GOTO 19
7        CONTINUE
19       IFREE=I2
         DO 9 J1=1,4
          IF(IESID(J1).NE.2) GOTO 9
          XE2(IFREE)=XEXIT(J1)
          YE2(IFREE)=YEXIT(J1)
          IROW2(IFREE)=II+1
          JCOL2(IFREE)=JJ
          ISD2(IFREE)=2
          IFREE=IFREE+1
9        CONTINUE
         RETURN
C SAVE THE EXIT POINTS OF SIDE 4
8        ISD4=4
         K=1
         DO 12 J2=1,4
          IF(IESID(J2).NE.4) GOTO 12
          XE4(K)=XEXIT(J2)
          YE4(K)=YEXIT(J2)
          K=K+1
12       CONTINUE
         RETURN
C SAVE & TRACE ON SIDE 1/3
11       IF(IS1.EQ.2) GOTO 22
         XSAVE=XX2
         XX2=XX1
         XX1=2.*XX1-XSAVE
         ISD4=3
         GOTO 33
22       YSAVE=YY2
         YY2=YY1
         YY1=2.*YY1-YSAVE
         ISD4=1
33       K=1
         DO 14 J2=1,4
          IF(IESID(J2).EQ.1.OR.IESID(J2).EQ.3) GOTO 155
```

```
         GOTO 15
155      XE4(K)=XEXIT(J2)
         YE4(K)=YEXIT(J2)
         K=K+1
15       XEXIT(J2)=0.
         YEXIT(J2)=0.
         IESID(J2)=0
14      CONTINUE
        NEXITS=0
        CALL SPCREC(ISP,II,JJ,CONT,MNSTEP,XSTEP,YSTEP,NEXITS)
        RETURN
        END
C
C****************CLIP LINES OUTSIDE THE LIMIT******************
C
        SUBROUTINE CCLIP(X,Y)
C CLIPS THE LINE TO THE RECTANGULAR EDGE
C USING THE CONCEPT OF FINDING THE INERSECTION POINT OF
C TWO STRAIGHT LINES
        COMMON/LIMIT/XX1,YY1,XX2,YY2,XEXIT(4),YEXIT(4),IESID(4),CONT,
&                               XXL,YYL,ITRACE
        XDIF1=X-XX1
        XDIF2=XX2-X
        YDIF1=Y-YY1
        YDIF2=YY2-Y
C ANY INTERSECTION WITH EDGES ?
       IF(XDIF1.LT.0..OR.XDIF2.LT.0..OR.YDIF1.LT.0..OR.YDIF2.LT.0.)GOTO
        RETURN
C COMPUTE THE GRADIENT AND THE CONSTANT TERM OF THE EQUATION
C OF THE STRAIGHT LINE
1       SLOPE=(Y-YYL)/(X-XXL)
        CONST=YYL-SLOPE*XXL
C CHECK WHICH SIDE?
        IF(XDIF1.LT.0.) GOTO 2
        IF(XDIF2.LT.0.) GOTO 3
        IF(YDIF1.LT.0.) GOTO 4
        IF(YDIF2.LT.0.) GOTO 5
        RETURN
C X-DIRECTION
2       X=XX1
6       Y=SLOPE*X+CONST
        RETURN
3       X=XX2
        GOTO 6
4       Y=YY1
7       X=(Y-CONST)/SLOPE
        RETURN
C Y-DIRECTION
5       Y=YY2
        GOTO 7
        END
```

APPENDIX 3.2

THE USER INTERFACE SUBROUTINES

```
C                        ****************
C                        * APPENDIX 3.2 *
C                        ****************
C
C THIS IS THE INTERACTIVE  PROGRAM WHICH HANDLES THE DISPLAY
C IMAGES GENERATED.IT CONSISTS OF THREE MAIN DISPLAYS:
C                1.CONTOUR PARAMETER ENTRY
C                2.CONTOUR TRACING
C                3.ZOOMING
C
C
C
C************MAIN PROGRAM SEGMENT*******************
C
        COMMON/IO/IN,IOUT
        COMMON/REGION/S1,S2,S3,S4
        COMMON/STATCS/NS,NEF1,NEF2
        REWIND 9
        CALL SETFIL(9,"TTYI22")
        NS=0
        NEF1=0
        NEF2=0
        IN=5
        IOUT=6
C PARAMETER ENTRY DISPLAY
1       CALL CPARAM(CONTOUR,XW,YW,XSTEP,YSTEP,IG,IC)
        GOTO(2,1,5),IC
C CONTOUR DRAWING
2       CALL CDRAW(CONTOUR,XW,YW,XSTEP,YSTEP,IG,IC)
        GOTO(1,5,5,5,5,2,5),IC
C       RUN TIME STATISTICS
5       WRITE(9,7777)NS,NEF1,NEF2
7777    FORMAT("TOTAL NO. OF STEP IN TRACING=",I6/"NO. OF FUNCTION
& EVALUTION IN TRACING ONLY",I6/"NO. OF FUNCTION EVALUATION
& OTHER THAN IN TRACING",I6)
        CALL ALPHMD
        STOP
        END
C
C*************CONTOUR PARAMETER & INTRODUCTION DISPLAY************
C
        SUBROUTINE CPARAM(CONTOUR,XW,YW,XSTEP,YSTEP,IG,IC)
        COMMON/IO/IN,IOUT
        COMMON/REGION/S1,S2,S3,S4
        DATA MNTXT1/"+ NEXT    + RESTART + EXIT    "/
        DATA MNTXT2/"+CON.LEVEL          + REGION           + G.WIDTH
&          + X,Y-STEP          + DISP.GRD"/
        LOGICAL*1 MNTXT1(30),MNTXT2(90)
C INITIALISATION
        CALL TXOPEN
1       CALL TXCLER
        ICONT=0
        IREG=0
        IWIDTH=0
        ISTEP=0
        IG=0
C OUTPUT DISPLAY INSTRUCTION
        CALL TEXTUP("CTXT",10)
        CALL CURPOS(1.,510.)
```

```
        WRITE(IOUT,20)
20      FORMAT("CONTOUR PARAMETER:-"//)
        WRITE(IOUT,30)
30      FORMAT("1-CONTOUR LEVEL TO BE TRACED INITIALLY :-"//
&       "2-REGION COORDINATES(BOTTOM L.H. & TOP R.H.)"/
&       "  ENTER MINIMUM(X,Y) & MAXIMUM(X,Y) :-"/
&       "3-RECTANGULAR GRID SIZE FOR REGION"/
&       "  DISCRETISATION. ENTER X-WIDTH & Y-WIDTH :-"/
&       "4-STEP LENGTH USED IN TRACING THE CONTOUR"/
&       "  LINE. ENTER X-STEP & Y-STEP :-"/
&       "5-DISPLAY GRID LINES :-"//
&       "NOTE:-NORMALY X,Y-STEP(X,Y-WIDTH(REGION SIZE"/
&       "* DEFAULT VALUES:        REGION SIZE = 15*WIDTH"/
&       "                         WIDTH       = 10*STEP")
        CALL MNOPEN(875.,715.,1)
        CALL MNDISP(MNTXT1,3,10,1)
        CALL FRAME(870.,733.,3)
        CALL MNOPEN(700.,450.,2)
        CALL MNDISP(MNTXT2,9,10,2)
        CALL FRAME(695.,470.,9)
        X1=695.
        X2=840.
        Y=430.
        DO 111 I1=1,4
         CALL TXMOVE(X1,Y)
         CALL TXDRAW(X2,Y)
         Y=Y-40.
111     CONTINUE
        CALL DTEXT(850.,360.,"*",1)
        CALL DTEXT(850.,320.,"*",1)
2       CALL MNPICK(J,ICHAR,MNO)
        IF(MNO.EQ.2) GOTO 22
5       CALL CONFRM(ICHAR)
        IF(ICHAR.EQ.78) GOTO 2
        IF(ICHAR.NE.89) GOTO 5
        IF(J.EQ.2.OR.J.EQ.3) GOTO 212
        IF(ICONT.EQ.0.OR.IREG.EQ.0) GOTO 2
        IF(IWIDTH.EQ.0) GOTO 77
312     IF(ISTEP.EQ.0) GOTO 88
212     GOTO(11,1,15),J
C NEXT
11      IC=J
        RETURN
C CONTOUR PARAMETER
22      GOTO(31,2,33,2,35,2,37,2,39),J
C CONTOUR INITIAL LEVEL
31      CALL CURPOS(1.,140.)
        CALL MESSAG("£ CONTOUR LEVEL?  ^")
        READ(IN,40)CONTOUR
40      FORMAT(G0.0)
        ICONT=1
        GOTO 2
C REGION COORDINATES
33      CALL CURPOS(1.,118.)
        CALL MESSAG("£ REGION COORDINATES,MIN(X,Y) & MAX(X,Y)?  ^")
        READ(IN,50)S1,S2,S3,S4
50      FORMAT(4G0.0)
        IREG=1
        GOTO 2
C GRID WIDTH
35      CALL CURPOS(1.,96.)
        CALL MESSAG("£ GRID SIZE(X-WIDTH & Y-WIDTH)?  ^")
        READ(IN,60)XW,YW
60      FORMAT(2G0.0)
        IWIDTH=1
        GOTO 2
```

```
C X/Y STEP
37      CALL CURPOS(1.,74.)
        CALL MESSAG("£ X-STEP & Y-STEP?  ^")
        READ(IN,70)XSTEP,YSTEP
70      FORMAT(2G0.0)
        ISTEP=1
        GOTO 2
C DISPLAY GRID LINES
39      IG=1
        GOTO 2
C DEFAULT GRID WIDTH
77      XW=(S3-S1)/15.
        YW=(S4-S2)/15.
        GOTO 312
C DEFAULT X-STEP & Y-STEP
88      XSTEP=XW/10.
        YSTEP=YW/10.
        GOTO 212
15      STOP
        END
C
C*************CONTOUR DISPLAY ROUTINE*****************
C
        SUBROUTINE CDRAW(CONTOUR,XW,YW,XSTEP,YSTEP,IG,IC)
        COMMON/REGION/S1,S2,S3,S4
        COMMON/GRID/NLX,NLY,XGRID(31),YGRID(31)
        COMMON/IO/IN,IOUT
C DEFINE MENU ITEMS
        DATA MNTXT1/"+ PREVIOUS+ GRAPH   + CURSOR *+ ZOOM    + GRID
&+ RESTART + EXIT    "/
        DATA MNTXT2/"+CON.LEVEL+ X-STEP  + Y-STEP  "/
        DIMENSION CONLVL(11),XSTP(11),YSTP(11)
        LOGICAL*1 MNTXT1(70),MNTXT2(30)
11      ICONT=1
        ISTEP=1
C DRAW GRID LINES & SET UP COMMAND MENU
        CALL TXCLER
        WRITE(IOUT,10)
10      FORMAT("CONTOUR DISPLAY:-")
        CALL MNOPEN(875.,715.,1)
        CALL MNDISP(MNTXT1,7,10,1)
        CALL FRAME(870.,733.,7)
        CALL DTEXT(780.,558.,"* TYPE F- FINISH",16)
        CALL MNOPEN(875.,518.,2)
        CALL MNDISP(MNTXT2,3,10,2)
        CALL FRAME(870.,536.,3)
        CALL CGRID(XW,YW,S1,S2,S3,S4,IG)
4       CALL LMTARA
        CALL MNPICK(J,ICHAR,MNO)
        IF(MNO.EQ.2) GOTO 22
C COMMAND CONFIRM
17      CALL CONFRM(ICHAR)
        IF(ICHAR.EQ.78) GOTO 4
        IF(ICHAR.NE.89) GOTO 17
        GOTO(1,2,3,6,77,1,1),J
C PREVIOUS/RESTART
1       IC=J
        RETURN
C GRAPH CONTOUR LINE BY SCANNING THE WHOLE REGION
2       CALL GRAPH(XSTEP,YSTEP,XW,YW,CONTOUR,S1,S2,S3,S4)
        IF(ICONT.GT.11)ICONT=11
        IF(ISTEP.GT.11)ISTEP=11
        CONLVL(ICONT)=CONTOUR
        ICONT=ICONT+1
        XSTP(ISTEP)=XSTEP
        YSTP(ISTEP)=YSTEP
```

```
      ISTEP=ISTEP+1
      GOTO 4
C USE CURSOR TO TRACE THE CONTOUR LINE
3     CALL LMTSCL(S1,S2,S3,S4)
      IF(ICONT.GT.11)ICONT=11
      IF(ISTEP.GT.11)ISTEP=11
      CONLVL(ICONT)=CONTOUR
      ICONT=ICONT+1
      XSTP(ISTEP)=XSTEP
      YSTP(ISTEP)=YSTEP
      ISTEP=ISTEP+1
7     CALL TXCURS(CX,CY,ICHAR)
      IF(ICHAR.EQ.70) GOTO 4
C FIND THE INDEX J
      J=INDEXG(CX,XGRID)
      XMIN=XGRID(J)
      XMAX=XGRID(J+1)
C FIND ROW INDEX
      I=INDEXG(CY,YGRID)
      YMIN=YGRID(I)
      YMAX=YGRID(I+1)
C TRACE LINE IN THIS RECTANGLE
      CALL RECANG(XMIN,XMAX,YMIN,YMAX,XSTEP,YSTEP,CONTOUR,I,J)
      GOTO 7
C ZOOMING
6     CALL ZOOM(XW,YW,XSTP,YSTP,CONLVL,IG,ICONT,IC)
      GOTO(11,5,8),IC
8     IC=7
      RETURN
C DRAW GRID LINES
77    IG=1
      CALL CGRID(XW,YW,S1,S2,S3,S4,IG)
      GOTO 4
C CONTOUR PARAMETER
22    GOTO(31,32,33),J
C CONTOUR LEVEL
31    CALL CLEVEL(CONTOUR,ICONT)
      GOTO 4
C X-STEP
32    CALL XYSTEP(XSTEP,ISTEP)
      GOTO 4
C Y-STEP
33    CALL XYSTEP(YSTEP,ISTEP)
      GOTO 4
5     STOP
      END
C
C************SET UP GRID LINES*****************
C
      SUBROUTINE CGRID(XW,YW,S1,S2,S3,S4,IG)
      COMMON/GRID/NLX,NLY,XGRID(31),YGRID(31)
C SET VIEWPORT & WINDOW
      CALL LMTSCL(S1,S2,S3,S4)
C HORIZONTAL GRID LINES
      Y=S2
      NLY=IFIX((S4-S2)/YW+.5)+1
      DO 1 I=1,NLY
       IF(IG.EQ.0) GOTO 11
       CALL TXMOVE(S1,Y)
       CALL TXDRAW(S3,Y)
11     YGRID(I)=Y
       Y=Y+YW
       YDIF=Y-S4
       IF(YDIF.GT.0.) GOTO 3
1     CONTINUE
C VERTICAL GRID LINES
```

```
3       X=S1
        NLX=IFIX((S3-S1)/XW+.5)+1
        DO 2 I=1,NLX
         IF(IG.EQ.0) GOTO 12
         CALL TXMOVE(X,S2)
         CALL TXDRAW(X,S4)
12       XGRID(I)=X
         X=X+XW
         XDIF=X-S3
         IF(XDIF.GT.0.) GOTO 4
2       CONTINUE
4       RETURN
        END
C
C*********** CONTOUR LEVEL *****************************
C
        SUBROUTINE CLEVEL(C,K)
        COMMON/IO/IN,IOUT
        IF(K.GT.10) K=10
        Y=450.-22.*(K-1)
        CALL CURPOS(760.,Y)
        CALL MESSAG("£ CON.LEVEL?^")
        READ(IN,10)C
10      FORMAT(G0.0)
        RETURN
        END
C
C**********DRAW COMPLETE CONTOUR LINE**************
C
        SUBROUTINE GRAPH(XSTEP,YSTEP,XW,YW,CONTOUR,S1,S2,S3,S4)
        COMMON/GRID/NLX,NLY,XGRID(31),YGRID(31)
C DRAWS CONTOUR LINE BY SCANNING THE WHOLE REGION
        CALL LMTSCL(S1,S2,S3,S4)
        NLY1=NLY-1
        NLX1=NLX-1
        YMIN=S2
        YMAX=S2
        DO 10 I=1,NLY1
         YMIN=YMAX
         YMAX=YMAX+YW
         XMIN=S1
         XMAX=S1
         DO 20 J=1,NLX1
          XMIN=XMAX
          XMAX=XMAX+XW
C DRAW SEGMENT OF THE CONTOUR LINE IN THE SPECIFIED RECTANGLE
          CALL RECANG(XMIN,XMAX,YMIN,YMAX,XSTEP,YSTEP,CONTOUR,I,J
20       CONTINUE
10      CONTINUE
        RETURN
        END
C
C***********FIND I/J INDEX**************
C
        FUNCTION INDEXG(XY,A)
        DIMENSION A(1)
C COMPUTE THE INDEX OF THE CHOSEN RECTANGLE
        DO 11 J1=1,20
         XYDIF=XY-A(J1)
         IF(XYDIF.LE.0.) GOTO 12
11      CONTINUE
12      INDEXG=J1-1
        RETURN
        END
C
C************ X/Y [illegible] ****************************
```

```
C
        SUBROUTINE XYSTEP(XYSP,ISTP)
        COMMON/IO/IN,IOUT
        IF(ISTP.GT.10)ISTP=10
        Y=230.-22.*(ISTP-1)
        CALL CURPOS(760.,Y)
        CALL MESSAG("£ X/Y-STEP?^")
        READ(IN,10)XYSP
10      FORMAT(G0.0)
        RETURN
        END
C
C************** ZOOM PART OF THE CONTOUR LEVELS****X**********
C
        SUBROUTINE ZOOM(XW,YW,XSTP,YSTP,CONLVL,IG,ICONT,IC)
        COMMON/REGION/S1,S2,S3,S4
        COMMON/IO/IN,IOUT
        COMMON/GRID/NLX,NLY,XGRID(30),YGRID(30)
        DATA MNTXT1/"+ PREVIOUS+ RESTART + EXIT     "/
        DIMENSION XSTP(1),YSTP(1),CONLVL(1)
        LOGICAL*1 MNTXT1(30)
C SET UP CURSOR TO PICK UP COORDS. OF ZOOMED REGION
        CALL LMTSCL(S1,S2,S3,S4)
        CALL TXCURS(ZX1,ZY1,ICHAR)
1       CALL TXCURS(ZX2,ZY2,ICHAR)
        IF(ZX1.EQ.ZX2.OR.ZY1.EQ.ZY2) GOTO 1
        XJ1=AMIN1(ZX1,ZX2)
        XJ2=AMAX1(ZX1,ZX2)
        YI1=AMIN1(ZY1,ZY2)
        YI2=AMAX1(ZY1,ZY2)
        J1=INDEXG(XJ1,XGRID)
        J2=INDEXG(XJ2,XGRID)
        I1=INDEXG(YI1,YGRID)
        I2=INDEXG(YI2,YGRID)
        ZSCL1=XGRID(J1)
        ZSCL2=YGRID(I1)
        ZSCL3=XGRID(J2+1)
        ZSCL4=YGRID(I2+1)
3       CALL TXCLER
        CALL LMTARA
        WRITE(IOUT,10)
10      FORMAT("ZOOMING:-")
        CALL MNOPEN(875.,715.,1)
        CALL MNDISP(MNTXT1,3,10,1)
        CALL FRAME(870.,733.,3)
        CALL CGRID(XW,YW,ZSCL1,ZSCL2,ZSCL3,ZSCL4,IG)
        NCONT=ICONT-1
        DO 11 K1=1,NCONT
         XSTEP=XSTP(K1)
         YSTEP=YSTP(K1)
         CONTOUR=CONLVL(K1)
         CALL GRAPH(XSTEP,YSTEP,XW,YW,CONTOUR,ZSCL1,ZSCL2,ZSCL3,ZSCL4)
11      CONTINUE
        CALL LMTARA
20      CALL MNPICK(J,ICHAR,MNO)
22      CALL CONFRM(ICHAR)
        IF(ICHAR.EQ.78)GOTO 20
        IF(ICHAR.NE.89) GOTO 22
        GOTO(2,3,2),J
C PREVIOUS/EXIT
2       IC=J
        RETURN
        END
C
C*************** LIMIT GRAPHIC AREA ON YHESCREEN***************
C
```

```
      SUBROUTINE LMTARA
      CALL TXVPRT(0.,0.,1023.,780.)
      CALL TXWIND(0.,0.,1023.,780.)
      RETURN
      END
C
C************* LIMIT GRAPHIC SCALE "WINDOW"***************
C
      SUBROUTINE LMTSCL(SCL1,SCL2,SCL3,SCL4)
      CALL TXVPRT(0.,0.,750.,750.)
      CALL TXWIND(SCL1,SCL2,SCL3,SCL4)
      RETURN
      END
C
C************** CONFIRM THE COMMAND*************************
C
      SUBROUTINE CONFRM(ICHAR)
      COMMON/IO/IN,IOUT
      CALL CURPOS(800.,750.)
      WRITE(IOUT,10)
10    FORMAT("CONTINUE(Y/N)?")
      CALL TXCURS(X1,Y1,ICHAR)
      RETURN
      END
```

APPENDIX 3.3

THE CONTOURING ALGORITHM SUBROUTINES USING THE RECTANGULAR SUBDIVISIONS

```
C                           ****************
C                           * APPENDIX 3.3 *
C                           ****************
C
C THIS IS THE SECOND VERSION OF THE CONTOUR TRACING ALGORITHM
C USING THE RECTANGULAR CELL SUBDIVISION.
C
C
C
C***************DRAWS THE CURVE FROM AGIVEN RECTANGLE SIDE**********
C
      SUBROUTINE DCRVSD(X1,Y1,X2,Y2,XS,YS,M,N,ISIDE)
      COMMON/LIMIT/XX1,YY1,XX2,YY2,XEXIT(4),YEXIT(4),IESID(4),CONT,
&                     XXL,YYL,ITRACE
      COMMON/CORNERS/CNRLT(2),CNRRT(2),CNRLB(2),CNRRB(2),ISIGNS(4)
&                     ,FUN(4)
      COMMON/STATCS/NS,NEF1,NEF2
      DIMENSION CLD(2),CRD(2),CLU(2),CRU(2)
      LOGICAL RFOUND
C CHECK FOR SPECIAL CASE
      IT=1
      IF(ITRACE.EQ.2.OR.ITRACE.EQ.4) GOTO 33
C FIND SINGLE ROOT ALONG THE GIVEN SIDE
      CALL ROOT(X1,Y1,X2,Y2,XS,YS,ISIDE,RFOUND)
      IF(RFOUND.EQ..FALSE.) RETURN
C CHECK FOR EXIT POINTS
33    DO 2 K=1,N
       XDIF=ABS(CNRLT(1)-XEXIT(K))
       YDIF=ABS(CNRLT(2)-YEXIT(K))
       XYSTEP=XS+YS
       XYDIF=XDIF+YDIF-XYSTEP
       IF(XYDIF.LT.0.) RETURN
2     CONTINUE
      IDIR=1
C MOVE BEAM TO THE LINEAR INTERPOLATED POINT OF THE FIRST
C BASE LINE
C CHECK FOR SPECIAL CASE
      IF(ITRACE.EQ.2.OR.ITRACE.EQ.4) GOTO 333
      XLT=CNRLT(1)
      YLT=CNRLT(2)
      FLT=FUN(1)
      XRT=CNRRT(1)
      YRT=CNRRT(2)
      FRT=FUN(2)
      IF(ISIDE.GT.2) GOTO 22
C COMPUTE X-COORDINATE BY  INTERPOLATION ALONG THE EDGE
      X=(XRT*FLT-XLT*FRT)/(FLT-FRT)
      Y=YLT
      GOTO 212
22    X=XLT
C COMPUTE Y-COORDINATE BY  INTERPLOATION ALONG THE EDGE
      Y=(YRT*FLT-YLT*FRT)/(FLT-FRT)
212   CALL TXMOVE(X,Y)
      XXL=X
      YYL=Y
C TRACE THE CURVE FURTHER IN THE CURRENT RECRANGLE
333   DO 3 NSTEP=1,M
       X3=CNRLB(1)
       Y3=CNRLB(2)
       F3=F(X3,Y3)-CONT
       FUN(3)=F3
       ISIGNS(3)=1
       IF(F3.LT.0.)ISIGNS(3)=-1
       X4=CNRRB(1)
       Y4=CNRRB(2)
```

```
      F4=F(X4,Y4)-CONT
      FUN(4)=F4
C RUN TIME STATISTICS
      NEF1=NEF1+2
      NS=NS+1
      ISIGNS(4)=1
      IF(F4.LT.0.)ISIGNS(4)=-1
      CLD(1)=CNRLT(1)
      CLD(2)=CNRLT(2)
      CRD(1)=CNRRT(1)
      CRD(2)=CNRRT(2)
      CLU(1)=CNRLB(1)
      CLU(2)=CNRLB(2)
      CRU(1)=CNRRB(1)
      CRU(2)=CNRRB(2)
      ICASE=0
C TEST FOR WHICH SIDE CURVE CROSSES AND TEST FOR DEGENERATE
C CELL(ICASE=6)
      IF(ISIGNS(3).NE.ISIGNS(4)) ICASE=ICASE+1
      IF(ISIGNS(1).NE.ISIGNS(3)) ICASE=ICASE+2
      IF(ISIGNS(2).NE.ISIGNS(4)) ICASE=ICASE+3
      IF(ICASE.EQ.6) GOTO 4
      GOTO(10,20,30),ICASE
C CURVE PASSES OUT OF TOP OF CELL
10    FL=FUN(3)
      FR=FUN(4)
      CALL DLINE(CLU,CRU,CLD,CRD,XL,YL,FL,FR)
      ISIGNS(1)=ISIGNS(3)
      FUN(1)=FUN(3)
      ISIGNS(2)=ISIGNS(4)
      FUN(2)=FUN(4)
      IDIR=1
      GOTO 7
C CURVE PASSES OUT LEFT HAND SIDE OF CELL
20    FL=FUN(1)
      FR=FUN(3)
      CALL DLINE(CLD,CLU,CRD,CRU,XL,YL,FL,FR)
      ISIGNS(2)=ISIGNS(3)
      FUN(2)=FUN(3)
      IDIR=2
      GOTO 7
C CURVE PASSES OUT OF RIGHT HAND SIDE OF CELL
30    FL=FUN(4)
      FR=FUN(2)
      CALL DLINE(CRU,CRD,CLU,CLD,XL,YL,FL,FR)
      ISIGNS(1)=ISIGNS(4)
      FUN(1)=FUN(4)
      IDIR=3
7     XLD1=XL-XX1-0.001
      XLD2=XX2-XL-0.001
      YLD1=YL-YY1-0.001
      YLD2=YY2-YL-0.001
C REACHED THE EDGE THE RECTANGLE
      IF(YLD1.LT.0..OR.YLD2.LT.0..OR.XLD1.LT.0..OR.XLD2.LT.0.) GOTO 9
3     CONTINUE
      RETURN
C SET EXIT POINTS
9     IF(IT.EQ.1) GOTO 31
      N=N+1
C MARK EXIT POINT SIDE FOR SPECIAL CASE
      WRITE(9,7777)NSTEP
7777  FORMAT(I3)
      IF(YLD1.LT.0.) IESID(N)=1
      IF(YLD2.LT.0.) IESID(N)=2
      IF(XLD1.LT.0.) IESID(N)=3
      IF(XLD2.LT.0.) IESID(N)=4
```

```
      XEXIT(N)=XL
      YEXIT(N)=YL
      RETURN
31    IT=0
      GOTO 3
C ICASE=6
4     GOTO(10,30,20),IDIR
      RETURN
      END
C
C************DRAW STAIGHT LINE IN A CELL*******************
C
      SUBROUTINE DLINE(CNRL,CNRR,CNROL,CNROR,X,Y,F1,F2)
      COMMON/CORNERS/CNRLT(2),CNRRT(2),CNRLB(2),CNRRB(2),ISIGNS(4)
&                    ,FUN(4)
      COMMON/LIMIT/XX1,YY1,XX2,YY2,XEXIT(4),YEXIT(4),IESID(4),CONT,
&                    XXL,YYL,ITRACE
      DIMENSION CNRL(1),CNRR(1),CNROL(1),CNROR(1)
      XLT=CNRL(1)
      YLT=CNRL(2)
      XRT=CNRR(1)
      YRT=CNRR(2)
      XDIF=XRT-XLT
      IF(XDIF.EQ.0.) GOTO 2
      X=(XRT*F1-XLT*F2)/(F1-F2)
      Y=YLT
      GOTO 22
2     X=XLT
      Y=(YRT*F1-YLT*F2)/(F1-F2)
22    CALL CCLIP(X,Y)
      XXL=X
      YYL=Y
      CALL TXDRAW(X,Y)
      DO 1 I=1,2
       CL=CNRL(I)
       CR=CNRR(I)
       COL=CNROL(I)
       COR=CNROR(I)
       CNRLT(I)=CL
       CNRRT(I)=CR
       CNRLB(I)=2.*CL-COL
       CNRRB(I)=2.*CR-COR
1     CONTINUE
      RETURN
      END
C
C*********FINDS ASINGLE ROOT ALONG THE SIDE OF RECTANGLE********
C
      SUBROUTINE ROOT(X1,Y1,X2,Y2,XS,YS,ISIDE,RFOUND)
      COMMON/LIMIT/XX1,YY1,XX2,YY2,XEXIT(4),YEXIT(4),IESID(4),CONT,
&                    XXL,YYL,ITRACE
      COMMON/CORNERS/CNRLT(2),CNRRT(2),CNRLB(2),CNRRB(2),ISIGNS(4)
&                    ,FUN(4)
      COMMON/STATCS/NS,NEF1,NEF2
      LOGICAL RFOUND
C TEST WHETHER THE CONTOUR LINE INTERSECT THE EDGE?
      F1=F(X1,Y1)-CONT
      FUN(1)=F1
      ISIGNS(1)=1
      IF(F1.LT.0.) ISIGNS(1)=-1
      F2=F(X2,Y2)-CONT
      FUN(2)=F2
C RUN TIME STATISTICS
      NEF2=NEF2+2
      ISIGNS(2)=1
      IF(F2.LT.0.)ISIGNS(2)=-1
```

```
C IF THE CONTOUR LINE DOES NOT INTERSECT RETURN WITH FALSE
C VALUE
        IF(ISIGNS(1).EQ.ISIGNS(2)) GOTO 1
        RFOUND=.TRUE.
        IF(ISIDE.GT.2) GOTO 2
        NINTS=IFIX((X2-X1)/XS+.5)
        WINT=(X2-X1)/NINTS
        GOTO 3
2       NINTS=IFIX((Y2-Y1)/YS+.5)
        WINT=(Y2-Y1)/NINTS
3       INTLR=0
        INTRR=NINTS
7       INTC=IFIX((INTLR+INTRR)/2+.5)
        IF(ISIDE.GT.2) GOTO 8
        W=X1+INTC*WINT
        FW=F(W,Y1)-CONT
        GOTO 9
8       W=Y1+INTC*WINT
        FW=F(X1,W)-CONT
9       ISIGNC=1
        IF(FW.LT.0.) ISIGNC=-1
        IF(ISIGNS(1).EQ.ISIGNC)GOTO 11
        INTRR=INTC
        FUN(2)=FW
C RUN TIME STATISTICS
111     NEF2=NEF2+1
        IF((INTRR-INTLR).GT.1) GOTO 7
C SET CORNER COORDINATES OF THE SIDE
        GOTO (10,10,20,20),ISIDE
11      INTLR=INTC
        FUN(1)=FW
        GOTO 111
C SIDE 1 OR 2
10      IF(ISIDE.EQ.1)YSS=YS
        IF(ISIDE.EQ.2)YSS=-YS
        CNRLT(1)=X1+INTLR*WINT
        CNRLT(2)=Y1
        CNRRT(1)=CNRLT(1)+WINT
        CNRRT(2)=Y1
        CNRLB(1)=CNRLT(1)
        CNRLB(2)=Y1+YSS
        CNRRB(1)=CNRRT(1)
        CNRRB(2)=Y1+YSS
        RETURN
C SIDE 3 OR 4
20      IF(ISIDE.EQ.3) XSS=XS
        IF(ISIDE.EQ.4) XSS=-XS
        CNRLT(1)=X1
        CNRLT(2)=Y1+INTLR*WINT
        CNRRT(1)=X1
        CNRRT(2)=CNRLT(2)+WINT
        CNRLB(1)=X1+XSS
        CNRLB(2)=CNRLT(2)
        CNRRB(1)=X1+XSS
        CNRRB(2)=CNRRT(2)
        RETURN
C ROOT NOT FOUND
1       RFOUND=.FALSE.
        RETURN
        END
C
C************** CHECK SPECIAL CASE & P@OCESSES********************
C
        SUBROUTINE SPCREC(ISP,II,JJ,CONTO[illegible]WSTEP,XSTEP,[illegible]TS)
        COMM[illegible]IMIT/XX1,YY1,XX2,YY2,XEX[illegible],YEXIT(4),[illegible],
&                             XXL,YYL,ITRACE
```

```
      COMMON/SPCSD2/XE2(10),YE2(10),IROW2(10),JCOL2(10),ISD2(10),
&             IPNTR
      COMMON/SPCSD4/XE4(2),YE4(2),ISD4
      COMMON/CORNERS/CNRLT(2),CNRRT(2),CNRLB(2),CNRRB(2),ISIGNS(4),
&                     FUN(4)
      COMMON/STATCS/NS,NEF1,NEF2
      DO 1 K=1,10
       IF(ISD2(K).NE.2) GOTO 1
       IF(IROW2(K).EQ.II.AND.JCOL2(K).EQ.JJ) GOTO 5
1     CONTINUE
      IF(ISD4.EQ.4.OR.ISD4.EQ.1.OR.ISD4.EQ.3) GOTO 4
      ISP=0
      RETURN
C TOW EXITS ON SIDE 2 ,ESTABLISH A CELL ALD TRACE CURVE
5     DO 8 L=1,2
       CNRLT(1)=XE2(K)-XSTEP/2.
       CNRLT(2)=YE2(K)
       CNRRT(1)=XE2(K)+XSTEP/2.
       CNRRT(2)=YE2(K)
       CNRLB(1)=CNRLT(1)
       CNRLB(2)=YE2(K)+YSTEP
       CNRRB(1)=CNRRT(1)
       CNRRB(2)=CNRLB(2)
       XL1=CNRLT(1)
       YL1=CNRLT(2)
       FL1=F(XL1,YL1)-CONT
       FUN(1)=FL1
       ISIGNS(1)=1
       IF(FL1.LT.0.) ISIGNS(1)=-1
       XR1=CNRRT(1)
       YR1=CNRRT(2)
       FR1=F(XR1,YR1)-CONT
       FUN(2)=FR1
       NEF2=NEF2+2
       ISIGNS(2)=1
       IF(FR1.LT.0.) ISIGNS(2)=-1
       IF(ISIGNS(1).NE.ISIGNS(2)) GOTO 117
       FDIF=ABS(FL1)-ABS(FR1)
       IF(FDIF.LT.0.)ISIGNS(1)=-ISIGNS(1)
       IF(FDIF.GE.0.)ISIGNS(2)=-ISIGNS(2)
117    CALL TXMOVE(XE2(K),YE2(K))
       ITRACE=2
       CALL DCRVSD(XX1,YY1,XX2,YY2,XSTEP,YSTEP,MNSTEP,NEXITS,2)
       ISD2(K)=0
       K=K+1
8     CONTINUE
      ITRACE=0
      IF(ISD4.EQ.4)GOTO 4
      GOTO 10
C TWO EXIT ON SIDE 4/3/1
4     DO 9 L=1,2
       IF(ISD4.EQ.1) GOTO 44
       CNRLT(1)=XE4(L)
       CNRLT(2)=YE4(L)-YSTEP/2.
       CNRRT(1)=XE4(L)
       CNRRT(2)=YE4(L)+YSTEP/2.
       IF(ISD4.EQ.4)CNRLB(1)=CNRLT(1)+XSTEP
       IF(ISD4.EQ.3)CNRLB(1)=CNRLT(1)-XSTEP
       CNRLB(2)=CNRLT(2)
       CNRRB(1)=CNRLB(1)
       CNRRB(2)=CNRRT(2)
       GOTO 77
44     CNRLT(1)=XE4(L)-XSTEP/2.
       CNRLT(2)=YE4(L)
       CNRRT(1)=XE4(L)+XSTEP/2.
       CNRRT(2)=YE4(L)
```

```
      CNRLB(1)=CNRLT(1)
      CNRLB(2)=YE4(L)-YSTEP
      CNRRB(1)=CNRRT(1)
      CNRRB(2)=CNRLB(2)
77    XL1=CNRLT(1)
      YL1=CNRLT(2)
      FL1=F(XL1,YL1)-CONT
      FUN(1)=FL1
      ISIGNS(1)=1
      IF(FL1.LT.0.) ISIGNS(1)=-1
      XR1=CNRRT(1)
      YR1=CNRRT(2)
      FR1=F(XR1,YR1)-CONT
      FUN(2)=FR1
      NEF2=NEF2+2
      ISIGNS(2)=1
      IF(FR1.LT.0.) ISIGNS(2)=-1
      IF(ISIGNS(1).NE.ISIGNS(2)) GOTO 17
      FDIF=ABS(FL1)-ABS(FR1)
      IF(FDIF.LT.0.)ISIGNS(1)=-ISIGNS(1)
      IF(FDIF.GE.0.)ISIGNS(2)=-ISIGNS(2)
17    CALL TXMOVE(XE4(L),YE4(L))
      ITRACE=4
      CALL DCRVSD(XX1,YY1,XX2,YY2,XSTEP,YSTEP,MNSTEP,NEXITS,ISD4)
9     CONTINUE
      ISD4=0
      ITRACE=0
10    ISP=1
      RETURN
      END
```

APPENDIX 4

TMG - PROGRAM LISTING

APPENDIX 4.1

TMG - BATCH PROGRAM

```
JOB BAKIR,CU,BH1609
LUFORTRAN
RUN ,,1000
VOLUME 10000
****
DOCUMENT SOURCE
       PROGRAM(M352)
       COMPACT
       OUTPUT 2 = LPO
       OUTPUT 7 = TPO
       INPUT 1 = CRO
        COMPRESS INTEGER AND LOGICAL
        USE 5=/ARRAY
       END
       MASTER POISSON
       LOGICAL ERRORM,AXHEAT,AXMAG,PLOT MESH,INT
       INTEGER TYPE,BIT,PARITY,TAG
       COMMON/KLP/KMAX,LMAX,PARITY
       COMMON/STAGE/NPROB,IP
       COMMON/EROR/ERRORM
       COMMON/TP/TYPE,AXMAG,AXHEAT
       COMMON/SC/ZS,RS
       COMMON/INDX/I1,I2,I3,I4,K1,K2,IA,IB,IC,L,MM,JJ(7),IR,INT
      1/ST/GN(200),TITLE(80)
      1/BLK1/AINTEGRAL,M(2),RR,TRNGL,SOURCE,COEFF,NN(2),A,B,C,T4,HP,FPI3
       DIMENSION PHI(650),TAG(650),CPU(650),CPR(650),CPL(650),SCT(650)
      1,R(650),Z(650)
      1,BUF(10),REGC(10,4),FILENAME(2),PICNAME(2)
       READ(1,1)NPROB
       WRITE(2,20)NPROB
20        FORMAT(40H1  THE NUMBER OF PROBLEMS ON THIS RUN IS ,I5
      1/2X,38(1H=))
       CALL DEFBUF(5,80,BUF)

       DO 5 IP=1,NPROB
       WRITE(7,100)
100    FORMAT(/////////////)
       READ(1,6)TITLE
6      FORMAT(80A1)
       WRITE(2,7)IP,TITLE
7      FORMAT(1H0,22H TITLE OF PROBLEM NO. ,I2,5X
      1,80A1/28X,82(1H=))
       AXHEAT,AXMAG=.FALSE.
       ERRORM=.FALSE.
       READ(1,2)PLOT MESH,NEQPTL,ZS,RS
2      FORMAT(L1,4X,I0,2F0.0)
       READ(1,3)NREG,NREGC
3      FORMAT(2I0)
       CALL INPUT(REGC,NREG,NREGC,BUF,NPT)
       DO 25 I=1,NPT
       TAG(I)=0
25     CPU(I),CPR(I),CPL(I),SCT(I),PHI(I),R(I),Z(I)=0.
10     CALL TOPLGY(REGC,NREG,NREGC,TAG,PHI,NPT,R,Z)
       IF(ERRORM)GOTO5
15      CALL MESH RELXN(REGC,NREG,NREGC,TAG,R,Z,NPT)
       CALL PARAM(REGC,NREG,NREGC,TAG,R,Z,CPU,CPR,CPL,SCT,NPT,PHI)
       CALL RELX(TAG,CPU,CPR,CPL,SCT,PHI,R,Z,NPT)
       CALL PLOTT(R,Z,PHI,TAG,NPT,NEQPTL,PLOT MESH)
5      CONTINUE
       STOP
```

```
1       FORMAT(I0)
        END
        SUBROUTINE INPUT(REGC,NREG,NREGC,BUF,NPT)
        LOGICAL END
        INTEGER PARITY
        COMMON/INDX/I,J,ICARD,JJ,M,END,IA,IB,IC,L,K,II(6),IG,IR,KK
       1/ST/GM(200),CHAR(80)/KLP/KMAX,LMAX,PARITY
        DIMENSION REGC(NREG,NREGC)
          DIMENSION BUF(10)
C   READ NUMBER OF COLUMNS AND ROWS; ALSO THE MESH PARITY(-1 L//Z),(+1 K
        READ(1,1)KMAX,LMAX,PARITY
        NPT=(KMAX+1)*(LMAX+1)
C   CALCULATE NUMBER OF POINTS IN MESH
        WRITE(2,20)NPT,KMAX,LMAX,PARITY
20      FORMAT(1H0,80(1H*)/37H THE TOTAL NUMBER OF MESH POINTS IS  ,I5,
       113H.  THERE ARE ,I3,8H COLUMNS/8H   AND  ,I3,
       128H  ROWS.  THE MESH PARITY IS ,I3/80(1H*))
        IG =1
        ICARD=1
10      DO 50 IR=1,NREG
CONSTANTS FOR REGION AND DEFINING BOUNDARY POINTS ARE READ IN
        READ(1,5)(REGC(IR,I),I=1,NREGC)
        CALL GEOMETRY(BUF)
50      CONTINUE
        RETURN
1       FORMAT(3I0)
2       FORMAT(4I0)
5       FORMAT(4E0.0)
        END
        SUBROUTINE GEOMETRY(BUF)
        LOGICAL END
        COMMON/INDX/I,J,ICARD,JJ,M,END,IA,IB,IC,L,K,II(6),IG,IR,KK
       1/ST/GM(200),CHAR(80)
        DIMENSION COUNT(10),FMT(5),SNTL(3),BUF(10)
        DATA COUNT,XFLD,AFLD,EFLD,SNTL,BLANK,ALEFT/5H0    ,5H1    ,
       15H2    ,5H3    ,5H4    ,5H5    ,5H6    ,5H7    ,5H8    ,5H9    ,
       15HX,   ,5HA1)  ,5HE0.0),
       15HA    ,5HB    ,5HG    ,5H     ,5H(    /
C    A CARD MAY BE MISSING AT THIS POINT
C   FREE FIELD ROUTINE;OBJECT-TIME FORMATS SET UP FOR ALPHANUMERIC FIELD
50      READ(1,1)BUF
1       FORMAT(10A8)
        READ(5,2)CHAR
        WRITE(2,60)ICARD,CHAR
2       FORMAT(80A1)
60      FORMAT(10H  CARD NO.,I4,5X,1H(,80A1,2H ))
        ICARD=ICARD+1
        IB=1
        JJ=1
        END=.FALSE.
        FMT(1)=ALEFT
        DO 45  I=1,80
        CALL COMP8(CHAR(I),BLANK,J)
        GOTO(40,5)J
5       GOTO(10,55)IB
10      IB=2
        DO 15 K=1,3
        CALL COMP8(CHAR(I),SNTL(K),J)
        GOTO(20,15)J
15      CONTINUE
```

```
      L=1
      FMT(5)=EFLD
      GOTO25
20    IF(K.EQ.3)END=.TRUE.
      FMT(5)=AFLD
      L=2
25    J=I/10
      FMT(4)=XFLD
      IF(J.NE.0)GOTO35
      FMT(2)=BLANK
      IF(I.NE.1)GOTO30
      FMT(3)=BLANK
      FMT(4)=BLANK
      GOTO65
30    FMT(3)=COUNT(I)
      GOTO65
35    M=I-J*10
      IF(M)100,100,105
100   FMT(2)=COUNT(J)
      FMT(3)=COUNT(10)
      GOTO65
105     FMT(2)=COUNT(J+1)
      FMT(3)=COUNT(M)
65    READ(5,FMT)GM(IG)
85    IG=IG+1
      IF(END)RETURN
      GOTO55
40    IF((I-JJ).GT.1)IB=1
      GOTO45
55    JJ=I
45    CONTINUE
      GOTO50
      END
      SUBROUTINE CODE(I,J,PHI,IG,SAME)
      LOGICAL SAME
      COMMON/ST/GM(200),TEMR(80)
      DATA BNDRY/5HB    /
      CALL COMP8(BNDRY,GM(IG),II)
      IF(II.EQ.2)RETURN
      SAME=.FALSE.
      IF(GM(IG+1).LT.0..OR.GM(IG+2).LT.0.)SAME=.TRUE.
      I=ABS(GM(IG+1))
      J=ABS(GM(IG+2))
      PHI=GM(IG+3)
      IG=IG+4
      RETURN
      END
       SUBROUTINE TOPLGY(REGC,NREG,NREGC,TAG,PHI,NPT,R,Z)
      LOGICAL ERROR,SAME,SAMEO
      INTEGER BIT,ARCFIN,PARITY,PK(6),TAG(NPT)
      COMMON/ST/GM(200),TEMP(80)/KLP/KMAX,LMAX,PARITY
      COMMON/EROR/ERROR
     1/INDX/I1,I2,I3,I4,I5,I6,IA,IB,IC,L,K,PK,IG,IR,ARCFIN
     1/BLK1/L1,L2,K1,K2,R1,R2,Z1,Z2,LO,KO,RO,ZO,PHIO,PHI1,PHI2,PHIAV
     1/BLK2/LAO,KAO,RAO,ZAO,THETAO,LA2,KA2,RAXIS,ZAXIS,KK,THETA,IA5,IA6
      DIMENSION REGC(NREG,NREGC),R(NPT),Z(NPT),PHI(NPT),A(2)
      DATA ARC SENTL,END GEOM,A/5HA     ,5HG     ,5HFREE ,5HFIXED/
      IT(I)=I-2*(I/2)
      IG=1
C   REGN. LOOP:
```

```
      DO 145 IR=1,NREG
      I0,J0,I1,I2=1
      PHI0=0.
CHECK FOR THE TYPE OF BOUNDARY IF INDICATED.
      CALL CODE(I0,J0,PHI0,IG,SAME0)
      I5=I0
      I6=J0
      PHIAV,PHI2,PHI1=PHI0
C  CHECK IF ARC COMMENCES AT FIRST PT.:
      CALL COMP8(GM(IG),ARC SENTL,II)
      IF(II.EQ.2)GO TO 30
      I2=2
      IG=IG+1
C  FIRST REGN. PT:
30    L1,L0=GM(IG)
      K1,K0=GM(IG+1)
      R1,R0=GM(IG+2)
      Z1,Z0=GM(IG+3)
      WRITE(2,505)IR,L0,K0,R0,Z0,A(I6),A(I5),PHI0
505   FORMAT(20H0 FIRST PT OF REGION  ,I3,2X,3H  (,2I3,
     12F12.6,2X,A8,2X,A8,5X,F12.6,1H))
      WRITE(2,501)
501    FORMAT(17H  SUBSEQUENT PTS:)
34    IG=IG+4
C  CHECK FOR SUCCEEDING ARCS:
      CALL COMP8(GM(IG),ARC SENTL,II)
      IF(II.EQ.2) GO TO 31
      I1=3
      GOTO36
C   ALSO FOR END OF REGN. GEOM:
31    CALL COMP8(GM(IG),END GEOM,II)
      IF(II.EQ.2)GO TO 40
      I1=2
36    IG=IG+1
40    IF(I1.NE.2)GOTO45
      L2=L0
      K2=K0
      R2=R0
      Z2=Z0
      I5=I0
      I6=J0
      PHI2=PHI0
      IF(SAME0)PHI1=PHI2
      GOTO48
45    CALL CODE(I5,I6,PHI2,IG,SAME)
      IF(SAME)PHI1=PHI2
      IF(I2.EQ.2)GOTO200
      L2=GM(IG)
      K2=GM(IG+1)
      R2=GM(IG+2)
      Z2=GM(IG+3)
      WRITE(2,502)L2,K2,R2,Z2,A(I6),A(I5),PHI2
502   FORMAT(5X,2I4,5X,2F12.6,2X,A8,2X,A8,5X,F12.6)
C   LINEAR INTERPOLATION OF BOUNDARY POINTS; TAGGING OF POINTS AND SIDES
48    CALL BSET(R,Z,PHI,TAG,NPT)
      IF(ERROR)RETURN
      GOTO(34,55,190)I1
CELLS ARE TAGGED WITH THEIR APPROPRIATE REGION NUMBERS.
55    CALL SET REGION(REGC,NREG,NREGC,TAG,PHI,R,Z,NPT)
      IF(ERROR)RETURN
```

```
145       CONTINUE
        RETURN
190     IG=IG+4
200     CALL ARCSET
        IF(ERROR)RETURN
        CALL ARC(TAG,PHI,R,Z,NPT)
        IF(ERROR)RETURN
         GOTO54
        END

        SUBROUTINE BSET(R,Z,PHI,TAG,NPT)
        LOGICAL ERROR
        INTEGER BIT,ARCFIN,PARITY,PK(6),TAG(NPT)
        INTEGER BT(5)
         COMMON/ST/GM(500),TEMP(80)/KLP/KMAX,LMAX,PARITY
       1/INDX/I1,I2,I3,I4,I5,I6,IA,IB,IC,L,K,PK,IG,IR,ARCFIN
       1/BLK1/L1,L2,K1,K2,R1,R2,Z1,Z2,L0,K0,R0,Z0,PHI0,PHI1,PHI2,PHIAV
       1/STAGE/NPROB,IP/EROR/ERROR
        DIMENSION PHI(NPT),R(NPT),Z(NPT)
        IT(I)=I-2*(I/2)
5       L21=L2-L1
        K21=K2-K1
        IB=1
        L,K=1
CHECKING OF PERMITTED LOGICAL LINES IS CARRIED OUT UNTIL LABEL  25.
        IF(L21.LE.0)L=-1
        IF(K21.LE.0)K=-1
        IF(L21.NE.0)IB=IB+1
        IF(K21.NE.0)IB=IB+2
        GOTO(250,20,30,25)IB
25      IF((K21-(L2+1)/2+(L1+1)/2).NE.0)IF(K21+L2/2-L1/2)400,20,400
20      INC=(KMAX+1)*L
        LMT=L21
        GOTO35
30      INC=K
        LMT=K21
35      IB=IB-1
        LMT=IABS(LMT)
        IA=L2*(KMAX+1)+K2+1
        R(IA)=R2
        Z(IA)=Z2
        PHI(IA)=PHI2
        PHIAV=PHIAV+PHI(IA)
        I3=I3+1

C     TAG A BOUNDARY POINT( DIRICHLET=2; FREE=1)
        CALL PACK(TAG(IA),I5,1)
        IF(K21)40,45,60
45      IF(IT(L2))60,60,40
C  TAG A SIDE ON THE BOUNDARY(UPPER=1;LOWER=2; BOTH=3)
40      IF(L21.NE.0)CALL PACK(TAG(IA),1+(1+L)/2,4)
60      IF(LMT.EQ.1)GOTO180
        R21=(R2-R1)/LMT
        Z21=(Z2-Z1)/LMT
        P21=(PHI2-PHI1)/LMT
C IN THIS LOOP PERFORM AFOREMENTIONED TAGGING OF POINTS AND SIDES; ALSO
C INTERPOLATE COORDINATES AND POTENTIALS FOR INTERMEDIATE POINTSOF LINE
        DO 105 IC=1,LMT-1
        IA=IA-INC
        IF(IB.EQ.3)IA=IA+IT(IC+L2)-(K+1)/2
```

```
      CALL PACK(TAG(IA),I6,1)
      GOTO(90,100,85)IB
85    CALL PACK(TAG(IA),2-IABS(K+L)/2,4)
      GOTO100
90    IF(IT(IC+L2).GT.0)CALL PACK(TAG(IA),3,4)
100    R(IA)=R2-IC*R21
      Z(IA)=Z2-IC*Z21
      PHI(IA)=PHI2-IC*P21
      PHIAV=PHIAV+PHI(IA)
      I3=I3+1
105   CONTINUE
180   IF(K21)130,120,110
120   IF(IT(L1))130,130,110
110   IA=L1*(KMAX+1)+K1+1
C   TAG A SIDE
      IF(L21.NE.0)CALL PACK(TAG(IA),1+(1-L)/2,4)
130   R1=R2
      Z1=Z2
      L1=L2
      K1=K2
      PHI1=PHI2
135   ARCFIN=0
      RETURN
C  =================================
C  ERROR  PRINTS)=
210   WRITE(2,1)
1     FORMAT(40H1 CONN. ARCS ,SAME L AND K, DIFF. COORDS)
      ERROR=.TRUE.
      RETURN
250   GO TO (251,135,252)I1
252     IF(ARCFIN.EQ.0)GO TO 251
      IF(ABS(R2-R1).GT.1.E-12.OR.ABS(Z2-Z1).GT.1.E-12)GOTO210
      RETURN
251   WRITE(2,2)L1,L2,K1,K2
2     FORMAT(12H1 NO NEW K,L,13H L1,L2,K1,K2=,4I3)
      ERROR=.TRUE.
      RETURN
400   WRITE(2,3)L1,L2,K1,K2
3     FORMAT(38H1 L1,K1 AND L2,K2 NOT ON SAME LOG, LIN/
     1   13H L1,L2,K1,K2=,4I3)
      ERROR=.TRUE.
      RETURN
      END
      SUBROUTINE ARCSET
      LOGICAL ERROR,SAME
      INTEGER SW(6),ARCFIN,PARITY
      COMMON/ERUR/ERROR/STAGE/NPROB,IP
     1/ST/GM(500),TEMP(80)/KLP/KMAX,LMAX,PARITY
     1/INDX/I1,I2,I3,I4,I5,I6,IA,IB,IC,L,K,SW,IG,IR,ARCFIN
     1/BLK1/L1,L2,K1,K2,R1,R2,Z1,Z2,L0,K0,R0,Z0,PHI0,PHI1,PHI2,PHIAV
     1/BLK2/LA0,KA0,RA0,ZA0,THETA0,LA2,KA2,RAXIS,ZAXIS,KK,THETA,IA5,IA6
      IT(I)=I-2*(I/2)
      HFPI=1.5707963
C   PREPARE INPUT FOR ARC)
C  LOG, CENTRE OF ARC)
200   LA0=GM(IG)
      KA0=GM(IG+1)
C  LOG, ANGLE)
      THETA0=HFPI*GM(IG+2)
      IF(THETA0)205,400,210
```

```
205     THETAU=-THETAU
        SW(6)=2
        GOTO215
210        SW(6)=1
215       IGC=IG+3
        CALL CODE(I5,I6,PHI2,IGC,SAME)
        IA5=I5
        IA6=I6
        I5,I6=IA6
        IG=IGC-3
        IF(SAME)PHI1=PHI2



        LA2=GM(IG+3)
        KA2=GM(IG+4)
        I4=1
        IF(LA0.EQ.L1)I4=I4+1
        IF(PARITY.LT.0)I4=I4+2
        IF(IT(I4).NE.0)GOTO219
        KK=KA2
        IF(KA0.NE.KA2)GOTO415
        GOTO221
219     IF(KA0.NE.K1)GOTO415
        KK=K1
        IF(LA0.NE.LA2)GOTO405
CALCULATE POINT COORDINATES AT LOGICAL CENTRE OF THE  ARC; ALSO,
CALCULATE LENGTHS OF SEMI AXES;SET INITIAL POLAR ANGLE.
221     GOTO(220,225,225,220)I4
220     RA0=R1
        ZA0=GM(IG+6)
        THETA=0.
        RAXIS=GM(IG+5)
        ZAXIS=Z1
        GOTO230
225     RA0=GM(IG+5)
        ZA0=Z1
        RAXIS=R1
        THETA=HFPI
        ZAXIS=GM(IG+6)
C   ERROR CHECKS)
230      RAXIS=RAXIS-RA0

        ZAXIS=ZAXIS-ZA0
        IF(LA2.EQ.L1)GOTO420
        IF(KA2.EQ.K1)GOTO425
        IF(PARITY.EQ.0)GOTO430
        RETURN
400     WRITE(2,401)
401     FORMAT(16H1 ZERO ARC ANGLE)
        GOTO500
405     WRITE(2,406) LA0,LA2
406     FORMAT(12H1 LA0.NE.LA2,2I4)
        GOTO500
415     WRITE(2,416)KK,KA0
416     FORMAT(15H1 IN ARC,KK.KA0,2I2)
        GOTO500
420     WRITE(2,421)LA2
421     FORMAT(15H1 IN ARC,LA2=L1,I2)
        GOTO50(
```

```
425    WRITE(2,426)KA2
426    FORMAT(16H1 IN ARC,KA2=K1=,I2)
       GOTO500
430    WRITE(2,431)
431    FORMAT(13H1 ZERO PARITY)
500     ERROR=.TRUE.
       RETURN
       END
       SUBROUTINE ARC(TAG,PHI,R,Z,NPT)
       LOGICAL ERROR
       INTEGER BIT,ARCFIN,PARITY,SW(6),TAG(NPT)
        COMMON/ST/GM(500),TEMP(80)/KLP/KMAX,LMAX,PARITY
      1/BLK1/L1,L2,K1,K2,R1,R2,Z1,Z2,L0,K0,R0,Z0,PHI0,PHI1,PHI2,PHIAV
      1/BLK2/LA0,KA0,RA0,ZA0,THETA0,LA2,KA2,RAXIS,ZAXIS,KK,THETA,IA5,IA6
      1/INDX/I1,I2,I3,I4,I5,I6,IA,IB,ICM,N,SW,IG,IR,ARCFIN
       COMMON/EROR/ERROR
       DIMENSION PHI(NPT),R(NPT),Z(NPT)
       COMMON/STAGE/NPROB,IP


       IT(I)=I-2*(I/2)
       HFPI=1.5707963
       L,K=1
       LTOT=LA2-L1
       KTOT=KA2-K1
       IF(LTOT.LT.0)L=-1
       IF(KTOT.LT.0)K=-1
       KTOT=IABS(KTOT)
       L2=L1
       LTOT=IABS(LTOT)
       K2=K1
       IF(L1.NE.LA0)GOTO300
       SW(1)=2
       SW(2),SW(3),SW(4),SW(5)=1
240    LC=LA2
       IF(SW(5).EQ.2)LC=LC-L
C  SET INCREMENT IN POLAR ANGLE)-
       DTHETA=(1-2*IT(I4))*(THETA0-HFPI*(1+PARITY)/2)/LTOT
       ID=L1
CALCULATE COORDINATES ON LOGICAL SLANT LINE OF THE ARC
245    L2=L2+L
       K2=K2+K*IABS((1+K)/2-IT(ID))
       THETA=THETA+DTHETA
       R2=RAXIS*SIN(THETA)+RA0
       Z2=ZAXIS*COS(THETA)+ZA0
       CALL BSET(R,Z,PHI,TAG,NPT)
       IF(ERROR)RETURN
       ID=ID+1
       IF(L2.NE.LC)GOTO245
       IF(SW(1)-2)335,306,335
300    SW(1)=1
       SW(2),SW(3),SW(4),SW(5)=2
306    A=FLOAT(KTOT)-LTOT/2.
       DTHETA=(2*IT(I4)-1)*(THETA0-HFPI*(1-PARITY)/2)/A
       MX=KTOT-LTOT/2
       IF(SW(4).EQ.1)MX=MX-1
       IF(KK.GE.KA0.OR.KA2.GE.KA0)GOTO310
       MX=MX-(1-IT(LA0))*IT(L2)
       GOTO315
310    MX=MX-(1-IT(L2))*IT(LA0)
```

```
315   KC=K2+MX*K
      ID=0
      IF(SW(3).EQ.1)IF(MX)320,330,320
      IF(SW(6).EQ.2)IF(IT(L2))320,320,325
320     ID=1
CALCULATE COORDINATES ON ROW SECTION OF ARC
325   K2=K2+K
      IF(ID.EQ.0)DTHETA=DTHETA/2.
      THETA=THETA+DTHETA
      R2=RAXIS*SIN(THETA)+RA0
      Z2=ZAXIS*COS(THETA)+ZA0
      IF(ERROR)RETURN
      IF(ID.EQ.0)DTHETA=2.*DTHETA
      CALL BSET(R,Z,PHI,TAG,NPT)
330   IF(K2.NE.KC)GOTO320
      IF(SW(2)-2)335,240,335
335     I1,I2=1
      I5=IA5
      I6=IA6
      L2=GM(IG+3)
      K2=GM(IG+4)
      R2=GM(IG+5)
      Z2=GM(IG+6)
      CALL BSET(R,Z,PHI,TAG,NPT)
      IF(ERROR)RETURN
      IG=IG+3
      ARCFIN=1
      RETURN
      END


      SUBROUTINE SET REGION(REGC,NREG,NREGC,TAG,PHI,R,Z,NPT)
      INTEGER BIT,PARITY,ARCFIN,PK(6),TAG(NPT)
      LOGICAL ERROR
      COMMON/KLP/KMAX,LMAX,PARITY/EROR/ERROR
     1/INDX/I1,I2,I3,I4,I5,I6,IA,IB,IC,L,K,PK,IG,IR,ARCFIN
     1/BLK1/L1,L2,K1,K2,R1,R2,Z1,Z2,L0,K0,R0,Z0,PHI0,PHI1,PHI2,PHIAV
      DIMENSION PHI(NPT),R(NPT),Z(NPT)
      DIMENSION REGC(NREG,NREGC)
      IT(I)=I-2*(I/2)

      JR=REGC(IR,1)
55    L=0
      IA=1
      IB=IA+KMAX+1
56    K1,K2,I1,I2=0
      IF(IT(L))90,65,90
C  IS THE LOWER SIDE ON A BOUNDARY ?
65    IF(BIT(TAG(IB),4,0).GE.2)IF(I2)80,85,80
      I1=1
      IF(I2)75,70,75
80    I2=0
      GOTO70
85    I2=1
C  PACK NEW CELL NO)
75    CALL PACK(TAG(IB),JR,3)
C    ERASE ANY LOWER SIDE TAG.
70    IF(I1.EQ.0)TAG(IB)=BIT(TAG(IB),4,2)
C
      IB=IB+1
```

```
      IF(K1.EQ.KMAX)GOTO115
      I1=0
      K2=K2+1
C  IS  THE UPPER SIDE ON A BOUNDARY?
90    IF(BIT(TAG(IA),4,0).EQ.1)IF(I2)95,110,95

      I1=1
      IF(I2)105,100,105
95    I2=0
      GOTO100
110   I2=1
C  NEW TAG)
105   CALL PACK(TAG(IA),JR,2)
C    ERASE ANY UPPER SIDE TAG.
100   IF(I1.EQ.0)TAG(IA)=BIT(TAG(IA),4,1)
      IA=IA+1
      IF(K2.EQ.KMAX)GOTO115
      I1=0
      K1=K1+1
      GOTO65

115   IF(L.EQ.LMAX-1)GOTO120
      L=L+1

      IB=IB+1
      IA=IA+1
      GOTO56
C  SET AVERAGE POTENTIAL FOR REGION.
120   PHIAV=PHIAV/I3
       IF(PHIAV.EQ.0.)PHIAV=0.5
       DO 140 L=1,LMAX-1
       I6=1-IT(L)
       DO 140 K=1,KMAX-1
       IA=L*(KMAX+1)+K+1
       IB=(L+1)*(KMAX+1)+K+I6
       IC=(L-1)*(KMAX+1)+K+I6
      IF(BIT(TAG(IA),1,0).NE.0)GOTO140
CHECK IF ALL CELLS ASSOCIATED WITH AN INTERNAL POINT HAVE THE SAME TAG
      PK(1)=BIT(TAG(IA),2,0)
      PK(2)=BIT(TAG(IB),3,0)
      PK(3)=BIT(TAG(IA-1),2,0)
      PK(4)=BIT(TAG(IA-1),3,0)
      PK(5)=BIT(TAG(IC),2,0)
      PK(6)=BIT(TAG(IA),3,0)
      DO 135 I1=2,6
      IF(PK(1).NE.PK(I1))GOTO390
135   CONTINUE
       IF(PK(1).NE.IR)GOTO 140
C      GIVE PT.A REGN. NO.
C     SET APPROXN. TP POTL
       PHI(IA)=PHIAV
        R(IA),Z(IA)=0.
140    CONTINUE
       RETURN



390        WRITE(2,391)PK,L,K
391     FORMAT(25H1 TRIANGLE LABELING ERROR,6I10,8H AT ROW ,I2,
    18H COLUMN ,I2,1H.///)
```

```
500    ERROR=.TRUE.
      RETURN
      END
```

```
      SUBROUTINE MESH RELXN(REGC,NREG,NREGC,TAG,R,Z,NPT)
      LOGICAL PRINT
      INTEGER BIT,ARCFIN,PARITY,PK(6),TAG(NPT)
      COMMON/KLP/KMAX,LMAX,PARITY
     1 /BLK1/L1,L2,K1,K2,DOLDR,DOLDZ,DNEWR,DNEWZ,LO,KO,SUMR,SUMZ,EPSR
     1,EPSZ,DR,DZ/BLK2/EIGJACR,EIGJACZ,EIGSORR,EIGSORZ,ETOR,ETAOZ
     1/INDX/I,PRINT,ITN,I4,I5,I6,IA,IB,IC,L,K,PK,IG,IR,ARCFIN
      DIMENSION REGC(NREG,NREGC),R(NPT),Z(NPT),W(6)

      IT(I)=I-2*(I/2)
      CALL TIME(T1)
      WRITE(2,10)T1
10    FORMAT(1H1,30X,48(1H*)/30X,
     138H*  TIME ON ENTRY TO  MESH RELAXATION   ,A8,3H   */30X
     1,48(1H*)//)
      I1,I2=1
      PRINT=.FALSE.
      IMXH=0.25*NPT
      EPSCM=.00001
      EPSR,EPSZ=1.
      DOLDR,DOLDZ,EIGJACR,EIGJACZ,EIGSORR,EIGSORZ,ETAOR,ETAOZ=0.
       RHOR,RHOZ=1.

20    ITN=0
30    DNEWR,DNEWZ,SUMR,SUMZ,RESIDMXR,RESIDMXZ=0.
      DO 45 L=1,LMAX-1
      I5=IT(L)
      DO 40 K=1,KMAX-1
      IA=L*(KMAX+1)+K+1
CHECK IF POINTIS IN DUMMY REGION OR ON DIRICHLET BOUNDARY.
      IF(TAG(IA).EQ.0.OR.BIT(TAG(IA),1,0).NE.0)GOTO40
      IR=REGC(BIT(TAG(IA),2,0),4)
      I6=1
      IF(IR.EQ.1)I6=1-I5
      IB=(L+1)*(KMAX+1)+K+I6
      IC=(L-1)*(KMAX+1)+K+I6
      IF(IR.EQ.2) GO TO 31
      DR=RHOR*((R(IB+1)+R(IB)+R(IA-1)+R(IA+1)+R(IC+1)+R(IC))
     1         *.1666666667-R(IA))
      GOTO34
31    DR=RHOR*((R(IB)+R(IA-1)+R(IC)+R(IA+1))*0.25-R(IA))
34    R(IA)=R(IA)+DR
      SUMR=SUMR+R(IA)*R(IA)
      DNEWR=DNEWR+DR*DR
      IF(.NOT.PRINT)GOTO35
      DR=ABS(DR/R(IA))
      IF(DR.GT.RESIDMXR)RESIDMXR=DR
35    IF(IR.EQ.2)GOTO61
      DZ=RHOZ*((Z(IB+1)+Z(IB)+Z(IA-1)+Z(IA+1)+Z(IC+1)+Z(IC))
     1*.16666667-Z(IA))
      GOTO64
61    DZ=RHOZ*((Z(IB)+Z(IA-1)+Z(IC)+Z(IA+1))*0.25-Z(IA))
64    Z(IA)=Z(IA)+DZ
      SUMZ=SUMZ+Z(IA)*Z(IA)
      DNEWZ=DNEWZ+DZ*DZ
      IF(.NOT.PRINT)GOTO40
      DZ=ABS(DZ/Z(IA))

      IF(DZ.GT.RESIDMXZ)RESIDMXZ=DZ
40     CONTINUE
```

```
45      CONTINUE
        IF(PRINT)GOTO400
        ITN=ITN+1
        IF(ITN.GT.IMXM)GOTO390
        I=0
        IF(ITN.GT.1)CALL SOR(DNEWR,DOLDR,SUMR,RHOR,EIGJACR,EIGSORR,ETAOR
       1,EPSR)
        DOLDR=DNEWR
        IF(EPSR.GE.EPSCH)GOTO55
        I=I+1
 55          CONTINUE
        IF(ITN.GT.1)CALL SOR(DNEWZ,DOLDZ,SUMZ,RHOZ,EIGJACZ,EIGSORZ,ETAOZ
       1,EPSZ)
        DOLDZ=DNEWZ
        IF(EPSZ.GE.EPSCH)GOTO50
        I=I+1
50      CONTINUE
        ITN=ITN+1

        IF(I.LT.2)GO TO 30
390       PRINT=.TRUE.
        GO TO 30
400     WRITE(2,401)ITN,RESIDMXR,RESIDMXZ,RHOR,RHOZ,EPSR,EPSZ
401     FORMAT(45H0 TOTAL NO. OF ITERATIONS IN MESH RELXN IS         ,I4
       1,35H )MAXIMUM R AND Z RESIDUALS ARE.        ,2F12.6/
       135H THE FINAL OVER-RELXN FACTORS ARE.    ,2F12.6
       1,40H   AND THE CONVERGENCE FACTORS ARE.              ,2F12.6)
        CALL TIME(T1)
        WRITE(2,15)T1
15      FORMAT(1H0,30X,48(1H*)/30X,
       138H*  TIME ON EXIT FROM MESH RELAXATION    ,A8,3H  */30X
       1,48(1H*)//)
        RETURN
        END



        SUBROUTINE PARAM(REGC,NREG,NREGC,TAG,R,Z,CPU,CPR,CPL,SCT
       1,NPT,PHI)
          LOGICAL INT
        INTEGER BIT,ARCFIN,PARITY,PK(6),TAG(NPT)
         COMMON/ST/GM(500),TEMP(80)/KLP/KMAX,LMAX,PARITY
        COMMON/INDX/I1,I2,I3,I4,K1,K2,IA,IB,IC,L,MM,JJ(7),IR,INT
       1/BLK1/AINTEGRAL,M(2),RR,TRNGL,SOURCE,COEFF,NN(2),A,B,C,T4,HP,FPI3
       1/BLK2/LL(2),COTT(3),KK(2),AA(2)
        DIMENSION REGC(NREG,NREGC),CPU(NPT),CPR(NPT),CPL(NPT),SCT(NPT)
       1,R(NPT),Z(NPT),PHI(NPT)
C THE COUPLING COEFFICIENTS AND SOURCE TERMS ARE EVALUATED  'CELLWISE';
C EACH CELL CONTRIBUTES TO THREE ADJACENT COUPLINGS AND SOURCES
        IT(I)=I-2*(I/2)
C   NON-STANDARD
        FPI=1.
        FPI3=FPI/3.
        L=0
        IA=1
        IB=IA+KMAX+1
10      K1,K2=0
        IF(IT(L))20,20,50
CALCULATE CONTRIBUTION   FROM THE LOWER CELL AT POINT  IB
20      IF(TAG(IB).NE.0)CALL TERMS(CPU,CPR,CPL,SCT,TAG,REGC,NREG,NREGC,
```

```
     1NPT,IB,IA,1,R,Z,PHI)

      IB=IB+1
      IF(K1.EQ.KMAX)GO TO 55
      K2=K2+1
CALCULATE THE CONTRIBUTION FROM UPPER CELL AT  IA.
50    IF(TAG(IA).NE.0)CALL TERMS(CPU,CPR,CPL,SCT,TAG,REGC,NREG,NREGC,
     1NPT,IA,IB,2,R,Z,PHI)
      IA=IA+1
      IF(K2.EQ.KMAX)GO TO 55
      K1=K1+1
      GOTO20
55    IF(L.GE.LMAX-1)RETURN
      L=L+1
      IA=IA+1
      IB=IB+1
      GOTO10
      END

      SUBROUTINE TERMS(CPU,CPR,CPL,SCT,TAG,REGC,NREG,NREGC,
     1NPT,I,J,K,R,Z,PHI)
      INTEGER BIT,ARCFIN,PARITY,PK(6),TAG(NPT)
      LOGICAL AXMAG,AXHEAT,INT
      COMMON/INDX/I1,I2,I3,I4,K1,K2,IA,IB,IC,L,MM,JJ(7),IR,INT
      COMMON/TP/TYPE,AXMAG,AXHEAT
     1/KLP/KMAX,LMAX,PARITY
     1/BLK1/AINTEGRAL,M(2),RR,TRNGL,SOURCE,COEFF,NN(2),A,B,C,T4,HP,FPI3
     1/BLK2/LL(2),COTT(3),KK(2),AA(2)
      DIMENSION REGC(NREG,NREGC),CPU(NPT),CPR(NPT),CPL(NPT),SCT(NPT)
     1,R(NPT),Z(NPT),PHI(NPT)
      IT(I)=I-2*(I/2)
      I1=IT(K)
      I2=I+1
      I3=2+I1

C     FIND REGION NUMBER OF CELL
      IR=BIT(TAG(I),13,0)
      IF(IR.EQ.0)RETURN
      RR=1.
      TRNGL=0.5*PARITY*(R(IB)*(Z(IA)-Z(I2))+R(I2)*(Z(IB)-Z(IA))
     1+R(IA)*(Z(I2)-Z(IB)))
      IF(TRNGL.LE.1.E-12)GOTO60
      IF(REGC(IR,3).LE.0.)GOTO50

      SOURCE=FPI3*TRNGL*REGC(IR,3)
      SCT(IA)=SCT(IA)+SOURCE
      SCT(IB)=SCT(IB)+SOURCE
      SCT(I2)=SCT(I2)+SOURCE
50    IF(REGC(IR,2).LE.1E-10)RETURN
      COEFF=0.125*REGC(IR,2)/TRNGL
C    MODIFICATIONS FOR AXIAL SYMMETRY(E.G-HEAT CONDUCTION AND MAGNETIC)
      IF(AXMAG)COEFF=COEFF*3./(R(IA)+R(IB)+R(I2))
      IF(AXHEAT)COEFF=COEFF*(R(IA)+R(I2)+R(IB))/3.
      A=(R(IA)-R(IB))**2+(Z(IA)-Z(IB))**2
      B=(R(IB)-R(I2))**2+(Z(IB)-Z(I2))**2
      C=(R(IA)-R(I2))**2+(Z(IA)-Z(I2))**2
      COTT(1)=(B+C-A)
      COTT(2)=(A+B-C)
      COTT(3)=(A+C-B)
CONTRIBUTIONS TO COUPLINGS(E.G- PERMITTIVITY*COTANGENT OF ANGLE)
```

```
      CPU(IA)=CPU(IA)*COEFF*COTT(I1+1)
      CPR(I)=CPR(I)+COEFF*COTT(I3)
      CPL(IB)=CPL(IB)+COEFF*COTT(3-2*I1)
      RETURN
60    WRITE(2,65)
65    FORMAT(40H0   TRIANGLE AREA ZERO OR NEGATIVE          )
      RETURN
      END


      SUBROUTINE RELX(TAG,CPU,CPR,CPL,SCT,PHI,R,Z,NPT)
      LOGICAL PRINT
      INTEGER TAG(NPT),PARITY,BIT
      COMMON/INDX/PRINT,IRCNT,IA,IB,IC,L,K,I,ITN
     1/KLP/KMAX,LMAX,PARITY
     1/BLK1/DOLD,DNEW,SUM,EPS,WOPT,DPHI,CPSUM
     1/BLK2/EIGJAC,EIGSOR,ETAO
      DIMENSION CPU(NPT),CPR(NPT),CPL(NPT),SCT(NPT),PHI(NPT)
     1,R(NPT),Z(NPT)
      IT(I)=I-2*(I/2)
      IMX=0.25*NPT
      PRINT=.FALSE.
      EPS=1.
      EPSC=.000001
      DOLD,ETAO,EIGJAC,EIGSOR=0.
      RHOI=1.5
      WOPT=1.5
      CALL TIME(T1)
      WRITE(2,400)TI,RHOI
400    FORMAT(1H0,' ENTRY AT  ',A8,'TO RELAXATION SOLUTION OF POISSON P'
     1,'ROBLEM IN A NON-UNIFORM TRIANGULATION, INITIAL VALUE OF ACCELERA'
     1,'TION FACTOR TAKEN AS        ',F6.3)
      ITN=0
5     DNEW,SUM=0.
       DO 100 L=1,LMAX-1
       I=1-IT(L)
       IA=(KMAX+1)*L+1
       IB=(KMAX+1)*(L+1)+I
       IC=(KMAX+1)*(L-1)+I
10      IF(PRINT)WRITE(2,70)L
70    FORMAT(1H0,11(1H*)/6H *ROW ,I3,2H */ 1H ,11(1H*)//6X,1HR,8X,
     11HZ,10X,'RESIDUAL     POTENTIAL      MESH POINT      COLUMN   TAG'//)
       DO 90 K=1,KMAX-1
       IA=IA+1
       IB=IB+1
       IC=IC+1
      IF(TAG(IA).EQ.0)GOTO90
      IF(BIT(TAG(IA),1,0).EQ.2)GOTO90
      CPSUM=CPU(IA)+CPL(IB)+CPR(IA-1)+CPU(IC)+CPL(IA)+CPR(IA)
      DPHI=WOPT*((CPU(IA)*PHI(IB+1)+CPL(IB)*PHI(IB)+CPR(IA-1)*PHI(IA-1)
     1+CPU(IC)*PHI(IC)+CPL(IA)*PHI(IC+1)+CPR(IA)*PHI(IA+1)
     1+SCT(IA))/CPSUM-PHI(IA))
      PHI(IA)=PHI(IA)+DPHI
      SUM=SUM+PHI(IA)*PHI(IA)
      DNEW=DNEW+DPHI*DPHI
      IF(.NOT.PRINT)GOTO90
      IF(PRINT)WRITE(2,40)R(IA),Z(IA),DPHI,PHI(IA),IA,K,TAG(IA)
40    FORMAT(1X,F8.3,1X,F8.3,5X,F12.6,5X,F12.6,5X,I6,5X,I6,5X,I6)
90     CONTINUE
```

```
100     CONTINUE
105       IF(PRINT)GOTO50
      IF(ITN.NE.0)CALL SOR(DNEW,DOLD,SUM,WOPT,EIGJAC,EIGSOR,ETAO,EPS)
      DOLD=DNEW
      WRITE(2,200)ITN,WOPT,EPS,EIGJAC,EIGSOR
200   FORMAT(2X,I3,5X,F12.6,5X,F12.6,10X,F12.6,5X,F12.6)
      IF(EPS.LT.EPSC)GOTO0150
      ITN=ITN+1
      IF(ITN.LT.IMX)GOTO5
150    PRINT=.TRUE.
      GOTO5
50    CALL TIME(T1)
      WRITE(2,55)T1
55    FORMAT(1H0,30X,48(1H*)/30X,'*  TIME ON EXIT FROM  RELAXATION
     1',A8,3H  */30X,48(1H*)//)
      RETURN
      END

       SUBROUTINE SOR(DNEW,DOLD,SUM,WOPT,LAMDA,MU,MUO,EPS)
       REAL LAMDA,MU,MUO
       DATA DIFF,RHOO,BETA/0.01,0.01,0.05/
       EPS=SQRT(DNEW/SUM)
       MU=SQRT(DNEW/DOLD)
       IF(ABS(MU-MUO).GE.DIFF) GOTO 50
       IF(MU.LE.WOPT-1.) GOTO 50
       LAMDA=(MU+WOPT-1.)/(WOPT*SQRT(MU))
       IF(LAMDA.LT.1.)WOPT=BETA*(2./(1.+SQRT(1.-LAMDA*LAMDA))-RHOO)
     1+(1.-BETA)*WOPT
50     MUO=MU
       RETURN
       END


      SUBROUTINE PLOTT(R,Z,PHI,TAG,NPT,NEQ,PLOT MESH)
      LOGICAL PLOT MESH ,FIRST
      INTEGER BIT,TAG(NPT),PARITY
      COMMON/BLK2/L1,L2,K1,K2,R1,R2,Z1,Z2,L0,K0,R0,Z0,PHI0,PHI1,PHI2
      COMMON/STAGE/NPROB,IP
     1/KLP/KMAX,LMAX,PARITY
      DIMENSION R(NPT),Z(NPT),PHI(NPT),EQ(20)
      IT(I)=I-2*(I/2)
      FIRST=.FALSE.
      DO 9 I=1,NPT
      IF(TAG(I).EQ.0)GOTO9
      IF(FIRST)GOTO5
      PHIMIN,PHIMAX=PHI(I)
      ZMIN,ZMAX=Z(I)
      RMIN,RMAX=R(I)
      FIRST=.TRUE.
      GOTO9
5     PHIMIN=AMIN1(PHIMIN,PHI(I))
      PHIMAX=AMAX1(PHIMAX,PHI(I))
      ZMIN=AMIN1(ZMIN,Z(I))
      RMIN=AMIN1(RMIN,R(I))
      ZMAX=AMAX1(ZMAX,Z(I))
      RMAX=AMAX1(RMAX,R(I))
9       CONTINUE
      WRITE(2,199)
199    FORMAT(47H  MAXIMA AND MINIMA OF PLOTTING  QUANTITIES     )
```

```
      WRITE(2,200)ZMIN,RMIN,ZMAX,RMAX,PHIMIN,PHIMAX
200     FORMAT(45H  ZMIN   RMIN   ZMAX   RMAX   PHIMIN   PHIMAX/
     14F7.3,2F9.3)
      DPHI=(PHIMAX-PHIMIN)/NEQ
      WRITE(2,201)NEQ
201     FORMAT(18H NO. OF EQUPTLS IS,I3)
      NEQ1=NEQ+1
      DO 13 I2=1,NEQ1
13    EQ(I2)=PHIMAX-(I2-1)*DPHI
      L=0
6     DO 15 K=1,KMAX
      IA=L*(KMAX+1)+K
15    CALL LINET(2,IA,IA+1,R,Z,TAG,NPT,PLOT MESH)
20    IF(L-LMAX)30,25,25
25    RETURN
30    K1=K2=KMAX
      IA=(L+1)*(KMAX+1)
      IB=IA+KMAX+1
      IF(IT(L))80,35,80
35    CALL LINET(3,IA,IB,R,Z,TAG,NPT,PLOT MESH)
      K1=K1-1
      IF(K1)120,40,40
40    IF(DPHI.GT.0..AND.BIT(TAG(IA-1),270).NE.0)
     1CALL EQUPLT(IA-1,IA,IB,PHI,R,Z,NPT,EQ,NEQ1)
   70 IA=IA-1
80    CALL LINET(4,IA,IB,R,Z,TAG,NPT,PLOT MESH)

      K2=K2-1
      IF(K2)120,81,81
81    IF(DPHI.GT.0..AND.BIT(TAG(IB-1),370).NE.0)
     1CALL EQUPLT(IA,IB-1,IB,PHI,R,Z,NPT,EQ,NEQ1)
110   IB=IB-1
      GOTO35
120   L=L+1
      GOTO6
      END


        SUBROUTINE EQUPLT(I,J,K,PHI,R,Z,NPT,EQ,NEQ1)
      INTEGER PSENTL
      COMMON/BLK2/L1,L2,K1,K2,R1,R2,Z1,Z2,L0,K0,R0,Z0,PHI0,PHI1,PHI2
      DIMENSION R(NPT),Z(NPT),PHI(NPT),EQ(NEQ1)
      Z0=Z(I)
      R0=R(I)
      Z1=Z(J)
      R1=R(J)
      Z2=Z(K)
      R2=R(K)
      PHI0=PHI(I)
      PHI1=PHI(J)
      PHI2=PHI(K)
45    IF(PHI0-PHI1)50,55,55
50    CALL INTCHN(R0,Z0,R1,Z1,PHI0,PHI1)
55    IF(PHI1-PHI2)65,60,60
65    CALL INTCHN(R1,Z1,R2,Z2,PHI1,PHI2)
      GOTO45
60    DO 75 I2=1,NEQ1
       JJ=I2+1
      PHIS=EQ(I2)
```

```
      CALL INTCPT(PHIS,RA,ZA,RB,ZB,PSENTL)
      IF(PSENTL)74,75,74
74    CALL LINE(JJ,ZA,RA,ZB,RB)
75    CONTINUE
      RETURN
      END



      SUBROUTINE INTCPT(PHIS,RA,ZA,RB,ZB,IPSENTL)
      COMMON/BLK2/II(4),R1,R2,Z1,Z2,JJ(2),R0,Z0,PHI0,PHI1,PHI2,PHIAV
      IF(PHIS-PHI0)10,10,20
10    IF(PHIS-PHI2)20,25,30
25    P=0.
      GOTO35
30    P=(PHIS-PHI2)/(PHI0-PHI2)
35    ZA=Z2+P*(Z0-Z2)
      RA=R2+P*(R0-R2)
      IF(PHIS-PHI1)40,45,50
40    P=(PHIS-PHI2)/(PHI1-PHI2)
      ZB=Z2+P*(Z1-Z2)
      RB=R2+P*(R1-R2)
      GOTO60
45    ZB=Z1
      RB=R1
      GOTO60
50    P=(PHIS-PHI1)/(PHI0-PHI1)
      ZB=Z1+P*(Z0-Z1)
      RB=R1+P*(R0-R1)
60    IPSENTL=1
      RETURN
20    IPSENTL=0
      RETURN
      END



      SUBROUTINE LINET(INDX,I,J,R,Z,TAG,NPT,PLOT MESH)
       LOGICAL PLOT MESH
       INTEGER TAG(NPT),BIT
      DIMENSION R(NPT),Z(NPT)
      IT(I)=I-2*(I/2)
CHECK FOR POINTS IN DUMMY REGION.
      IF(TAG(I).EQ.0.OR.TAG(J).EQ.0)RETURN
      IFUNC=I-IT(INDX)
      WRITE(2,800) NPT,I,J,INDX,IFUNC
800   FORMAT(1H ,5I10)
      IU=BIT(TAG(I-IT(INDX)),2,0)
      WRITE(2,800) IU
      K=I
      IF(INDX.GT.2)K=J-1+IT(INDX)
      IL=BIT(TAG(K),3,0)
C      DRAW  A MESH LINE.
      IF(PLOT MESH.OR.(IU.NE.IL))CALL LINE(1,Z(I),R(I),Z(J),R(J))
      RETURN
      END


      SUBROUTINE INTCHN(RA,ZA,RB,ZB,PHIA,PHIB)
```

```
      Z=ZA
      R=RA
      PHI=PHIA
      ZA=ZB
      RA=RB
      PHIA=PHIB
      ZB=Z
      RB=R
      PHIB=PHI
      RETURN
      END
      SUBROUTINE LINE(I,Z1,R1,Z2,R2)
      WRITE(7,10)Z1,R1,Z2,R2,I
      WRITE(2,10)Z1,R1,Z2,R2,I
10    FORMAT(4(F8.3),2X,I2)
      RETURN
      END
```

APPENDIX 4.2

TMG - DISPLAY PROGRAM

```
C                          ****************
C                          * APPENDIX 4.2 *
C                          ****************
C
C THIS IS THE DISPLAY PROGRAM USED IN PLOTTING THE TRIANGULAR
C MESH AND EQUIPOTENTIAL LINES OF THE  PROBLEM REGION.
C IT GENERATES THE FOLLOWING DISPLAYS:-
C       1.INTRODUCTORY AND DATA ENTRY
C       2.TRIANGULAR MESH AND EQUIPOTENTIAL LINES
C       3.ZOOM
C
C
C
C *********** MAIN PROGRAM SEGMENT *************
        COMMON/IO/IN,IOUT
        INTEGER FN(3)
        IN=5
        IOUT=6
        CALL TXOPEN
C INTRODUCTORY DISPLAY
3       CALL DISP1(FN,IC)
        GOTO(1,2,2),IC
C TRIANGULAR MESH DISPLAY
1       CALL DISP2(FN,IC,IZ)
        GOTO(2,3,2,2,2,2,2),IC
2       STOP
        END
C
C************INTRODUCTORY DISPLAY &DATA ENTRY *********
        SUBROUTINE DISP1(FILNAM,NEXT)
        COMMON/IO/IN,IOUT
        COMMON/SCALING/SCL1,SCL2,SCL3,SCL4
        DATA MNTXT/"+ NEXT    + RESTART + EXIT    "/
        LOGICAL*1 MNTXT(30)
        INTEGER FILNAM(3)
15      CALL TXCLER
C SET UP INSTRUCTION TO THE USER AND THE MENU OPTIONS
        CALL TEXTUP("TXTFL1",25)
        CALL MNOPEN(875.,715.,1)
        CALL MNDISP(MNTXT,3,10,1)
        CALL FRAME(870.,733.,3)
        CALL CURPOS(1.,150.)
        CALL ALPHMD
C INPUT DATA FILE NAME
20      CALL MESSAG("£ DATA FILE NAME?^")
        READ(IN,30)FILNAM
30      FORMAT(2A4,A2)
40      CALL MESSAG("£ ENTER PLOTTING SCALE LIMIT(MIN.&MAX.OF X,Y)?^")
        IF(IERROR(110).NE.0)GOTO 40
        READ(IN,50)SCL1,SCL2,SCL3,SCL4
50      FORMAT(4G0.0)
2       CALL MNPICK(J,ICHAR,MNO)
5       CALL CONFRM(ICHAR)
        IF(ICHAR.EQ.78) GOTO 2
        IF(ICHAR.NE.89) GOTO 5
        GOTO(60,15,80),J
60      NEXT=J
        RETURN
80      CALL ALPHMD
```

```
      STOP
      END
C
C***********DISPLAY 2 FOR PLOTTING ****************
      SUBROUTINE DISP2(FILNAM,NEXT,IZOOM)
      COMMON/IO/IN,IOUT
      COMMON/SCALING/SCL1,SCL2,SCL3,SCL4
      COMMON/ZOMSCL/ZSCL1,ZSCL2,ZSCL3,ZSCL4
      DATA MNTXT/"+ NEXT    + PREVIOUS+ EQUIP'L + MESH    + ZOOM
&+ RESTART + EXIT    "/
      DIMENSION XEQP1(1000),YEQP1(1000),XEQP2(1000),YEQP2(1000)
      LOGICAL*1 MNTXT(70)
      INTEGER FILNAM(3),FIRST
C FIRST TIME DATA FILE IS READ
      FIRST=1
1     CALL TXCLER
      IZOOM=0
      WRITE(IOUT,70)
70    FORMAT("TRIANGULAR MESH GENERATION:-")
      CALL MNOPEN(875.,715.,1)
      CALL MNDISP(MNTXT,7,10,1)
      CALL FRAME(870.,733.,7)
2     CALL LMTARA
      CALL MNPICK(J,ICHAR,MNO)
5     CALL CONFRM(ICHAR)
      IF(ICHAR.EQ.78)GOTO 2
      IF(ICHAR.NE.89) GOTO 5
      GOTO(21,21,23,22,24,1,21),J
      GOTO 2
21    NEXT=J
      RETURN
C GENERATE MESH
22    K=1
      FIRST=2
      NEQP=0
      REWIND 9
      CALL SETFIL(9,FILNAM)
      CALL LMTSCL(SCL1,SCL2,SCL3,SCL4)
11    IF(IERROR(110).NE.0) GOTO 200
C INPUT LINE COORDINATES FROM THE DATA FILE
      READ(9,10)Z1,R1,Z2,R2,IEQP
10    FORMAT(4(F8.3),2X,I2)
      IF(IEQP.EQ.1) GOTO 20
C SAVE EQUIPOTENTIAL LINES COORDINATE
      XEQP1(K)=Z1
      YEQP1(K)=R1
      XEQP2(K)=Z2
      YEQP2(K)=R2
      K=K+1
      GOTO 11
20    IF(J.EQ.3) GOTO 11
      CALL TXMOVE(Z1,R1)
      CALL TXDRAW(Z2,R2)
      GOTO 11
200   IF(J.EQ.3) GOTO 23
      IZOOM=IZOOM+2
      GOTO 2
C GENERATE EQUIPOTENTAIL LINES
23    IF(FIRST.EQ.1) GOTO 22
      CALL LMTSCL(SCL1,SCL2,SCL3,SCL4)
      IZOOM=IZOOM+1
      K=K-1
      DO 100 I=1,K
       CALL TXMOVE(XEQP1(I),YEQP1(I))
       CALL TXDRAW(XEQP2(I),YEQP2(I))
100   CONTINUE
```

```
        GOTO 2
        RETURN
C ZOOMING
24      CALL LMTSCL(SCL1,SCL2,SCL3,SCL4)
C INPUT THE COORDINATES OF THE ZOOMING WINDOW
        CALL TXCURS(XZ1,YZ1,ICHAR)
36      CALL TXCURS(XZ2,YZ2,ICHAR)
C CHECK FOR TWO OPPOSITE CORNERS OF THE ZOOMED WINDOW
        IF(XZ1.EQ.XZ2.OR.YZ1.EQ.YZ2)GOTO 36
C DRAW ZOOMED WINDOW
        CALL TXMOVE(XZ1,YZ1)
        CALL TXDRAW(XZ2,YZ1)
        CALL TXDRAW(XZ2,YZ2)
        CALL TXDRAW(XZ1,YZ2)
        CALL TXDRAW(XZ1,YZ1)
        ZSCL1=AMIN1(XZ1,XZ2)
        ZSCL2=AMIN1(YZ1,YZ2)
        ZSCL3=AMAX1(XZ1,XZ2)
        ZSCL4=AMAX1(YZ1,YZ2)
        CALL LMTARA
32      CALL CONFRM(ICHAR)
        IF(ICHAR.EQ.78) GOTO 24
        IF(ICHAR.NE.89) GOTO 32
C CALL ZOOM ROUTINE
        CALL ZOOM(FILNAM,IC,IZOOM,K,XEQP1,YEQP1,XEQP2,YEQP2)
        GOTO(1,42,42),IC
C PREVIOUS
42      STOP
        END
/*      */
/*      */
/* THIS C-PROGRAM READS IN THE PAPER TAPE PRODUCED ON THE ICL 1904S */
/* MACHINE AND STORE ITS DATA ON FILE SPECIFIED BY THE USER USING...*/
/* UNIX INPUT COMMAND:                          */
/*           %PRTAPE(/DEV/PR)AFILE                              */
/* WHERE 'PRTAPE' IS THE NAME OF THE EXCUTABLE MODULE OF THIS PROGRAM*/
/* AND AFILE IS THE NAME OF THE FILE INTO WHICH THE DATA WILL BE */
/* STORED*/
        MAIN(){
        INT I;
        CHAR C;
        WHILE(1){
                I=READ(0,&C,1);
                IF(C != 0) BREAK;
        }
        C =& 0177;
        WRITE(1,&C,1);
        WHILE(1){
                I = READ(0,&C,1);
/* END OF FILE */
                IF(I == '\0')BREAK;
/* STRIP THE PARITY BIT FROM THE ICL PAPER TAPE */
                C =& 0177;
/* NOCKS OUT THE NULL CHARACTERS WHICH SIGNIFY END OF STRING  */
/* UNDER UNIX */
                IF(C == '\0')CONTINUE;
                I = WRITE(1,&C,1);
        }
        }
```