COMPUTER AIDED DESIGN AND MANUFACTURE FOR THREE DIMENSIONAL MILLING

by

ASHRAF ABD ELHALIM BADR

A thesis submitted for the Degree of
DOCTOR OF PHILOSOPHY

to

THE UNIVERSITY OF ASTON IN BIRMINGHAM

Department of Mechanical and Production Engineering

MAY 1986

DECLARATION

THE WORK DESCRIBED IN THIS THESIS HAS BEEN CARRIED OUT

INDEPENDENTLY AND SUBMITTED FOR NO OTHER DEGREE.

A. BADR.

THE UNIVERSITY OF ASTON IN BIRMINGHAM

COMPUTER AIDED DESIGN AND MANUFACTURE FOR THREE DIMENSIONAL MILLING

Ashraf Abd Elhalim Badr

A Thesis submitted for the Ph.D Degree 1986

SUMMARY

Advances in both computer technology and the necessary mathematical models capable of capturing the geometry of arbitarily shaped objects has led to the development in this Thesis of a surface generation package called 'IBSCURF' aimed at providing a more economically viable solution to free-form surface manufacture.

A suit of computer programs written in FORTRAN 77 has been developed to provide computer aids for every aspect of work in designing and machining free-form surfaces. A vector-valued parametric method was used for shape description and a lofting technique employed for the construction of the surface.

The development of the package 'IBSCURF' consists of two phases. The first deals with CAD. The design process commences in defining the cross-sections which are represented by uniform B-spline curves as approximations to give polygons. The order of the curve and the position and number of the polygon vertices can be used as parameters for the modification to achieve the required curves. When the definitions of the sectional curves is complete, the surface is interpolated over them by cubic cardinal splines. To use the CAD function of the package to design a mould for a plastic handle, a mathematical model was developed.

To facilitate the integration of design and machining using the mathematical representation of the surface, the second phase of the package is concerned with CAM which enables the generation of tool offset positions for ball-nosed cutters and a general post-processor has been developed which automatically generates NC tape programs for any CNC milling machine. The two phases of these programs have been successfully implemented, as a CAD/CAM package for free-form surfaces on the VAX 11/750 super-minicomputer with graphics facilities for displaying drawings interactively on the terminal screen. The development of this package has been beneficial in all aspects of design and machining of free form surfaces.

KEYWORDS: Computer Aided Design, Computer Aided Manufacture, Mathematical Model, B-Splines, N.C Part Programming

ACKNOWLEDGEMENTS

I wish to express my gratitude to everyone involved in the successful completion of this research. In particular, I would like to thank:

Dr D.A. Milner, for his continuous help, guidance, advice and encouragement as supervisor throught the research project.

The University of Aston in Birmingham and the Science and Engineering Research Council, for financially supporting me throughout the course with a grant.

Professor R.H. Thornley, for permitting this research to be carried out in the Production Division of the Department of Mechanical and Production Engineering.

Last, but not least, Iloria for typing the thesis.

Special thanks go to my wife for her assistance, patience and continuous understanding.

CONTENTS

		Page
Summary		i
Acknowledgements		ii
List of Figures, Plates, Tables and Charts		x
CHAPTER 1		
1. INTRODUCTION		1
1.1	Computer Aided Design and Manufacture - State of the Art	1
1.2	Industrial Application	2
1.3	Research Objectives	3
CHAPTER 2		
2.	GENERAL VIEW OF CAD/CAM GEOMETRIC MODELLING DEVELOPMENT	4
2.1	Wire Frame Modelling	4
2.2	Surface Modelling	5
2.2.1	Sculptured Surface Modelling	6
2.3	Solid Modelling	7
2.3.1	Idealised Model	9
2.3.2	Approximated Model Faceted Model	9
2.3.3	Complete Model	10
2.3.3.1	Boundary Representation (B-rep)	10
2.3.3.2	Constructive Solid Geometry (C.S.G)	10

Page

CHAPTER 3

3. A REVIEW OF DEVELOPMENTS IN NUMERICAL CONTROL 11

3.1 Milestones in Numerical Control 11

3.2 Introduction to Numerical Control 12

3.3 Methods of Tool Positioning Control 12

3.4 Classification of NC Machine Tools 12
a) Positional (point to point) 12
b) Paraxial (straight line) 12
c) Continuous path (contouring) 13

3.5 Types of Existing Control Systems 13
3.5.1 Conventional NC of Malchine Tools 13
3.5.2 Computer Numerical Control of Machine Tools 13
3.5.3 Direct Numerical Control of Machine Tools 14
3.5.3.1 DNC Behind the Tape Reader System 14
3.5.3.2 DNC Machine Control Unit (MCU) 15
3.5.4 Microprocessor Numerical Control of Machine Tools 15

3.6 Integrated Manufacturing Systems 16

CHAPTER 4

4. A REVIEW OF COMPUTER AIDED DESIGN OF FREE FORM SURFACES

4.1 Considerations of design 25
4.1.1 Traditional Approach to Design 25
4.1.2 Computer Aided Design 25

4.2 CAD hardware 26
4.2.1 Interactive Devices 26
4.2.2 Types of Graphics Devices 27
4.2.2.1 Cathode Ray Tube Graphics Devices 27
4.2.2.2 Plotters 28

Page

4.2.3 Classification of Graphics Devices 29

4.3 Computer Aided Design Software 30
4.3.1 Introduction to Numerical Geometry 30
4.3.2 Parametric Description of curves and surfaces 31
4.3.3 Curve Design 32
4.3.3.1 Ferguson Cubic Curve 33
4.3.3.2 Spline Functions 36
4.3.3.3 B-spline Curve 39
4.3.3.4 Bezier Curves 39
4.3.4 Surface Design 42
4.3.4.1 Coons Patches 42
4.3.4.2 Ferguson Surface 48
4.3.4.2.1 APT Surface Fitting Routine 50
4.3.4.3 Bezier Surfaces 50

CHAPTER 5

5. DEVELOPMENT OF THE COMPUTER AIDED SURFACE DESIGN PACKAGE "IBCSURF"

5.1 Introduction to 'IBCSURF' 60

5.2 Mathematical Basis of 'IBCSURF' 61
5.2.1 Definition of Spline Functions 61
5.2.2 B-Spline Basis 61
5.2.2.1 B-Spline versus Bernstein Basic Function 62
5.2.2.2 Evaluation of B-Spline Basis Function 63
5.2.3 B-Spline Curve 65
5.2.4 Interpolating with B-splines 66
5.2.4.1 Inversion Technique 67
5.2.5 Cubic Cardinal Splines 69

5.3 Curve Design Procedure 72

5.4 Curve Fitting 74
5.4.1 Least Squares B-spline Curve Fitting 74

Page

5.4.2 Implementation of Inversion Technique in Curve Fitting 76

5.5 Surface Design Procedure 77

5.6 Justification of Approach 79

5.7 Method to be used for Machining 82
5.7.1 Analytical Method of Computing Normals 83
5.7.2 Numerical Method of Calculating Tool Offset 85

5.8 Interference Checking 87
a) ANAOFF 87
b) NUMOFF 88

CHAPTER 6

6. THE MATHEMATICAL MODEL OF THE HANDLE

6.1 Introduction 95
6.1.2 Program "SECT" 95
6.1.2.1 Stage One 95
6.1.2.2 Stage Two 96
6.1.2.3 Stage Three 100
6.1.2.4 Intersection Points 102
6.1.2.5 Intersection of Circle with Circle 103
6.1.2.6 Intersection of line inclined at angle not equal to II/2 with Circle Radius R 104
6.1.2.7 Calculation of the Co-ordinates Z_2, Z_3, Z_4 105

CHAPTER 7

7. SYSTEM OVERVIEW

7.1 Introduction 110

Page

7.2.1 Software Objectives 111

7.3 The Computer Workstation 112

7.3.1 Design for Portability 112

7.4 Starrag Rigid Milling Machine 113

7.4.1 Control System 114

7.4.2 Tally 464 Paper Tape Reader 115

7.5 Bridgeport CNC Machine 115

7.5.1 Machine Concept 115

7.5.2 Brief Description of Control 116

7.6 N.C. Part-Program Pattern 117

7.7 Software Functions 117

7.8 The Processing Commands 120

CHAPTER 8

8. COMPUTER AIDED DESIGN SOFTWARE FINCTIONS

8.1 Introduction 124

8.2 Program 'BSLSQ' 124

Page

8.4 Program 'BCSURFI' 125

8.5 Program 'INVERT' 125

8.6 Program 'BSCURV' 125

8.7 Program 'BCSURF' 125

8.8 Design using 'IBCSURF' Package 126

8.8.1 Curve Design with 'BSCURV' 126

8.8.2 Flexibility of 'BSCURV' in Shape Control 126

8.8.3 Curve Fitting 128

8.8.4 Surface Design 129

CHAPTER 9

9. COMPUTER AIDED MANUFACTURE FUNCTIONS

9.1 Introduction 146

9.2 Program 'BCSURN' 147

9.3 Program 'NUMOFF' 147

9.4 Program 'ANAOFF' 147

9.5 Post-processor 147

9.5.1 Introduction 147

9.5.2 Post-processor Program 'PREPARE 3D' 148

Page

9.6 Computer Aided Machining with 'IBCSURF' 152

9.6.1 Flow Data in the System 153

9.6.2 Practical Applications 154

CHAPTER 10

10. CONCLUSION AND FUTURE WORK 161

10.1 Conclusion and Remarks 161

10.2 Suggestions for Future Work 167

APPENDICES 169

Appendix 1 169

(a) Cross Sectional Curve for Upper and Lower of the Mould

(b) Top and Bottom Mould for the Plastic Handle viewed from different angles

Appendix 2 194

The Packaged Programs

REFERENCES 284

LIST OF FIGURES

		Page
Fig. (3.1)	Conventional Method of Defining Linear and Rotary Displacement of Machine Tools.	18
Fig. (3.2)	Closed Loop Displacement Control	19
Fig. (3.3)	Open Loop Displacement Control	19
Fig. (3.4)	Block Diagram of a Typical Hard-Wired NC System	20
Fig. (3.5)	Types of NC Malchine Tools and Equipment	20
Fig. (3.6)	Block Diagram of CNC Systems (a) Full Computer Control (b) System Incorporating Hard-Wired Interpolator	21
Fig. (3.7)	Block Diagram of a DNC Behind the Tape Reader System	22
Fig. (3.8)	A DNC Machine Control Unit System	23
Fig. (3.9)	MCU Modified Minimum Cost DNC System	23
Fig. (3.10)	Function Diagram of a Decentralised Approach in Design of NC Systems	24
Fig. (3.11)	Flexible Machining Cells in an Integrated Manufacturing System	25
Fig. (4.1)	Block Diagram of a Computer Aided Design System	54
Fig. (4.2)	Parametric Method in Curve Description	55
Fig. (4.3	Parametric Method in Surface Description	55
Fig. (4.4)	The Effect of End Tangents on Ferguson's Curve	56

		Page
Fig. (4.5)	Construction of Spline Curves	56
Fig. (4.6)	A Bezier Curve Segment	57
Fig. (4.7)	Continuity of Bezier Curve Segments	57
Fig. (4.8)	A Parametric Surface Patch	58
Fig. (4.9)	The Characteristic Polyhedron for a Cubic Bezier Patch	58
Fig. (4.10)	Characteristic Polyhedra for Construction of Composite Surfaces with Positional and Slope Continuity	59
Fig. (5.1)	B-spline Basis Functions for Different Degrees	91
Fig. (5.2)	B-splines for Periodic and Non-Periodic Curves	91
Fig. (5.3)	Surface Description Method	92
Fig. (5.4)	Equal Parametric Distances with Uneven Distribution	92
Fig. (5.5)	The Effect of Vertex Numbering on Parametrisation	93
Fig. (5.6)	Tool Offset on a Multi-faced Polyhedron	93
Fig. (5.7)	Data Point Organisation for NUMOFF	94
Fig. (5.8)	Examination of Planes within the Radial Distance of the Tool	94
Fig. (6.1)	Mathematical Basis of the Handle	107
Fig. (6.1)	Cross-section of the Handle	108
Fig. (6.3)	(a) Case One (b) Case Two	108

		Page
Fig. (6.4)	Intersection between the Circles	109
Fig. (6.5)	Positioning of the Circles on the Surface.	109
Fig. (8.1)	B-spline Curves of Different Order	133
Fig. (8.2)	Local Modifications on B-spline Curves	134
Fig. (8.3)	Curve Modification by Insertion of Vertices	135
Fig. (8.4)	The Effect of Multiple Vertices	136
Fig. (8.5)	Embedding of Linear Segments	137
Fig. (8.6)	A 3rd Order Closed B-spline Curve	138
Fig. (8.7)	A 4th Order Closed B-Spline Curve	138
Fig. (8.8)	A 3rd Order Closed B-spline Curve	139
Fig. (8.9)	A 4th Order Closed B-Spline Curve	139
Fig. (8.10)	A 3rd Order Closed B-spline Curve	140
Fig. (8.11)	A 4th Order Closed B-Spline Curve	140
Fig. (8.12)	The Effect of Selected Vertices on the Fitted Curve	141
Fig. (8.13)	B-spline Curve Fitting Using Different Orders	142
Fig. (8.14)	Surface Design Procesure in the System	143
Fig. (8.15)	Cross-sectional Curves for Upper and Lower Parts of the Mould	144
Fig. (8.16)	Upper Part of the Mould Viewed from Different Angles	144

Page

Fig. (8.17)	Lower Part of the Mould	145
Fig.(9.1)	Outputed Formats from Post-processor Program 'PREPARE 3D'	159
Fig. (9.2)	Block Diagram of Dataflow Using 'IBCSURF'	160

PLATES

Plate 7.1	A Tektronix 4107 Workstation	121
Plate 7.2	A Tektronix 4113 Workstation	121
Plate 7.3	The DEC Professional	122
Plate 7.4	Starrag Rigid Milling Machiane	122
Plate 7.5	Bridgeport CNC Machine	123
Plate 8.1	Cross-sectional Curve for Lower Part of the Mould	131
Plate 8.2	Cross-sectional curve for upper part of the Mould	132
Plate 9.1	The Mould for Plastic Handle Machine on Olivetti NC machine	156
Plate 9.2	The Mould for Plastic Handle Machine on Bridgeport CNC Machine	156

CHAPTER 1

CHAPTER 1

1.1 COMPUTER AIDED DESIGN AND MANUFACTURE - STATE OF THE ART

Manufacturing industry is entering a new era of advanced creativity and productivity made possible by computer aided manufacture. This augmentation of the human intellect is a product of man - computer synergism; a partnership between man and the computer, combining the best qualities of each to form a capability of great power. The progress expected in this field in the next decade has been likened in magnitude to the progress already achieved by the use of the digital computer itself since its introduction.

During the last decade the application of computer technology in the manufacturing process has concentrated on plant control. Peripheral activities such as process planning and design were not greatly affected. The recognition of a need for improvements in these area of CAD (Computer Aided Design) and CAM (Computer Aided Manufacture) has been a feature of the last few years [1]. The advances now being introduced in CAM will redress the balance in technological levels in different parts of the manufacturing process.

CAD/CAM systems available today provide either an aid to the engineer which enables him to combine his skill interactively with the speed, memory and computational ability of the computer, or a computerised and automated batch processing facility which, from a relatively small amount of input data will complete a sequence of tasks previously undertaken by the engineer. In the case of the interactive approach, the system tends to be adaptable to a very wide range of tasks, on the basis that much of the critical decision is carried out by the operator. Whereas in the case of the data-input systems the very fact that the computer system has to make an often complex series of decisions tends to limit their application to specific types of activity [2].

Integration of CAD and CAM takes place through using the stored geometry of compounds and the CAD/CAM system at many other stages in the production cycle. While 'integration' sounds straightforward enough in theory, the practical realities for any particular company are not always so clear or easy [3]. Many companies see how to use CAD/CAM systems for carrying out design, for the automatic production of drawings, and for the preparation of NC tapes. Companies are not however, so aware or able to appreciate how to exploit or develop the design-manufacturing link. For many, it is still a missing link.

The major rewards which can be reaped from computer aided manufacture come from integration [4] . It can shorten lead time in all activities dealing with design and manufacture, and small or medium sized manufacturing companies can gain large benefits from this technology by developing according to their own specifications, a tailor-made computer aided software system on a low cost mini-computer.

1.2 INDUSTRIAL APPLICATION

A major area of interest in computer aided geometrical design is in representation (CAD) and manufacture (CAM) of three dimensional objects composed in part or in total of free-form curved surfaces.

The design and manufacture of free-form surfaces has always presented difficulties to the manufacturing engineer, some of the difficulties and design details are traditionally recorded on a drawing as geometric description of part together with other relevant information, such as materials and manufacturing information. However, using this method, information is not normally given about the topology of the component and therefore, parts with curved surfaces cannot be adequately described. The manufacturing engineer, who traditionally has sought a solution to this problem by the construction and subsequent COPY MILLING of a manually made model.

Advances in both computer technology and the necessary mathematical models capable of capturing the geometry of arbitarily shaped objects has allowed the development of surface generation package aimed at providing a more economic solution to the problem of free-form surface manufacture.

1.3 RESEARCH OBJECTIVES

This research is concerned with the development of computer aided design and manufacture for 3-D milling. In essence, the following activities are considered:

1. Overview of computer aided design and manufacture geometric modelling development.

2. Computer aided geometric design of free-form surfaces, and a study on the design of using mathematical models.

3. Implementation of the computer aided design software on the computer workstation.

4. Design and development of the package editor with interactive graphics facilities, capable of displaying all the edited object from different angles.

5. The mathematical model of the handle.

6. Design and development of a general post processor for NC milling machines.

CHAPTER 2

CHAPTER 2

2. GENERAL VIEW OF CAD/CAM GEOMETRIC MODELLING DEVELOPMENT

Central to the design and production of mechanical and other discrete engineering components is Geometry. Engineering drawings are used to specify geometry between the functions of design and production.

The general widespread use of mini-computers, plotters and graphic display units initiated development of computer aided drafting systems in the 1970's . These systems generally improved quality and productivity of drafting. This is on the increase and is the beginning of Computer Integrated Manufacture (C.I.M). Geometric modelling is seen as the key element in the development of CIM.

The following describes some of the main techniques used in geometric modelling systems.

2.1 WIRE FRAME MODELLING

Wire frame [5] technique is used in many commercial CAD systems. They are usually simple 2-D designs for printed circuit boards (PCB), large scale integrated circuit (LSI), 2-D draftings and other similar tasks.

These systems store the geometry as data structures containing lists of points, lines and arcs. Applications programs scan through data structures to produce photomasks or engineering drawings. The systems usually require only simple analytic geometry co-ordinate transformations for scaling, rotation, translation and projection.

Gradually 3-D wire systems were introduced and used especially in Mechanical

Engineering because of the following advantages:

1) Model creation is fast and simple.
2) Model viewing is fast.
3) Geometric entities (points and edges) can be updated quickly.
4) Computer resource (ie power and storage) is low.

Limitation are:

1) Ambiguity - wire frame model is ambiguous.
2) Impossible object - wire frames seem to tolerate impossible or meaningless objects.
3) Hidden line removal: the hidden line in a model can only be removed manually.
4) Silhouette line: the silhouette line of the model (view dependent) cannot be generated automatically.
5) Sectioning: sectioning of the model cannot be done automatically.
6) Mass properties: the total surface area and volume of the model cannot be generated automatically.
7) Interference checking: no automatic interference facility.

Above mentioned deficiencies is due to wire frame models lack of geometric completeness.

2.2 SURFACE MODELLING

Surface modelling overcomes some of the problems of wire frame modelling as they provide more information describing the surface. They can be created by attaching surface elements to the edge frame.

Most common types of surface models are:

1) Plane: a flat plane model created between two parallel straight lines
2) Tabulated cylinder - a curved plane model between two arbitrary parallel curves.
3) Ruled surface: a model interpolated of two curves.
4) Surface of revolution: a model created by revolving an arbitrary curve through an arc about an axis.
5) Sweep surface: a model created by sweeping an arbitrary curve through another arbitrary arc.
6) Fillet surface: a model having a cylindrical face joining two other adjacent surfaces in a continuous manner.
7) Sculptured surface: this model is a general scheme to represent complex sculptured surfaces.

2.2.1 Sculptured Surface Modelling

Sculptured surface modelling evolved from work done by Ferguson, Coons, Bezier and Reisenfeld (6). The technology was initially developed to replace the lofting process used in the design of sculptured objects such as turbine blades, ships hulls, aircraft and car bodies etc.

Ferguson developed one of the earliest techniques. The Ferguson patch in 1963. The techniques used parametric rather than cartesian co-ordinates in curve and surface representation. The surface could be thus defined mathematically and designed and transformed easily by the computer via matrix algebra.

Coons of MIT in 1964 developed a general theory of surface patches (Coon's Patch). He showed how four arbitrary boundary curves can be blended to a smooth

patch and inter-patch continuity of gradient and curvature could be achieved.

Coon's work was significant but difficult for a non-mathematician to use. In 1971 Bezier introduced UNISURF (6). His practical system was based on Ferguson's work and allows curves and surfaces to be designed by non-mathematicians.

Gordon and Reisenfeld in 1974 began to exploit properties of B-spline in surface design. They incorporated local modification of curves and surfaces without affecting other existing curves and surfaces.

Sculptured surface work has mainly been concerned with interpolation and approximation rooted in the mathematical theory of spline and numerical analysis. Sculptured surface theory has had little influence on solid modelling theory, algorithms and systems.

Some convergence of these two powerful technologies is envisaged. Surface modellers create models fast, edit quickly, allow surface modification and need only a small amount of data storage. There are however, some disadvantages:

1) Lack of connective information between surfaces gives ambiguity.
2) Mass properties calculations are limited to single surfaces.
3) Calculation of intersection between sculptured surfaces is complex.
4) Interference between surface models relies on user detection.
5) Sculptured surface machining is still being developed.

2.3 SOLID MODELLING

The solid modelling technique was developed to overcome the above mentioned difficulties .

The development of solid modelling begun in Britain by Dr. Ian Braid of Cambridge University CAD Group. In 1973 they developed BUILD - 1 (7) system, in 1978, BUILD-2 (8), this continued in the Engineering Department of Cambridge University . The ROMULUS geometric modeller was produced by Dr Ian Braid working with 'Shape Data Limited' in 1978. Other significant developments were the Geometric Modelling Project at Leeds University and the MEDUSA system of Cambridge Interactive System (CIS).

Baumgart in U.S.A. influenced geometric modelling development in applying Euler operator and wing-edge data structure which helped in Eastman's GLIDE - 1 at Carnegie Mellon University and BUILD - 2 at Cambridge University. PADL - 1 was developed in

1972 in Rochester University and later PADL - 2. (9) These influenced the development of GMSOLID. In 1976 CAM - 1 launched its Geometric Modelling Project.

In Japan, Hokkaido University started TIPS - 1 (10); Tokyo University launched GEOMAP in 1973. TIPS was further developed by Cornell University under a CAM - 1 contract (11).

In Europe in 1969 Berlin Technical University developed COMPAC system, followed by PROREN system at the Ruhr University. EUCLID was launched in France and EUKLID in Switzerland.

Commercial developments were rapid. In 1980 Evan and Sunderland marketed ROMULUS, followed by many geometric modelling product announcements by CAD/CAM vendors like APPLICON (based on Synthia vision) and Computer vision (Solid Design) etc.

There are at present three major classes of solid models:

1) Idealised
2) Approximate
3) Complete

2.3.1 Idealised Model

At present this is in the experimental stages, but it is a form of solid object representation which is not completely evaluated. Examples are the sheet model (for sheet metal work) which represents the solid object by a set of single surfaces which

associated surface thickness. A graph model represents a piping system by a description of centre lines of pipes and associated cross sections of pipes.

2.3.2 Approximate Model (Faceted Model)

This model contains only planar faces therefore, it is very simple. The technique is often favoured by CAD/CAM vendors. It uses polygonal schemes to approximate solid objects with polyhedra, with hundreds or thousands of planar polygonal faces. In specifying the number of faces used the user has some control over accuracy.

The accuracy of the model will remain within acceptable limits even when a large number of faces are used, but data storage requirements increase and picture generation and manipulation becomes very slow. Although any surface can be approximated, including sculptured surface, for application that require high accuracy as NC machining, a complete representation of the solid model is preferred.

2.3.3 Complete Model

This is the most advanced representation of a solid object. A complete and unambiguous description of the object is stored in the computer. Having geometric completeness, automation of various applications based on the technique are possible. A complete model contains all boundary surfaces, is made up of faces which meet in edges located between vertices. Each edge has two vertices and meets only two adjacent faces.

The two main classes of complete models are:

2.3.3.1 Boundary Representation (B-rep)

This contains geometric entities and topology elements within the data base. The geometric entities are mains, points, curves and surfaces. The Topology elements are vertices, edges and faces. The topology relates various geometries together. All geometric information in B-rep is evaluated.

2.3.3.2 Constructive Solid Geometry (C.S.G)

This represents solid objects in terms of Boolean algebra (i.e. union, intersection and differences etc.) operated on volume elements (primitives). C.S.G. models are unevaluated trees.

CHAPTER 3

CHAPTER 3

A REVIEW OF DEVELOPMENTS IN NUMERICAL CONTROL

3.1 Milestones in Numerical Control

From the inception of mass production methods developed in the U.S. at the turn of the century there was no significant change in the methods of machining a component until the evolution of the NC machine in the U.S.in the 1950's.

The need for new developments becomes apparent in the aerospace industries after world war II. Initially, a machine tool was needed to machine curved surfaces of large components (12).

The U.S. airforce sponsored a project for a prototype NC milling machine with three axes control which was build in 1952 by the Massachusetts Institute of Technology (13). Major machine tool manufacturers continued the development based on the M. I. T. projet.

In 1959 the second generation control systems apppeared these comprised discrete electronic components which replaced earlier systems based on large space consuming vacuum tubes. In 1965 the third generation control systems reliant on integrated circuitry was introduced. Two factors made viable the used of mini-computers for control : the technical advances in the production of integrated circuit chip and the rapid fall in their production cast. From 1964 general purpose mini computers had been used in control systems, but in 1970 the fourth generation came into being.(14)

In 1974 a fifth generation or at least on extension to the fourth NC system was developed. Microprocessors were used especially in open loop systems,to standardise control systems and as a progression from large scale integrated circuits (15).

3.2 INTRODUCTION TO NUMERICAL CONTROL

In an operator controlled machine all processes are monitored by the operator - data processing position input, position feed back and compensation functions.

In an NC machine position command information is fed to slideway transmission elements by the control unit. The component drawing must be put into a form acceptable to the control.

One of the most widely accepted definitions of NC is given by the Electronic Industry Association (EIA) it states:

> "A system in which actions are controlled by the direct insertion of numerical data at some point, the system must automatically interpret at least some portion of this data".

The numerical data is given to the system through a stored input medium such as punched card, punched paper tape, magnetic tape, magnetic card or even manually [16].

3.3 METHODS OF TOOL POSITIONING CONTROL

A component is produced to the design requirements by a machine tool which alters its position relative to the workpiece by means of machine slides. Machine slides are under the control system. There are at least two slides ways on most machine tools at right angles a long which slides are displaced. Rotary displacement about the axes is usual on more complex NC machines. Fig (3.1).

The slide displacement can be controlled by open or closed loop systems. In a closed loop system, Fig (3.2). The command signal is constantly compared with the true position and the difference is fed through an amplifier so the drive motor adjusts until the difference between the signal and true positon is zero. A/D convertors are essential to most closed loop control systems as input to the servo system is in analogue form so digital signals mut be converted.

In an open loop system Fig (3.3) there is no check on whether moving parts such as the machine table or spindle are in the exact position commanded. If machine dynamics and function can be preset within permitted tolerances and the machine tool is accurate then it is possible to achieve the correct result.

Stepping motors can be used to drive and measure. These reduce costs [17] eliminate the need for A/D or D/A convertors [18].

3.4 Classification of NC Machines Tools

These may be classified into three types according to the work performed by the machine.

a) Positional Point to Point

The cutter and workpiece are placed at fixed relative positions cutting takes place at speed along a prescribed path. Suitable for machines for drilling, boring and tapping.

b) Paraxial(Straight line).

The cutter and workpiece are placed in parallel and cutting occurs along a straight line. Used mainly for surface milling or contouring.

c) Continuous Path (Contouring)

The workpiece moves with respect to the cutter while cutting takes place. These have some similarities with the point to point machine but it would be uneconomical to use them only in this way. These machines are for example; milling, routing machines, flame cutters etc. The machine table can move in any direction as they have independent control of the motors driving each axis [19].

3.5 TYPES OF EXISTING CONTROL SYSTEMS

3.5.1 Conventional NC of Machine Tools

In conventional NC a specific logic circuit designed as a hard-wired controller translates a small amount of input data to a larger output to control machine tools so achieving the automatic control of machine tools [14,20].

Conventional NC machines increase production and reduce setting up time but they also have drawbacks eg:

a) Satisfactory production of input data tapes requries great effort
b) A tape reader seen as the weakest link in NC machines is needed for each machine tool.
c) A relatively inflexible NC controller is required for each machine [21].

Application of NC machines is wide and impressive (13) and progress with these types of NC machine has been considerable, but they are not very numerous. Fig (3.5).

3.5.2 Computer Numerical Control of Machine Tools

CNC is a system in which a mini-computer replaces the electronics of a

conventional numerically controlled system and carried out some or all basic functions of a machine or number of machine tools Fig (3.6). under the control of programs stored in the mini-computer.

It was felt (22) that computer control was inevitable but only with the advent of low cost, highly modular, integrated circuit reliable low maintenance mini-computers has CNC been possible [23].

Two major advantages were hoped for in developing CNC.

1) The standardisation of hardware
2) The availability of memory dependent operational facilities [24].

With computer numerical control there is a change of design emphasis from hardware to software. Originally their was limited flexibility of system architecture due to the use of hardware techniques so a specific system was designed for a specific machine.

In Using a mini-computer it was though machine dependent functions in software would lead to standardised hardware and that future operational improvements would be implemented in software[25] so guaranteeing against absolescence. Mini-computers with relatively cheap memory could offer numerous operational facilities which, due to technical impracticalities or high cost were not possible in hardware systems such as:

a) Part-program storage or shop floor edit
b) Multiple number of tool offset and compensation
c) Large capacity operation message/instruction display
d) Input part program tape compatibility
e) Machine tool interface compatibility

3.5.3 DIRECT NUMERICAL CONTROL OF MACHINE TOOLS

In a DNC system, a main computer controls a set of machines simultaneously. It can take over machine logic or supervise the co-ordination of data distribution and storage. It is mainly a communication system which can issue current reports on down time, cost of components, maintenance requirements, inventory control etc.

The first DNC appeared in the U.S. in 1968 other countries such as Japan and Germany showed great interest.

DNC systems are usually installed in large companies which can provide a large capital outlay and technical expertise to run the system.

Systems tend to differ in design as ther eare three different contributory supplies to a DNC system machine tool manufacturers, machine tool users and builders of control system. Each has their own idea on design and criteria for the level of the control computer to operate machine tools in the system.

Generally, there are two main approaches to the design of DNC systems namely 1) "maximum flexibility" 2) "minimum cost" [28], based on (considering the role of the computer).

1) Behind the Tape Reader Method (BTR)
2) Systems with built in Machine Control Unit (MCU)

3.5.3.1 DNC Behind the Tape Reader System (BTR)

In the BTR system the main computer transfers data to several machines in parallel. The tape reader is by-passed in the interfacing of the mini-computers to

the controller but tape and conventional controller are retained Fig (3.7). The system offers maximum flexibility as each machine tool can be used as an independent NC system or as part of a DNC system.

The only disadvantage of the system is the high initial capital outlay but the system has all the advantages of conventional control plus the benefits of computer control.

3.5.3.2 DNC Machine Control Unit (MCU)

In this system most control functions are performed in the software that is in machine control unit, cost can be cut if nearly all electronic equipment at the machine tool is cut to the minimum retaining only parts such as serro drive motors, and other equipment that may be needed for data transmission and distribution. Here, control functions and machine logics for each machine tool are implemented in the control computer which services individual machines at specific times Fig (3.8).

There are disadvantages in this system, mainly a high performance computer with large core memory capacity will be required. Fewer machines can be controlled simultaneously than by using the BTR system. Costs of complicated software, and non-standardised interface must be taken into account. The second category DNC machine control unit called MCU modified cost Fig (3.9) has been developed to solve these problems.

This system allows some hardware to be kept at the machine end and some control functions such as interpolation to be performed from the computer. By reducing the rate of data requirements more machines can be connected to a single computer. In an MCU system designed by Warner and Swasey Co, circular interpolation is carried out in two stages. First a coarse interpolation is

carried out and intermediate points established. The path between these points which the cutting tool will follow is determined by the fine interpolator at the machine tool.

3.5.4 Microprocessor Numerical Control of Machine Tools

The first integrated circuit became a reality in 1970, when Intel Corp, Santa Clora C.A. produced the 4004 "Microprocessor" [29]. The circuit was designed as the control element for a cash register but was organised like a computer.

The first microprocessors were all 4-bit. The next generation produced by 1973 Intel 8080 were 8-bit with longer arithmetic registers and data words. Since 1973 National Motorola and Texas Instruments and others have produced different types of 8 and 16 bit microprocessors. These allow 'by the slice' whatever word length is desired to be built [30].

There are significant differences in the structure of microprocessors and mini-computers. Shorter word lengths in microprocessors limits the number of bits and devices to be addressed. They also take longer to execute instructions and their interrupt capability is simpler.

Microprocessors have opened up possibilities for cheaper NC systems and a different approach to system design bearing in mind that the advantage of CNC is achieved only at the most complex of machine tool spectrum.

Kean, G.C. and Savage, R. [24,31]. analyse an NC system into three different areas: data input, computation and sequence control. Control in these areas can be achieved by using a microprocessor, dedicated hardware or other control techniques. Each area shows characteristics that will affect design. Data handling and distribution needs 8-bit byte manipulation capabilities. The second area needs

systems (Westinghouse W2560). Successive machine tool exhibitions since 1977 show that micro-processor base CNC systems are the most dominant of control systems. 'Playback' systems are also a result of the development of micropros: In this system, a skilled machine operator machines the first component of the batch using console controls. The instructions are recorded and used to machine the rest automatically. Systems such as Kremer 100 self programming control using magnetic cassettes was one of the earliest. Siemens have MATE-M 3 axis system with semi-conductor memory which can store up to 230 instructions.

Recent systems [34] have extensive diagnostic testing facilities, more compact control design, reduction in hardware costs [33] and flexibility.

3.6 INTEGRATED MANUFACTURING SYSTEMS

Machining centres were among the first approaches to integrate all machining operations to be performed at one setting, such centres tend to be expensive and time is wasted on operations such as drilling and tapping which could be done on smaller, less complex, cheaper machines

The trend seems to be towards an integrated automatic factory.

In Japan and the USA [35] to name but two countries, DNC M/C tools, handling devices and transportation conveyors are integrated into a system with software to control timing relation between this equipment and advance production schedule.

Most DNC systems can be justified economically and technologically, so have a future in the automated manufacturing system. Group technology involving production and component systems of classification has developed from the need to run numerically controlled machine tools fully loaded so that production components are economically channelled to a specific machine tool for specific operations.

Recent developments in group technology include the 'Flexible Machining Cell' (FMC) [36]. A single CNC unit comprising controls, machine tools and work handling devices which perform all operations required on a family of components. Cells are linked hierarchically to form a flexible manufacturing system Fig (3.11).

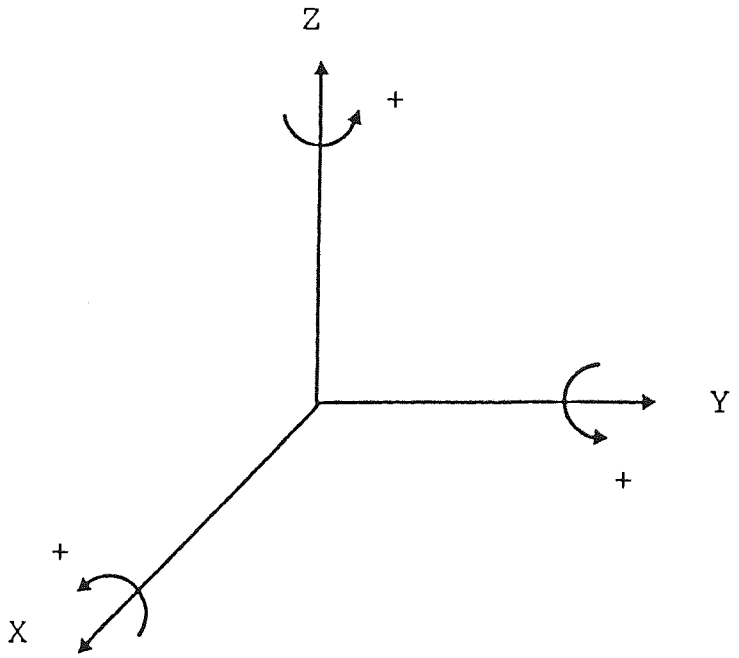


Fig. (3.1) Conventional Method of Defining Linear and Rotary Displacement of Machine Tools.

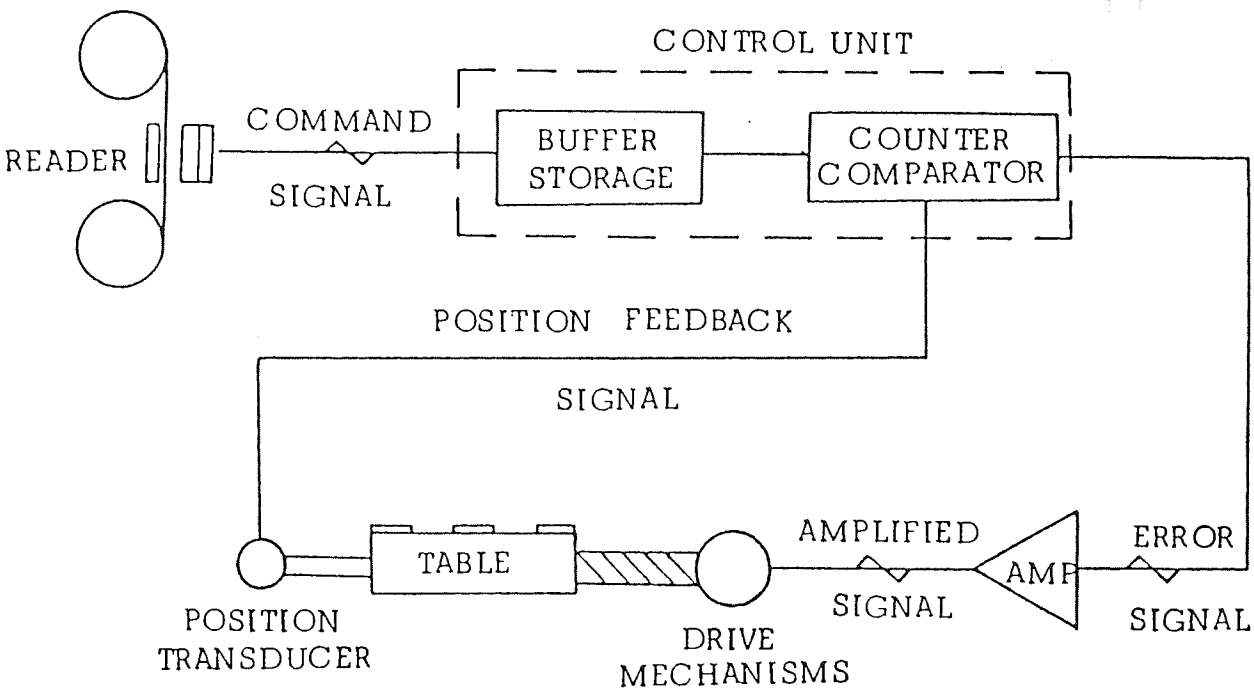


Fig. (3.2) Closed Loop Displacement Control

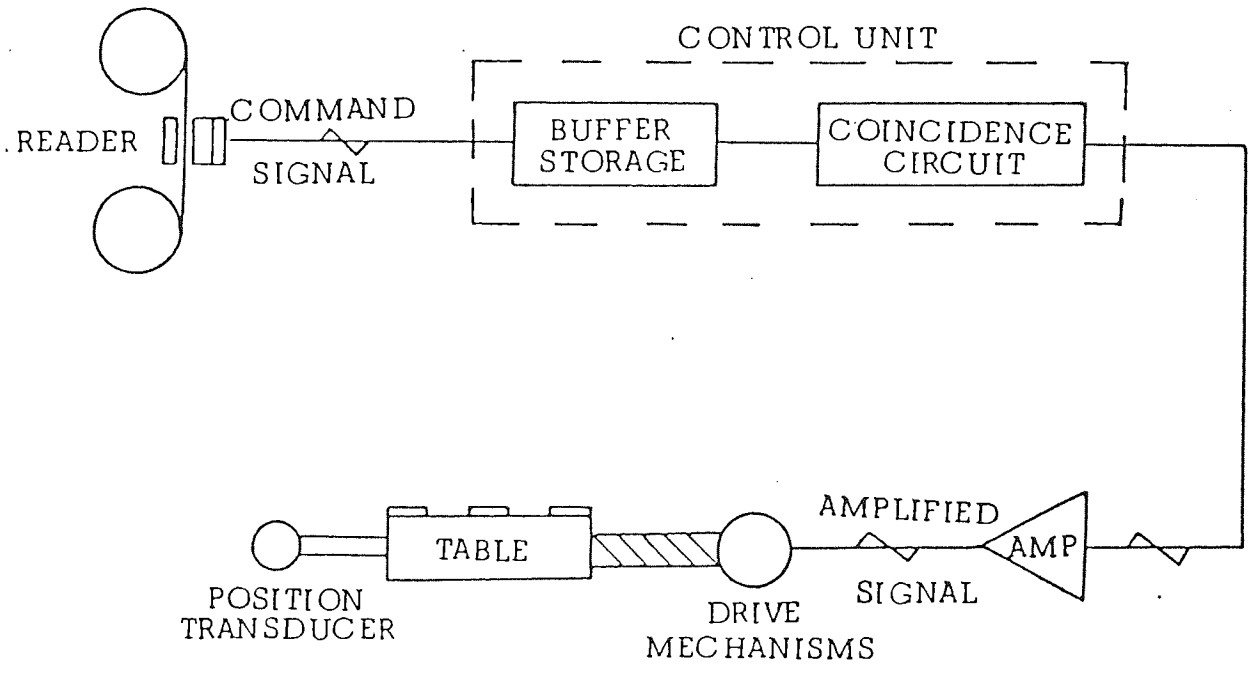


Fig. (3.3) Open Loop Displacement Control

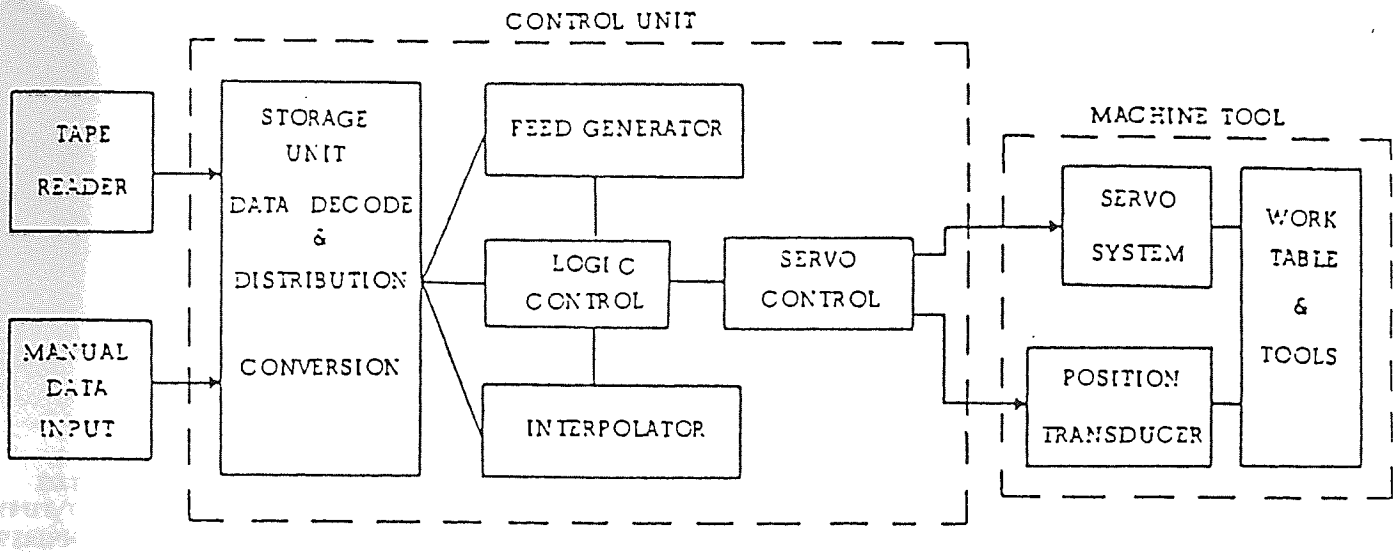


Fig. (3.4). Block Diagram of a Typical Hard-Wired NC System

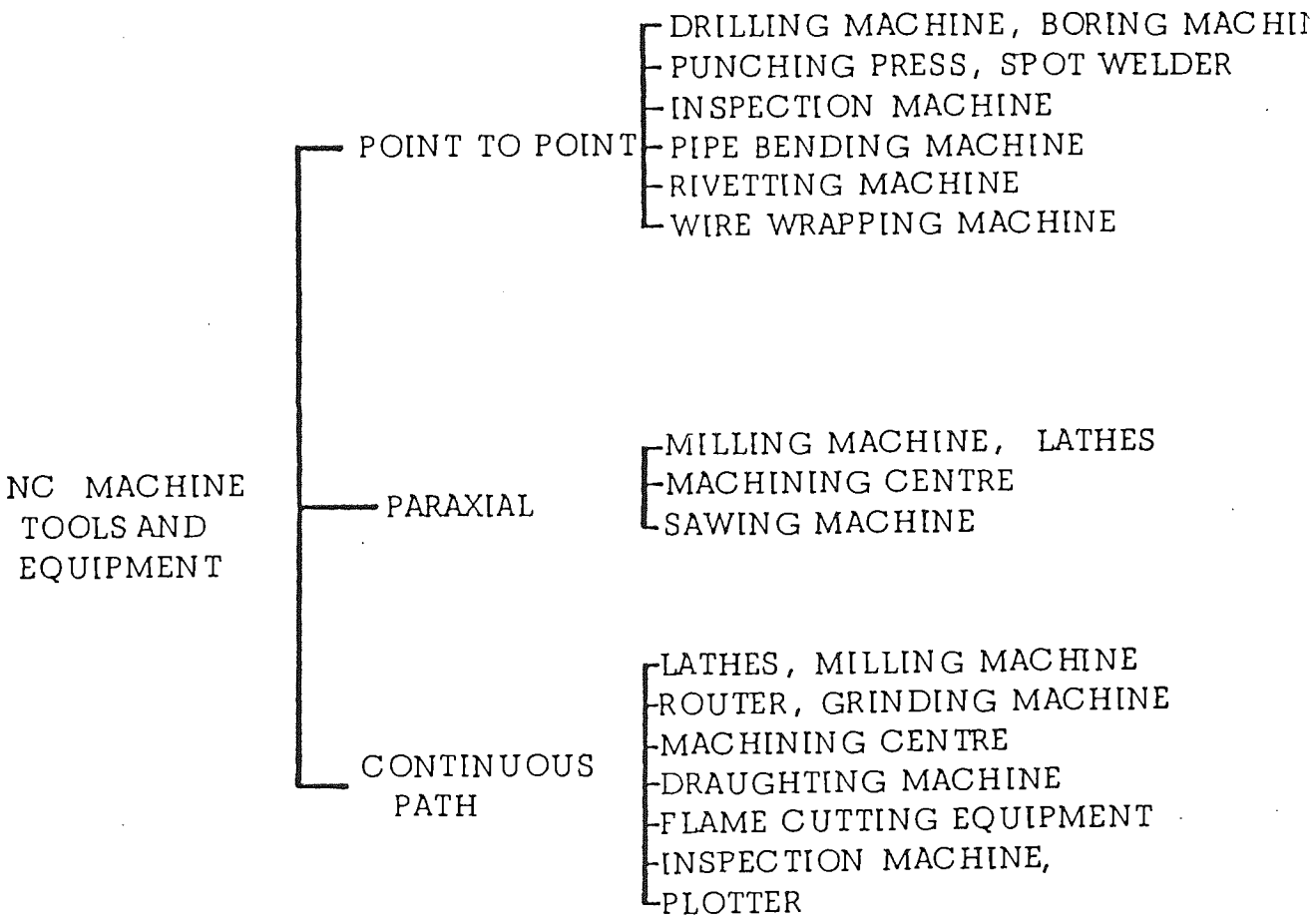
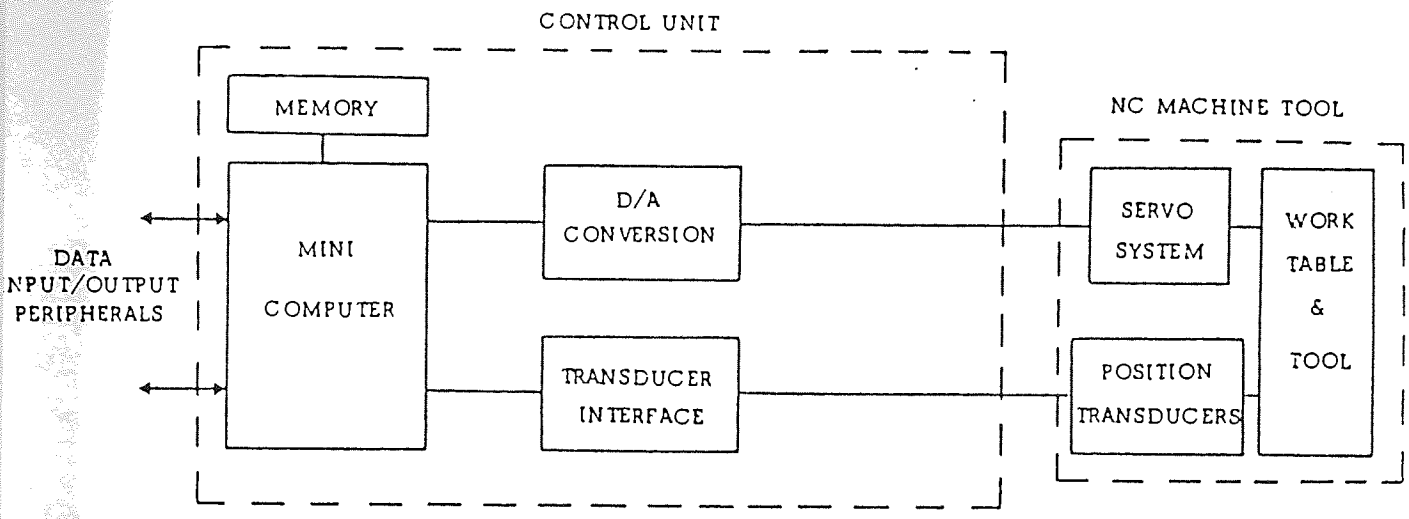
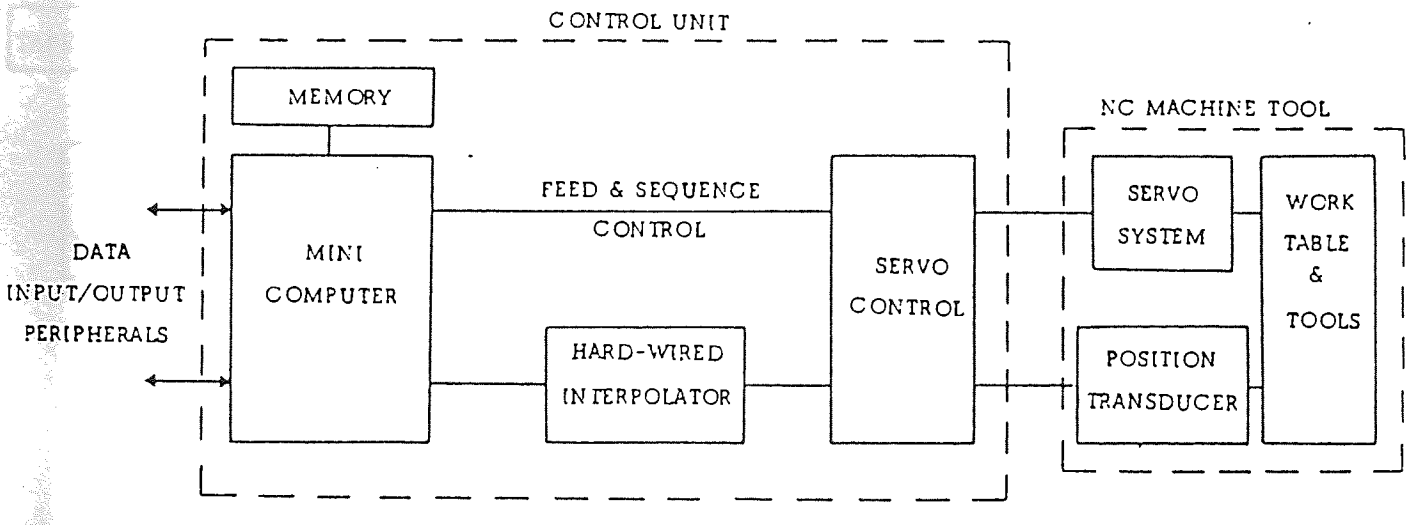


Fig. (3.5) Types of NC Machine Tools and Equipment



(a)



(b)

Fig. (3.6) Block Diagrams of CNC Systems

(a) Full Computer Control

(b) System Incorporating Hard-Wired Interpolator

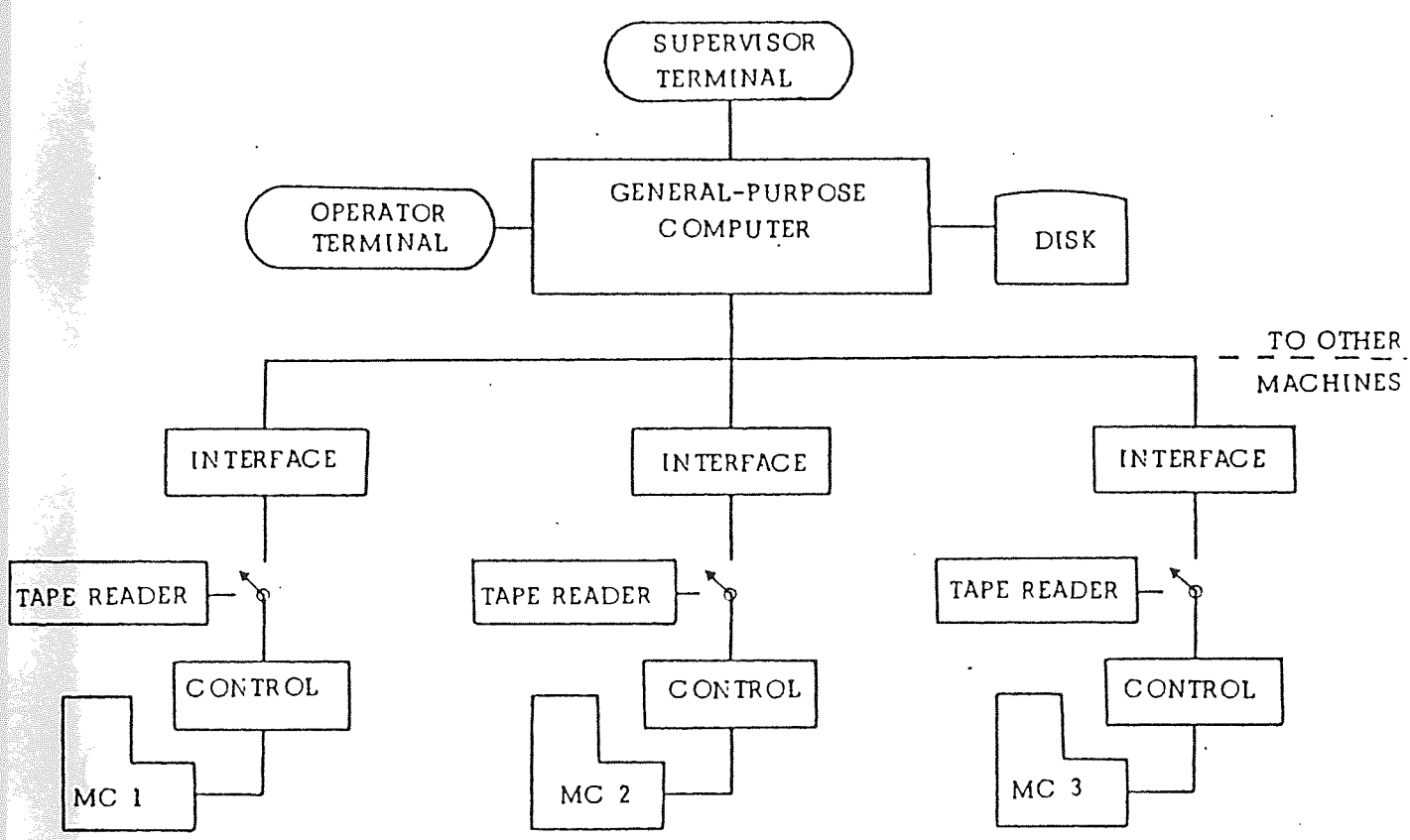


Fig. (3.7) Block Diagram of a DNC Behind the Tape Reader System

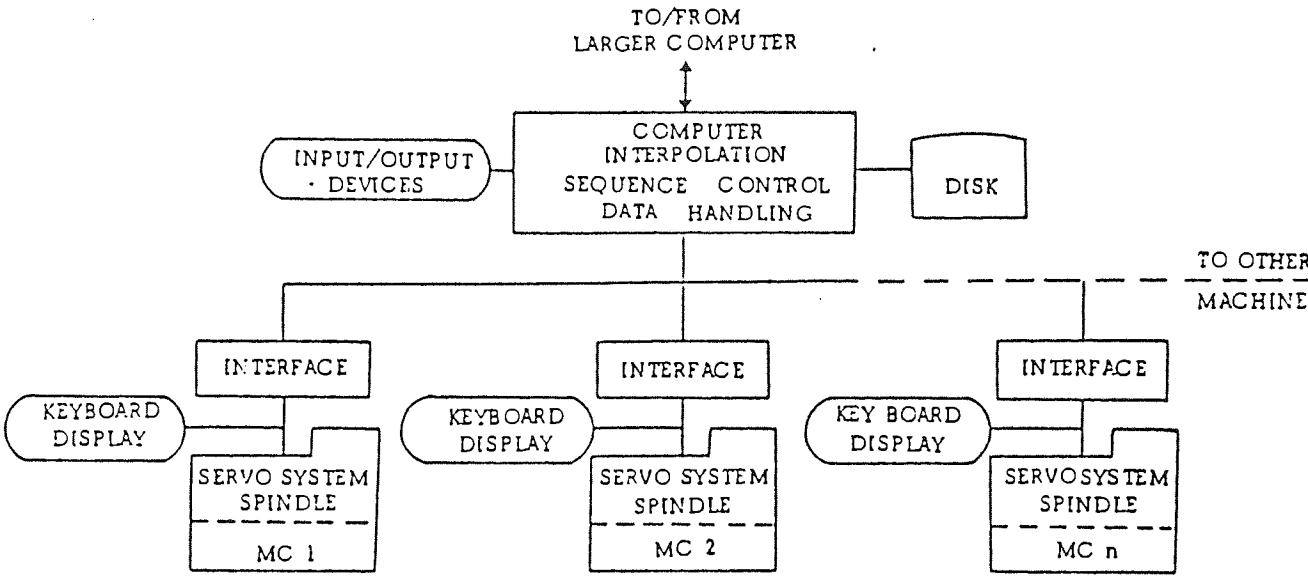


Fig. (3.8) A DNC Machine Control Unit System

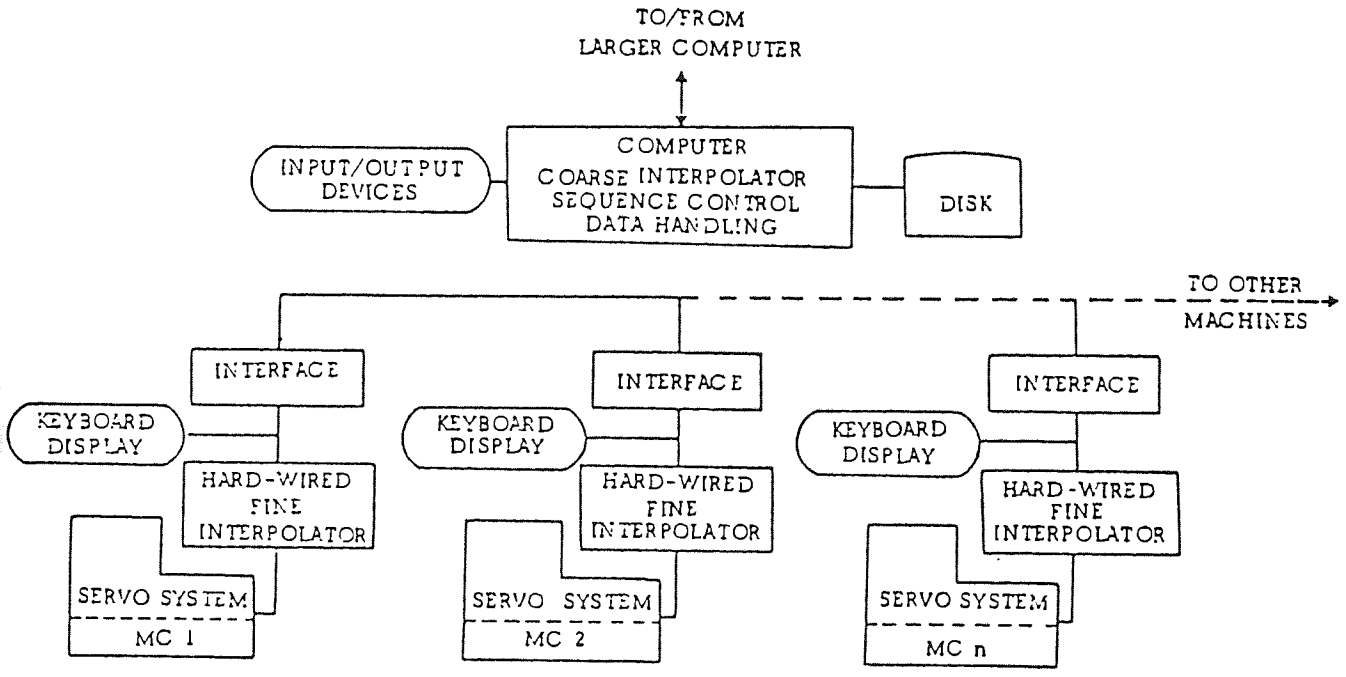


Fig. (3.9) MCU Modified Minimum Cost DNC System

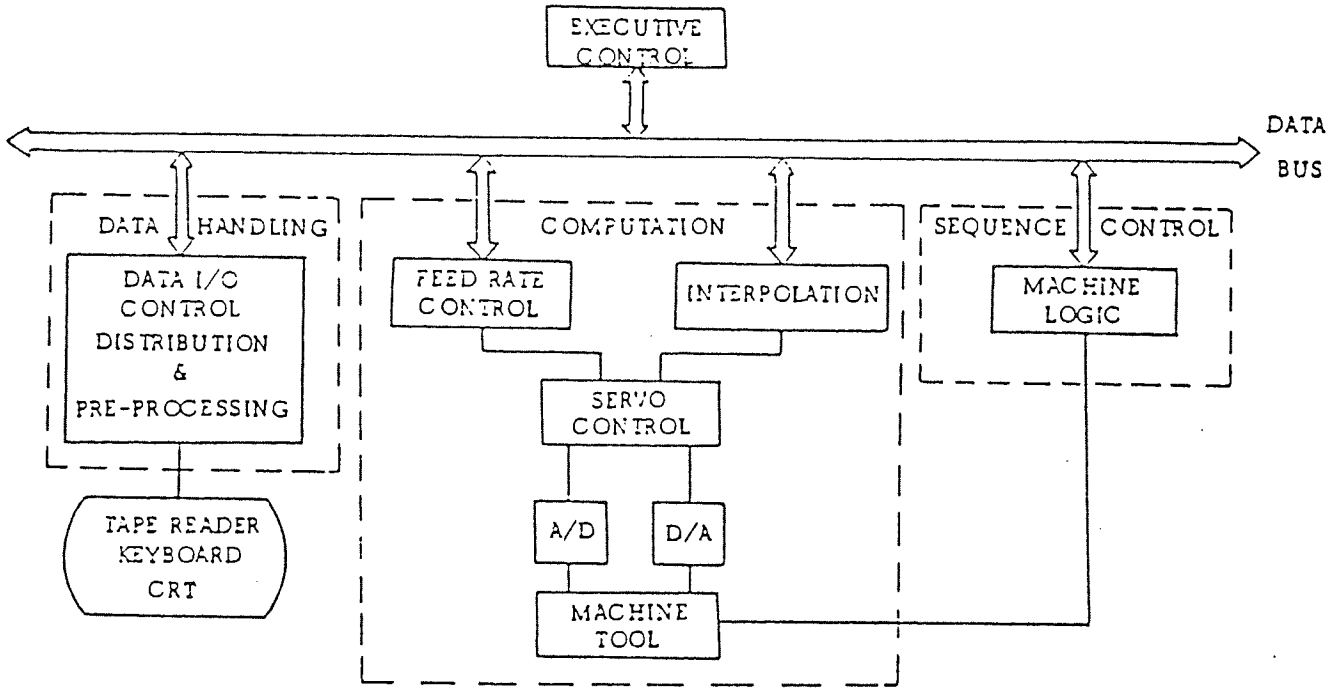


Fig. (3.10) Function Diagram of a Decentralised Approach in Design of NC Systems

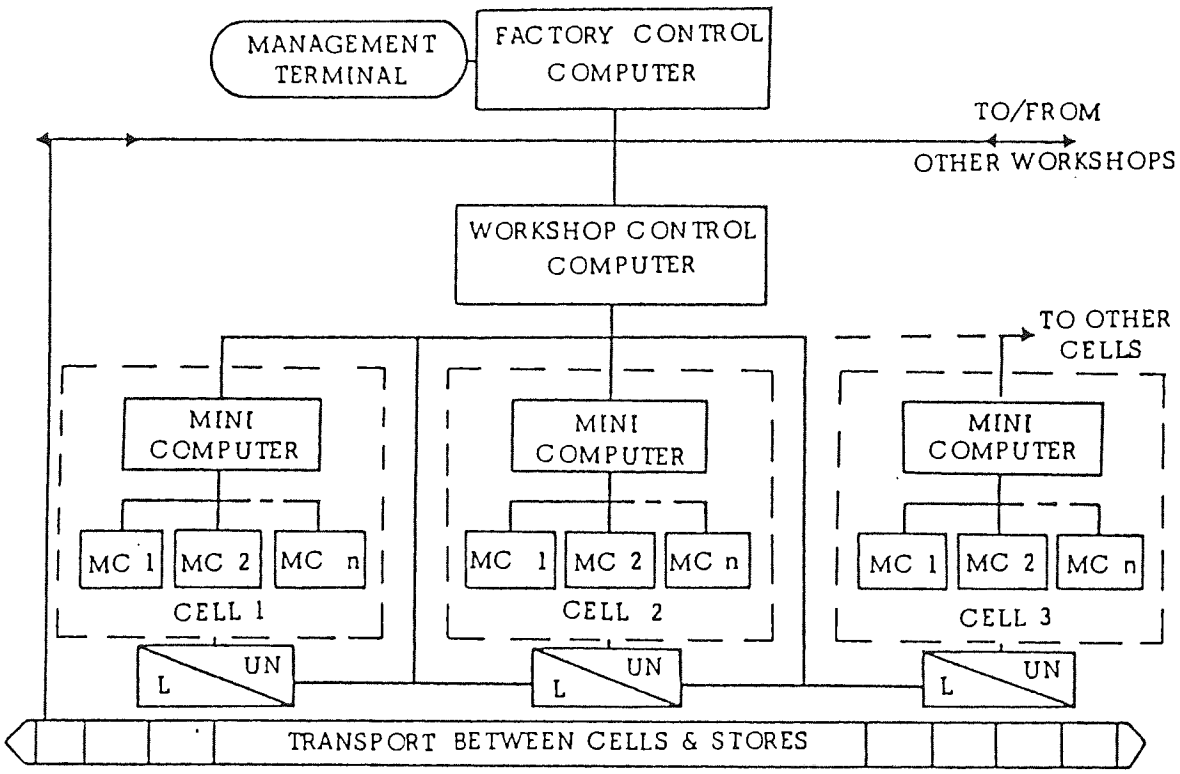


Fig. (3.11) Flexible Machining Cells in an Integrated Manufacturing System

CHAPTER 4

CHAPTER 4

A REVIEW OF COMPUTER AIDED DESIGN OF FREE-FORM SURFACES

4.1 CONSIDERATIONS OF DESIGN

The design of mechanical object begins with the study of aesthetic or scientific constraints and a shape is produced to satisfactory product shape, mechanical properties and aerodynamic performance [37].

Analysis of constraints and other design processors can be done using traditional methods or using a computer aided technique.

4.1.1 Traditional Approach to Design

The designer traditionally designed his object in the following forms:

a) A model

b) An analytical surface

c) A free hand sketch

d) Conventional engineering drawing

e) A tabular specification of points which lie on the devised surface

f) Displacement and slope requirements at selected points on the surface

The main disadvantages of traditional design are that it is time consuming, often ambiguous and may require special skills such as artistic and analytical ability.

4.1.2 Computer Aided Design

It has been said that CAD is a technique in which man and machine are blended

into a problem solving team, intimately coupling the best characteristics of each, so that this team works better than either along [38]. When a mechanical object is designed using a computer, the object is modelled in the computer and can give a better representation which helps the study of design parameters.

The CAD system can be divided into hardware and software. The hardware comprising the support computer and graphics devices - interactive or otherwise. The software is divided into two parts. First the software system that provides the interface between the normal drawing functions and the graphics devices - a device dependent software written for use on a specific device.

The second is the data manipulating software used for geometrical description of the time to be designed. The mathematical basis for this part of CAD is the main topic of this chapter. Because graphics devices are an integral part of any CAD system, some more important types are also briefly discussed (Fig 4.1).

4.2 C.A.D. HARDWARE

4.2.1 Interactive Devices

Interactive devices communicate with the program while it is running so interrupting it where necessary to include new or different information. Many devices have been used for this the simplest of which is the alpha-numeric keyboard. More sophisticated devices include light pens, joysticks, control dials and analogue tablets.

Taking a keyboard as an interactive device, alphabetic numeric and control information can be given to the program by typing in the data. The light pen, which contains a sensitive photo-electric cell and its circuitry, can be positioned over a line of CRT and activated. The position of the pen is sensed so causing an interrupt to

be sent to the computer.

By moving a control in the joystick and other similar devices, two dimensional positioning can be communicated to the computer. This is an analogue device and does not give precise positional information.

The most accurate and versatile device for communicating positional information is the analogue tablet and it's associated pen whose position can be sensed as it moves over the surface. The pen's position and its relative position on the picture - display area directly affects the movement of a cursor on that display. The analogue tablet which is an easily accessible horizontal surface is used to perform a pointing function indicated on the database not in the display file, so programming is simplified.

4.2.2 Types of Graphics Devices

There are many types of graphics devices. Mainly three different types of CRT (storage tube, refresh and rastor scan) pen and ink plotter and dot matrix plotter will be described.

4.2.2.1 Cathode Ray Tube Graphics Devices

The direct view storage tube display (bistable storage tube) is similar to an oscilloscope with a very long persistence phosphor. A line or character will remain visible for up to one hour before being erased. Storage tube displays have several advantages such as; flicker free display, low cost, a quickly obtainable hard copy. Conceptually they are easier to program and more suitable for time sharing applications than refresh and rastor scan displays. The main disadvantage is that the screen cannot be cleared selectively so that the whole picture must be redrawn to change any defail. This also means that display of

dynamic motions is not possible. This characteristic also slows down interaction between the user and the display.

A refresh CRT graphics tube is based on a television like cathode ray tube. The phospor used, fades rapidly so the complete picture must be reconstructed many times per second. This is called the refresh rate which must be at a minimum of 30 to avoid flicker. The drawing instruction is contained in the display buffer and the complexity of the picture is limited by this to include each new element of the design. The complete picture must be re-drawn each time and new elements can be added or deleted.

Dynamic motion is possible, but it is difficult to obtain a hard copy of the picture in the display. For dynamic motion in real-time or rapid interaction this is the best graphics display unit.

The rastor scan CRT graphics display uses a standard television monitor as the display console. The picture display is composed of series of dots which are traced out using the dual rastor scan technique. The basic electrical signal used is

an analogue signal, the modulation of which represents the intensity of individual dots. Lines and character information have to be converted to a form compatible with rastor representation. These displays are generally slower and selective erasure is difficult to implement, but it is possible if it is interfaced to a closed circuit television system.

4.2.2.2 Plotters

Digital incremantal plotters can provide high quality hard copy of graphical output. These pen and ink plotters are of two general types; either flat based, or drum. They are quite slow compared to CRT so are not used for real time

interactive graphics.

The electrostatic dot-matrix printer/plotter operates by depositing particles of a toner onto small electrostatically charged areas of special paper. The particles are attracted to charged areas and graphics are displayed. The electro-static dot-matrix, plotter/printer is a rastor scan device so it presents information one line at a time. This requires a considerable amount of computer storage. Although it is reliable and excellent drawings can be produced at high speed, the accuracy and resolution are quite low.

4.2.3 Classification of Graphics Devices

There are various methods of classifying graphical devices, each based on a different point of view and objectives.

Some devices are passive. Such devices allow the computer to communicate with the user graphically and a computer controlled picture is drawn. Examples are teletype, line printers, storage tube CRT's and plotters. Other devices are active and allow the user to communicate with the computer as well. Such a device can usually reposition the cursor and read the new position. Examples are analogue tablets, light pens, joysticks etc. All of which usually require some supporting passive graphics device.

Devices can be classified according to whether it is point plotting or line (vector) drawing. All storage tube CRT's and most refresh CRT's and all pen plotters are line drawing. While the rastor scan CRT, high speed line printers and electrostatic plotters are point drawing devices.

A further method of classifying graphics devices depends on determining whether a device can accept 3-D data or whether 3-D data should be converted to

2-D data by applying some projective transformation to present it to the system. Essentially the method requires determining whether a graphics device has 2 or 3 registers to hold co-ordinate data [39].

4.3 COMPUTER AIDED DESIGN SOFTWARE

4.3.1 Introduction to Numerical Geometry

As in other areas of science and technology, numerical geometry first became important during the Second World War when the pressure of production - especially in the aircraft and shipbuilding industries stimulated the development of new design techniques. The old methods were mainly graphical but the new methods were based on analytical curves, conic in particular.

The development of electronic computers opened the way for more ambitious techniques. Ferguson [40] developed one of the first computer aided geometry systems which used parametric rather than cartesian co-ordinates in its curve and surface definition. This has become standard usage.

Progress in numerical geometry was helped by the development of computer graphics, draughting machines and interactive computer peripherals. Numerically controlled machine tools were also becoming available and so complemented the production process to the manufacturing stage.

Numerous important mathematical developments followed Ferguson's system [41]. The properties of spline curves have been studied widely resulting in the development of an efficient method of spline curve construction called B-spline or 'fundamental splines' by Curry and Shoenberg [41]. These have the property of allowing local modifications to be made to a shape during the design process without having to recompute the entire process.

Coon's Theory in surface design which blended curves into a single smooth patch having inter-patch continuity [42] was another major development.

Bezier [43] introduced the UNISURF system in 1971. A major step in the field of CAD UNISURF is a good example of a successful interactive system. A designer starts with defining a polygon. This is then approximated by a smooth curve, this is modified until the desired curve is produced. In 1974 Gordon and Reisenfeld [44] approached the design of curves and surfaces using defining polygons from a different angle. They used B-spline instead of simple polynomial as Bezier had done.

Polynomial based functions provide continuity but are mostly oscillatory. Some new ideas have been put forward which endeavour to avoid this such as spline under tension [45] and non-linear spline used by the AUTOKON system [46].

Various systems for CAD have been developed in recent years. In the design of sculptured or free form surfaces, the better known ones, apart from those already mentioned are, N.M.G. (Numerical Master Geometry) at the British Aircraft Corporation [47], POLYSURF [48], DUCT [5] at Cambridge, BSURF [49] at the University of Leeds, OKISURF [50] in Japan.

A wide range of products that can be designed by these programs include propellers, car body panels, boilers, air ducts, pipes and exhaust systems, shoe ship lasts, ship hulls, aircraft bodies, domestic appliances etc.

4.3.2 Parametric Description of Curves and Surfaces

The path of a moving point in three dimensional space may be described by the value of a position vector r at successive instants of time. Therefore, r is a

function of time and the relationship can be shown as $r = r(t)$ which is equivalent to:

$X = X(t)$, $Y = Y(t)$, $Z = Z(t)$ in terms of coordinates.

The time in the above relationship only refers to a point on the path, so can be replaced by any parameter, "U". It is better that any given value for u represents only one point on the curve Fig (4.2).

If the curve $r = r(u)$ moves in space, the successive position of this curve generates a surface where each point is distinguished by the time t and parameter u. Again t can be substituted for any other parameter, V for example. Then vector function $r = r(u,v)$ of two variables will represent a surface Fig(4.3).

There are several reasons for using parametric curves and surfaces for CAD. In the non-parametric method, to alter one point specifying a fitted surface means the entire surface must be recomputed, so all parts of the surface, excepting data points, will be changed positionally to some extent. So the effect of modification is not localised [40]. Twisted curves in 3-dimensions can be represented more simply and closed curves with vertical tangen can be represented easily in a fixed coordinate system.

Another significant advantage is that translation or rotation of the axes or object can usually be carried out by translating or rotating the vectors defining the curve without modifying the functions as the mathematical representation is independent of the coordinate system. The parametric method simplifies the computation of cutter offsets and similar related curves for numerical control purposes [51].

4.3.3 Curve Design

In industry, most curves are defined piecewise and surfaces in parametric

patches as it is unusual to find many objects with simple shapes that can be defined by simple analytical functions.

Many non-parametric methods are used in curve and surface design, the following sections only deal with parametric piecewise description of curves and parametric patch design of surfaces.

4.3.3.1 Ferguson Cubic Curve

Ferguson [40] working at Boeing Company designing aircraft bodies, introduced the use of parametric representation of curves and surfaces.
Considering a cubic, the segments of these curves are described as follows:

$$r = r(u) = a_0 + ua_1, + u^2 a_2 + u^3 a_3 \qquad (4.1)$$

One way to determine the values of a_0, a_1, a_2 and a_3 is to find the value of r and dr/du at both ends of the segments. Taking the parameter u between 0 and 1, u + 0 and u + 1 refer to both ends of the segments.

Denoting dr/du by r (u)

$$a_0 = r(0),$$

$$a_1 = \dot{r}(0), \qquad (4.2)$$

$$a_0 + a_1 + a_2 + a_3 = r(1),$$

$$a_1 + 2a_2 + 3a_3 = r(1)$$

Solving for a_0, a_1, a_2, a_3

$$a_0 = r(0),$$

$$a_1 = \dot{r}(0) \tag{4.3}$$

$$a_2 = 3[r(1) - r(0)] - 2\dot{r}(0) - \dot{r}(1),$$

$$a_3 = 2[r(0) - r(1)] + \dot{r}(0) + \dot{r}(i)$$

By substituting these values in (4.1), r can be described in terms of $r(0)$, $r(1)$, $\dot{r}(0)$, $\dot{r}(1)$, i.e.

$$r = r(u) = r(0)\ (1 - 3u^2 + 2u^3) + r(1)\ (3u^2 - 2u^3)$$

$$+ \dot{r}(0)\ (u - 2u^2 + u^3) + \dot{r}(1)\ (- u^2 + u^3) \tag{4.4}$$

or in matrix form

$$r(u) = [1\ u\ u^2\ u^3] \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & -2 & 1 & 1 \end{bmatrix} \begin{bmatrix} r(0) \\ r(1) \\ \dot{r}(0) \\ \dot{r}(1) \end{bmatrix} \tag{4.5}$$

Note that $r(0)$, $r(1)$, $\dot{r}(0)$, $\dot{r}(1)$ are all vectors. The derivatives $\dot{r}(0)$ and $\dot{r}(1)$ are therefore proportional to the unit tangent vectors $T(0)$ and $T(1)$ at the end points. So we can write :

$$\dot{r}(0) = \alpha_0\ T(0),\quad \dot{r}(1) = \alpha_1\ T(1)$$

The magnitude of the end tangent vectors must not become too large as an unwanted curve will be produced Fig(4.4). A safe rule is that the magnitudes should not be much greater than the chord length of the segment.

If we wish to construct a composite curve by joining segment r 1 (u 1), $0 \leq u\ 1 \leq 1$ to segment r2 (u2), $0 \leq u2 < 1$. Normally the constraint is to have continuity at the joint and continuity of slope there. Therefore,

$$r\,1\,(1) = r2\,(0) \tag{4.6}$$

$$\dot{r}\,1\,(1) = \alpha_1, T$$

$$\dot{r}\,2\,(0) = \alpha_2\, T \tag{4.7}$$

In obtaining curvature continuity, Ferguson matches $r, \dot{r}, \ddot{r}$ across the joints so that:

$$\alpha_1 = \alpha_2$$

$$\ddot{r}\,1\,(1) = \ddot{r}\,2(0) \tag{4.8}$$

If $\ddot{r}\,1\,(1)$ and $\ddot{r}\,2\,(0)$ are evaluated from (4.4) conditions (4.6) and (4.8) yield that:

$$6\,r\,1\,(0) + 2\,\dot{r}\,1\,(0) + 4\,\dot{r}\,1(1) = 6\,r2\,(1) - 4\,\dot{r}\,2(0) - 2\dot{r}\,2\,(1)$$

To fit a Ferguson curve through a set of points r_0, r ,..... r_n with tangents at those points t_0, t_1, ... t_n the last equation gives:

$$t_{i-1} + 4t_i + t_{i+1} = 3(r_{i+1} - r_{i-1}),\ i = 1, 2, \ldots\ldots n-1$$

From this recursive relation it is only necessary to specify t_0 and t_n to obtain the remaining tangents in terms of positional data using the following matrix.

$$\begin{bmatrix} 4 & 1 & 0 & \ldots\ldots & 0 \\ 1 & 4 & 1 & 0 \ldots\ldots & 0 \\ 0 & 1 & 4 & 1 \; 0 & 0 \\ \cdot & \cdot & \cdot & \cdot\ \cdot & \cdot \\ 0 & . & . & 0 \; 1 \; 4 & 1 \\ 0 & . & . & . \; 0 \; 1 & 4 \end{bmatrix} * \begin{bmatrix} t_1 \\ t_2 \\ t_3 \\ : \\ \\ t_{n-1} \end{bmatrix} = \begin{bmatrix} 3(r_2 - r_o) - t_o \\ 3(r_3 - r_1) \\ 3(r_4 - r_2) \\ : \\ : \\ 3(r_n - r_{n-2}) - t_n \end{bmatrix}$$

4.3.3.2 Spline Functions

A spline is a simple mechanical devise consisting of a long narrow strip of flexible material such as wood, metal or plastic. heavy objects called 'ducks' are placed at specified points on the spline is passed through to obtain a 'fair' curve.

Physical spline can be seen as a thin elastic beam so that Euler's equation (52) gives:

$$M(x) = \frac{EI}{R(x)}$$

Where M(x) : bending movement, E: Young's module

R(x) : radius of curvature, I: moment of inertia

If we denote by Y = Y(x), the equation of the curvature produced by the

spline:

$$I / R = \ddot{Y} / (1 + \dot{y}^2)^{3/2}$$

for small deflections small slopes, Y can be neglected (53)

then

$$R \approx 1 / \ddot{Y} \qquad \text{therefore} \qquad \ddot{Y} = \frac{M(x)}{EI}$$

Assuming that the ducks act as a simple support, then M(x) is a linear function between the supports. Taking M(x) = AX + B and integrating the resulting equation twice, shows that the physical spline is described by Cubic polynomials between supports.

Generally, the mathematical spline is a piecewise polynomial of degree n with continuity of derivatives of order n - 1 at the joints.

There are many methods of constructing cubic splines. One of the more usual is as follows (54). Taking s (u) = cubic spline function $u_{i-1} < u < u_i$. Since $\ddot{S}$ (u) is cubic S (u) is linear in U over this span. If $\ddot{S}(U_{i-1}) = a_{i-1}$ and $\ddot{S}(u_i) = a_i$ then any point on $\ddot{S}$ (u) is calculated by

$$S(u) = \frac{a_{i-1}(u_i - u) + a_i (u - u_{i-1})}{h_i}$$

Integrating the above equation twice:

$$S(u) = \frac{a_{i-1}(u_i - u)^3 + a_i (u - u_{i-1})^3 + C_i u + C_2}{6h_i}$$

To determine the values of C_i and C_2. The end conditions can be used resulting in:

$$S(u) = \frac{a_{i-1}(u_i - u)^3 + a_i (u - u_{i-1})^3}{6h_i}$$

$$+\left[\frac{r_{i-1}}{h_i} - \frac{a_{i-1} h_i}{6}\right](u_i - u) + \left[\frac{r_i}{h_i} - \frac{a_i h_i}{6}\right](u - u_{i-1})$$

$$u_{i-1} \leq u \leq u_i \qquad (4.9)$$

To evaluate a_{i-1} and a_i the property of first order continuity at the joint is utilized. Differentiating (4.9) and taking $u = u_i$

$$\dot{S}(u_i) = \frac{r_i - r_{i-1}}{h_i} + \frac{a_i h_i}{3} + \frac{a_{i-1} h_i}{6} \qquad (4.10)$$

If in (4.9) i is substituted by i + 1, the function represents the next span $u_i \leq u \leq u_{i+1}$. Differentiating this function and setting $u = u_i$

$$\ddot{S}(u_i) = \frac{r_{i+1} - r_i}{h_{i+1}} - \frac{a_i h_{i+1}}{3} - \frac{a_{i+1} h_{i+1}}{6} \qquad (4.11)$$

The right hand side of (4.10) and (4.11) are equal then:

$$h_i a_{i-1} + 2(h_i + h_{i+1}) a_i + h_{i+1} a_{i+1} =$$

$$\frac{6(r_{i+1} - r_i)}{h_{i+1}} - \frac{6(r_i - r_{i-1})}{h_i} \qquad (4.12)$$

Considering (4.12) over n spans of a spline a curve built on $r_0, r_1 \ldots\ldots r_n$, there will be, n - 1 functions and n + 1 unknowns, these being a_0, a, ...a_n, therefore, we need to know two further relations for a. These relations depend on the specific application [55,56]. Some possible relations are relaxed end (natural spline), i.e. $\ddot{S}(u_0) = \ddot{S}(u_n) = 0$, or a clamped end where the end tangents are specified (Fig 4.5).

4.3.3.3 B-Spline Curve

This is one of the most interesting mathematical methods introduced for C.A.D. and was used in the system developed in this work. The theory and advantages of B-spline over other methods will be discussed later in Chapter 5.

4.3.3.4 Bezier Curves

Curves such as Ferguson's passing through all data points are not very effective in 'ab-initio' interactive design. This is due to the fact that the curve shape when controlled by numerical specifications of the direction and magnitude of the tangent derivatives does not always have the 'feel' necessary for curve design. In otherwords, there is not always a clear relationship between the numbers and the shape of the curve.

Bezier's [43,57] method of curve design allows the user to be the artist or designer as there is a greater feel for the relationship between input and output.

The curve can be altered until the desired shape is achieved.

Using Bezier's method the curve is defined by an open polygon. Only the first and last vertices of this characteristic polygon are on the curve. The other points define the derivatives, order and shape of the curve. Any change of vertices within a span will only affect the curve within that span, therefore local modification is easily achieved.

The mathematical basis of the Bezier curve is a polynomial blending function which interpolates between the first and last vertices. The Bezier curve is defined by recombining the terms in Ferguson parameterisation.

A cubic Bezier curve in simple form is defined as follows:

$$r = r(u) = (1 - u)^3 r_o + 3u(1 - u)^2 r_1 + 3u^2 (1 - u) r_2 + u^3 r_3$$

where $0 \leq u \leq 1$ for any given segment.

Compared to Ferguson's form it can be seen that:

$$a_0 = r_0,$$

$$a_1 = 3(r_1 - r_0),$$
$$a_2 = 3(r_2 - 2r_1 + r_0),$$

and $$a_3 = r_3 - 3r_2 + 3r_1 - r_0$$

Therefore

$$r(0) = r_0,$$

$$r(1) = r_3,$$

$$\dot{r}(0) = 3(r_1 - r_o), \qquad (4.13)$$

and $\dot{r}(1) = 3\,(r_3 - r_2)$

From (4.13) it can be seen that the curve passed through the points r_0, r_3 and has tangents at r_0 in the direction from r_0 to r_1 and at r_3 in the direction from r_2 to r_3 (Fig 4.6).

The Bezier polynomial is related to the Bernstein polynomial so the Bezier curve is said to have a Bernstein basis. This basis function is given by:

$$B_{n,i}(u) = \frac{n!}{1!\,(n-1)!}\; u^i\,(1-u)^{n-i}$$

n is the degree of polynomial, i refers to the specific vertex from 0 to n. Generally an n degree polynomial is defined by n + 1 vertices.

The general form of the Bezier curve is then defined by:

$$r = r(u) = \sum_{i=0}^{n} \frac{n!}{i\,n!\,(n-1)!}\; u^i\,(1-u)^{n-1}\, r_i$$

With the above definition, curves of any degree can be constructed curves with a higher order, allow several orders of continuity, but the higher the order, the less indicative of the curve shape is the characteristic polygon.

If a second curve r2 (u2) is required to be joined to an existing curve, r1 (u1) with slope and curvature continuity at the joint, then it is first required that:

$$r1_3 = r2_0 \qquad (4.14)$$

Since $r(0) = r_0$ and $r(1) = r_3$. It is also required that $r1(1) = r2(0)$.

From (4.13) $\dot{r}(0) = 3(r_1 - r_0)$ and $\dot{r}(1) = 3(r_3 - r_2)$

Therefore $r1_3 - r1_2 = r2_1 - r2_0$ (4.15)

All parameters in (4.15) are vectors (4.14) and (4.15) imply that $r1_2, r1_3 = r2_0$, and $r2_1$ must be collinear . To achieve curvature continuity it is proved [54] that for a cubic $r1_1, r1_2, r1_3 = r2_0, r2_1$ and $r2_2$ must be Coplanar (Fig 4.7).

4.3.4 Surface Design

4.3.4.1 Coons Patches

To appreciate Coons' [42] method of patch description start by describing a lofted or ruled surface. Assume that two boundary curves associated with the opposite sides of the unit square in the a.v. plane are known, say r (u,0), r(u,1). A ruled surface is then obtained by linearly interpolating between these two curves. The linear interpolation scheme is:

$$Q_1(u,v) = r(u,0)(1 - v) + r(u,1)\, v \qquad (4.16)$$

From the above method it is clear that the edges of the interpolated surface coincide with the given data curves, i.e.

$$Q_1(u,0) = r(u,0)$$

$$Q_1(u,1) = r(u,1)$$

Alternatively, if curves on the other two sides of the patches are known as, r (0,v) and (1,v).

$$Q_2 (u,v) = r (0,v) (1 - u) + r (1,v) u \qquad (4.17)$$

again we obtain;

$$Q_2 (0,v) = r (0,v)$$

$$Q_2 (1,v) = r (1,v)$$

A Coons patch is defined as a simple sum of the lofted surfaces in the two directions (Fig 4.8):

$$Q = Q_1 + Q_2$$

$$Q (u,v) = r (u,0) (1 - v) + r (u,1) v + r (o,v) (1 - u) + r (1,v) u \qquad (4.18)$$

However, (4.18) shows that at the corners

$$Q (0,0) = r (0,0) + r (0,0) \text{ etc.}$$

and at the edges

$$Q (0,v) = r (0,0) (1 - v) + r (0,1) v + r (0,v) \text{ etc.} \qquad (4.19)$$

Therefore, if a surface patch $Q_3 (u,v)$ can be found whose boundaries are the unwanted linear interpolations, then the patch can be defined by forming

$Q_1 + Q_2 - Q_3$. From (4.19) it is clear that the unwanted boundary curve for $u = 0$ is

$$r(0,0)\,(1,v) + r(0,1)\,v$$

and for $u = 1$

$$r(1,0)\,(1 - v) + r(1,1)\,v$$

and a further linear interpolation in a direction then gives

$$Q_3(u,v) = r(0,0)\,(1 - u)(1 - v) + r(1,0)\,u\,(1 - v) + r(0,1)\,(1,u)\,v + r(1,1)\,uv \qquad (4.20)$$

The surface Q is then obtained by $Q_1 + Q_2 - Q_3$ which is expressed below in matrix form:

$$Q(u,v) = \begin{bmatrix} 1 - u & u & 1 \end{bmatrix} \begin{bmatrix} -r(0,0) & -r(0,1) & r(0,v) \\ -r(1,0) & -r(1,1) & r(1,v) \\ r(u,0) & r(u,1) & 0 \end{bmatrix} \begin{bmatrix} 1 - v \\ v \\ 1 \end{bmatrix} \qquad (4.21)$$

Linear Coons patch as in (4.21) is the most elementary functions u, 1 - u, v, and 1 - v are called blending functions as they blend the boundary curves to produce the internal surface shape. These linear blending functions can be replaced by other functions. Suppose that 1 - u is replaced by $A_0(u)$ and U by

$A_1(u)$ these blending functions should then satisfy the conditions:

$$1 - A_1 = A_0$$

$$A_0(0) = 1, \quad A_0(1) = 0, \quad A_1(0) = 0$$

and $A_1(1) = 1$

Blending functions are usually chosen to be continuous , therefore polynomials are widely used.

If it is intended to construct a surface of these patches only positional continuity can be obtained on the boundaries. In most practical cases, slope continuity is essential. One method to achieve this is to impose additional conditions on the blending functions such as:

$$\dot{A}_0(0) = \dot{A}_0(1) = \dot{A}_1(0) = \dot{A}_1(1) = 0$$

By so doing, the slope across the boundary, for example;

r (u,0) takes the form

$$Q(u,0)_v = r_v(0,0)\ A_0(u) + r_v(1,0)\ A_1(u)$$

Here the subscript refers to the slope, i.e. $Q(u,0)_u = \dfrac{\partial\,(Q(u,0))}{\partial_u}$

Therefore the slope across the boundaries depends on the end tangent vectors across the boundary and the blending function. Two patches with the same

blending function r2 and common boundarycurve, i.e. $r\,1(u,1) = r2\,(u,0)$ are continuous in slopes if

$$r\,1_u\,(0,1) = K.\,r2_u\,(0,0) \text{ and } r\,1_v\,(1,1) = K.\,r2_v\,(1,0)$$

Generally a patch is the sum of one or more types of Coons patches (58).

$$P\,(u,v) = Q\,(u,v) + R\,(u,v) + S\,(u,v) + \ldots.$$

Q (u,v) is called the first conical form according to Forrest, other types can be seen as correction surfaces altering the internal shape and boundary conditions of the basic patch Q (u,v).

In this approach, continuity of first and second derivatives can be effectively implemented across the boundaries. We have, for slope continuity:
P (u,v) = Q (u,v) + R (u,v). The function R (u,v) will change the slope and higher order boundary conditions without changing the shape of the boundary curves.

Similar to (4.21), R (u,v) is defined as below:

$$R\,(u,v) = [B_0\,(u)\;\; B,(u)\;\; 1] \begin{bmatrix} -r_{uv}\,(0,0) & -r_{uv}\,(0,0) & r_u\,(0,v) & B_0\,(v) \\ -r_{uv}\,(1,0) & -r_{uv}\,(1,1) & r_u\,(1,v) & B_1(v) \\ r_v\,(u,0) & r_v\,(u,1) & 0 & 1 \end{bmatrix} \tag{4.22}$$

B_0 and B_1 are slope bending functions. To ensure the correction surface has zero positioned value on the boundaries but has the necessary boundary

slope the following conditions should be satisfied:

$$B_0(0) = B_0(1) = B_1(0) = B_1(1) = 0$$

and $$\dot{B}_0(0) = 1, \dot{B}_0(1) = 0, \dot{B}_1(0) = 0, \dot{B}_{,1}(1) = 1$$

Finally (4.21) and (4.22) can be combined in a matrix form to give a compact representation:

$$P(u,v) = Q(u,v) + R(u,v) = -\left[-1 A_0(u) \; (A_1(u) \; B_0(u) \; B_1(u)\right] M \begin{bmatrix} -1 \\ A_0(v) \\ A_1(v) \\ B_0(v) \\ B_1(v) \end{bmatrix}$$

Where $$M = \begin{bmatrix} 0 & r(u,0) & r(u,1) & r_v(u,0) & r_v(u,1) \\ r(0,v) & r(0,0) & r(0,1) & r_v(0,0) & r_v(0,1) \\ r(1,v) & r(1,0) & r(1,1) & r_v(1,0) & r_v(1,1) \\ r_u(0,v) & r_u(0,0) & r_u(0,1) & r_{uv}(0,0) & r_{uv}(0,1) \\ r_u(1,v) & r_u(1,0) & r_u(1,1) & r_{uv}(1,0) & r_{uv}(1,1) \end{bmatrix} \quad (4.23)$$

Coons method is very general as well as surface design, it can also be used for surface fitting. Two families of curves can be fitted onto the data points which can then be used as patch boundaries.

Some disadvantages appear when using this method with C.A.D. systems. Three quantities exist; position, tangent and twist vector all must be specified. Corner twist vectors are difficult to comprehend using this method, so are often assumed to be zero. This gives a flatness [59] to the patch corners of a composite surface. Forrest [58] suggests that twist vectors are computed in terms of other data values to overcome this problem.

Lastly, the user has no intuition and cannot predict the effect of a change when creating a surface by changing elements in the defining matrix.

4.3.4.2 Ferguson Surface

As discussed, a Ferguson curve can be shown as:

$$r = r(u) = r(0)\,A_0(u) + r(1)\,A_1(u) + \dot{r}(0)\,B_0(u) + \dot{r}(1)\,B_1(u) \qquad (4.24)$$

Where $A_0(u) = 1 - 3u^2 + 2u^3$

$A_1(u) = 3u^2 - 2u^3$

$B_0(u) = u - 2u^2 + u^3$

$B_1(u) = -u^2 + u^3$ (4.25)

or functions in (4.25) maybe written in matrix form as:

$$F(u) = [\,A_0(u) \quad A_1(u) \quad B_0(u) \quad B_1(u)\,] = [1 \; u \; u^2 \; u^3] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & -2 & 1 & 1 \end{bmatrix}$$

$$= UN \qquad (4.26)$$

In a Ferguson surface, the blending functions are assumed to be the same as elements used in the boundary curves. If the sum of two lofted surfaces substracted by additional elements to satisfy the boundary conditions are used; i.e. the same procedure used to construct Coons patches - then the final result will be:

$$P(u,v) = [\, A_0(u)\ A_1(u)\ B_0(u)\ B_1(u)] \begin{bmatrix} r(0,0) & r(0,1) & r_v(0,0) & r_v(0,1) \\ r(1,0) & r(1,1) & r_v(1,0) & r_v(1,1) \\ r_u(0,0) & r_u(0,1) & r_{uv}(0,0) & r_{uv}(0,1) \\ r(1,0) & r_u(1,1) & r_{uv}(1,0) & r_{uv}(1,1) \end{bmatrix} \begin{bmatrix} A_0(v) \\ A_1(v) \\ B_0(v) \\ B_1(v) \end{bmatrix}$$

(4.27)

$$\text{taking} \begin{bmatrix} A_0(v) \\ A_1(v) \\ B_0(v) \\ B_1(v) \end{bmatrix} = N^T [1\ v\ v^2\ v^3\,]^T$$

(4.27) can be more compactly presented as:

$$P(u,v) = [1\ u\ u^2\ u^3]\ N\ R1\ N^T\ [1\ v\ v^2\ v^3]^T = U\ N\ R1\ N^T\ V^T$$

where N is the square matrix in (4.26) and R1 the square matrix in (4.27).

Ferguson surface is actually a particular case of Coons method as boundary curves are limited to polynomials containing the same elements as blending functions. Recalling the previous sections to eliminate some of the problems,

Ferguson [40] originally produced a simplified version known as an F-patch which is sometimes used and where twist vectors are assumed to be zero, therefore, only first order continuity is possible.

4.3.4.2.1 APT Surface Fitting Routine

A practical application of F-patch is in F-Mill [60] used in APT for surface fitting. In F-patch [54] twist vectors are assumed zero and gradients r_u and r_v at mesh intersections are calculated using positional data. Directions are taken as parallel to the chord line joining preceeding and following data points. Magnitude is taken as the length of the chord joining the point concerned to the preceeding or the following point. To avoid loops in the curve in case of a high magnitude for the tangent the smaller value is chosen.

4.3.4.3 Bezier Surfaces

The Bezier like its corresponding curve design is based on an approximation, not fitting, of a set of data points. The Bezier patch is designed in terms of a 'characteristic polyhedron'. A cubic patch is defined by its 16 vertices. The patch is actually an approximation to the polyhedron only eight points of which lie on the surface.

It is usually assumed that in surface description, information such as position, tangent and twist vectors is available. The information may be difficult to provide. The user doesn't have adequate 'feel' indicating the effect of changes in the defining input on the shape of the curve. The Bezier surface description is one of the most successful methods to overcome these difficulties. Gradient and twist vectors need not be specified and the configuration of the polyhedron gives the user a good indication of the general shape of the patch.

Bezier patch using the binomial representation given in section 4.3.3.2 for a Bezier curve can be represented [6] (Fig 4.9):

$$P(u,v) = [(1-u)^3 \;\; 3u(1-u)^2 \;\; 3u^2(1-u) \;\; u^3]\, R^2 \begin{bmatrix} 1(1-v)^3 \\ 3v(1-v)^2 \\ 3v^2(1-v) \\ v^3 \end{bmatrix}$$

$$\text{where } R2 = \begin{bmatrix} r_{00} & r_{01} & r_{02} & r_{03} \\ r_{10} & r_{11} & r_{12} & r_{13} \\ r_{20} & r_{21} & r_{22} & r_{23} \\ r_{30} & r_{31} & r_{32} & r_{33} \end{bmatrix} \qquad (4.28)$$

In matrix representation:

$$[(1-u)^3 \;\; 3u(1-u)^2 \;\; 3u^2(1-u) \;\; u^3] = [1 \;\; u \;\; u^2 \;\; u_3] \begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & -3 & 3 & 1 \end{bmatrix}$$

$$= UM \qquad (4.29)$$

Therefore,

$$P(u,v) = U\,M\,R2\;M^T[1 \;\; v \;\; v^2 \;\; v^3]^T = U\,M\;R2\;M^T\;V^T$$

Recalling 4.3.4.2 a Ferguson surface can be shown:

$$P(u,v) = U\ N\ R1\ N^T\ V^T$$

Comparing this with Bezier's

$$U\ N\ R1\ N^T\ V^T = U\ M\ R2\ M^T\ V^T$$

Hence $R1 = (N^{-1}M)\ R2\ (N^{-1}M)^T$ *

or

$$R1 = \begin{bmatrix} r(0,0) & r(0,1) & r_v(0,0) & r_v(0,1) \\ r(1,0) & r(1,1) & r_v(1,0) & r_v(1,1) \\ r_u(0,0) & r_u(0,1) & r_v(0,0) & r_{uv}(0,1) \\ r_u(1,0) & r_u(1,1) & r_{uv}(1,0) & r_{uv}(1,1) \end{bmatrix}$$

$$= \begin{bmatrix} r_{00} & r_{03} & 3(r_{01} - r_{00}) & 3(r_{03} - r_{02}) \\ r_{30} & r_{33} & 3(r_{31} - r_{30}) & 3(r_{33} - r_{32}) \\ 3(r_{10} - r_{00}) & 3(r_{13} - r_{03}) & 9(r_{00} - r_{10} - r_{01} + r_{11}) & 9(r_{02} - r_{12} - r_{03} + r_{13}) \\ 3(r_{30} - r_{20}) & 3(r_{33} - r_{23}) & 9(r_{20} - r_{30} - r_{21} + r_{31}) & 9(r_{22} - r_{32} - r_{23} + r_{33}) \end{bmatrix}$$

It can be seen that gradient and twist vectors are expressed in terms of the vertices of the characteristic polyhedron. Therefore, unsatisfactory assumptions such as $r_{uv} = 0$ are avoided.

---

* N^{-1} means Inverse of matrix N

To achieve continuity on the border of two adjacent patches of a composite surface, take the patches as r1 (u,v) and r2 (u,v).

Positioned continuity across the boundary will result if r1 (1,v) = r2 (0,v) for all values of v such that $0 \leq v \leq 1$. This is achieved when the 2 patches have a common boundary polygon between the two characteristic polyhedrons i.e.

$$r1_{3i} = r2_{0i} \quad i = 0, 1, 2, 3$$

Slope continuity can be achieved in two ways. First by Collineraity of four pairs of polyhedron edges meeting at the boundary. Fig (4.10 (a)) secondly by coplanarity of three edges as shown in Fig (4.10 (b)).

If higher degrees of continuity are required, cartesian product representation [54, 61] of Bezier surfaces should be used. Particularly the surface is given by:

$$P(u,v) = \sum_{i=0}^{n} \sum_{J=0}^{m} B_{n,i}(u)\ B_{m,i}(v)\ r_{ij}$$

Where $B_{n,i}(u)$ and $B_{m,j}(v)$ are Bernstein basis functions of degree n and m.

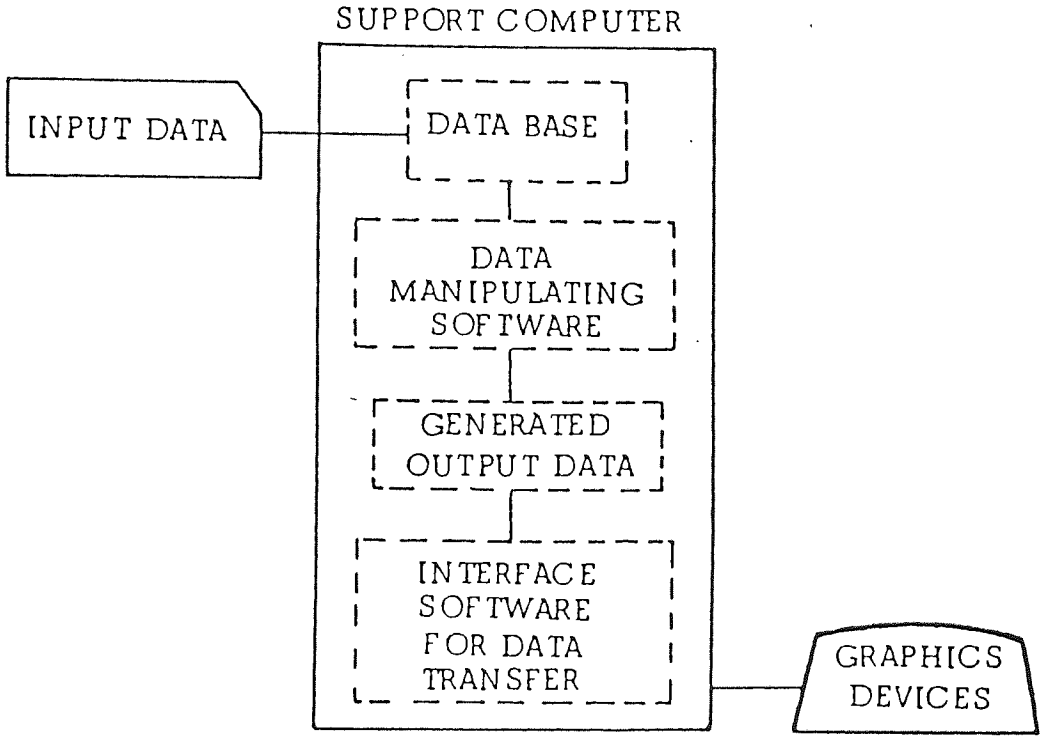


Fig. (4.1) Block Diagram of a Computer Aided Design System.

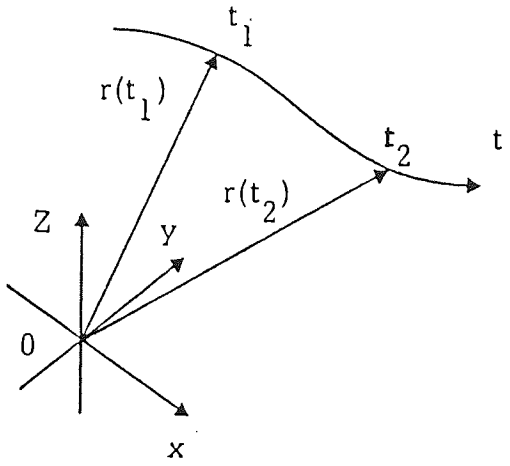


Fig. (4.2) Parametric Method in Curve Description

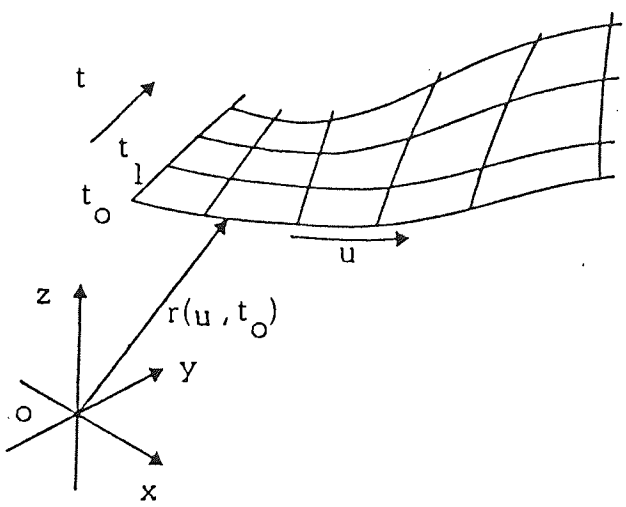


Fig. (4.3) Parametric Method in Surface Description

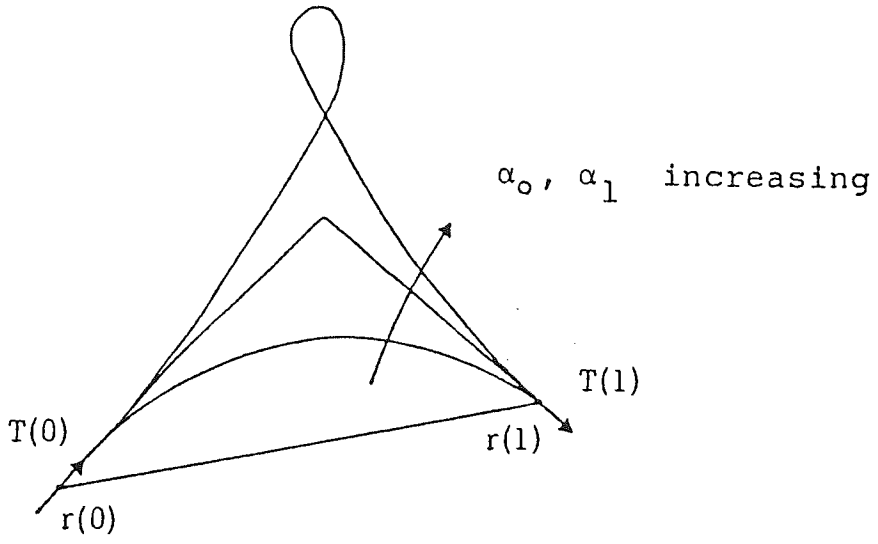
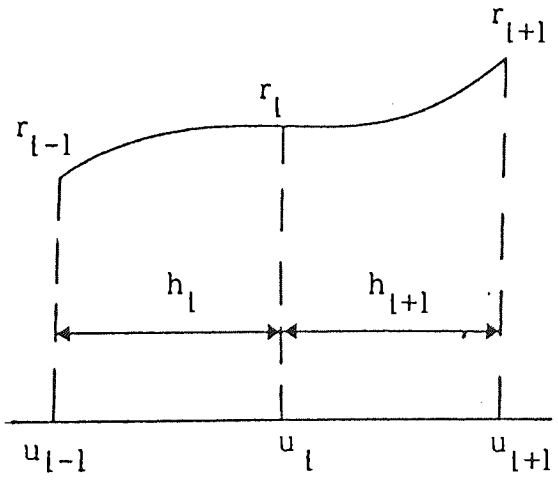


Fig. (4.4) The Effect of End Tangents on Ferguson's Curve



$h = u_l - u_{l-1}$

Fig. (4.5) Construction of Spline Curves

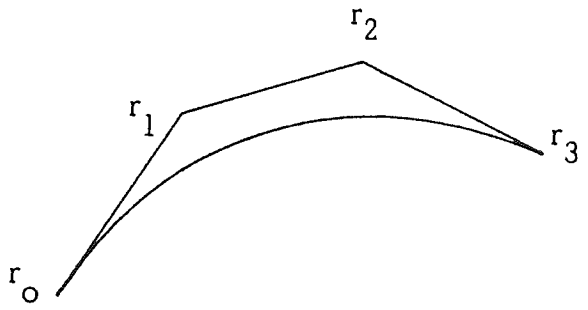
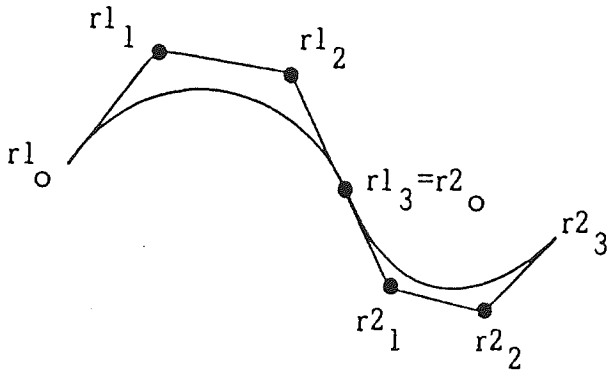


Fig. (4.6) A Bezier Curve Segment



● Coplanar Points for Curvature Continuity

Fig. (4.7) Continuity of Bezier Curve Segments

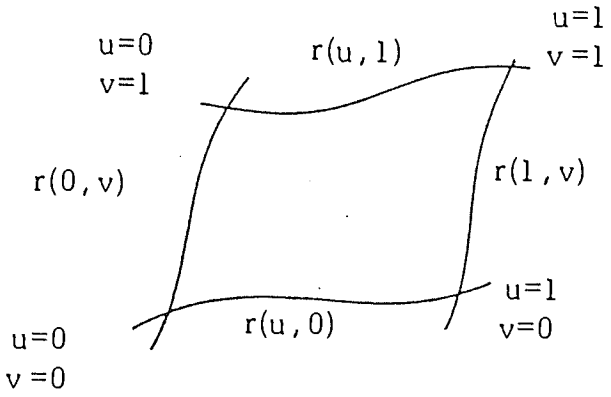
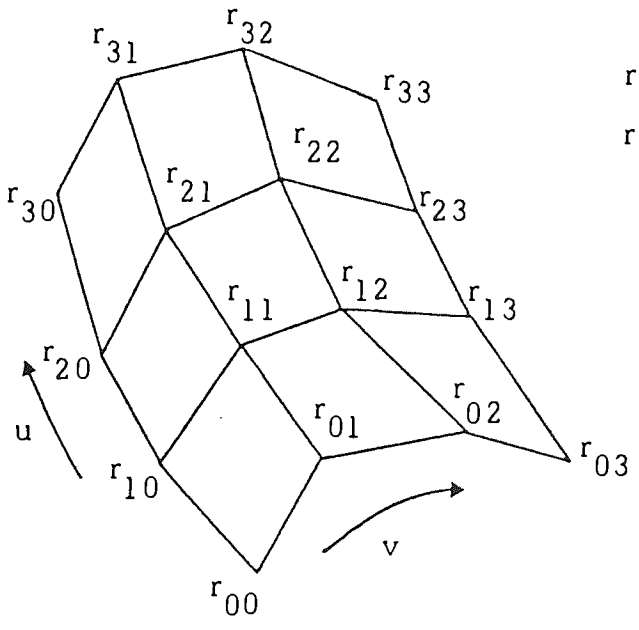


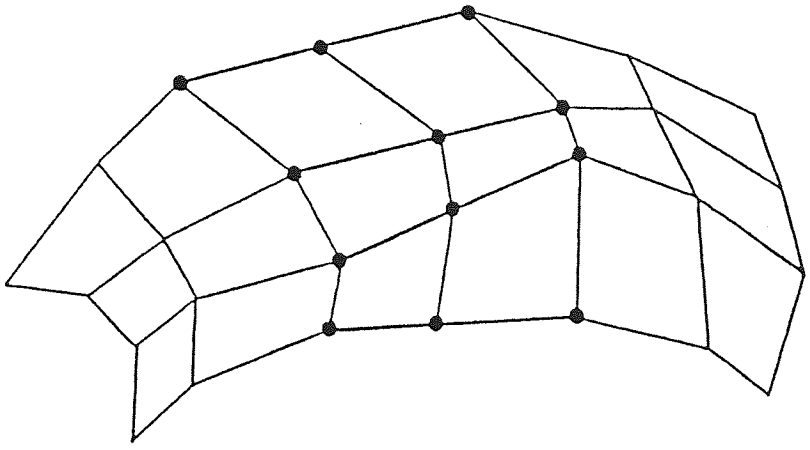
Fig. (4.8) A Parametric Surface Patch



Vertices lying on the surface are :

r_{00} r_{10} r_{20} r_{30}
r_{03} r_{13} r_{23} r_{33}

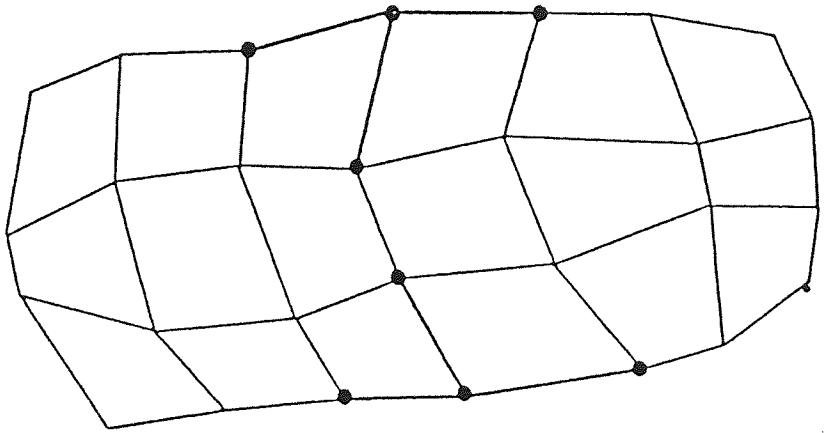
Fig. (4.9) The Characteristic Polyhedron for a Cubic Bezier Patch



Patch 1 Patch 2

Collinear Edges

(a)



Patch 1 Patch 2

Coplanar Edges

(b)

Fig. (4.10) Characteristic Polyhedra for Construction of Composite Surfaces with Positional and Slope Continuity

CHAPTER 5

CHAPTER 5

DEVELOPMENT OF THE COMPUTER AIDED SURFACE DESIGN PACKAGE "IBCSURF"

5.1 INTRODUCTION TO "IBCSURF"

Computer Aided Design of curved or free form surfaces is based on a mathematical description of the shape from which instructions or drawing can be given to a numerically controlled machine tool. There are three types of surface representation (61). First the cartesion product surface which interpolates only through point data. Examples of cartesian product surfaces are Bezier and B-spline surfaces.

Bernstein and B-spline polynomials are bivariate interpolants. The second type of surface representation is lofting. This utilizes one family of parametric curves over which the surface is interpolated. The third type is the transfinite surface representation which results from interpolation through two families of curves. The interpolants here are bi-orthonormals such as cardinal spline functions. Coons surfaces are a special form of the transfinite approach employing a Hermitian patch-like interplant.

"IBCSURF" an experimental surface description method has been developed in the present work and is explained in the subsequent sections. Vector valued parametric representation is adopted and the basic scheme employed is based on lofting technique. The cross-sectional curves are representated by uniform B-splines where curve order can be selected by the user as a 'handle' to achieve the desired shape for sectional curves. The

parametric method used, enables planar/non-planar, closed or open curves to be designed in 3D space without the need for sectional curves to be parallel. The surface between

cross-sectional curves is interpolated by cubic cardinal splines. Ring-like objects with periodic surfaces can be designed simply using closed parametric cardinal splines.

Programs in this package are written in FORTRAN 77 (62) and run on the VAX 11/750 super mini- computer with graphics facilities (63) for displaying drawings interactively.

5.2 MATHEMATICAL BASIS OF "IBCSURF"

5.2.1 Definition of Spline Functions

Given an increasing set of real numbers X_0, X_1, -------X_L, a function S is called a (polynomial) spline function of degree n - 1 (order n) in the following two conditions are satisfied.

1) In each interval (X_i, X_{i+1}), S is a polynomial of degree n - 1.

2) S and its derivatives of order 1,.2, ----n - 2 are continuous everywhere.

The points X_i are called the Knots, and the i^{th} span of the spline lies in the interval (X_i, X_{i+1}).

5.2.2 B-Spline Basis

Truncated power functions, cardinal splines and B-splines (64) are best known and given most attention as representative basis functions or splines.

In Fig (5.1) B-Spline basis functions are given for progressively increasing degree and knots at integers. B-spline is termed 'spline of minimal support as Fig (5.1) shows for degree n - 1. Basis functions have a finite local support function of width n, this is the number of spans over which a spline is

non-zero. Basis are local in that at every point, only a fixed number (equal to the order) of the B-spline is non-zero.

5.2.2.1 B-Spline versus Bernstein Basis Function

Schoenberg [53] first introduced B-spline in a paper dealing with smoothing and approximation of statistical data. From a mathematical viewpoint basis or weighting functions relate a curve to a defining polygon.

As mentioned in 4.3.3.4, Bernstein basis function produces Bezier curve and the B-spline basis is proved to contain Bernstein basis as a special case.

Bernstein basis function has several useful properties, but its characteristics limit the flexibility of the curve produced. First the order of the curve is determined by the number of defining polygon vertices. A cubic curve, for example, must be defined by a polygon with four vertices and a polygon with six vertices always produces a fifth degree curve. However, the number of vertices must be reduced or increased to reduce or increase the order of the curve.

Secondly, Bernstein basis function (5.1) is global or it is non-zero over an entire span of the curve.

$$B_{n,i}(u) = \frac{n!}{i!\,(n-i)!}\, u^{i}\,(1-u)^{n-i} \quad 0 < u < 1 \qquad (5.1)$$

where n is the degree of the polynomial and i the particular vertex.

Any point on a Bezier curve results from weighting the values of all defining vertices. A change in one vertex affects all, therefore it is impossible to make local changes within one span.

The Bezier curve is tangent to the polyhedron sides at the end points and it is possible to change the shape of a fine point polygon without altering the direction of the end slopes. Again, any change will affect the whole curve shape.

B-spline function is non-global as each vertex is associated with a unique B-spline basis function which is non-zero over a limited number of spans. A change in vertex changes only the local parameter values within which the function is non-zero.

B-spline basis also allows the curve shape to be altered without changing the number of polygon vertices as the curve order is not fixed by the number of polygon sides.

5.2.2.2 Evaluation of B-spline Basis Functions

B-spline basis can be explained in many ways but Cox [65] and deBoor [66] gives a stable and efficient way of evaluating B-spline basis functions.

For the i^{th} normalised B-spline basis curve or order n the weighting function $N_{i,n}(u)$ are defined by the recursive formulae:

$$N_{i,1}(u) = \begin{cases} 1 & x_i \leq u \leq x_{i+1} \\ 0 & \text{Otherwise} \end{cases}$$

$$N_{i,n}(u) = \frac{(u - x_i)\, N_{i,n-1}(u)}{x_{i+n-1} - x_i}$$

$$+ \frac{(x_{i+n} - u)\, N_{i+1,n-1}(u)}{x_{i+n} - x_{i+1}} \qquad \text{for } n > 1 \qquad (5.2)$$

The convention $0/0 = 0$ is assumed here.

The values of x_i, are elements of a knot vector. A knot vector $x = (x_0, x_L)$ may contain identical knots up to a multiplicity n. Multiple interior knots reduce the degree of differentiability of basis functions with M multiple knots at X_j, $N_{i,n}$ will have differentiability up to $C^{(n-m-1)}$ at this point.

Multiple knots also mean the induction of spans with zero length giving a reduction in the width of support of the basis function. Multiplicity M of the interiod knots generate M - 1 zero length spans.

A knot vector is a series of real-integers X_i such that $X_i \leq x_{i+1}$ for all X_i. For non-periodic basis multiple knots with multiplicity equal to the order of the curve must be defined at each end of the knot set. For a periodic basis, the know vector consists of a series of equally spanned numbers such as X = (0, 1 ....L) or a cyclic shift of the above.

In Fig (5.2) basis splines for a non-periodic and a periodic curve are given.

In a parametric curve, the values of X_i are considered to be parametric knots. They can be used to show the range of the parameter u which is used to generate a B-spline curve. For example, the vector (0 1 2 3) indicates that the parameter u varies from 0 to 3.

5.2.3. B-Spline Curve

A B-spline curve is an approximation to a defining polygon. Taking r (u) to be a position vector along the curve as a function of parameter u, a curve generated using B-spline basis is given by:

$$r(u) = \sum_{i=0}^{m} P_i \; N_{i,n}(u) \qquad (5.3)$$

Where P_i is m + 1 vertices of the defining polygon, and n the order of the curve.

The first step in generating a B-spline curve is to define the knot vectors. In this work the parametric B-spline basis as defined by Gordon and Reisenfeld [47] was used. Here knots are restricted to integer values.

The order of the curve and the number of spans of the defining polygon are reflected in the knot vector used to generate the curve. A duplicate intermediate knot value shows a multiple vertex (span of zero length) occurs at a point. When there are no multiplevertices, the parameter u varies from 0 to m - n + 2 over the cure. For example, for a further order curve, defined by vertices (P_0, P_1, P_2, P_3, P_4), this is given by $0 \leq u \leq 4 - 4 + 2 = 2$, and

the complete set of knots with a multiplicity of 4 at each end is given by:

$$\underbrace{0000}_{n} \quad 1 \quad \underbrace{2222}_{n}$$

For a closed (periodic) curve a cyclic set of knots must be defined (that is, $X_L = X_0$, $X_{L+1} = X_1$ ...etc), no multiple knots are necessary at the end points in this case.

5.2.4 Interpolating with B-splines

When interpolating with B-splines, it is usual to construct splines of order n on a set of points X_o, X_1, ...X_L that is expressed as a sum of multiples of B-splines defined on the same set of points now used as knots and extended by (n-1) additional knots at each end. Expression (5.3) can be rewritten in the form:

$$S(x) = \sum_{i=0} C_i \; N_{i,n} \; (x) \qquad (5.4)$$

where S(x) is any spline of degree n - 1 on the given set of points and C_i is numerical coefficients.

If the values to be interpolated at x_0, x_1....x_L are Z_o, Z_1, ...Z_L, then

$$Z_J = S(x_J) = \sum_{i=0} C_i \; N_{i,n} \; (x_J)$$

For simplicity if a cubic case is considered and the values of $N_{i,n}(x_j)$ are calculated using deBoor algorithm [64], for each X_J:

(J = O, ... L) only three values $N_{i,4}\ (x_J)$, $N_{i+1,4}\ (x_J)$, $N_{i+2,4}\ (x_J)$ are non-zero. Therefore from (5.4) we obtain;

$$Z_J = S(x_J) = C_i N_{i,4}(x_J) + C_{i+1} N_{i+1,4}(x_J) + C_{i+2} N_{i+2,4}(x_J)$$

in which C's are unknown. There will be one linear equation for each of the (L + 1) knots ($x_0, x_1 \ldots x_L$. There are (L + 3) B-splines ($N_{0,4}$ (x0), $N_{1,4}(x_i), \ldots N_{L+2,4}(X_2)$) so(L + 3) coefficients $C_o, C_1, \ldots C_{L+2}$ are to to be determined. Two extra items of information are requried to determine S(x) completely . These are usually derivative values. For example, if

$$\ddot{Z}_J = \ddot{S}(X_J) = 0, \text{ then for } J = 0;$$

$$\ddot{Z}_0 = \ddot{S}(x_0) = C; \ddot{N}_{i,4}(x_0) + C_{i+1} \ddot{N}_{i+1,4}(x_0) + C_{i+2} \ddot{N}_{i+2,4}(x_0)$$

given similar conditions at X_L, there is sufficient information to calculate coefficient values. Values of B-spline derivatives can be evaluated using deBoors method.

When all coefficients have been determined , then expression (5.4) can be used again to find the function values for any X.

5.2.4.1 Inversion Technique

This technique was introduced by Gordon and Riesenfeld [44] and solves the problem of interpolation more efficiently. This method used

in this work, allows degree curve to be interpolated on a given set of points without any need to put extra conditions at end points.

Given a function f: [O,K] $\rightarrow\Delta$ R, the spline approximation of order n to f is defined as:

$$S_n(s) = \sum_i f(\zeta_i)\, N_{i,n}(s) \qquad (5.5)$$

where

$$\zeta_i = \frac{1}{n-1}\,(x_{i+1} + X_{i+2} + X_{i+n-1}) \qquad (5.6)$$

the ζ_i's are the nodes.

If polygon P is viewed as a piecewise linear function F, so that the B-spline curve is the parametric B-spline approximation to F, then

$$F(\zeta_i) = P_i \qquad i = 0, 1, \ldots\ldots m \qquad (5.7)$$

Suppose a set of points lie on spline S, the problem is to find the unique polygon $P_o, P_1, \text{---} P_m$, the B-spline approximation corresponding to S. The nodes can easily be found using (5.6) by determining the knot set $X_0, X_1, \ldots X_L$ as required by S, now knots and nodes available to matrix N as defined below can be calculated.

$$N = N_{J,n}(\zeta_i) \qquad 0 \le i, \quad J \le m \qquad (5.8)$$

from (5.5), (5.7) and (5.8) a polygon can be found to satisfy

$$N \cdot (P_0, P_1 \ \ldots \ P_m)^T = (S(\zeta_0) \ \ldots \ S(\zeta\ 1) \ldots S(\zeta_m))^T.$$

The problem of interpolation can be solved by replacing the parametric values $S(\zeta_o)$, $S(\zeta_1)$, ... $S(\zeta_m)$ by the actual cartesian point data to be interpolated.

5.2.5 Cubic Cardinal Splines

Here particularly, the method of Nilson, E.N. [66] devised to construct cubic cardinal splines is examined. Two cubic arcs as defined below are used:

$$T1(\theta) = \theta - \theta^3, \quad T2(\theta) = (\theta^2 - \theta^3)/2$$

$R(\theta)$ with parameter α is also introduced,

$$R(\theta) = 1 - (\alpha + 3)\theta^2 + (\alpha + 2)\theta^3 \quad O \leq \theta \leq 1$$

Begin with arc $R(\theta)$ over [O, 1] to construct any cubic cardinal spline on a uniform mesh. Attach arcs comprising suitable linear combinations of $T1(\theta)$ and $T2(\theta)$.

Considering a mesh consisting of integer points $0, \pm 1, \pm 2, \ldots$ splines $A(\theta)$ with $A(0) = 1$ and $A(\pm n) = 0$; $n = 1, 2 \ldots$ are as follows:

on [0,1] $A(\theta) = R(\theta)$,

on [1,2] $A(\theta) = \dot{A}(1)\, T1\,(\theta - 1) + \ddot{A}(1).\, T_2(\theta - 1)$,

on [2,3] $A(\theta) = \dot{A}(2)\, T1\,(\theta - 2) + \ddot{A}(2)\, T_2(\theta - 2)$,

with $A(\theta) = A(-\theta)$ it is proved that on $[n, n+1]$, $n \geq 1$, $A(\theta)$ is defined by:

$$A(\theta) = (\alpha_n\ \alpha - 3\ \alpha_{n-1})\, T1\,(\theta - n) +$$

$$[2(2\ \alpha_n + \alpha_{n-1})\ \alpha + 6(\alpha_n + 2\ \alpha_{n-1})]T2\,(\theta - n) \qquad (5.9)$$

where $\alpha_n + 4\ \alpha_{n-1} + \alpha_{n-2} = 0$, $\alpha_0 = 0$ and $\alpha_1 = 1$

choice of constant α is determined by convenience.

A non-periodic spline on the mesh Δ: $X_0 < X_1 \ldots < X_N$ can be defined as:

$$S(x) = \sum_{i=0}^{N} f_i A_i(x) + b0\ B_0(x) + b_N B_N(x) \qquad (5.10)$$

Where f; (i = 0, ...N) are the ordinates.

and $B_0(X)$ and $B_N(X)$ curdinal splines for implementing the end conditions.

For "free -b ending ' ends, i.e.

$$\frac{d^2 S(X)}{dX^2} = 0 \text{ at, } X = X_0$$

and $X = X_N$ (5.10) can be reduced to

$$S(x) = \sum_{i=0}^{N} f_i \, A_i(x) \qquad (5.11)$$

This can be applied to periodic or non-periodic provided that $f_n = f_0$ for a periodical case.

Nilson [68] in developing this algorithm takes the value of to be B-splines in forming the surface.

$$= \frac{3\lambda - \lambda^{N-1},}{1 - \lambda^N} \qquad \lambda = -2 + 3$$

So on $[n, n+1]$, $1 \le n \le N-1$ using $\lambda^n = \alpha_n \lambda - \alpha_{n-1}$,

$$A(\theta) = \frac{3\lambda^n}{1-\lambda^N} \quad \frac{T(\theta - n) + 3\lambda^{N-n-1}}{1 - \lambda^N} \quad T(1 - (\theta - n)) \qquad (5.12)$$

and on $[0,1]$

$$A(\theta) = (1-\theta)^3 + 3\left[\frac{1}{1-\lambda^N} T(\theta) + \frac{N-1}{1-\lambda^N} T(1-\theta)\right] \qquad (5.13)$$

Where $T(\theta) = \theta - (\lambda + 2)\ \theta^2 + (\lambda + 1)\theta^3$

Where mesh is uniform Δ: XO, X_1, ...X_N, $h = X_J - X_{J-1}$, obtaining for x on $[X_K, X_{K+1}]$, $(K = 0, 1, \ldots N)$, and defining $\theta = (X - X_K) / h$,

the Expression:

$$S(x) = \frac{T(\theta)}{1-\lambda^N}\left[\sigma_{N-1}\ \lambda^{K+1} + \sigma_K\ (1-\lambda^N)\right] + (1-\theta)^3\ f_K$$

$$+ \frac{T(1-\theta)}{1-\lambda^N}\left[\tau_1\ \lambda^{N-K} + \tau_{K+1}\ (1-\lambda^N)\right] + \theta^3\ f_{K+1} \qquad (5.14)$$

Where $\sigma_K = 3[f_0\ \lambda^K + \ldots\ldots f_{K-1}\lambda + f_K]$,

and $\tau_K = 3[f_K + f_{K+1}\lambda + \ldots\ldots f_N\lambda^{N-K}]$

The parametric from of this expression was used to interpolate the cross sectional curves designed by the B - splines in forming the surface.

5.3 CURVE DESIGN PROCEDURE

In designing B-spline curves in the routines 'BSCURV", the most general form as shown by Expression (5.15) has been implemented and basis functions are evaluated using the recursive formulae given by (5.2).

$$r(u) = \sum_{i=0}^{M} P_i\ N_{i,n}(u) \qquad (5.15)$$

The method allows closed or open planner or space curves of a different order to be designed

'BSCURV' is written as nested subroutines where knot vectors for closed or open curves are defined in their respective subroutines using information about the defining

polygon. In both cases, knots are given at integers. For an open curve, the knot set $X_0, X_1, \ldots X_{m+n}$ is defined in which

$$X_0 = X_1 = \ldots X_{n-1} = 0,$$

$$\text{and } X_m = X_{m+1} = \ldots\ldots X_{m+n} = U_{max}$$

Multiple vertices are checked and corresponding multiple knots are inserted. Now, without any multiple vertex

$$X_i = X_{i-1} + 1, \quad i = n,m.$$

otherwise, if $P_{i-n} = P_{i-n+1}$

$$X_i = X_{i-1}$$

A closed curve method, slightly different from that described in section 5.2.3. was taken. Here, for the n^{th} order curve having the control polygon ($P_0, P_1 \ldots P_m$), $P_{m+1} = P_0$, $P_{m+2} = P_1$, ..., and $P_{m+n-1} = P_{n-2}$ are defined, therefore no multiple knots at the beginning and the end of the knot set are given, the new defining polygon (P_0, $P_1, \ldots P_m$, $P_{m+1} = P_0 \ldots P_{m+n-1} = P_{n-2}$) is regarded as open.

The first stage of designing a B-spline curve using BSCURV is to define or sketch the defning polygon. The numbers and co-ordinates of the polygon vertices with the order of the desired curve are input to BSCURV. User can also specify the number of points generated along the curve. Range and parameterisation (0 to Umax) divided by this

number will be the parametric intervals at which the B-spline curve is evaluated.

5.4 CURVE FITTING

In design problems, functional, aesthetic, or scientific constraints give some information affecting the shape of the object. This information may be some empirical data gained by statistically or experimentally regarding the geometrical shape, a physical model, photographic information, drawings sketches etc. Computer Aided Design based on a mathematical representation of a shape must be able to use the above data to produce a surface as close as possible to the existing shape. So curve and surface fitting are a necessary part of every CAD system. The curve is regarded as the basic element of design only curve fitting routines have been developed in this work. It was decided to base routines on the mathematics of B-spline because 1) the result could easily be used in the surface design process 2) schemes were very efficient.

5.4.1 Least Squares B-Spline Curve Fitting

The problem is to fit a B-spline curve to a set of data points (X_J, Y_J), J = 1,2 ..,.L. Assuming the spline S(x) is constructed on knot set $\lambda_1, \lambda_2, ..., \lambda_m$ where $\lambda_1 < \lambda_2 ..... < \lambda_m)$, $\lambda_{2-n} = \lambda_0 = \lambda_1$, and $\lambda_m = \lambda_{m+1} = \lambda_{m+n-1}$ need to be defined where n is the order of the B-spline.

With this augmented set of knots we can define m + n basis splines $N_{i,n}(x)$, i = 1, 2. .. m + n . According to Curry and Schoenberg [41] S(x) in the range

$X_1 < X < X_L$ can be shown in the form:

$$S(x) = \sum_{i=1}^{m+n} C_i \quad N_{i,n}(x)$$

The curve fitting problem is then a matter of finding values for the coefficients C_i as the least squares solution to the observation equations (67)

$$S(x_J) = \sum_{i=1}^{m+n} C_i \quad N_{i,n}(x_J) = Y_J \quad J = 1,2,....L \quad (5.16)$$

That is to minimise

$$\sum_{J=1}^{L} (S(x_J) - Y_J)^2$$

(5.16) may be written in matrix notation as

$$AC = Y \qquad (5.17)$$

Where A is the matrix Jx (m+n) whose element in column i row J is $N_{i,n}(x_J)$, and C and Y are the column vectors with elements C_i and Y_J respectively.

In program 'BSLSQ' for least squares B-spline curve fitting, knots are chosen at random within the data range ($\lambda_1 = X_1$, $\lambda_m = X_L$). Selection of the most suitable set of knots and the number of intervals is arrived at by trial and error, experience and general knowledge of the curve shape as shown by the data. BSLSQ is written as a series of nested subroutines. X and Y co-ordinates of data points, the knot set and order of curve to fit the data are all input to the subroutine 'COEFF' which calculates coefficients C_i . To evaluate the B-spline curve at a point, X_2

subroutine FUNCAL' is used. Calculation of basis functions are carried out in separate subroutines used by COEFF and FUNCAL.

Cholesky's elimination method was used to solve the simultaneous equation system of the least squares problem. The method is economical on the number of arithmetic operations and is therefore, the fastest basic elemination method [68].

n elements are adjacent and are non-zero in each row of matrix A. Where data points are arranged in the order of increasing X then the first n element in any row will never lie more to the left than that of the previous row. It is useful, to simplify the computer program, to refer to this as a band structure.

Defining λ_i's at equal intervals, the coefficients C_i give the Y values for defining polygon vertices at points λ_i in a two dimensional curtesian system which could be used later as input to the surface design program.

5.4.2 Implementation of Inversion Technique in Curve Fitting

It is necessary to locate and establish polygon vertices and data points through which a B-spline approximation will pass. If the data is selected points on an existing curve the number and position of selected points will affect the closeness of the approximated curve to the original.

To simplify the technique in use, the uniform B-spline method as discussed in 5.4.4.1 was used. B-spline curves of different order can be interpolated to the data points. A parametric method allows the deformination of defining polygons to non-planar and closed curves.

Input to subroute 'INVERT' is the number and coordinates of the data points and the order of the curve. Within 'INVERT' coordinate values of the vertices are calculated. A set of uniform knots must be defined to evaluate B-spline bases and the nodes (ζ_i). The determining factors are the number of data points and the order of the curve. The number of vertices evaluated by this method equals the number of selected input points. In a least squares fit' the number of vertices or coefficients C_i can be determined independently of the number of input points.

5.5 SURFACE DESIGN PROCEDURE

As stated in 5.2.5 cardinal spline curves can be defined by:

$$S(X) = \sum_{i=0}^{N} f_i \; A_i(x)$$

where f_i, ($i = 0, ...N$) are the ordinates and the $A_i(x)$ the cardinal spline basis functions in a parametric form with parameter V.

$$S(V) = \sum_{i=0}^{N} f_i \; A_i \; (V)$$

Let $B_i(u)$ be the parametric representation of the i^{th} sectional B-spline curve in three dimensional space. At a parametric position, U_J, the values of curve $B_i(u)$ are shown as $B_i(U_J)$. Replacing f_i with $B_i(U_J)$ one will have:

$$S_{uJ}(V) = \sum_{i=0}^{N} B_i \; (U_J) \; A_i \; (V)$$

Denoting $S_{uJ}(V)$ symbolically by $Q(U_J, V)$ and generalising the case for all values of u,

one may write:

$$Q(u,v) = \sum_{i=0}^{N} B_i(u) \; A_i(v) \qquad (5.18)$$

Here u means the same as in the definition of B-spline curves; i.e. each sectional curve is a function of parameter u at a constant value of V.

$$Q(u,V_i) = B_i(u)$$

Obviously (5.18) represents a two dimensional parametric space or a surface resulting from interpolation of the sectional curves using curdinal spline basis functions. Because cardinal spline curves have the characteristic property that the spline contains the controlling points, the surface will pass through all sectional curves.

Continuity of surface derivatives is dependent on the degree of curves used in defining sectional curves and degree of splines that interpolated them.

There is no need for the sectional curves to be parallel or planar.

It points $((U_J, V_i),\ = 1, n, i = 1, m)$ are generated at equal parametric distances using expression (5.18) then directly used in graphic displays, this may give a misleading visual representation of the surface because the number and location of defining polygon vertices are the major factor in parameterisation of sectional B-spline curves. Equal parametric distances on a curve may produce a completely uneven distributed of points at varying arc lengths as a result Fig (5.4).

Parameterisation depends on the ordering of polygon vertices (in closed corners), or depending on which vertex is regarded as first in the defining polygon, a different result

will be produced.

So, points identified by parametric value U_J could be non-planar, or even lie at opposite sides of the surface over a set of sectional curves . Cardinal splines joining these points. Q (U_j, V) may twist on the surface so that, a wrong impression is produced on a graphical display.

Avoiding this problem requires alternative strategies. Usually in a graphical surface representation , heights at the nodes of a finely spaced rectangular grid of points in the (X,Y) plane is produced so lines joining the adjoining points are isometrically projected. A drawing of the surface requires projected surface lines to be about equally spaced but this does not usually happen in the case of isoparametric lines.

'IBCSURF' tries to overcome these problems and simplify the cutter centre-line path. Using 'IBCSURF' each sectional curve is defined then digitised and a number of points of equal arc intervals are selected by the user and will be the same for all sectional curves. Curdinal splines are generated to interpolate selected points on the contours they pass through.

Curdinal spline curves are divided into equal arc intervals and intermediate contours produced. Finally , this mesh of points can be used to generate a centre-line centre path, or joined together and projected giving a resulting drawing of the shape. In 'IBCSURF' package this method is available for surface design.

5.6 JUSTIFICATION OF APPROACH

The choice of mathematical representation in Computer Aided Design depends on the application for the curve or surface design. usually simplicity, flexibility and means of

enforcing continuity of derivatives in the surface are featured in common. The C.A.D. system ideally should not require much mathematical knowledge of the formulation used, but he should be able to control curve and surface shape by control parameters offered. Modification of curves and surfaces should be easy and the effects of such modifications predictable.

The main problem in 3-dimensional surface design is man-machine communication. 3-D interactions are dealt with by typing a command into the keyboard. It is possible to input some 3-dimensional data points but using available hardware equipment it is very difficult to input a three dimensional curve. In some systems, positional data is specified using a tabled stylus or light pen. This is mainly used for 2-D or orthographic and perspective projections of 3-D objects. Effects have been made to use non-standard devices such as head mounted displays [69,70] and a 'wand' or 3-D input device enabling the designer to 'walk about' in a 3-D environment containing 3-D objects which can be modified using the wand.

The more important surface design methods such as Fergusons, Coons' patch design Bezier and B-spline surfaces (as reviewed in Chapter4) have some draw backs . Ferguson and Coons' methods do not allow user intuition so he is not readily able to predict the effect on the final shape of a change in the defining input. The more advanced Bezier and B-spline surface approaches allow some modification to be carried out with interaction in 3-dimensional space. Even given effective manipulative communication, this would be difficult to use when comparing the resulting approximation to the existing shape because of a lack of information about the original shape definition.

Lofting techniques by-pass many difficulties involved in communicating three dimensional data, as curves are regarded as the basic data elements for surface design. If limited to planar sections, available hardware devices can efficiently be used to input 2-D data and manipulate the resulting curves. This method has greater potential for the

comparison of approximations to the original as well as giving better manipulative control over the resulting shape.

In the lofting method of surface description, the manipulative power of the curve design procedure is at the centre of attention. Most mathematical representations of curves and surfaces so far devised are based on piecewise methods so the shape of real objects is generally non-analytical and normally, the shape is unaffected by local conditions. Design proceeds by a series of local changes, analytic functions do not have the desired properties, but piecewise functions appear to provide a satisfactory solution. Classical methods of piecewise curve construction are not really flexible enough to be used in computers. In using the Hermetian method for example; curve sections can be interpolated through two end points. The composite curve has continuity of specified derivatives at the joints but the process is tedious and the effect of end derivatives is too obscure to be visualised.

Constructing a piecewise curve using the Bezier method is easier and local modification can be carried out by changing the position of two vertices the curve order will increase in direct proportion to the number of polygon vertices used.

The most desirable properties in Computer Aided curve design can be offered by the B-spline representation of curves for the following reasons:

1) Curve shape can be easily manipulated by control points.

2) Any specified control point affects a limited portion of the entire curve (non-global characteristic).

3) B-spline representation has intrinsic derivative continuity across adjacent segment boundaries. Curve order can be selected according to application and is not, as in the Bezier curve, governed by the complexity of the defining polygon shape. Finally, a major consideration must be that the computational

aspect of B-spline seems to be more efficient than other schemes especially in an interactive design situation.

For these reasons, lofting technique was used in surface representation and B-spline applied for cross sectional curve design in this work.

Choosing cardinal splines to interpolate cross-sections in surface formation is justified because it provides a simple procedure for interpolation which is computationally efficient memory-wise and for speed. Secondly, the interpolated curve is usually free from undersired oscillations.

5.7 METHOD TO BE USED FOR MACHINING

Design and manufacture must be linked so that transfer from design to manufacture is easy. The design shape definition must be transformed to an acceptable format for the manufacturing system and the final product should vary only slightly from the original design shape.

Machining in CAD systems is usually performed by NC machines. Generally, when cutting surfaces the cutter is moved along a straight line, though some machines can produce curved contours. Point data is required if the cutting process is point to point or by linear interpolation. To cut the surface, whether defined by analytical equations or other means the data given to the manufacturing unit should be coordinates or a set of points on the surface.

There are several problems in cutting free from surfaces; cutter offset, accuracy and 'cutter interference'. Ball nosed cutters are usually used and cutter offset is calculated by finding the direction of normals on the surface, by analytical or numerical means. Once the

direction of normals has been determined, the cutter is offset by the radius of the cutter along the vector over the surface.

5.7.1 Analytical Method of Computing Normals

Take Q (u,v) to represent a surface. Tangent vectors of a point on the surface can be calculated by differentiating this equation with respect to the surface parameters u & v.

If TU_{iJ} and TV_{iJ} are taken to be the tangent vectors at point (U_J, V_i)

$$TU_{iJ} = \left[\frac{\partial Q}{\partial u}\right] \qquad v = vi$$

$$u = u_J$$

and

$$TV_{iJ} = \left[\frac{\partial Q}{\partial v}\right] \qquad v = vi$$

$$u = u_J$$

The direction of normal here is obtained by taking the cross product of these two vectors:

$$TN_{iJ} = TU_{iJ} * TV_{iJ}$$

Assuming a radius R for the ball-nosed cutter, then this must be offset by $TN_{ij} . R$.

If the process is repeated consistently for each surface point then the cutter

centre positions can be given in parametric representation by:

$$CQ(u,v) = Q(u,v) + TN \cdot R \qquad (5.19)$$

Where TN is the normal vector at point (u,v).

Partial differentiation of the surface equation as given by Expression (5.18) with respect to parameter u and v provides:

$$Tu = \frac{\partial Q}{\partial u} = \sum_{i=0} B_i(u) \; A_i(v) \qquad (5.20)$$

$$Tv = \frac{\partial Q}{\partial v} = \sum_{i=0} B_i(u) \; A_i(v) \qquad (5.21)$$

Both $\dot{B}_i(u) = \frac{dB(u)}{du}$ and $\dot{A}_i(v) = \frac{dA(v)}{dv}$ are

easy to find. If an n order B-spline curve is constructed over the knot set X_0, X_1, X_2 ..... and the defining polygon vertices are P_0, P_1, P_2 ....., P_m, deBoor (64) shows that the J^{th} derivative can be found using the following Expression:

$$S^{(J)}(u) = (n-1) \ldots (n-J) \sum_i P_i^{(J)} N_{i,n-J}(u) \qquad (5.22)$$

where $P_i^{(J)}$ is defined as :

$$\begin{cases} P_i^{(0)} = P_i \\ P_i^{(J)} = (P_i^{(J-1)} - P_{i-1}^{(J-1)}) / (X_{i+n-J} - X_i), \; J > 0 \end{cases}$$

and $N_{i,n-J}^{(u)}$ is the (n - J) order B-spline basis function at interval $X_i < u < X_{i+1}$.

The first derivative of an n order uniform B-spline curve with knots at integers can therefore be given by:

$$\dot{B}(u) = S^{(1)}(u) = (n-1) \sum_{i=1}^{m} (P_i - P_{i-1}) N_{i,n-1}(u) \qquad (5.23)$$

While computing the B-spline basis functions $N_{i,n}(u)$ for evaluating the B-spline curves using the recursive formula (5.2), $N_{i,n-1}(u)$ are automatically determined and only need to be saved for future use.

Expressions (5.12) and (5.13) need only to be differentiated to evaluate the curdinal spline basis function derivatives. Expression (5.14) which embodies (5.12) and (5.13) could be differentiated so determining the first derivative of the cubic curdinal spline. This in conjunction with (5.23) allows the computation of tangent vectors. This method is used in BCSURN to simplify the procedure. After determining the tangent vectors T_u and T_v they are normalised and their cross products are computed to obtain the direction of the normals on the surface.

5.7.2 Numerical Method of Calculating Tool Offset

Here, three data points are joined to form a triangular plane, a group of these form a multi-faced polyhedron that approximates to the desired shape. Approximate surface machining is obtained by directing a sphenically ended milling cutter to touch all the triangular surface facets in turn. Points of contact are at the centroid of the triangles and slightly within the surface nbu to a lesser extent as the vertices are more closely spaced.

Considering a triangular plane facet ABC, it has the general equation:

$$ax + by + cz + d = 0$$

The coefficients a, b, c, d are found knowing the coordinates of points A, B and C. Direction cosines of the normal to this plane are easily found by dividing a, b, c and d by $(a^2 + b^2 + c^2)^{1/2}$ to give for example; L, M, N, and P. Plane ABC represented in perpendicular form would be:

$$LX + MY + NZ + P = 0$$

Where P is the negative of the perpendicular distance of plane ABC from the origin.

With reference to Fig (5.6), if C is the centroid of plane A B C and T a point R from C on the normal to the plane through C then

$$X_T = X_c + L.R$$

$$Y_T = Y_c + M.R$$

$$Z_T = Z_c + N.R$$

This gives the coordinates of the centre of a spherical ended tool of radius R with its centre at T and tangential to plane ABC at its centroid. "IBCSURF" includes such a routine as described for numerical computation of tool offset. Most components require checks to be carried out on cutter interference.

Calculation of tool offsets and interference checking are combined in one routine which will be discussed in the following sections.

5.8 INTERFERENCE CHECKING

The purpose of interference checking is to see if the spherical ended cutter will remove any part of the adjoining area while cutting to a specified position. No interference will occur if the surface is planar or concave. The largest feasible tool should be used but, if interference occurs, this must be reduced in size.

Two programs have been developed to accomplish interference checking. These are 'NUMOFF' and 'ANAOFF' .

a) ANAOFF

Cutter offset is carried out here using components of the normals produced independently.

A simple, general approach was used in this routine which requires point coordinates specifying the surface with the elements of normals as inputs at the hegenerated surfaces. Routine BCSURF was modified to allow the computation of normals carried out analytically (see section 5.7.1).

First, coordinates of tool centre position (X_T, Y_T, Z_T) are determined using the input data for a given point on the surface. The distance between all surface points and point (X_T, Y_T, Z_T) is calculated. If the distance is smaller than (R-Tolerance), the tool is raised to the new Z_T position to avoid interference and the test is repeated for all remaining points. After checking with the largest tool, the same procedure is then carried out using the next largest tool. This continues until interference is eliminated or the smallest tool set by the user has been considered.

The routine also includes segments that generate the NC machine formatted part program using the interference checked cutter centre position data. All cutters used follow the parametric lines joining the data points and travel over the component in sequence to produce a better surface finish.

b) NUMOFF

Cutter offset is computed numerically here and a more comprehensive procedure of interference checking was adopted in this routine. It uses coordinates of points generated on the surface by the surface design program BCSURF. The data could have originated from elsewhere.

For example, being the result of digitisation of a physical model for the purpose of replicating an existing object. Mathematics used are based on algebriac geometry as discussed extensively by Duncan and Mair [37,71].

The function of the program and not the mathematical theory is briefly discussed here.

Program NUMOFF reads in the point data, organised in matrix form, radius of up to 8 cutters used starting with the largest, and control parameters. Triangular planes are constructed from the data taking points from adjoining rows and columns. The program has a control switch which can be set to control drawing of diagonals in the opposite direction (see Fig 5.7).
The equation of the plane $ax + by + cZ + d = 0$ is derived from the vertices of each triangle. A tool centre position (X_T, Y_T, Z_T) is taken at distance R = Tool radius along the normal of the centroid (X_c, Y_c, Z_c) of the triangle, the program then takes the indices of all the planes that lie within the radial distance of the tool at position (X_T, Y_T, Z_T). Let (i,J) be the row and column of the plane

under consideration. The distance of all three vertices of the neighbouring plane (i, J + 1) from the point (X_T, Y_T,) i.e.

$$(\Delta v)^2 = (X_T - X_v)^2 + (Y_T - T_v)^2$$

is computed and the minimum taken. If the distance is less than the tool radius, plane i, J + 1) is within the radial distance of the tool. Other planes (i, J + 2), (i, J + 3), are also tested. The process terminates when one plane is further than R from (X_T, Y_T). Also (i, J - 1), (i, J - 2) , and planes in the upper and lower columns (i - 1, J), (i - 2, J) ... , (i + 1, J), (i + 2J) are tested and indices for all planes threatened with under cutting arc saved.

The next stage is to eliminate planes that are co-planar or lie below the plane under examination. To do this, the plane being examined is represented in its perpendicular form by LX + MY + NZ + P = 0, substituting X, Y and Z by X1, Y1, Z1 at the right hand side of the above expression which produces the distance of the point (X1, Y1, Z1) from this plane. X_T, Y_T, Z_T can now be substituted and coordinates of the three vertices of the planes form previous tests suspected of being undercut in the above expression. If all three latter results are zero or of opposite signs to the first, then that plane is co-planar or lies below the plane under examination. The indices of the planes not eliminated are saved for further verification.

Further suspected planes are eliminated by testing the normal distance from the (X_T, Y_T, Z_T)to each plane in turn. There is no interference if it is greater than or equal to the radius less a tolerance specified by the user. Lastly, it must be determined whether the tool cuts planes as yet not eliminated and where interference is likely within the area bounded by the sides of the triangle.

By the end of these tests, remaining planes will be under-cut when the tool is at (X_T, Y_T, Z_T). Coordinates are found for all these planes to avoid interference. The maximum is taken to be the Z_T. The indices of the planes whose facets will be affected by interference are saved to be processed later by a smaller tool. The process continues until either the number of saved indices are zero or the smallest tool has been used.

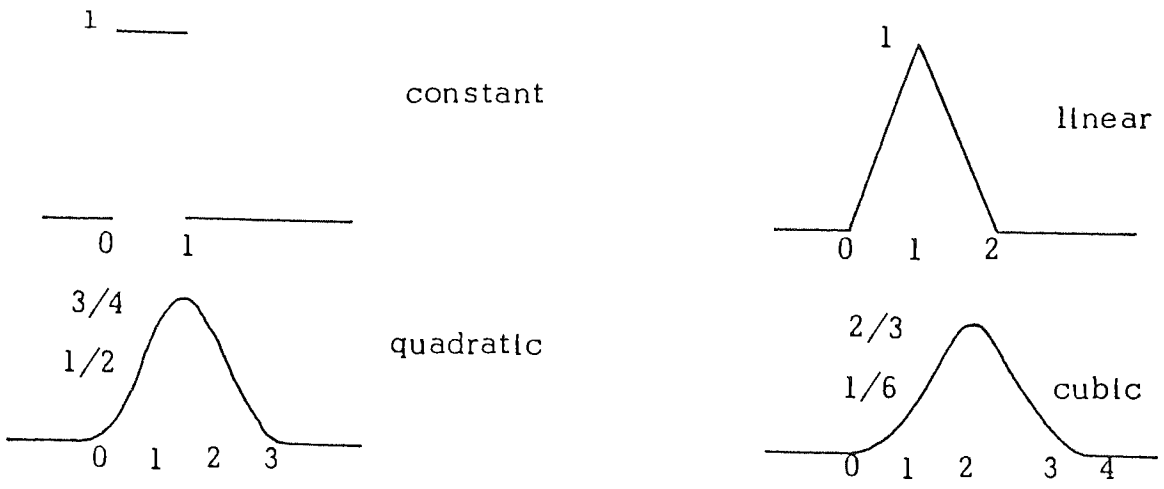


Fig. (5.1) B-spline Basis Functions for Different Degrees

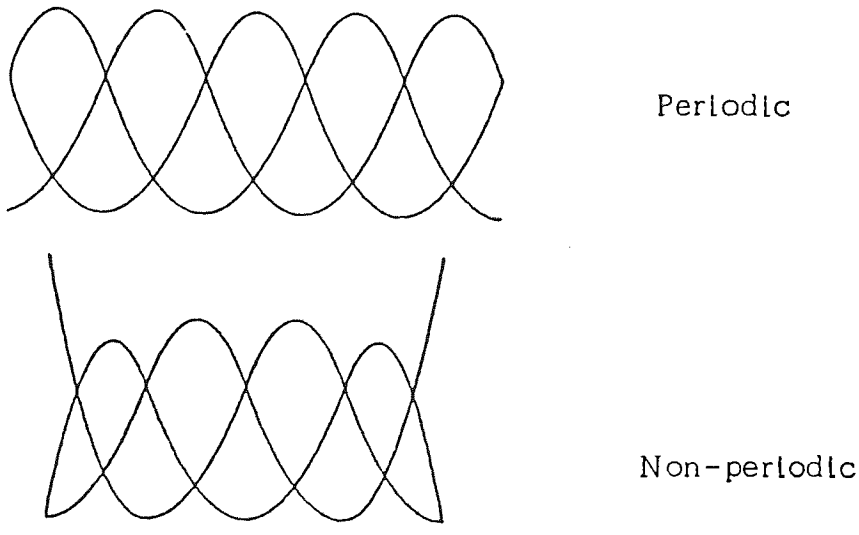


Fig. (5.2) B-Splines for Periodic and Non-periodic Curves

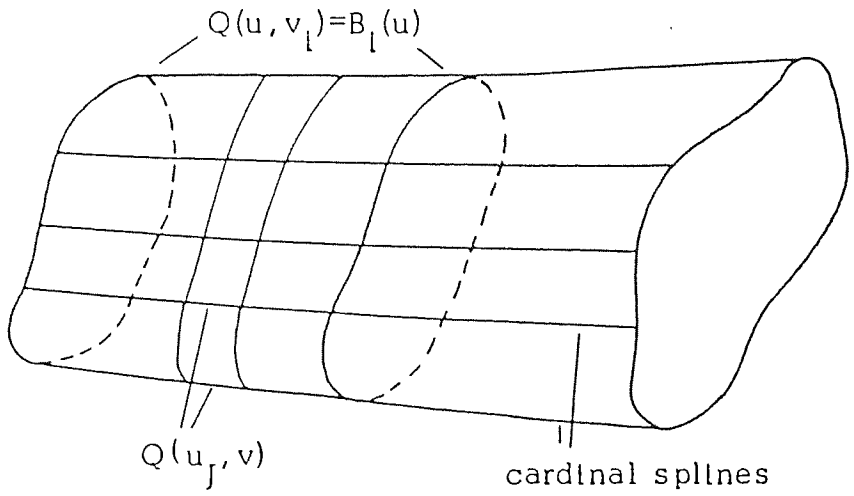


Fig. (5.3) Surface Description Method

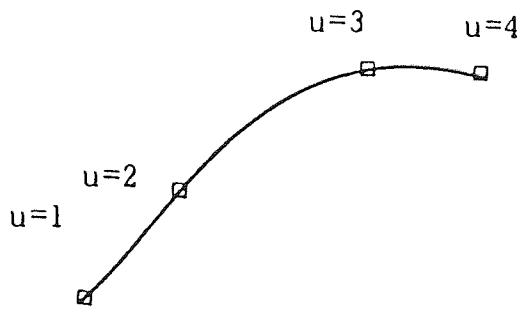


Fig. (5.4) Equal Parametric Distances with Uneven Distribution

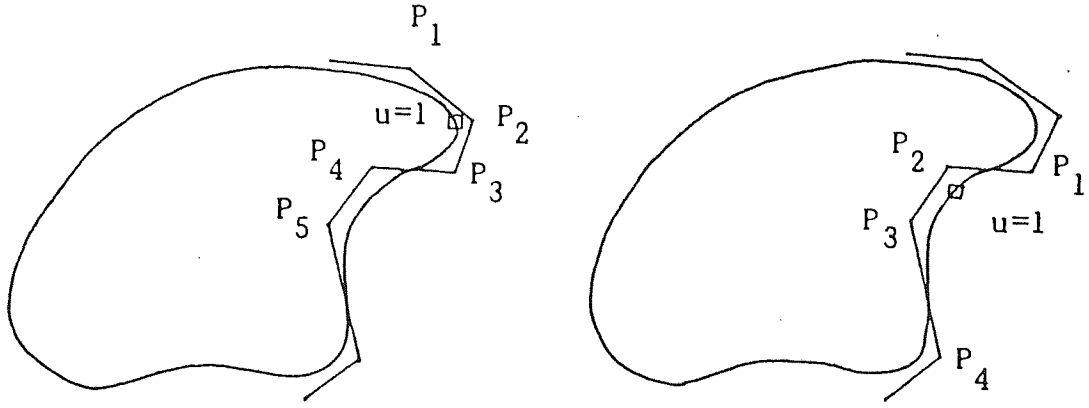


Fig. (5.5) The Effect of Vertex Numbering on Parametrisation

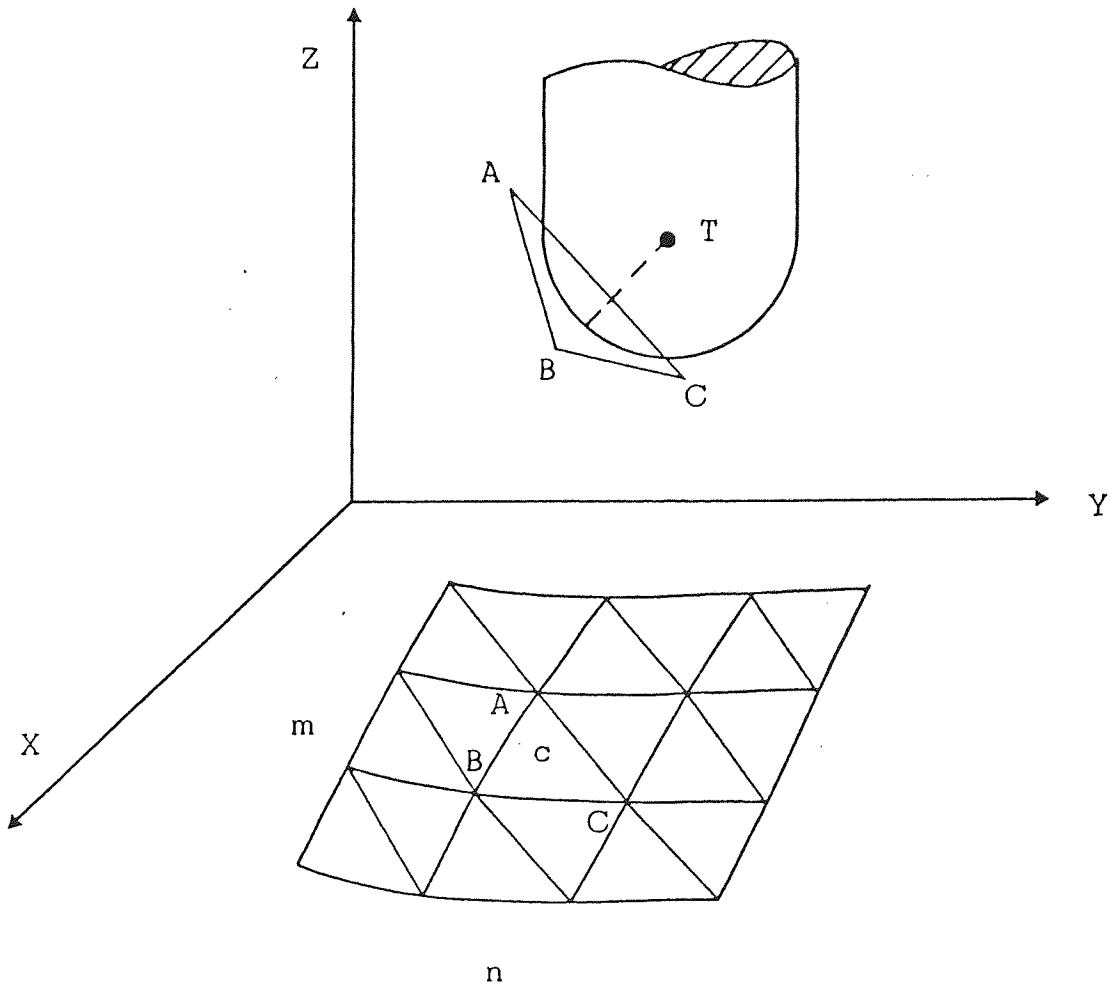


Fig. (5.6) Tool Offset on a Multi-faced Polyhedron

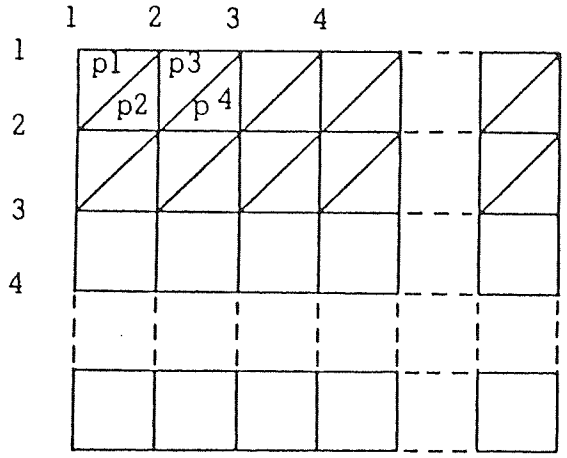


Fig. (5.7) Data Point Organisation for NUMOFF

cutting are saved.

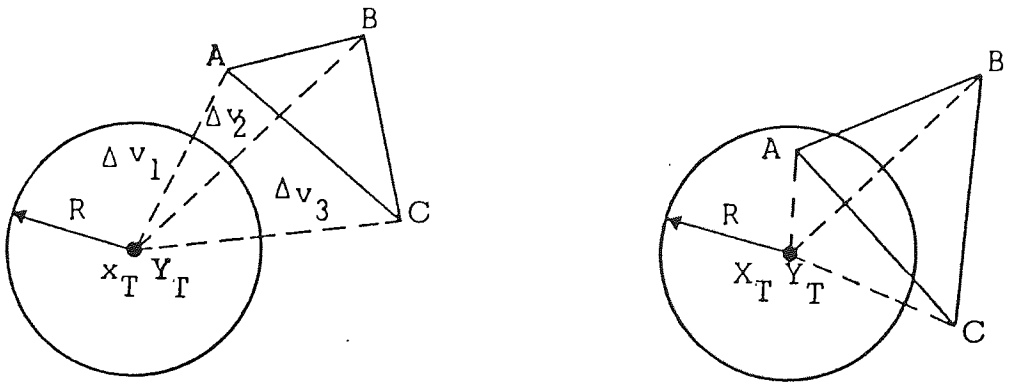


Fig. (5.8) Examination of Planes within the Radial Distance of the Tool

CHAPTER 6

CHAPTER 6

MATHEMATICAL BASIS OF THE HANDLE

6.1 INTRODUCTION

The surface of the handle is generated by blending together circles and lines. It consists of four basic sections along the Z-axis and two basic cross sections Fig (6.1).

The data points on the surface are calculated by means of two dimensional geometry and the cartesian co-ordinate system is used.

A computer program called 'SECT' written in FORTRAN 77 was developed for this purpose. Different shapes of the handle can be created by varying the input parameters.

6.1.2 Programme 'SECT'

There are three stages employed in the calculation of data points

6.1.2.1. Stage One

The positions along the z-axis and the widths along the x-axis for all cross-sections are calculated.

If N is the number of required cross-sections, then the width along the

x-axis of every cross-section 'I' and its position along the z-axis are given by:

$$Z(I) = \frac{Z(N)}{N - 1} \cdot (I - 1) \qquad (6.1)$$

and

$$X(I) = X(1) - \frac{X(1) - X(N)}{Z(N)} Z(I) \qquad (6.2)$$

where:

axes are taken as shown in Fig (6.2) Z(n) is the length of the handle, X(1), X(n) is the width of the handle at the rear and front end respectively and $1 \leq I \leq N$.

6.1.2.2 Stage Two

To calculate the heights (y - coordinates) of all cross-sections N, the y and z coordinates of the centre O_i of the circles C_i(i = 1, ... 6) and of the intersection points P_j (j = 1, ... 9) (Fig 6.1), between circle-circle and line-circle need to be precalculated.

For simple notation, let C_{ij} denote a circle centred at O_i and radius $R_i + R_j$, then O_3 is given by the intersection of circle C_{23} with line L_5 and 0_5 by the intersection of C_{45} with L_7.

The focus of circle C_1 is the line L_2, which is perpendicular to the z-axis and its z-coordinate Z_1 is one of the input parameters and is therefore, known. Consequently, point O_1 (z_1, y_1) is given by the intersection of L_2 and the translation of L_1 along the positive Y-axis by $\frac{-R_1}{\cos\theta}$ where the equation of L_1 is: $y = Z \tan \phi + Y_2$ with Y_2 known, and of L_2 is : Z = Z, then points

$P_1(z_1^*, y_1^*)$ $P_2(x_2^*, y_2^*)$ have coordinates:

$$Z_1^* = Z_1 - R_1 \sin \phi, \qquad y_1^* = y_1 + R_1 \cos \phi$$

$$Z_2^* = Z_1 + R_1 \sin \theta, \qquad y_2^* = y_1 + R_1 \cos \theta$$

Since the thickness of the handle at the section "Y" is known to be Y_1, then O_2 lies on the bisector L_4 of the angle between line L_3 and Line $y = Y_1$. It is given by the intersection of line L_4 with the translation of line L_3 along the positive Y-axis by $\dfrac{R}{\cos \theta}$

where equation of L_3 is:

$$y = (Z - Z_2^*) \tan (180 - \theta) + y_2^* \qquad (6.3)$$

Point B (Z', y') has co-ordinates:

$$Z' = \frac{y_2^* - Y_1}{\tan \theta} + Z_2^*, \quad y' = Y_1 \text{ and}$$

therefore, equation of bisector L_4 is:

$$y = (Z - Z') \tan \left(\frac{180 - \theta}{2} \right) + Y_1 \qquad (6.4)$$

Using (6.13) and (6.14) the coordinates of O_2 (z_2, y_2) are given by:

$$Z_2 = \frac{Z_2^{*} A - C + Y_1 - ZB}{A - B}, \quad y_2 = R_2 + Y_1$$

Where $A = \tan(180 -)$, $B = \tan\left(\frac{180 - \theta}{2}\right)$

and $C = y_2^{*} + \frac{R_2}{\cos \theta}$

Suppose the locii of circles C_3, C_4, C_5 are lines L_5: $Z = Z_2$, L_6: $z = Z_3$, L_7: $z = Z_4$, respectively, where the calculation of the coordinates Z_2, Z_3, Z_4 is discussed later in this chapter.

Intersection of circle C_{z3} with line L_5 gives O_3 (Z_3, y_3) and point P_4 (Z_4^{*}, Y_4^{*}) is given by the intersection of circle C_3 with Circle C_2.

Similarly, intersection of line L_6 with circle C_{34} gives O_4 (Z_4, y_4) and intersections of circle C_3 with circle C_4 gives point P_5 (Z_5^{*}, y_5^{*}).

Furthermore, O_5 (Z_5, y_5) is the point of intersection circle C_{45} and line L_7, and point P_6 (Z_6^{*}, y_6^{*}) is given by the intersection of circle C_4 with circle C_5).

Point $P_7 (Z_7^*, y_7^*)$ has coordinates:

$$Z_7^* = Z_5 + R_5 \sin\beta, \quad Y_7^* = Y_5 + R_5 \cos\beta$$

and therefore, the equation line L_8 is:

$$y^* = (Z - Z_7^*) \tan (180 - \beta) + y_7^*.$$

$O_6 (Z_6, y_6)$ is given by the intersection, of the translation of line L_8 along the positive z-axis by $\frac{- R_6}{\text{Sin}\,\beta}$ and the translation of line L_9 along the positive z-axis by $\frac{- R_6}{\sin\alpha}$ where equation of L_9 is:

$$y = (Z = Z_5) \tan (180 - \alpha).$$

Therefore, the coordinates of the points $P_8 (Z_8^*, y_8^*)$, $P_9 (Z_9^*, y_9^*)$ are given by:

$$Z_8^* = Z_6 + R_6 \cos (90 - \beta), \quad y_8^* = y_6 + R_6 \sin (90 - \beta)$$

$$Z_9^* = Z_6 + R_6 \sin (90 - \alpha), \quad y_9^* = y_9 + \cos (90 - \alpha)$$

The coordinates of all the other points on the surface are calculated using the appropriate equation, relating Y and Z, depending on the position of the points of the surface, so if,

$Y_2 \leq Z(I) \leq P_1$ equation of L_1 is used to calculate y (I)

$P_1 < Z(I) \leq P_2$ equation of C_1 is used to calculate Y (I)

$P_2 < Z(I) \leq P_3$ equation of L_3 is used to calculate Y (I)

$P_3 < Z(I) \leq P_4$ equation of C_2 is used to calculate Y (I)

$P_4 < Z(I) \leq P_5$ equation of C_3 is used to calculate Y (I)

$P_8 < Z(I) \leq P_9$ equation of C_6 is used to calculate y (I)

$P_9 < Z(I) \leq Z_5$ equation of L_9 is used to calculate y (I)

Where $1 \leq I \leq N$, $Z(1) = 0$, $Z(N) = Z(5)$ and $y(N) = 0$

6.1.2.3 Stage Three

At this stage, there are two cases to be considered, Fig (6.4). The idea is to calculate a fixed number M, of points on all cross-sections N, whose coordinates are represented by:

$(x(I,J), \; y(I,J))$ where,

$1 \leq I \leq N$ and $1 \leq J \leq M$.

In both cases, M is taken to be an odd number because of the symmetry of the cross-sections about the Y-axis and because point B (x,y), whose coordinates are already known from stage two, lies on the symmetry axis.

Therefore, only $\frac{J - 1}{2}$ points need to be calculated and their

reflection about the y-axis gives the other $\frac{J - I}{2}$ points.

(a) Case One Fig (6.4a)

By default, point C (X_1, y_1) is taken to have coordinates $X_1 = 0$, $y_1 = 39.00$ and therefore, point B (x_2, y_2) has coordinates:

$$X_2 = 0, \quad y_2 = y_1 - y(I).$$

Also the coordinates of O_1 (x_1, y_1) are $X_1 = 0$, $y_1 = y_2 + R_1$ where R_1 is the radius of circle C_1.

O_2 (x_2, y_2) is given by the intersection of the translation of line L, along the positive x-axis by $\frac{-R_2}{\cos \psi}$ with the circle radius $R_1 - R_2$, centred at O_1 (x_1, y_1), where equation of L_1 is $y = (X - X(I)) \tan (90 - \psi) + y_1$ and R_2 is the radius of circle C_2.

Points $P_1 (X_1^*, y_1^*)$, $P_2 (X_2^*, y_2^*)$ are given by the intersection of C_2 with L_1 and C, with C_2 respectively.

Furthermore,

$$y(I, 1) = y_1, \qquad X(I,1) = X(I)$$

$$y\left(I, \frac{J+1}{2}\right) = \bar{y}_2 \ , \ X\ I, \left(\frac{J+1}{2}\right) = 0$$

The position of all other points are preset by the main program and depend on the parameters R_1, R_2, ψ and were chosen so, to give the best possible representation of the surface for the B-spline fitting. In other words, more points are calculated on the arcs $\overset{\frown}{P_1 P_2}$, $\overset{\frown}{P_2 B}$ than on the line L_1.

(b) Case Two Fig 6.4b)

As in case one, O_2 (x_2, y_2) is given by the intersection of the translation of L_2 along the positive y-axis by (R_2) and the translation of L_1 by $\frac{-R_2}{\cos\psi}$ along the positive x=axis similarly, points P_1, P_2 and all intermediate points are calculated as above.

If R_1 is given to be zero then the program(s) executes case two and if R_1 is greater than zero, case one is executed.

Intersection Points

The intersection points between two circles or line and circle are calculated using specially written sub-routines.

6.1.2.5 Intersection of Circle with Circle

Suppose equations of the two circles are:

$$X^2 + y^2 + 2gx + 2fy + c = 0 \qquad (6.5)$$

$$X^2 + y^2 + 2Gx + 2Fy + S = 0 \qquad (6.6)$$

Where $R_1 2 = g^2 + f^2 - c$, $R_2{}^2 = G^2 + F^2 - S$

and the coordinates of their respective centres are (-g, -f), (-G, -F).

From (6.5), (6.6):

$$X = \frac{S - C - 2(f - F)y}{2(g - G)} \qquad (6.7)$$

and from (6.5)

$$(x + g)^2 = g^2 + f^2 - c - (y + f)^2 \qquad (6.8)$$

substitute for x in (6.8) to get:

$$(A + B_y)^2 - R_1{}^2 + (y + f)^2 = 0$$

or $y^2 (B^2 + 1) + y (2AB + 2f) + A^2 - R_1{}^2 + f^2 = 0,$ (6.9)

a quadratic in y where:

$$A = \frac{S - C}{2(g - G)} = g \text{ and } B = \frac{F - f}{g - G}$$

solving the quadratic equation (6.9) for y and substituting in (6.7) the coordinates of the common points are obtained.

6.1.2.6 Intersection of line inclined at an angle not Equal to $\Pi/2$ with circle radius R.

Suppose $y = (x - x_1)\, m + y_1$ (6.10)

and $(x - a)^2 + (y - b)^2 = R^2$ (6.11)

are the equations of the line and the circle respectively,

where m = tan angle

substituting for y in (6.11) one gets:

$(Xm - (X_1 m - y_1 + b))^2 = R^2 - (X^2 + a^2 - 2 \times a)$, or

$X^2 (m^2 + 1)\ x - 2a - 2m (x_1 m - y_1 + b)) - R^2 + a^2 + (x_1 m - y_1 + b)^2 = 0$

a quadratic in x.

Solving for x and substituting in (6.10) the intersection points are obtained.

If the angle is equal to $\Pi/2$ then the equation of the line is x = A and by direct substitution in (6.11).

$y = b \pm \sqrt{R^2 - (A - a)^2}$

6.1.2.7 Calculation of the Co-ordinates Z_2, Z_3, Z_4 (Fig 5.1)

In the mathematical modelling of the handle, it was necessary to position two circles, C_1 (i = 2, ..., 5) so that the intersection between them was always feasible.

In most cases, the conditions that two circles C_1, C_2 always intersect is that the distance between their centres should not exceed the sum of their radii, except for example in the case below (Fig.6.5).

Circle C_2 must always be above the base line (Fig 6.5) and preferably not have any common points with it. Therefore, the above condition does not apply, especially when the length of BP is very small, close to zero or even zero, where R_1 must be greater than R_2.

In Fig 6.5 the maximum distance between the centre of the two circles, which satisfy the above conditions is BM.

From triangle AFD:

$$\text{Sin } \Theta = \frac{FD}{R_1 + R_2} = \frac{FM - DM}{R_1 + R_2} \quad \text{or,}$$

$$\Theta = \text{Sin}^{-1} \frac{R_2 - AB}{R_1 + R_2}$$

and therefore;

$$\text{Cos } \Theta = \frac{AD}{R_1 + R_2} \quad \text{i.e.} \quad AD = (R_1 + R_2) \text{ Cos } \Theta = BM$$

The position of line L_2, the locus of C_2, can now be manipulated, so it always satisfied the above conditions. For example; suppose the new position of L_2 is through the point C, then:

$BC = AD - \varepsilon$, Where $\varepsilon \geq 0$ but smaller compared to R_2.

In other words, the program selects the position of the locus C_2, by subtracting the value of the parameter ε from the maximum distance. This way all the above conditions are satisfied and thus allow accurate positioning of the circles on the surface Fig (6.6).

Fig (6.6) shows how the parameters ε in form used by one program: DU(I) to position a small circle in relation to another small one and EU (I) to position a large circle in relation to a small one, from left to right.

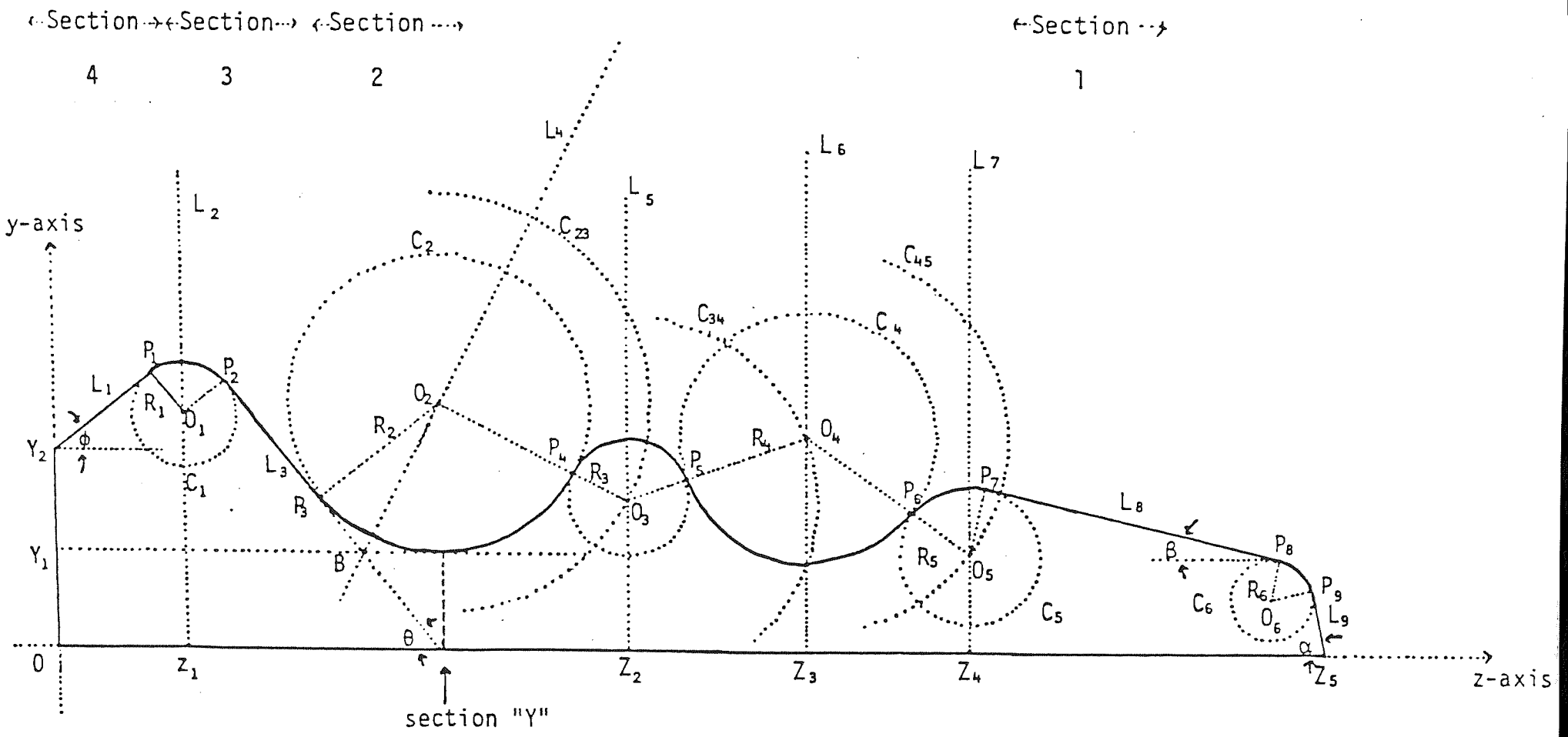


Fig 6.1 Mathematical Basis of the Handle

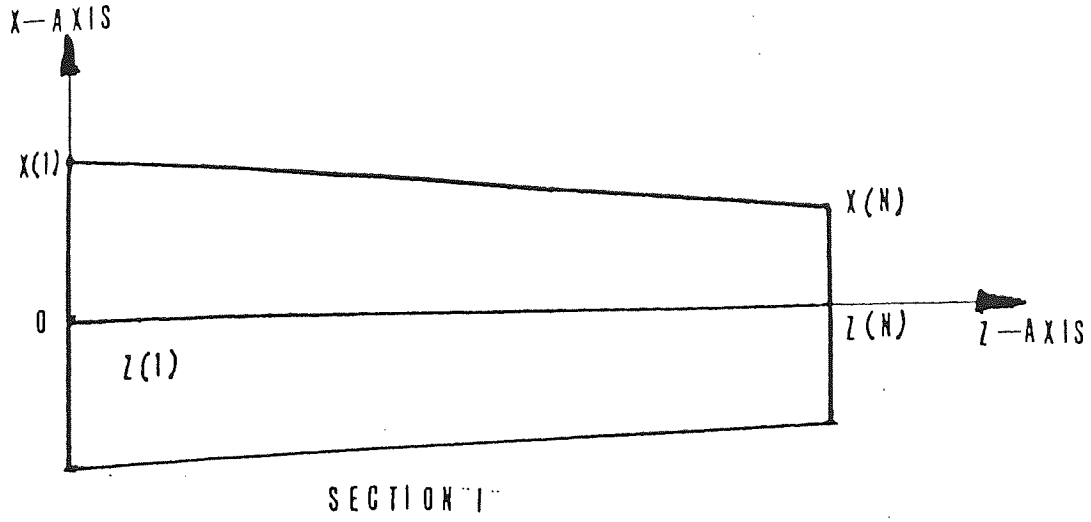


Fig 6.2 Cross-section of the Handle

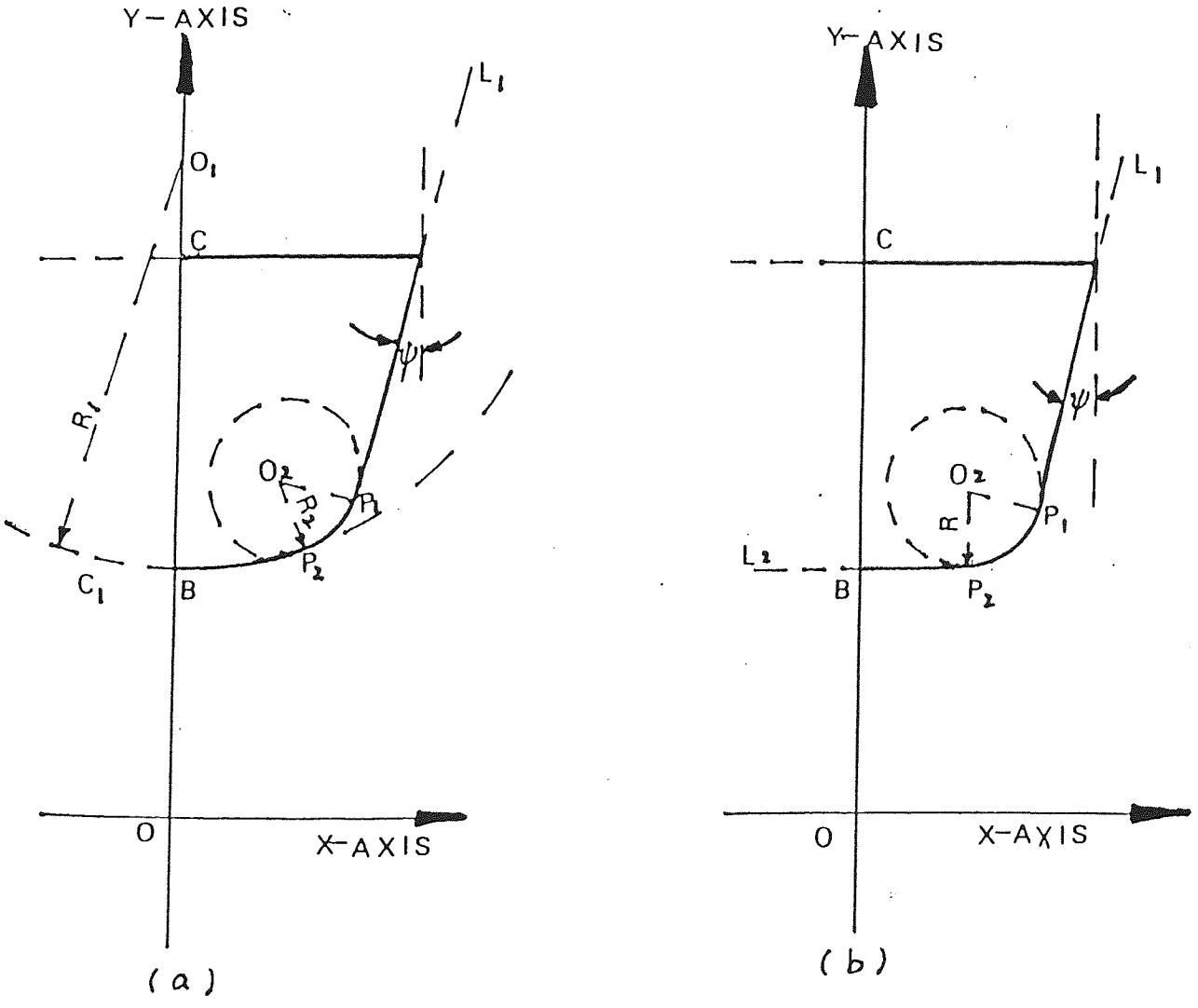


Fig 6.3 (a) Case One (b) Case Two

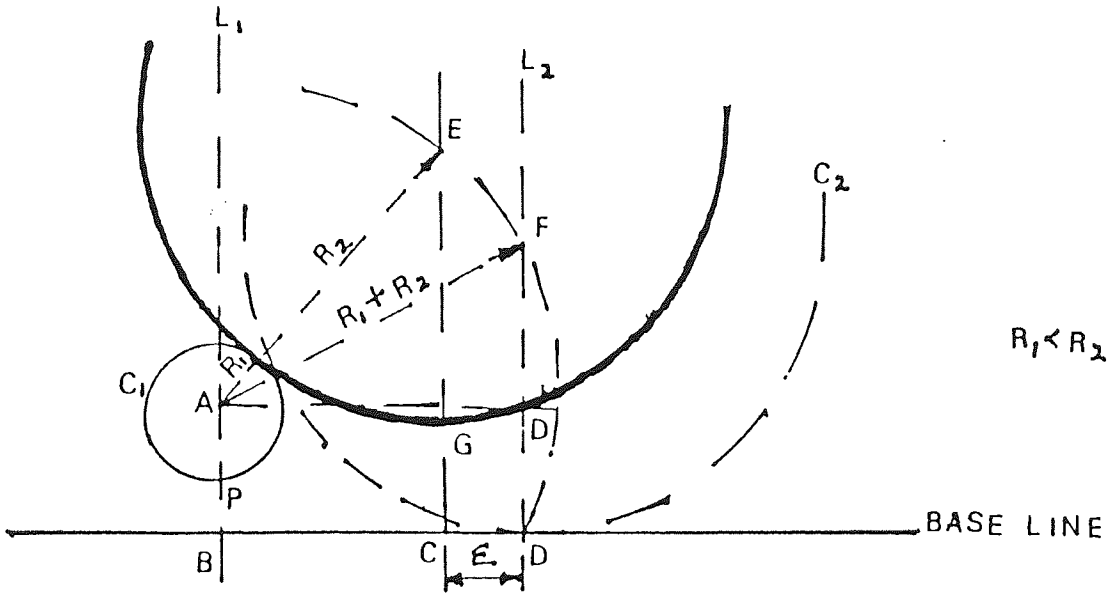


Fig 6.4 Intersection Between the Circles

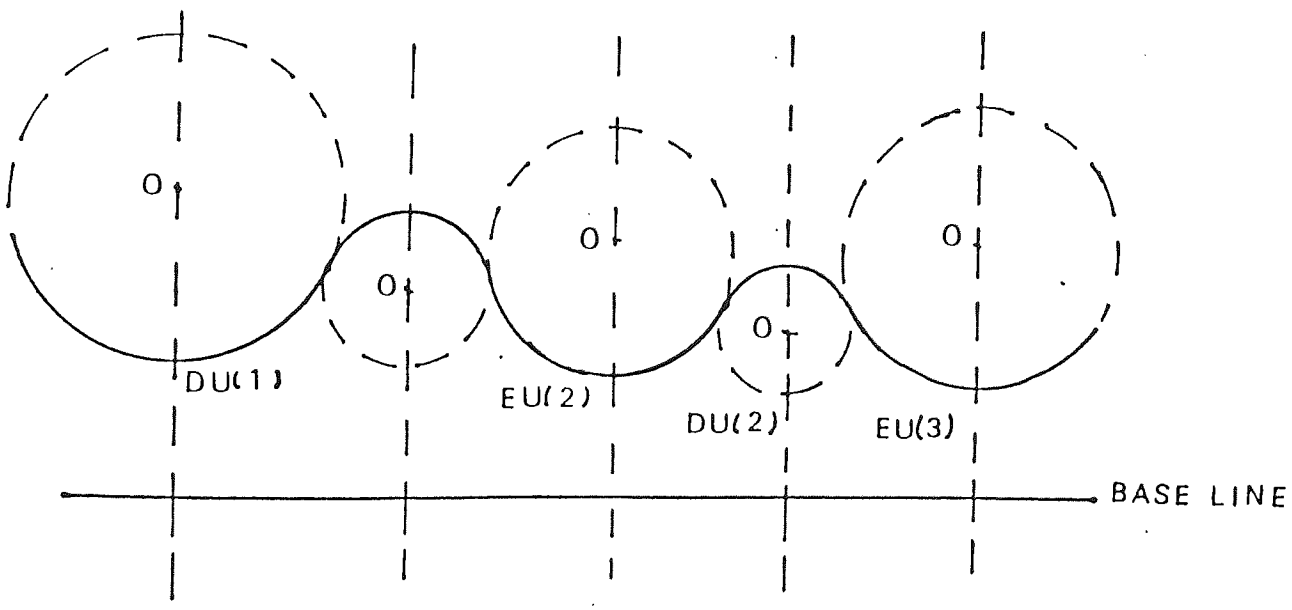


Fig 6.5 Positioning of the Circles on the Surface

CHAPTER 7

CHAPTER 7

SYSTEM OVERVIEW

7.1 INTRODUCTION

The traditional approach for designing and manufacturing three - dimensional objects composed in part or total of free-form curved surfaces as described in Chapter 1, relies heavily on the knowledge and expertise of the designer, manufacturing engineer, and on the skill of the machinist in selecting the appropriate machining instructions. This manual method lacks well defined scientific principles and may entail great expense and time.

The application of computer aided facilities can improve on the inefficiency in the final product which is largely due to human error. Careful planning is essential before undertaking such a complex task [72]. In the past, many unreliable and inflexible software packages have been developed because of a lack of scientific approach for software design and development, and frequently mismanaged software projects [73].

7.2 PLANNING FOR THE SOFTWARE DEVELOPMENT

All development work requires systematic planning. Software development is no exception, especially when that involved is of considerable scale and complexity. Software objectives and computerisation requirements have to be evaluated and established to guide the development process. These are based on existing needs against the constraints of available resources, facilities and know how.

For successful computerisation, a clear understanding of practices in industry and its needs was necessary. It was also necessary to have the knowledge available for designing three dimensional objects. For example, Healey Mouldings Ltd, a Birmingham based

company used traditional methods to produce moulds. These moulds were time consuming and uneconomical. The engineer had to prepare engineering drawings for every design and include all relevant information for specialists to cut the moulds. The designs were based on the designers own ability to visualise and describe the object on paper.

These problems were examined at the University by incorporating available CAD/CAM systems to develop systems to resolve the specific design of a plastic handle and wider applications for design and manufacture of free form surfaces.

7.2.1 Software Objectives

The software development program of this research was planned to integrate with previous phases of software development by Razavi (74), some of whose programs were run on the ICL 1904 computer written in FORTRAN IV and used graphical facilities.

According to this phase of software development, the following activities, as mentioned in section 1.3, were planned for computerisation.

1) Implementation of the CAD software on the computer workstation (VAX)
2) Design and development of the package editor with interactive graphics facilities capable of displaying all the edited object from different angles.
3) The mathematical model of the handle
4) Design and development of the post-processor for the Olivetti NC machine.
5) Design and development of a general post-processor for NC milling machine.

All the above software objectives have been successfully carried out according to the plan of this research.

7.3 THE COMPUTER WORKSTATION

The computer plays a dynamic part in the development of integrated manufacturing systems, and its processing capabilities are vital factors in the success of the system. All software was written in FORTRAN 77 [62] using GINO-F [63] library for graphics, the acquired syustem should therefore provide a FORTRAN compiler as well as the GINO-F graphics software library.

With these considerations in mind, the VAX 11/750 32-bit super mini-computer with 4 Mbytes of RAM, a 300 Mbyte disc drive and 75 in/s dual density tape drive, was chosen. The VAX 11/750 is supplied by the Digital Equipment Corporation and its high speed shortens program execution time.

The workstation is a Tektronix 4107 microprocessor controlled terminal [75] with a 640 x 480 pixel resolution (plate 7.1). The resolution is considerably enhanced by an addressable display matrix of 4096 x 4096 points. The user can zoom in on a portion of the display and the 4107 terminal will recompute the coordinate information and display the designated section in much greater detail. Rather than 'pixel replication', which simply enlarges the picture without providing further detail, this true zoom significantly increases resolution.

Other peripherals include a printer, a Tektronix 4113 graphics terminal and a Tektronix 4691 graphics plotter to provide hardcopy drawings, and a paper tape punch to produce NC tapes when necessary.

7.3.1 Design for Portability

Portability is considered to be an important quality characteristic in modern software design and development. The ability to convert a software system to suit

new operating environment is desirable. The situation tends to arise when developed software becomes popular, it is unlikely that all prospective users will have identical computer systems. It is not always possible to produce a universally portable software system in the present state of software practices, it is still possible to design the software so that there is minimal effort involved in modifying it for transference.

In this work, it is highly probable that the software developed may be implemented on a different system in the near future. This has to be considered and incorported as far as possible. Programming and graphic languages are basic areas of concern. Considerations have been based on previously developed software against facilities available. As all existing software was written in FORTRAN 77 a language available on the University's VAX 11/750 it was justified. FORTRAN 77 is a high level language which is highly portable and is actually one of the most popularly supported scientific languages used in most computer systems. The choice used in most computer systems. The choice of GINO-F graphic facilities was justified again, because of its portability.

It is envisaged that the system could soon be implemented on a stand-alone powerful microcomputer with 512 Kbytes of RAM and a 5 Mbyte disk drive has been chosen. The DEC Professional 350 (Plate 7.2) can be linked to the VAX 11/750, and all software can be down-loaded from one computer to the other.

7.4 STARRAG RIGID MILLING MACHINE (Plate 7.4)

Machine Details

Type	Vertical Milling Machine KAB-50N	
Table Size	500 mm x 1,100 mm	
Maximum Table Moment	Longitudinal	700 mm
	horizontal	350 mm
	Vertical	250 mm
Table Feed	Hydraulic 10 700 mm/min.	
Spindle Speeds	94.2, 120 in 10 step	

7.4.1 Control System

This machine has an Olivetti CNC numerical control system which controls the three axes of the machine simultaneously to perform the 3-D contour milling. There is a built in linear interpolator which allows a straight line to be machined between two positions at a constant feed rate in X and Y axis, but there is no circular interpolator for milling circular paths in the X - Y plane. The control unit is composed of two main parts:

a) Logic Unit:

The logic unit co-ordinates the operation of control and transfers digital data to obtain instantaneous position of the axis.

b) Servo Circuits:

These translate the digital output of the logical unit, to signals suitable to operate the servo motors. After the data is analysed, it is then transferred to each axis in the form of signals and finally converted into analogue form to drive the servo motors.

The path of the cutter is calculated by taking the difference between the present co-ordinate position, and dividing it into increments. The machine's input accuracy is to the nearest 0.0001 in., and thus all output data generated in the logic unit is correct to 4 decimal units.

To ensure position accuracy, the drive system of the axes possesses a position transducer which is able to measure instantaneous position differences between the actual and commanded position.

7.4.2 Tally 464 Paper Tape Reader

The Tally 464 paper tape reader can read any standard five to eight hole tape asynchronously at speeds up to 120 characters per second. To move the tape incrementally a D.C. pulse is applied in either forward or reverse directions of the Capstan escapement system.

The capstan escapement system is an electro-mechanical unit, over which the tape is passed.

7.5 BRIDGEPORT CNC MACHINE

7.5.1 Machine Concept

The Bridgeport Series I Interact further extends the capability for simple automation on the shop floor as introduced by the series I MDI (Manual Data Input) machine.

The Series I Interact has the additional capability for 2 axis simultaneous linear movement and circular contouring plus mirror image, datum shift and 3 axis positioning.

It provides the simplest form of CNC control and minimal training is required to enable operators to use the machine with very little adjustment to their traditional method of operation. Positioning is achieved through the interactive control which leads the operator step-by-step through the programme using question and answer techniques.

An electronic handwheel permits full manual operation in all three axes using the digital read out on the control display.

The Series I Interact enables full advantage to be taken of the latest technology for the machining of both simple complex parts. The machine has been designed specially with chrome ways, d.c. drives, ballscrews and a special saddle, table and knee assembly having large slideway area to provide the rigidity, accuracy and repeatability essential for operation under automatic control.

7.5.2 Brief Description of Control

The interactive CNC control provides direct programming on the machine by the machine operator. For this reason, some of its details purposely deviate from established programming standards. Special attention has been given to simple operation and a clear arrangement of the control panel. This includes separate displays for:

i) Plain language dialogue and program block

ii) Program block

iii) Entry data

iv) Position values

and keyboards for:

i) Entry values and axis selection

ii) Operating modes

iii) Programming and other functions.

The programming procedure is "dialogue-guided" i.e. the necessary data required for program entry is asked for by the control - in the correct sequence - via the plain

language dialogue display. Programming can be performed in the following ways:-

- with stationary machine in accordance with a workpiece drawing or program sheet
- by "keying-in" values prior to automatic machining of the first workpiece.
- external programming via the V-24 interface. This interface enables the connection of a magnetic tape storage unit, or a tape punch unit.

7.6 N.C. PART-PROGRAM PATTERN

In the NC part-program, details of machining operations have to be set down in sequence. Each line of the program should be numbered in sequence, details of the operation have to be stated and the X, Y and Z coordinates should be given. The basic information should then be supplemented by code numbers representing the preparatory functions; feed rate; spindle speed; required tool; and miscellaneous functions.

Nevertheless, the format or pattern for an NC part-program may vary according to the type and make of machine control system. Detail for the part-program format for the Olivetti and the Bridge-port Control system can be referred to the programming manuals However, a list of the preparatory functions (G-codes) and miscellaneous functions (M-codes) and some program format functions with more details concerning the NC part-programs which convert the cutter location data information into an N.C. tape program will be described in Chapter 9.

7.7 SOFTWARE FUNCTIONS

The development of the package consists of two phases. The first deals with CAD. This is further divided into two sections: Curve Design and Surface Design..

CURVE DESIGN

1) Program 'BSLSQ' used to fit curve to a larger number of empirical data points

2) Knot vectors determined using 'BSCURV'.

3) Shape modified by changing order of curve and position and number of defining polygon vertices.

4. Local modifications made by changing position of a vertex.

5) Shape is also controlled using multiple vertices.

6) Program 'SECT' gives mould cross-section. 'BSCURV' employed to fit B-spline on the point.

7) Program 'INVERT' used when more efficient approximation required on existing curve.

8) Cross-sections of the mould can be viewed interactively on the terminal screen.

SURFACE DESIGN

1) Program 'BCSURF' used to give a surface resulting from interpolating sectional curves using cardinal splines.

2) Using 'BCSURF' points at equal arc intervals are selected over each sectional curve.

3) Cubic cardinal splines are then generated in the other direction to interpolate the selected points.

4) The resulting mesh of points on the surface used for generation of graphical output.

5) Using program 'VIEW' the mould design for the plastic handle can be viewed.

The second phase of the package deals with CAM. The output from 'BCSURF' is fed to 'BCSURN'.

1) Using 'BCSURN' the direction cosines of normals to the surface are calculated.

2) Program 'ANAOFF' examines whole surface to select tool and cutter position.

3) The post-processor converts cutter location data ianto suitable NC tape program.

7.8 THE PROCESSING COMMANDS

COMMAND	TITLE OF PROGRAM
S1	BSLSQ
S2	INVERT
S3	SECT
S4	BCSURF1
BS5	BSCURV
TS5	BSCURV
S6	BCSURF
S7	VIEW
S8	BCSURN
S9	ANAOFF
S10	NUMOFF
S11	PREPARE 3-D
S12	General Post-Processor

For Input and Output Files, Refer to Appendix 2

Plate 7.1 The Tektronix 4107 Workstation

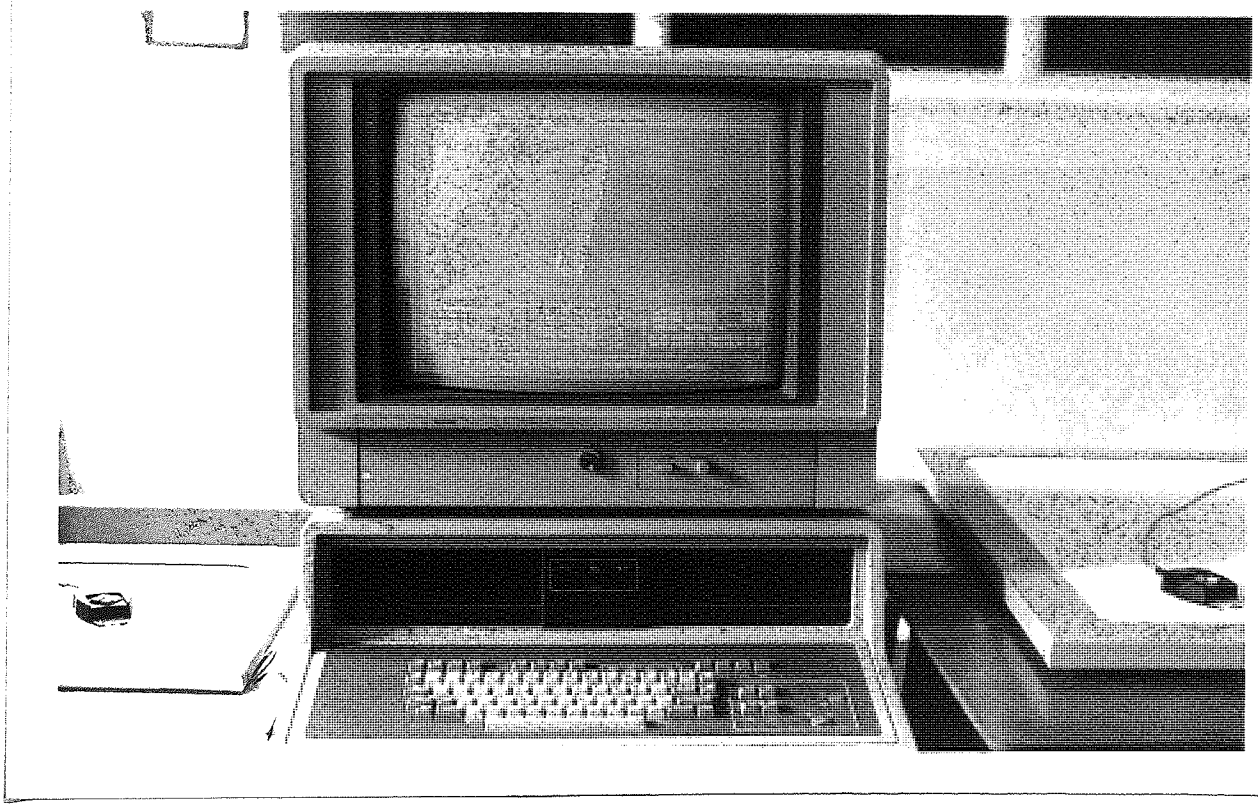


Plate 7.2 The Tektronix 4113 Workstation

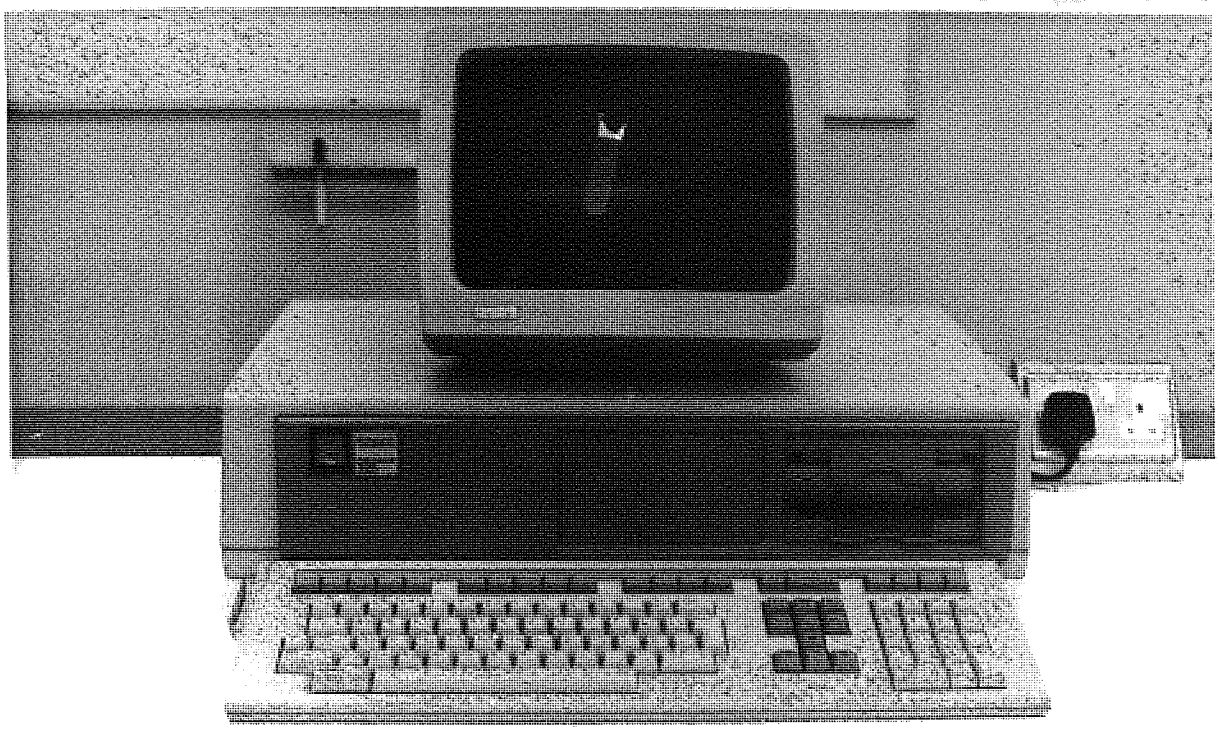


Plate 7.3 The DEC Professional Workstation

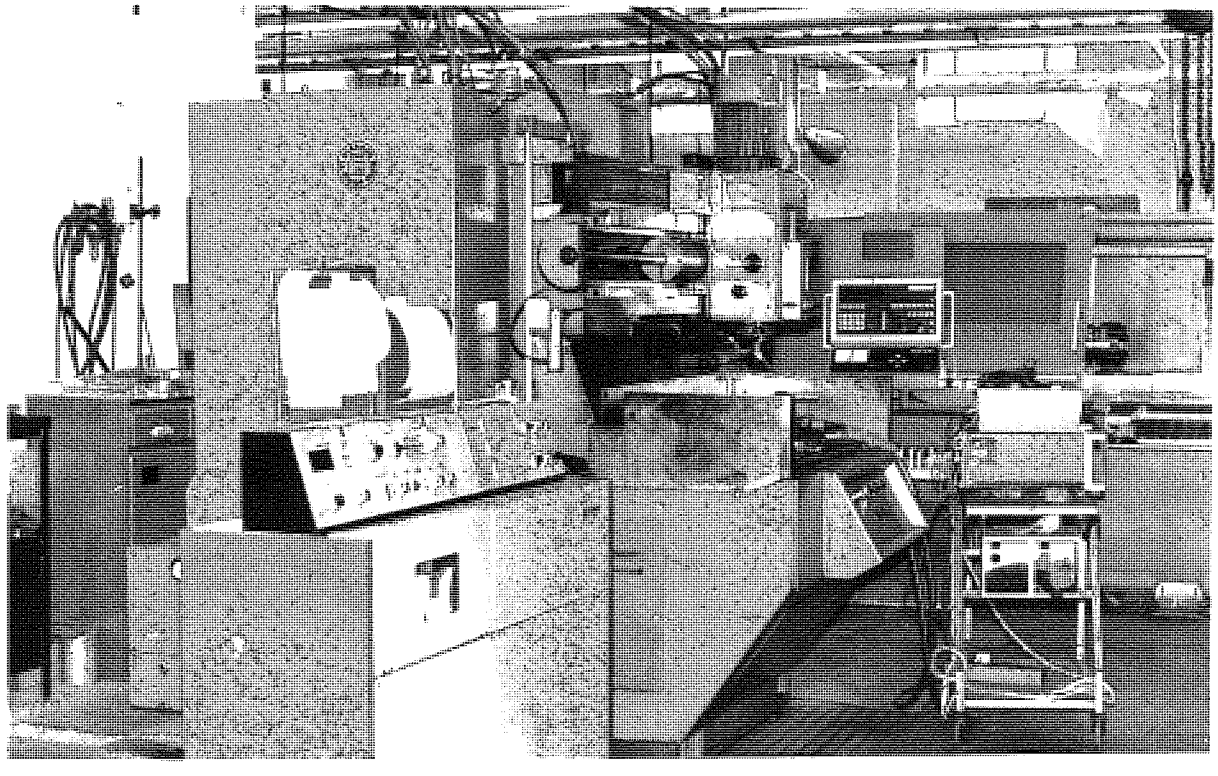


Plate 7.4 The Starrag Rigid Milling Machine

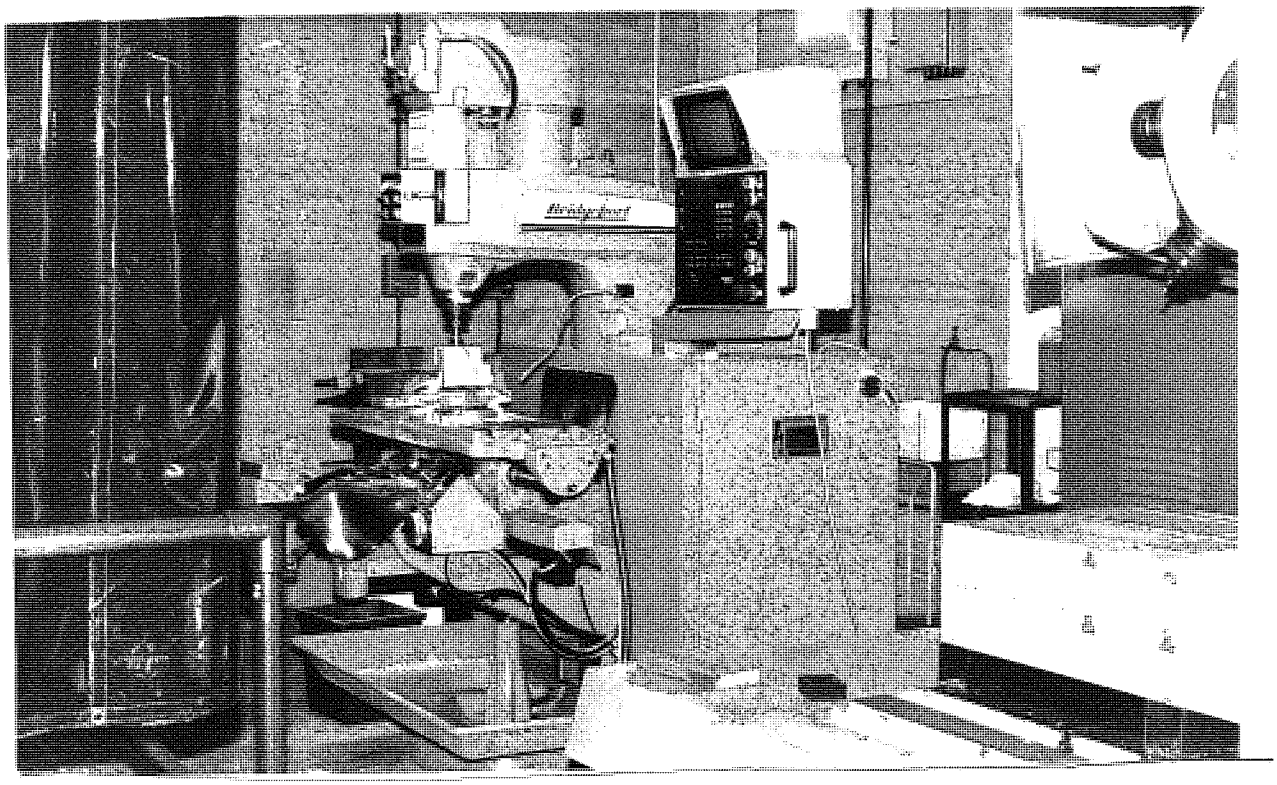


Plate 7.5 The Bridgeport CNC Machine

CHAPTER 8

CHAPTER 8

COMPUTER AIDED DESIGN SOFTWARE FUNCTIONS

8.1 INTRODUCTION

The aim of this chapter is briefly to outline some significant aspects of the programs with special emphasis on their implementation on the VAX workstation.

Some of these programs, originally written in FORTRAN 66, were developed for batch processing on the ICL 1904S mainframe computer system.

In this research, all these programs have been implemented on the VAX 11/750 and have therefore been modified accordingly to suit the VAX/VMS operating system. Data acquisition is achieved while running the programs interactively through a question / answer sequence via the terminal screen and keyboard. Simultaneously, the input information is stored in assigned data files which can be re-called when re-running the programs, to minimise keyboard entry.

8.2 PROGRAM 'BSLSQ'

This program by calling subroutine 'COEFF' finds least squares coefficients to set a B-spline curve to a set of given data, then by calling 'Funcal' generate the B-spline curve at intervals of 'OELX' finally plots the curve. The computed coefficients are stored separately for future use.

8.3 PROGRAM 'SECT'

Between 21 and 35 data points are required for each cross-section of the handle for

machining. Top and bottom have 50 cross-sections along the y-axis and the length of the workpiece is 140 mm. The top mould cavity is a first order curve and the bottom third order Fig (8.15).

8.4 PROGRAM 'BCSURFI'

This program reads the number of cross-sections and the number of cardinal B-splines required as well as the output from the program 'SECT' to produce a new file in the form required by the program 'BCSURF'.

8.5 PROGRAM 'INVERT'

The input for 'INVERT' is the output file from 'SECT' which produces vertices of a polygon, B-spline approximation which fits the given set of data points. 'INVERT' based on the inversion technique is computationally more effective and simpler to use. The fitted curve passes through all data points. 'INVERT' is more efficient in estimating an existing curve.

8.6 PROGRAM 'BSCURV'

This program by calling subroutine B-spline' approximate a B-spline curve to generate polygon and then plots the resulted curve.

8.7 PROGRAM 'BCSURF'

When the graphical routine 'VIEW' is represented with the data file from 'BCSURF' graphical output in the form of an axonometric projection of the surface is obtained. This can be rotated and viewed from different angles scaled up or down and/or sections magnified see Fig (8.16) and Fig (8.17).

8.8 DESIGN USING 'IBCSURF' PACKAGE

8.8.1 Curve Design with 'BSCURV'

In curve design using BSCURV all that is required as input data is the coordinates of the control polygon vertices, order of the curve and some control variables, see Appendix (2). The output is a series of points on the approximated curve at parametric intervals defined by the user. Gino-graph plotter routines were used to generate graphical output.

The curves produced using this design routine show that it fulfills many important requirements of computer aided curve design in engineering. Parametric representation makes the design of closed curves as simple as open curve design. With minimal extra computation space, curves can be produced if vertices of the defining polygon are given as points in 3-D space. The one disadvantage in using parametric representation is that equal parametric distances do not relate to equal scalar values at different sections of the curve. This means that more computing and data manipulation is necessary to produce points at equal arc intervals.

Unlike many global schemes, curves designed by this method do not have extraneous undulation. This can be seen in following figures. The shape and complexity of the curves is controlled using different 'handles' each producing an additional effect and more flexibility.

8.8.2 Flexibility of 'BSCURV' in Shape Control

The shape of a B-spline curve can be changed using different types of control in

the BSCURV routine. These are:

1) Changing the order of the curve
2) Changing the position and number of defining polygon vertices and using repreated vertices.

In Fig (8.1) the polygon defined by vertices (P_0, P_1, P_2, P_3) is used to construct different order curves. A second order curve comprises straight lines that join the vertices. The resulting curve becomes smoother and looks tighter but the end slopes are unaffected as the order of the curve increases.

the fourth order curve corresponds to the Bezier curve as the number of vertices is equal to the order of the curve.

Local modifications can be made simply by changing the position of a vertex. See Fig (8.2). The change only affects a limited part of the curve thus illustrating the non-global property of B-spline.

The Bezier curve requires the number of vertices to be fixed by the order of the curve, while B-spline only requries that:

ORDER of the curve $\leq$ no. of vertices. Therefore in constructing a curve, the shape can be modified by inserting additional vertices. In Fig (8.3) by inserting an extra vertex, the minor alteration caused can be seen.

Multiple vertices can be important in controlling curve shape. As shown in Fig (8.4) for a set of fifth order curves, double, triple and quadruple vertices are given at (P_2). The curve is pulled towards the vertex as multiplicity increases. The

quadruple vertex for the fifth order curve creates a knuckle at this vertex as the slope becomes discontinuous.

By defining an equal number of collinear vertices as the order of the curve, linear segments may be embedded in a curve as shown in Fig (8.5) and from Fig (8.6) to Fig (8.11).

Little computer storage is necessary to implement this routine. The greatest amount is needed by the graphical routines. Computational time is dependent on the number of vetices of the defining polygon and the order of the curve. The following figures show examples of curves designed with BSCURV.

8.8.3 Curve Fitting

the essential aim of the system was to include newly designed curves, existing curves and constraints presented elements in surface design. Curve fitting routines were developed in the package to accomplish this aim. The two fitting routines investigated were:

1) The least squares approximation and
2) Reisenfeld's Inversion technique.

Each showed its importance in difficult areas of application.

In looking for the best fit to a set of data points,using BSLSQ based on the least squares method, different order B-spline curves can be used and the knot set can be changed to produce the desired curve. A spline curve which fits the data and is force from spurious oscillation is better obtained if knots are chosen grouped

together more closely where the function (underlying the data) or its derivations change more rapidly than elsewhere.

Another handle in manipulating the shape of the fitted curve is the prescription of multiple knots where their multiplicity does not exceed the order of the curve. Where multiple knots are defined, the curve will more closely approximate the data points, but the degree of continuity of its derivatives is reduced simultaneously. There is negligible error in the computed values of the fitted curve compared with data points in most practical situations. The method is very satisfactory because of the flexibility of manipulation of the shape of the curve expecially if used in CAD systems based on B-splines Fig (8.12) and Fig (8.13).

The program INVERT which is based on the inversion technique is comptationally more efficient and simpler than BSLSQ. The fiitted curve passes through all data points. In INVERT, the same number of data points must be used as vertices to be evaluated. By increasing the number of points, the least squares approximation may give a better fit with fewer vertices. It can be said that BSLSQ is more easily used in fitting a curve to a large number of empirical data points, whereas INVERT is more efficient in approximating an existing curve.

8.8.4 Surface Design

In the design of pipe-like objects such as exhaust manifolds, plastic handles, bottles etc, and surfaces such as aircraft wings, ship hulls, turbine blades, car bodies etc. BCSURF is ideal.

The design procedure using BCSURF is simple, but alterations to the designed shape can only be achieved by modification, insertion or removal of cross-sectional curves. Even in an interactive situation it is difficult to see the immediate or dynamic

effect as the modification takes place. In the system, graphical routines used allow isometric projection of the surface to be rotated and viewed from different angles scaled up or down. Sections of the surface can be windowed and magnified for closer examinations. Batch processing does not permit the user on-line plotting command control - of prime importance in these circumstances. Therefore, at each stage in the process of surface design, multiple sets of data giving different control variables, plotting commands and defining values can be input. The result is in three forms:-

a) Printed output

b) Plotted result of the shape

c) Parts of the result to be used in the next stage of the process are saved in the file.

Fig (8.14) shows the flowchart of the surface design procedure.

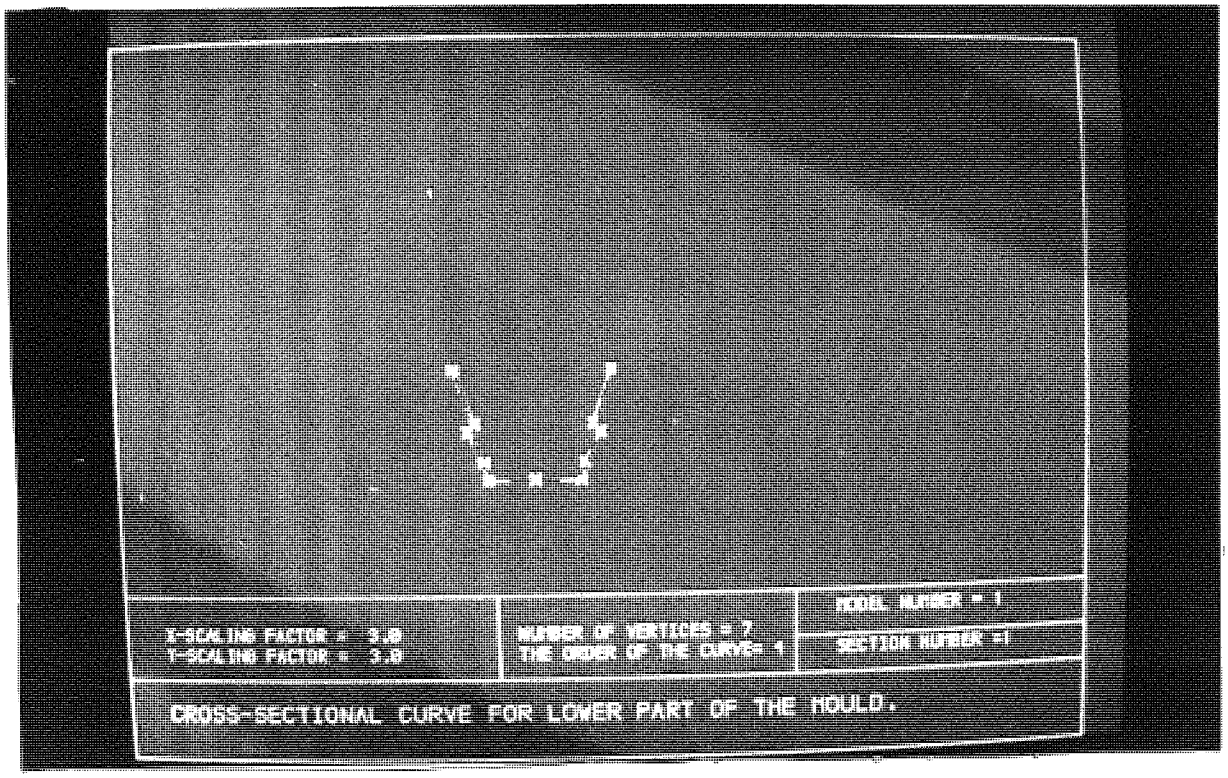


Plate 8.1 Cross-sectional Curve for Lower Part of the Mould

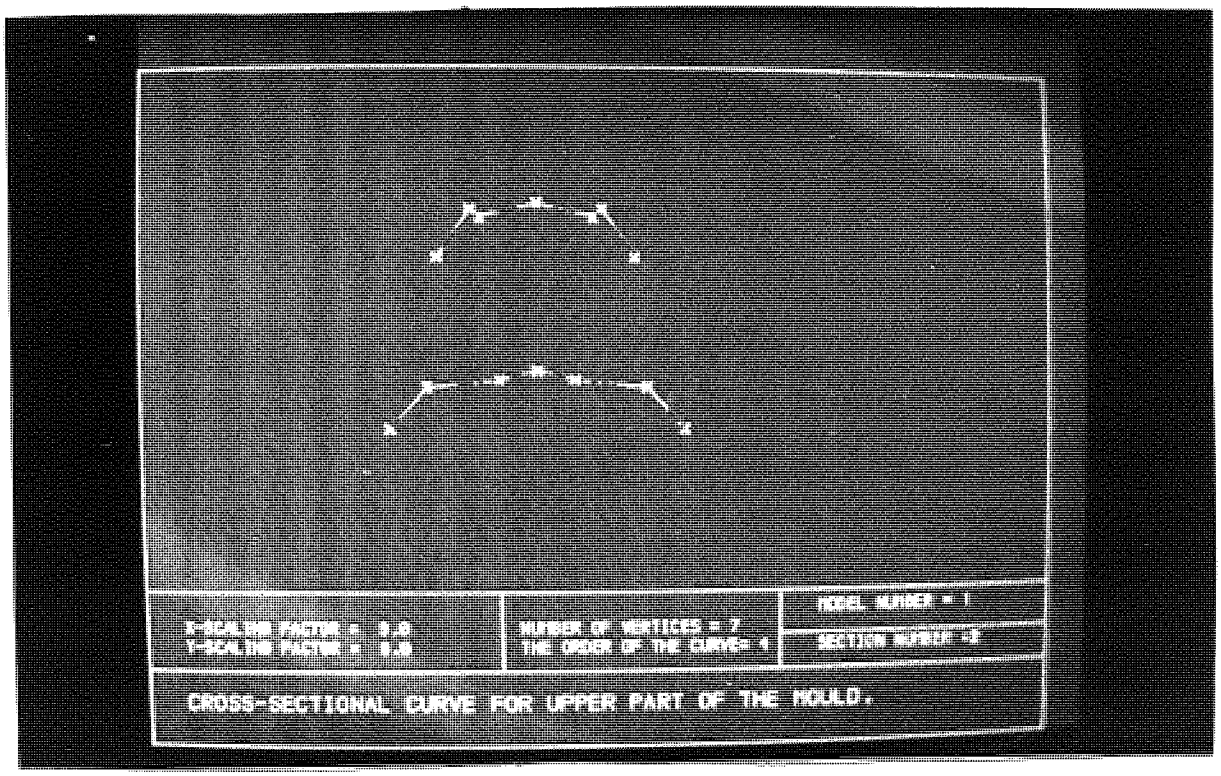


Plate 8.2 Cross-Sectional Curve for Upper Part of the Mould

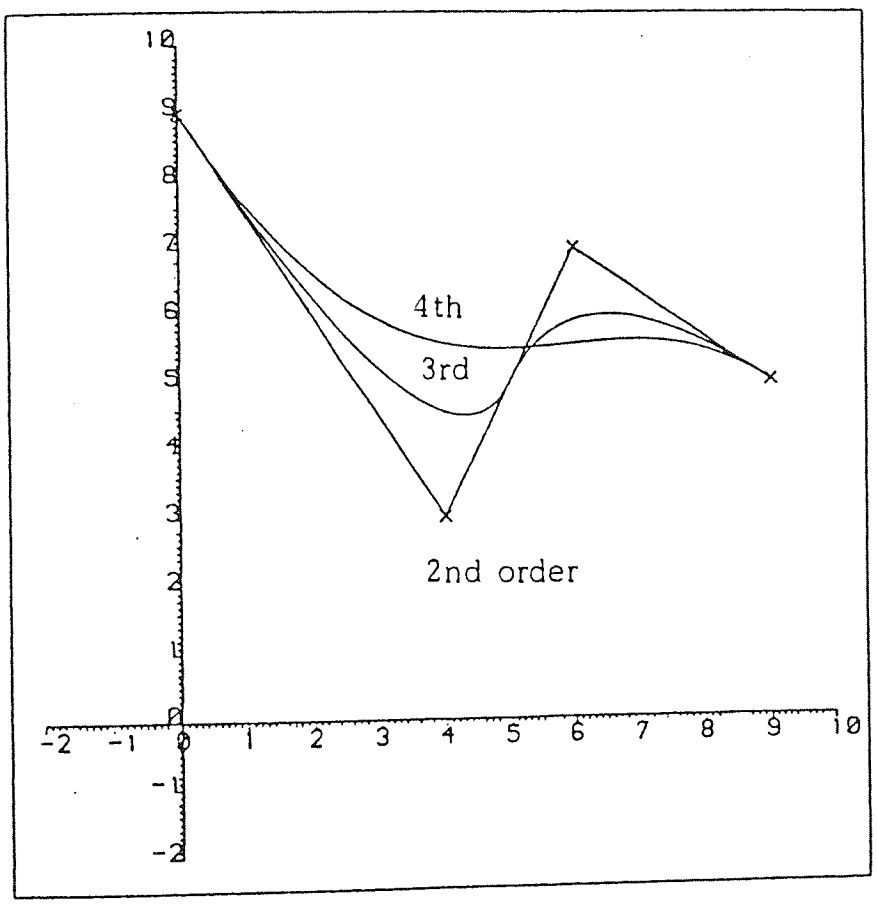


Fig. (8·1) B-spline Curves of Different Order

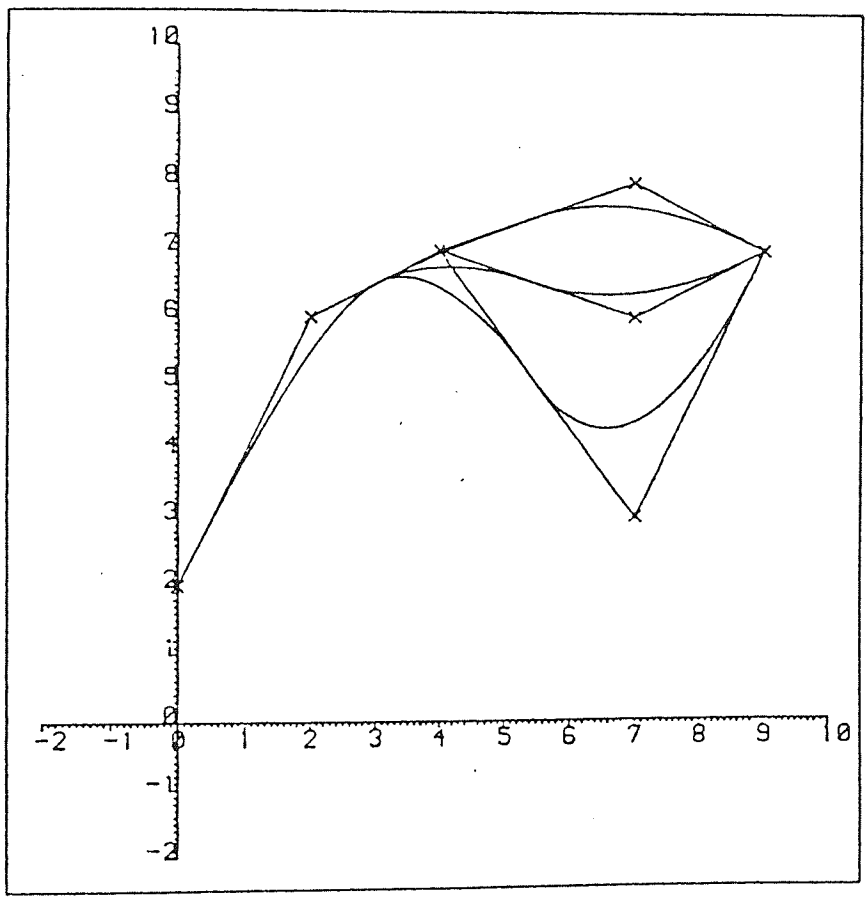


Fig. (8·2) Local Modification on B-spline Curves

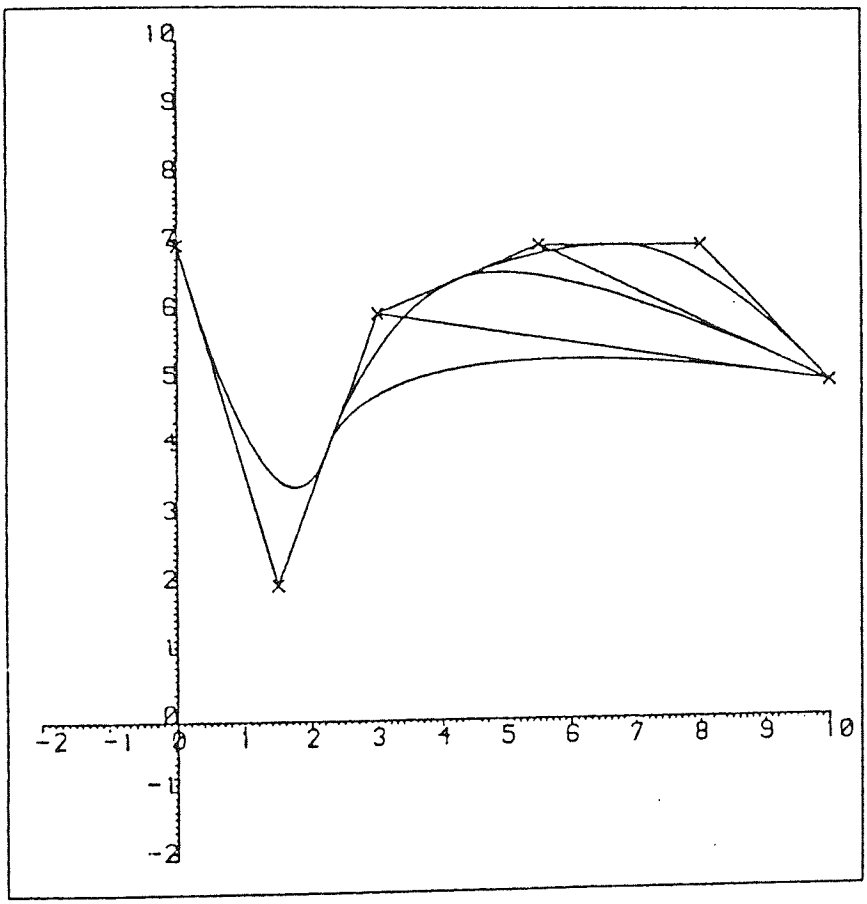


Fig. (8·3) Curve Modification by Insertion of Vertices

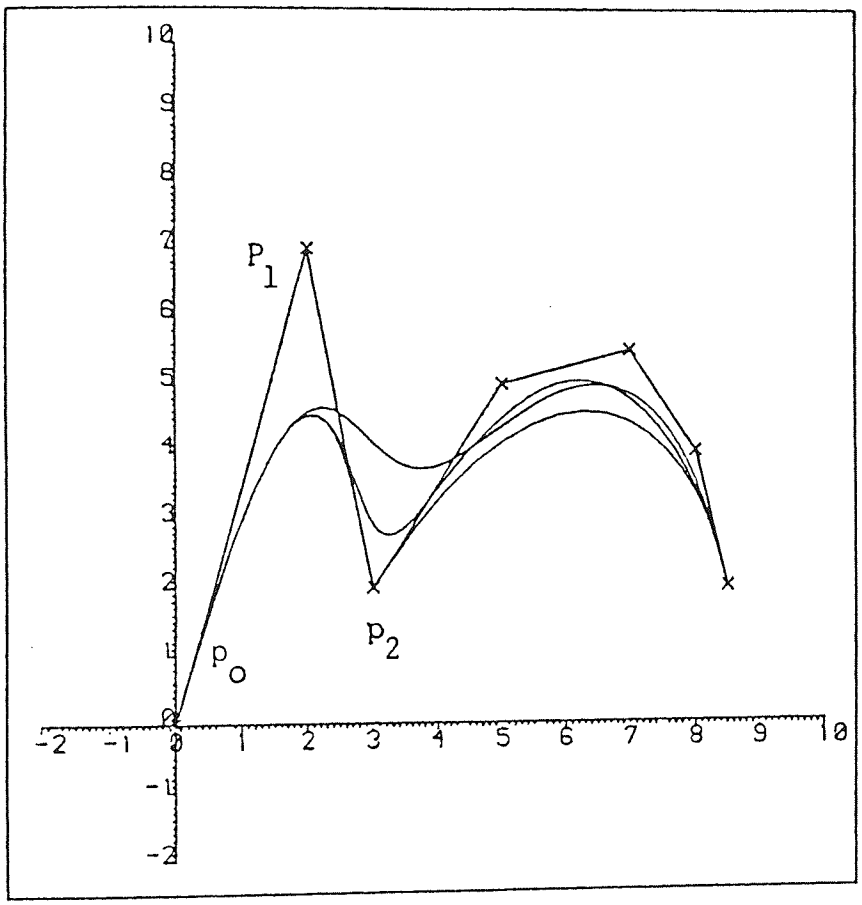


Fig. (8·4) The Effect of Multiple Vertices

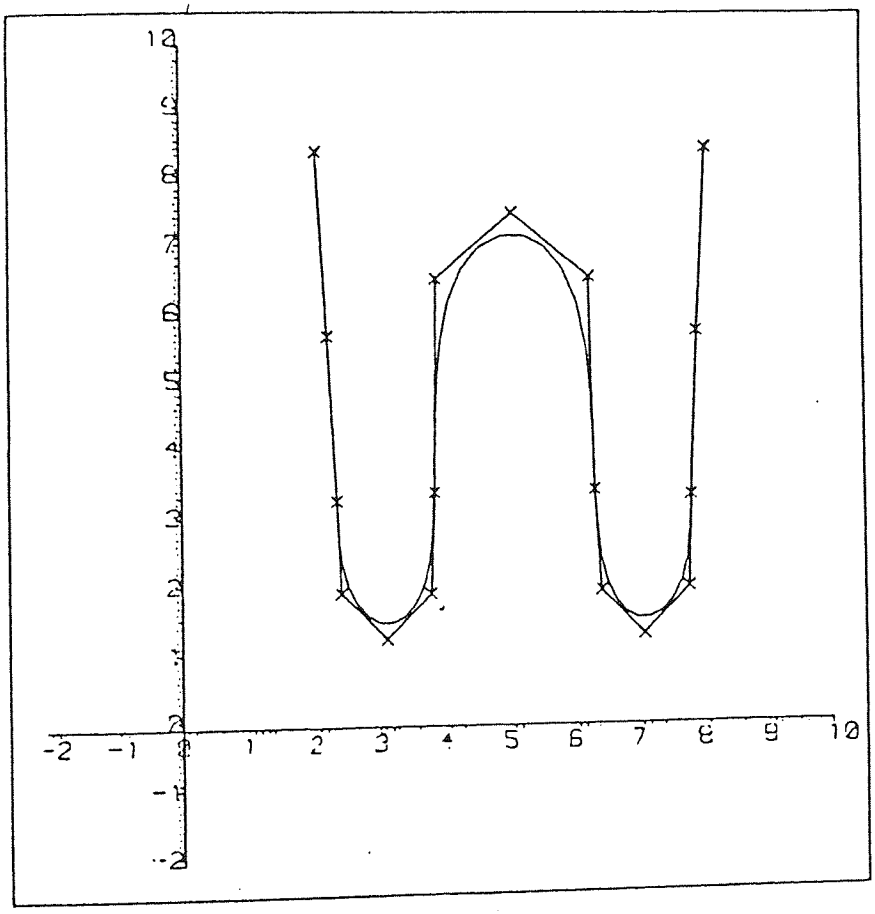
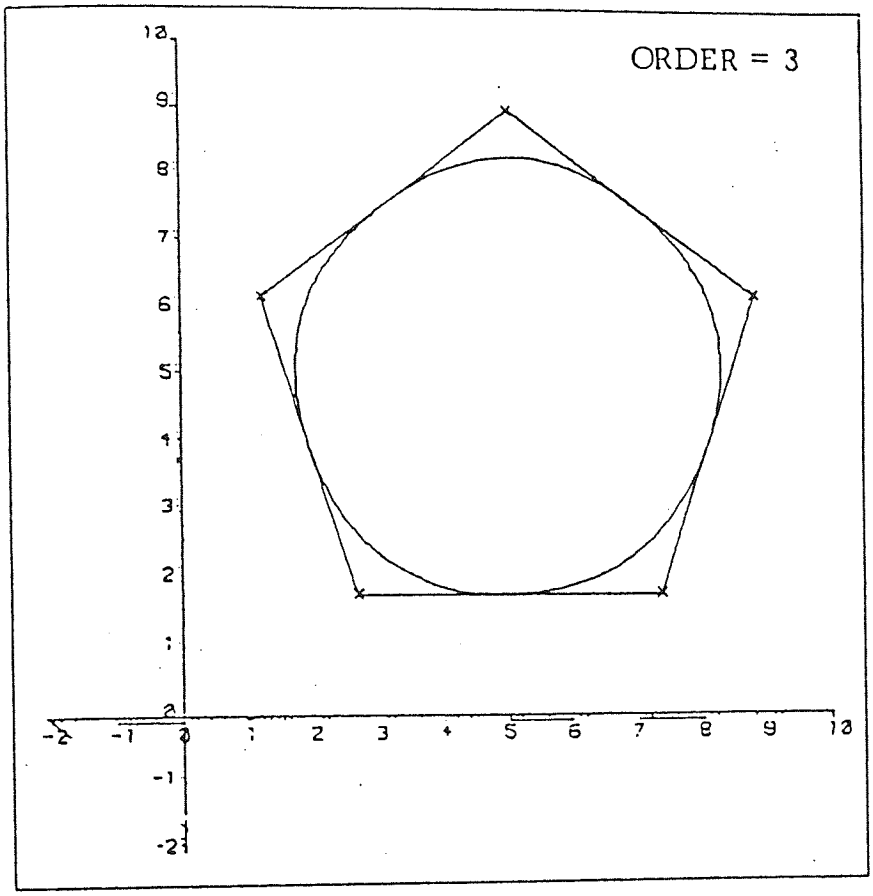
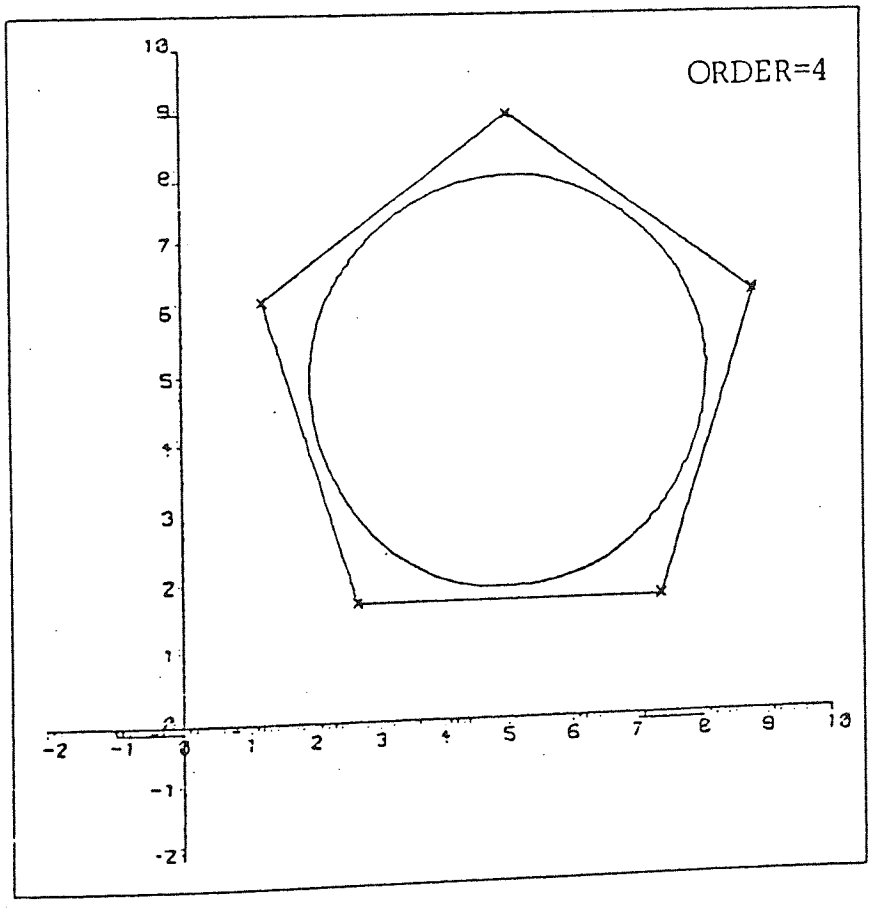


Fig. (8.5) Embedding of Linear Segments



Flg. (8·6) A 3rd Order Closed B-spline Curve



Flg. (8·7)

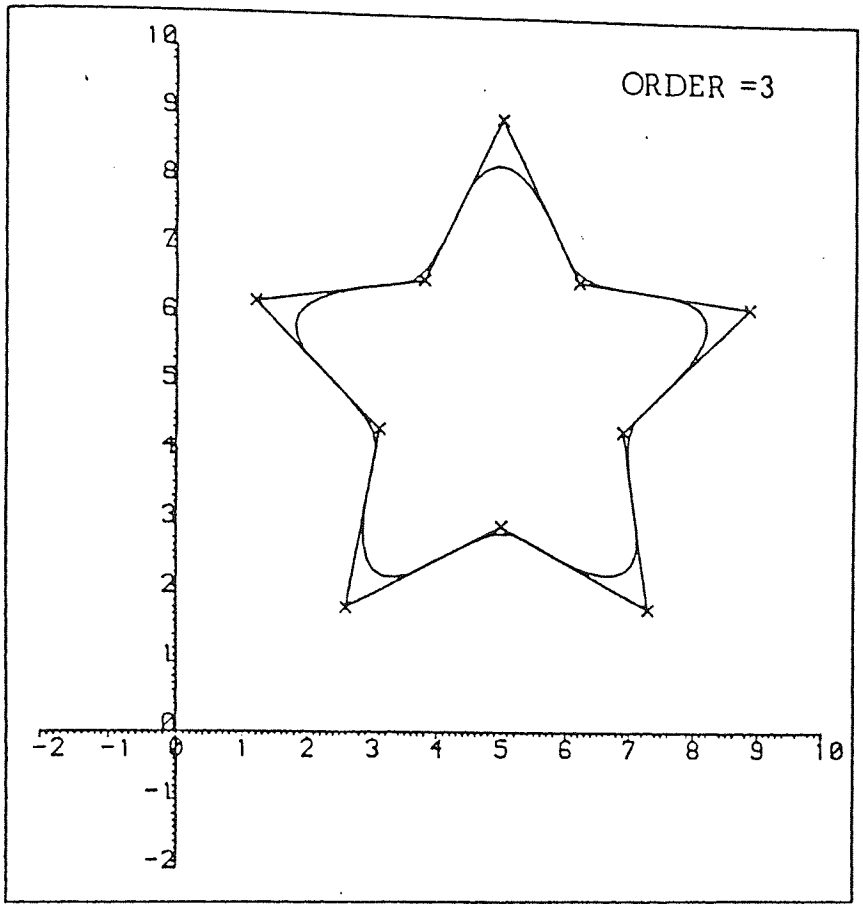

ORDER =3
10
9
8
7
6
5
4
3
2
1
0
-1
-2
-2 -1 0 1 2 3 4 5 6 7 8 9 10

Fig. (8·8)

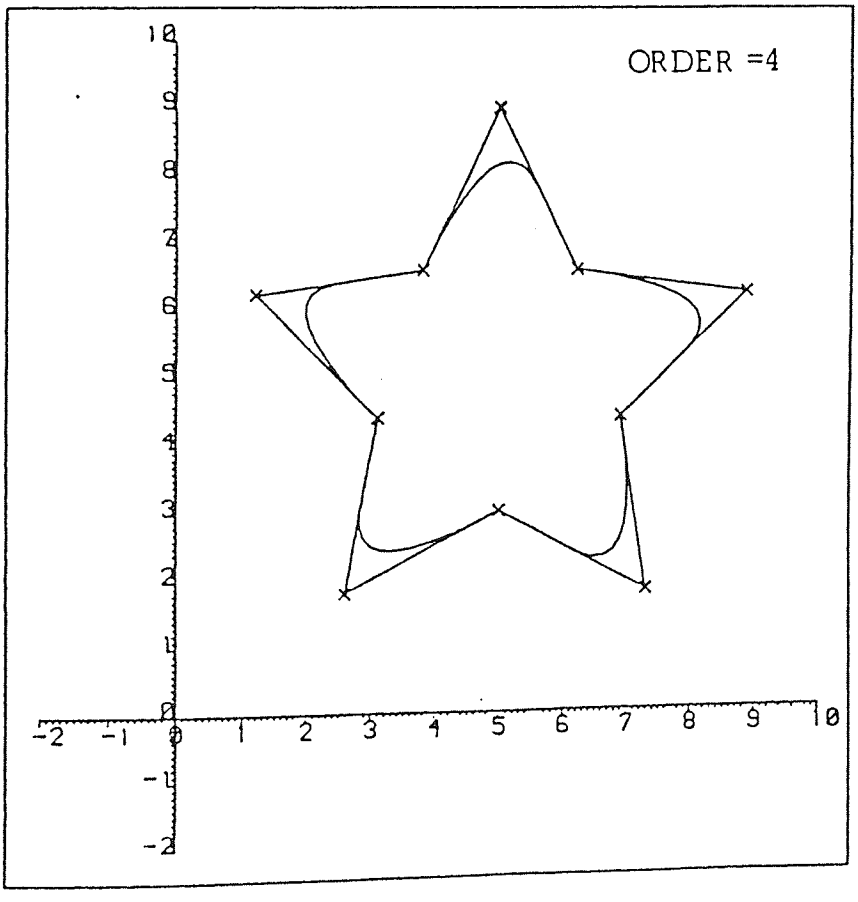

ORDER =4
10
9
8
7
6
5
4
3
2
1
0
-1
-2
-2 -1 0 1 2 3 4 5 6 7 8 9 10

Fig. (8·9)

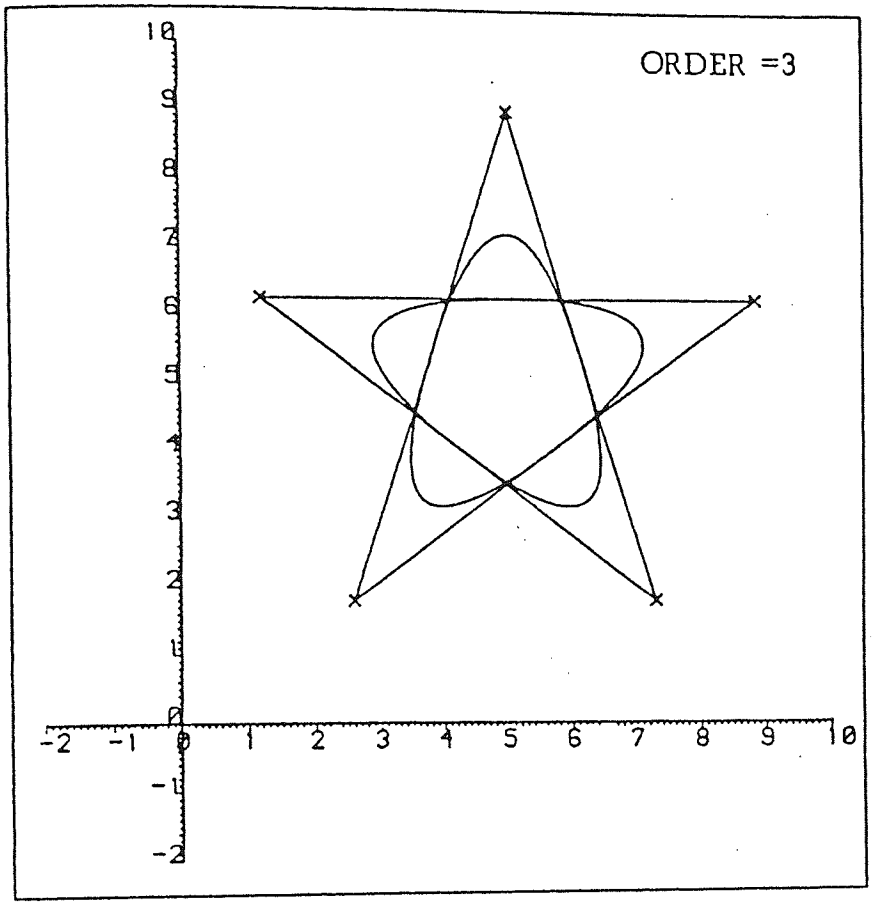


Fig. (8·10)

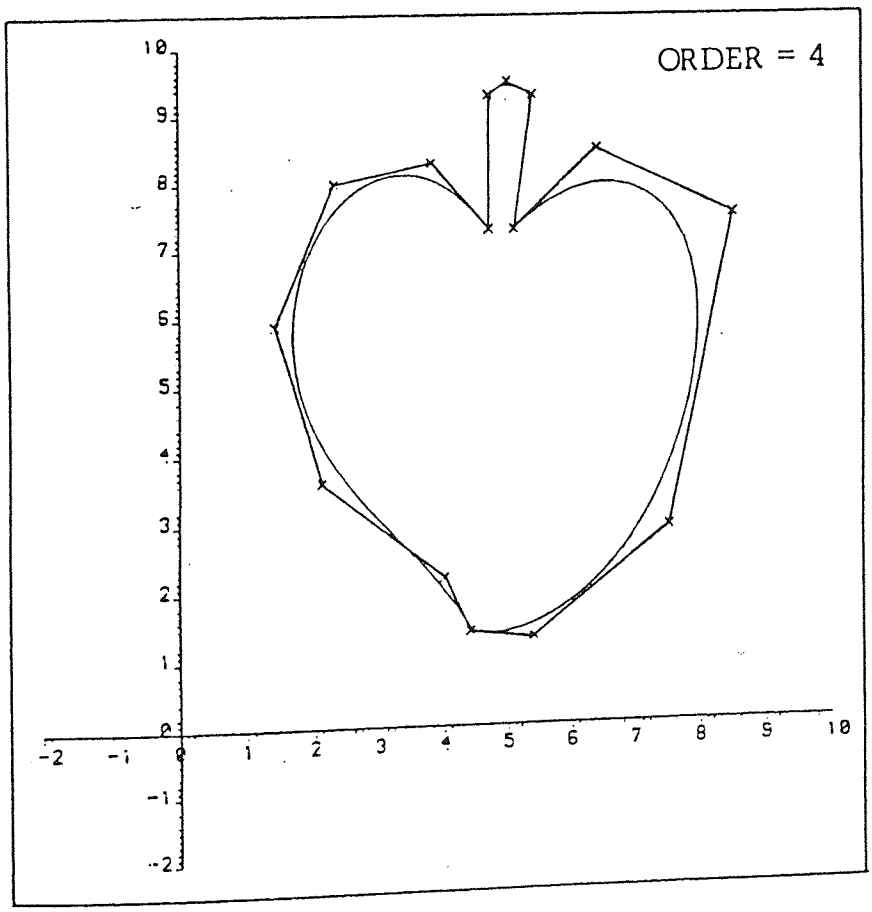
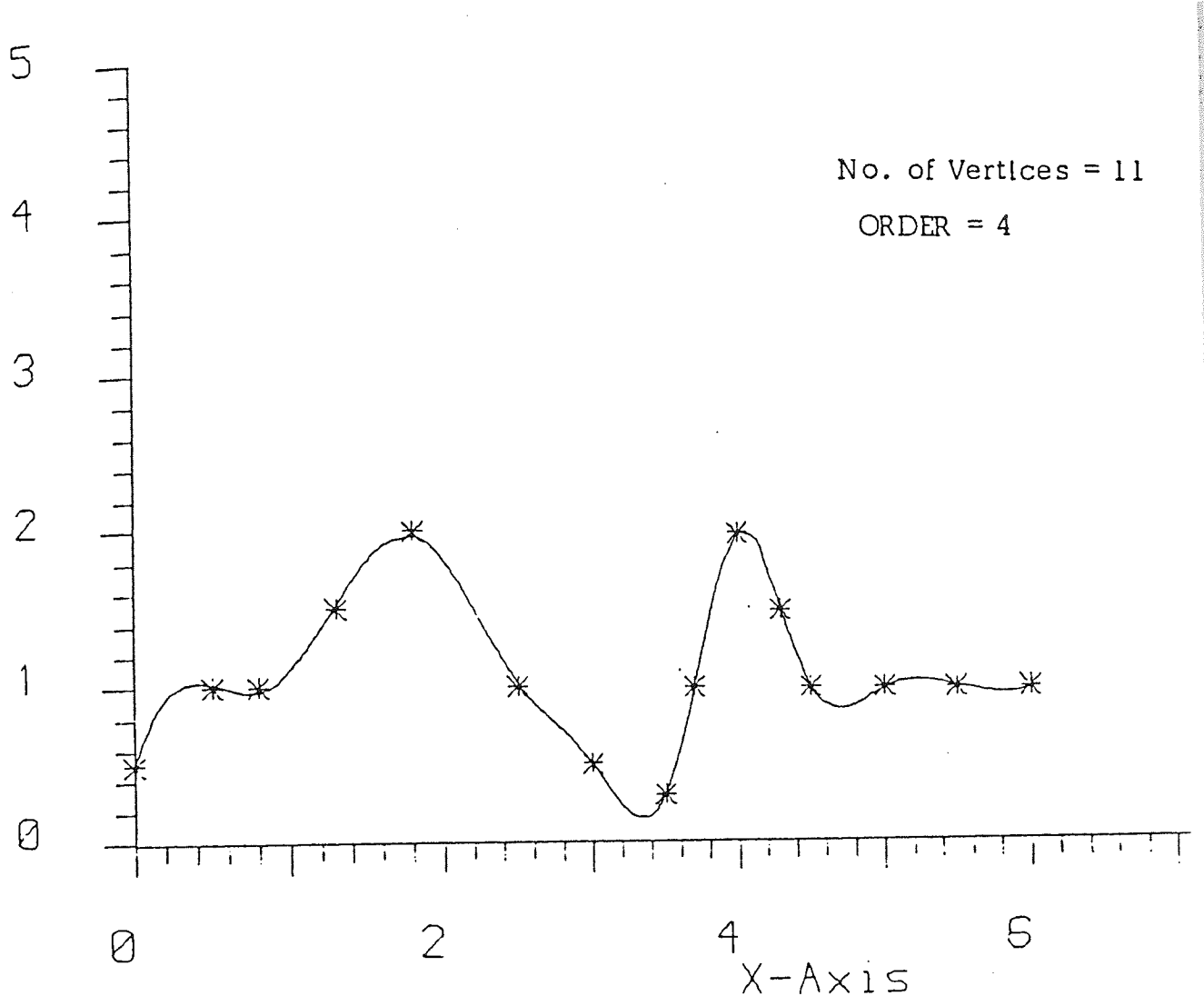


Fig. (8·11)



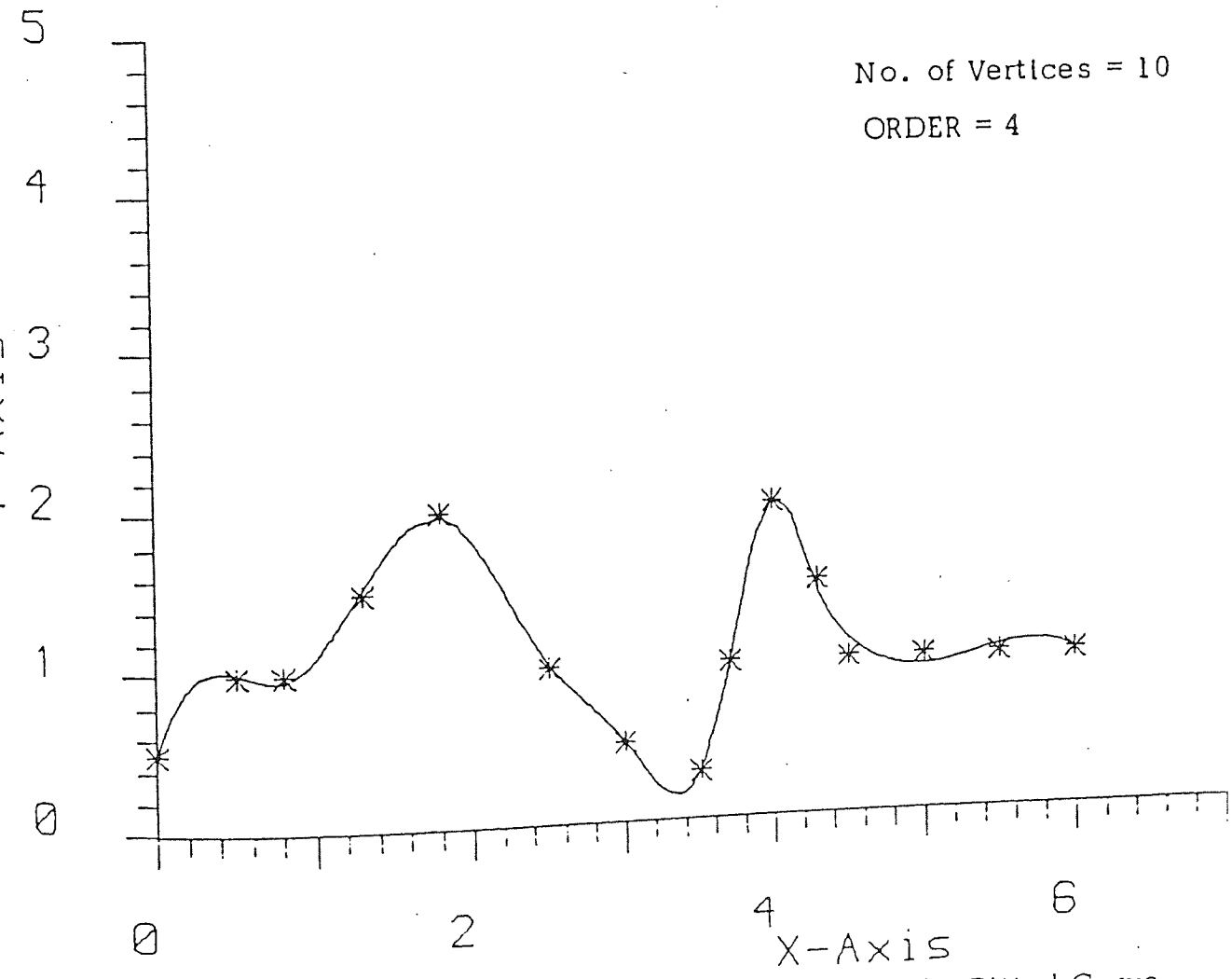
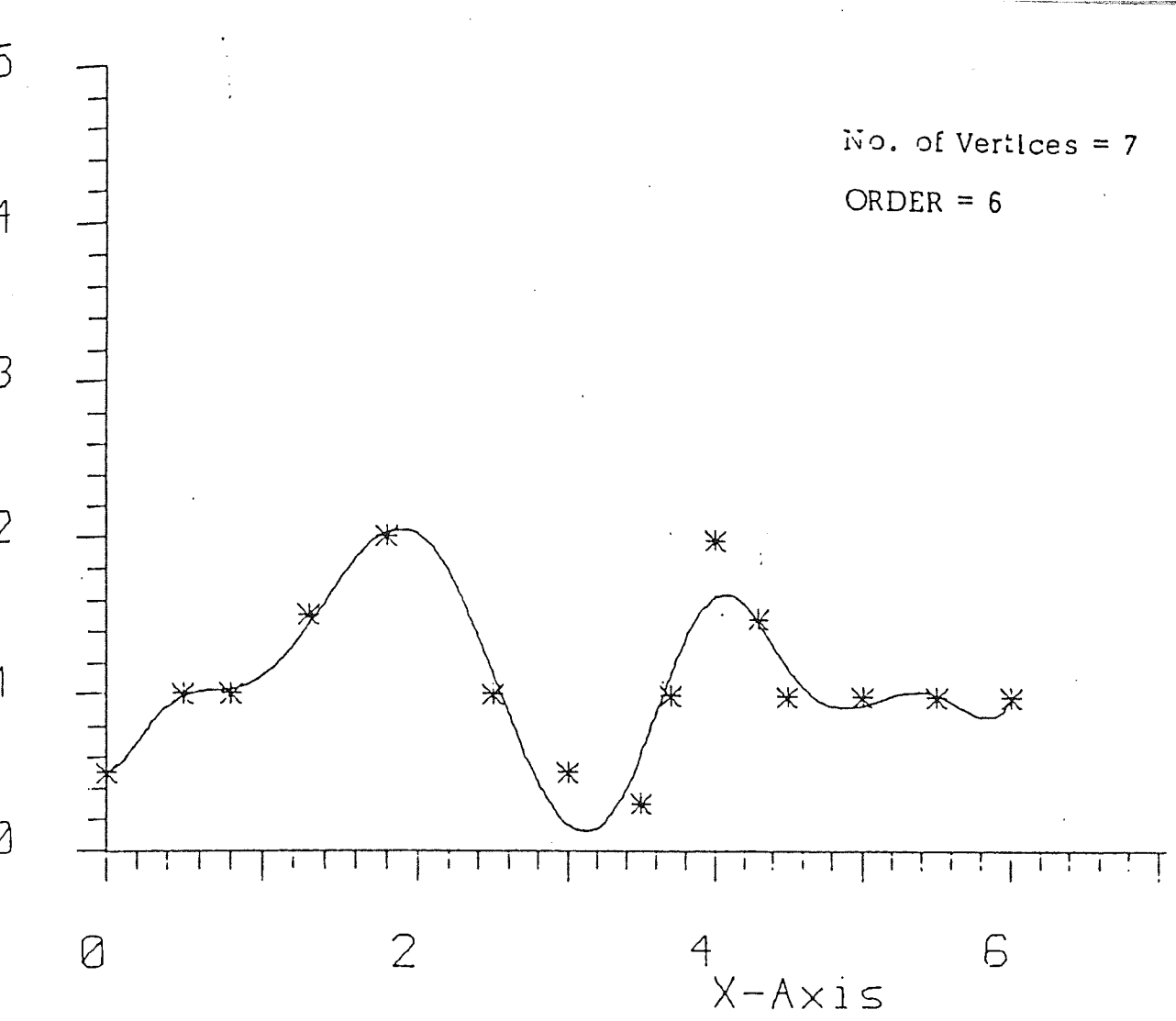
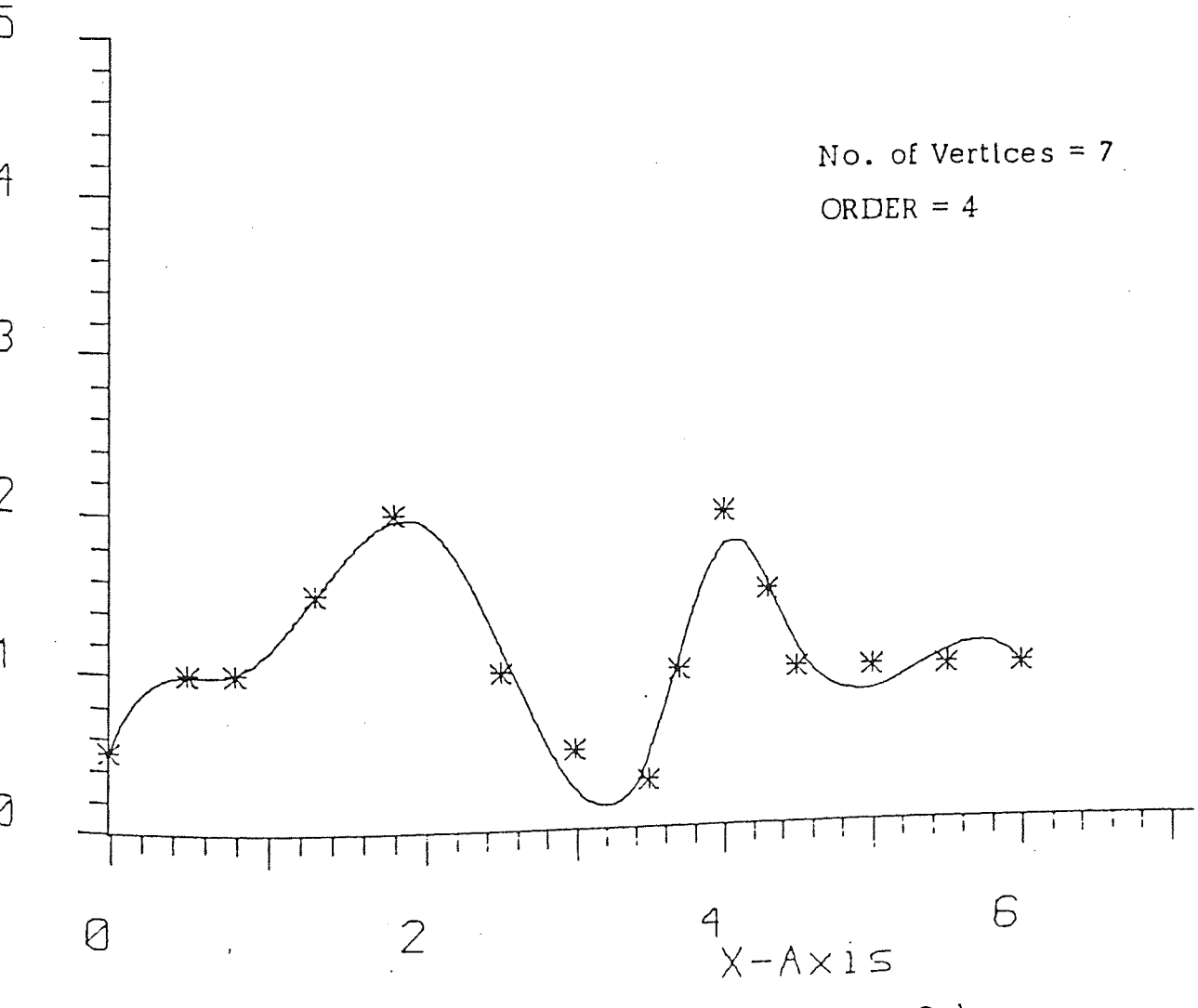


Fig. (8.12) The Effect of Selected Vertices on the Fitted Curve


No. of Vertices = 7
ORDER = 6
0
2
4
6
X-Axis


No. of Vertices = 7
ORDER = 4
0
2
4
6
X-Axis

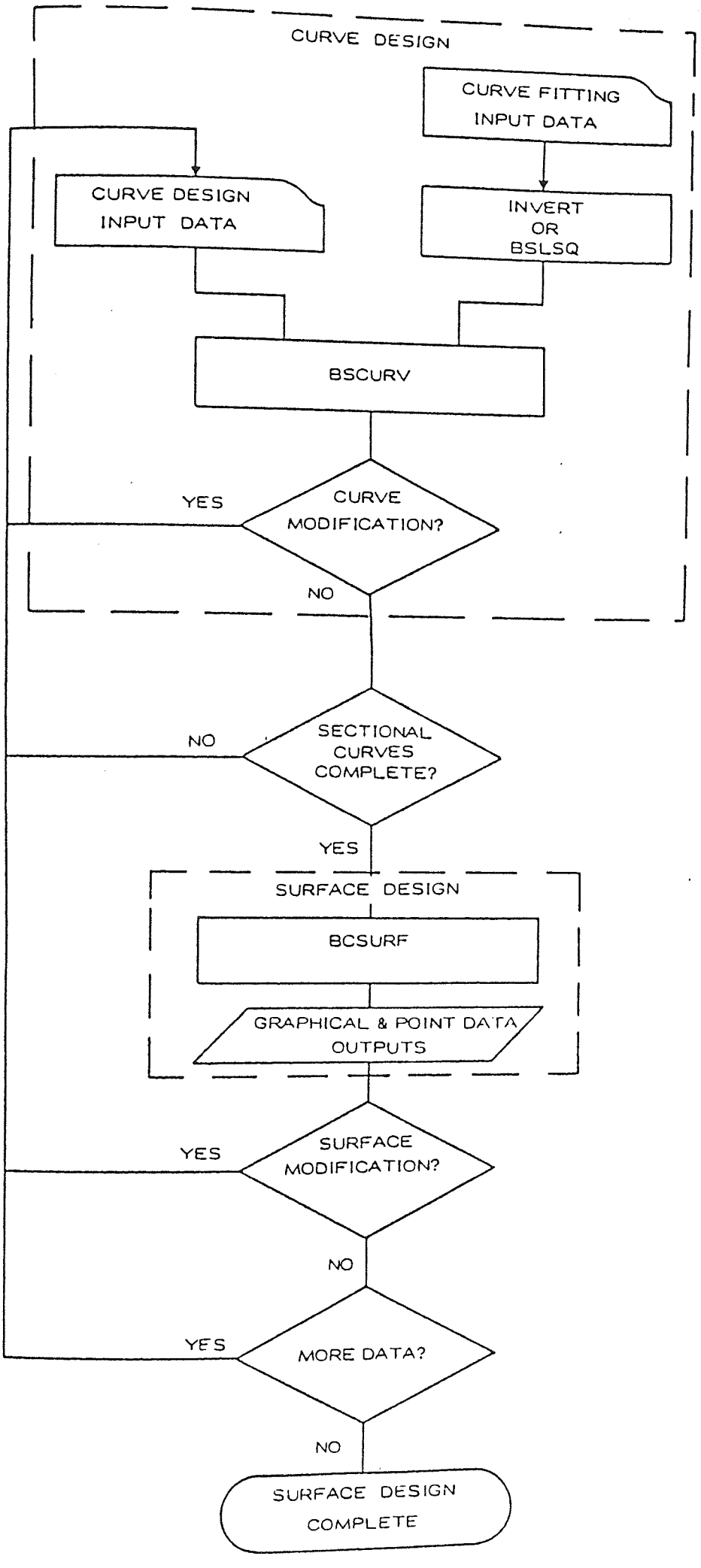


Fig. (8.14) Surface Design Procedure in the System

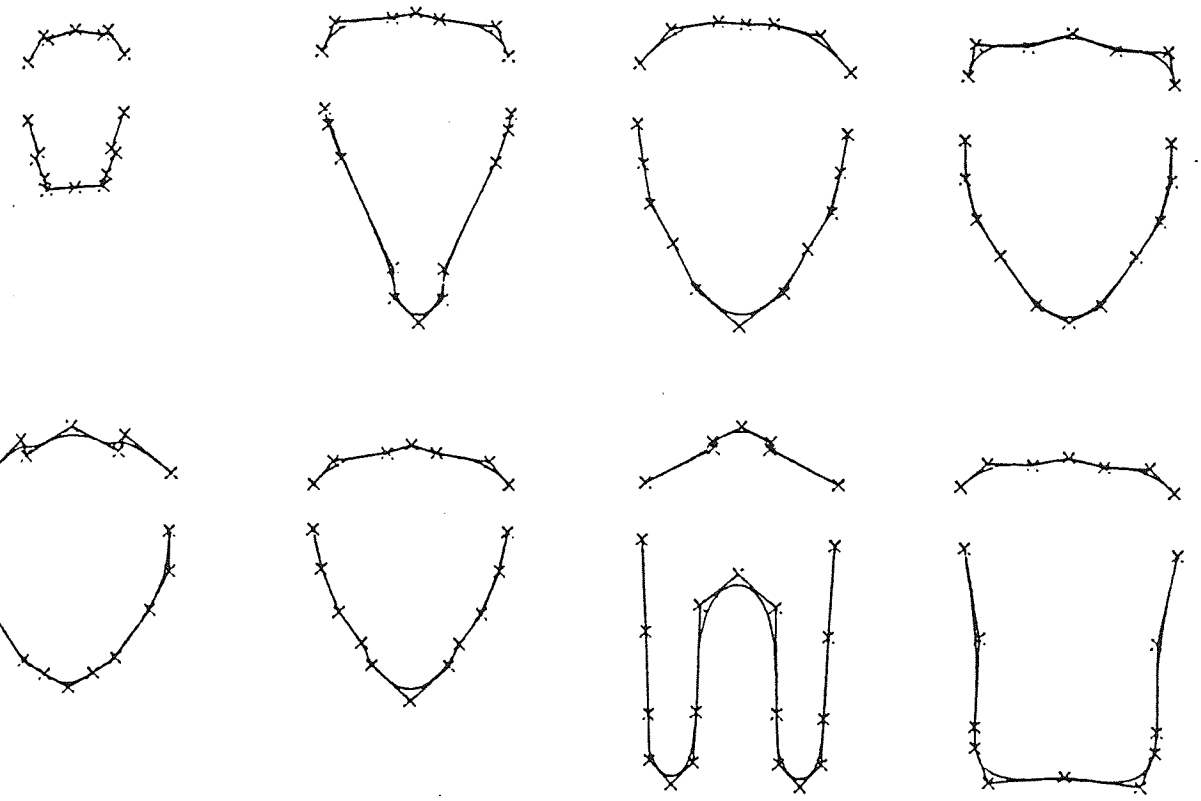


Fig. (8·15) Cross-sectional Curves for Upper and Lower Parts of the Mould

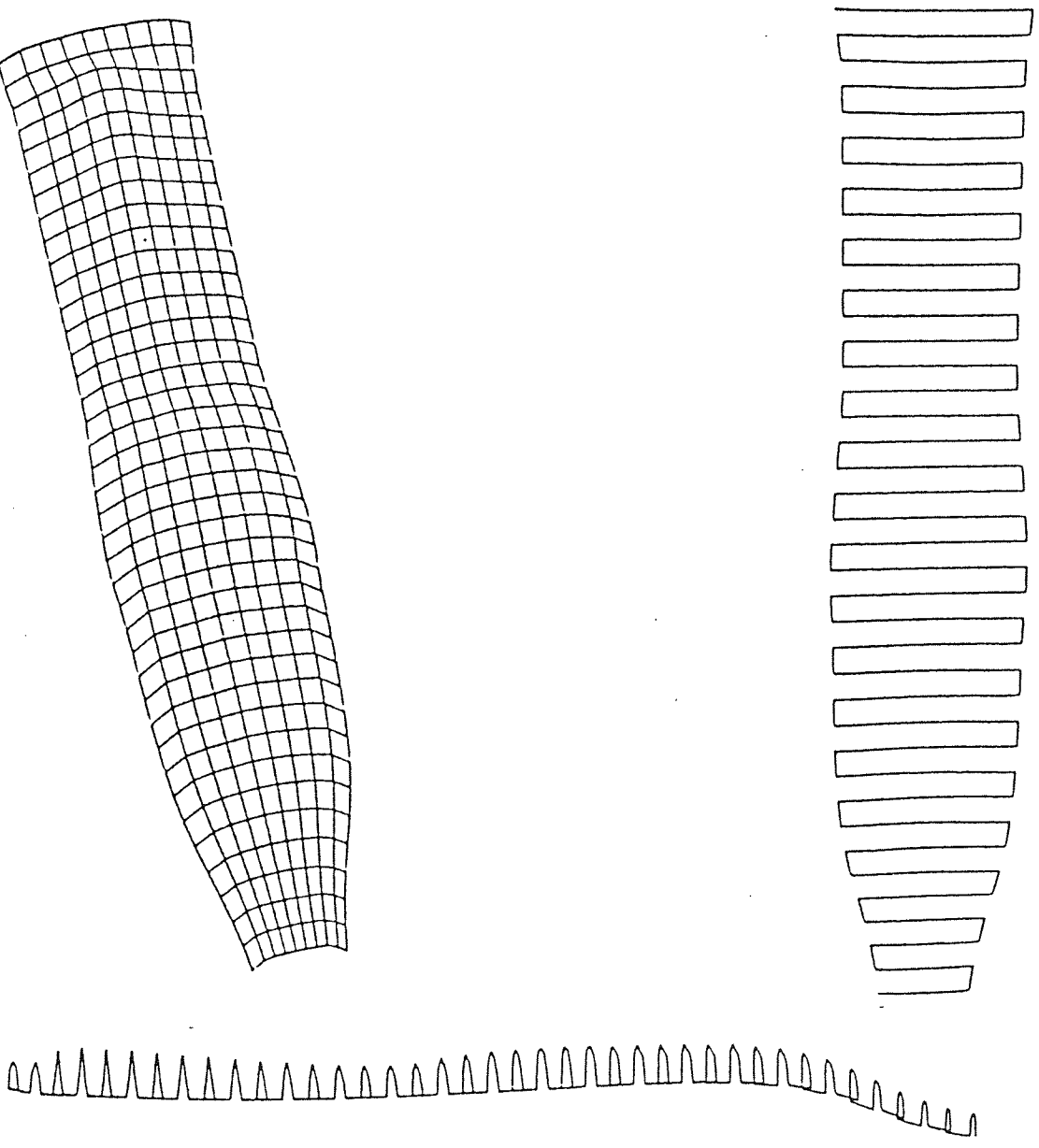
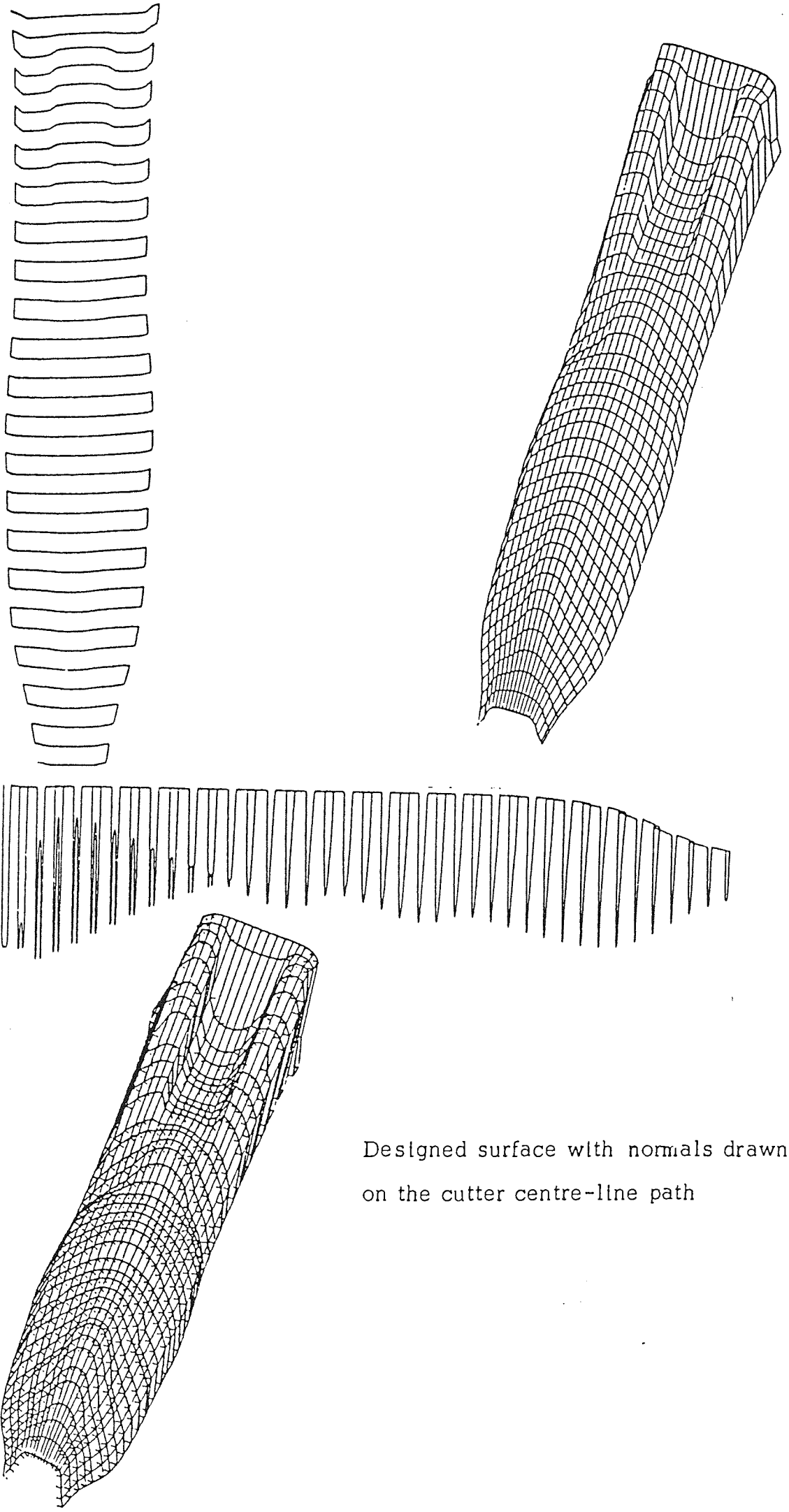


Fig. (8·16) Upper Part of the Mould Viewed from Different Angles



Designed surface with normals drawn on the cutter centre-line path

CHAPTER 9

CHAPTER 9

9.1 INTRODUCTION

The conventional manual method for producing three dimensional objects comprising partly or totally free-form curved surfaces relies heavily upon the skill of the machinist. The more complex the shape of the object profile the greater the demand on skill and time.

Numerical control machining was applied to improve on the traditional manufacturing method. This was enhanced with the development of computer aided part-programming facilities for the preparation of NC tapes for machining the object.

To integrate design and manufacture, the processes must be linked so that the results from the design stage can be easily transferred into the manufacturing system.

There are three main problems when cutting a free-form surface:

1. Cutter offset
2. Accuracy
3. Cutter Interference

A ball nosed cutter is usually used and the offset calculated by finding the direction of normals on the surface and the cutter offset by its radius along the vector over the surface.

The mathematical representation of surface in this work has allowed both analytical and numerical method to be employed in computing the cutter offset. (See chapter 5). A post-processor program to produce the required compatible format for the N.C. machine was written, as all others, in FORTRAN 77 and run interactively on the VAX 11/750 super mini-computer.

9.2 PROGRAM 'BCSURN'

This program by calling sub-routine 'REPART' first fits B-spline curves to cross sections of a three dimensional object, then by calling subroutine 'EQLT' interpolates selected points on the B-spline curves using cardinal cubic spline. Finally, it calculates direction cosines of normals to the surface (see Chapter 5).

9.3 PROGRAM 'NUMOFF'

Program 'NUMOFF' reads in the point data which are organised in the matrix form. The radius of the cutters used (up to eight tools from the largest available size) and control parameters (see Chapter 5).

9.4 PROGRAM 'ANAOFF'

This program by reading data points on a surface and direction cosine at those points, produces cutter centre-line path data along parameter u, and, or v. By setting 'INTCHCK' switch cutter interference check can also be carried out. In this process the size of the tool used is successively reduced to avoid excessive removal of adjacent area.

9.5 POST PROCESSOR

9.5.1 Introduction

Post processors are necessary for converting the cutter location data information into an NC tape program that suits the exact requirement of the particular NC system on which the component is to be machined. There are two types of post-processors; namely: Specific post - processors which output the exact code for particular machine

tool; and general post-processors which output a generalised format which needs to be edited to satisfy the requirements of a particular machine tool (75, 76, 77).

In this research, as it has been the intention to design an NC part-programming system to cope with the two types of post-processors as follows:

9.5.2 Specific Post-Processor "Program Prepare 3-D"

The cutter offset coordinate outputs from the machining routine 'ANAOFF' is in decimal value, but the Olivetti NC machine requires point coordinate as input data to the control unit.

A post-processor program called 'prepare 3-D' and 'prepare' are written in FORTRAN 77 to produce the required compatible format for the NC machine. The difference between program 'prepare 3-D' and 'prepare' is in the 'write' statement format. In program 'prepare 3-D', all the N, G, X, Y, Z, F, R, M values are suppressed so that they are printed in one block in the part-programme, and in programme 'prepare' all these values are printed in a single column for the punching of the paper tape.

This post-processor programme consists of two routines called 'REMOVE' and 'COMPLETE' respectively.

9.5.2.1 Routine 'REMOVE'

This routine provides the following functions:-

a) Removes decimal point from all the point coordinates.

b) Inserts leading zeros to all the point coordinates and the sequence number N.

c) Inserts canned cylce G54 to move the tool rapidly to the first cutting positions in X-Y plane and canned cycle G55 to cancel the rapid motion.

d) Inserts R-plane value.

e) Inserts miscellaneous function MO3 to switch on the spindle and MO8 to switch on the coolant.

f) Inserts appropriate feedrate for the 21/2 or three dimensional machining.

9.5.2.2 Routine 'Complete'

This routine provides the following functions:

a) Inserts RO to move tool to change position.

b) Inserts canned cycle G54 to move tool rapidly to the tool set point and canned cycle G55 to cancel the rapid motion.

c) Inserts miscellaneous function M05 to switch off spindle and coolant and M30 to rewind the paper tape.

9.5.2.3 Versatility of the Post-Processor Program

In addition to the above mentioned functions, a control variable K is employed in this programme to generate different part-program formats:-

K = 0, generate part-program for 3-D milling as shown in Fig (9.2 (a)).

K = 1, generate part-program for 2.5 -D milling as shown in Fig (9.2 (b)).

K = 2, generate part-program for contour milling on X - Y plane as shown in Fig (9.2 (c)).

K = 3, generate part-program for contour milling on Y - Z plane as shown in Fig (9.2 (d)).

K = 4, generate part-program for plain and pocket milling in X-direction as shown in Fig (9.2 (e)).

K = 5, generate part-program for plain and pocket milling in Y - direction as shown in Fig (9.2 (f)).

9.5.3 The Post-Processor

This is a general post-processor applicable to any CNC milling machine, the application of this post-processor was in this case, limited to machining the mould of the plastic handle, but could be adapted for other use.

The input for this program should be the cutter location data file.

There are two subroutines in addition to the subroutines 'REMOVE' and 'COMPLETE' in program 'PREPARE 3D'.

a) Subroutine 'CREATE'

This subroutine reads the input of machine data file (i.e. G-codes, M-codes and the format for NC programming) interactively via the terminal. All data input is stored in file for any further execution. An example of machine code data input using subroutine 'CREATE' follows:

G - Codes

(1)	Inch Mode	Metric Mode	Pure Absolute Programming	Reference Point Search

G-Codes contd..

(2)	Rapid Linear Interpolation	Linear Interpolation	Circular Interpolation (CW)	Circular Interpolation (CCW)
(3)	Spindle Cutting Surface Speed	Spindle Speed		
(4)	Feed per Time	Feed per Spindle Rotation	Radius Compensation (Tool to the left)	Radius Compensation Tool to the right)

M- Codes

(5)	Spindle Forward	Spindle Reverse	Coolant On	Spindle Stop	Spindle Forward and Coolant On	Spindle Off and Coolant Off
(6)	Program Stop	Optional Stop	End of Program	End of Program with Rewind		
(7)	Low Speed Range (Lower range and higher range)			High Speed Range Lower range and higher range)		

Other Program Formats

(8) X-Coordinate Code for the Centre Point of an Arc

Z-Coordinate Code for the Centre Point of an Arc

(9) Tool Number Code

Tool Compensation Code

(10) Code for Cancelling the Tool Offset

All the data supplied by the user is stored in a data file ("NAME". DATA), where "NAME" is the name of the CNC milling machine, which can be retrieved when further processing is required for the same machine.

b) Subroutine 'EXIST'

This subroutine reads the input of MACHINE DATA (i.e. G-codes, M-codes and the format for NC programming). The input is the machines NAME, DATA FILE, and which should be read through channel No. 6 for further details, see Appendix (2).

9.6 COMPUTER AIDED MACHINING WITH 'IBCSURF'

Two different methods for the computation of cutter offset and two different approaches for interface checking have been used in the two routines developed to aid machining in this package. The foolproof interference checking method adopted in ANAOFF enables the entire surface to be examined for each cutter position. This leads to higher computation cost. To reduce costs in interference checking, only the planes within the radial distance of the cutter need be considered. Assuming that data points are organised in matrix form points in a column or row are examined. The check moves to the following column or row when the

edge of the component is reached. With a 'U' shaped part it is possible that some points may be within the radial distance of the cutter even though the edge is reached. Generally, ANAOFF can be used for small components with fewer points to check but care must be taken to achieve a reliable result.

The cutting strategy in the routine allows the user to move the cutter longitudinally, cross-sectionally or both for a better finish. Longitudinal cutting lines are advantageous for tabular shapes such as pump volutes as the direction of cutting would be along lines of low curvature which means that finer tolerance can be achieved with the same number of NC steps as cross-sectional cutter movement. Also small ridges which remain after several passes are aligned with the direction of the duct, so that where there is fluid flow the ridges cause little resistance for objects such as pipe fittings.

In NUMOFF, the routine based on a numerical method of calculating cutter offset, as the tool is offset along the normal at the centroid of the triangle joining three neighbouring points which specify the surface, the ridges between successive passes are left over the data points. With ANAOFF, the cutter is directly offset over the data point position so all points will accurately lie on the machined surface. It would be reasonable to thus assume that the analytical method used in ANAOFF produces a better tolerance but surface finish is always a compromise between computing and NC machine costs, and putting the work into hand for polishing.

9.6.1 Flow of Data in the System

Surface design routines were implemented on the VAX 11/750 super mini-computer at the Computer Centre of Aston University.

As the input data goes through the package the output data transfer to the NC machine using punched paper tape as shwon in Fig (9.2).

9.6.2 Practical Applications

A component chosen to be designed and machined using the system, was the mould for a plastic handle. A local industry provided the engineering drawing. The component contained third degree projections and cross-sections in a few places were defined using circular arcs. As usual for objects with a free form surface, large areas of the shape were undefined to be completed by the pattern maker .

The number of cross sections given were not quite adequate to define the component using IBSCURF. Free-hand sketches of cross-sections at intermediate points were produced using other information available.(see chapter 6) .

The mould was designed in two parts so split positions at cross-sections were determined and data points from cross-sectional curves extracted and input to INVERT. BSCURV B-spline curves were used from defining polygon vertices generated by INVERT . After necessary modifications, cross-sectional curves were designed as shown in Fig (8.15).

Following usual procedure for surface design, final polygon vertices were input to BCSURF, the results of which are shown in Fig (8.16) and Fig (8.17).

The analytical method of computing the normals to the surface was used and interference checking was carried out using four tools starting with a 1/2" ball-nosed milling cutter. The mould was produced in wood and soft material as shown in Plates (9.1) and Fig (9.2). As no roughing tape was produced, the method applied was to sink the surface incrementally in two.

The result produced cavity cross-sections which correspond to the profiles specified at other orientations longitudinally, interpolating cardinal splines determine

the shere of the profile. Although they pass through cross-sections control points, prediction of their behaviour at intermediate positions is a matter of experience. As with any interpolating scheme some oscillation is inevitable but was found to be minimal in most situations in this work. In using the package it becomes apparent that the desired effect can be achieved and undulations minimised by changing the position and number of the cross-sections.

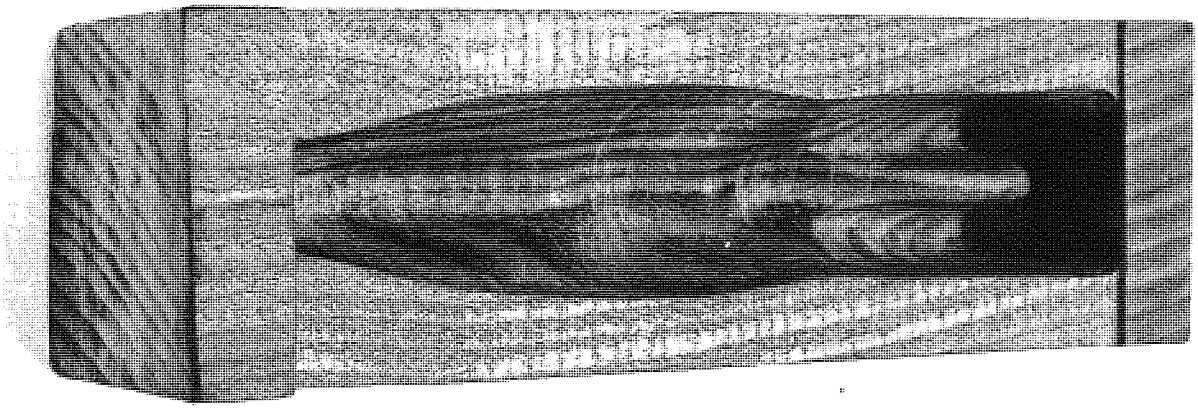


Plate 9.1 The Mould for Plastic Handle Machine

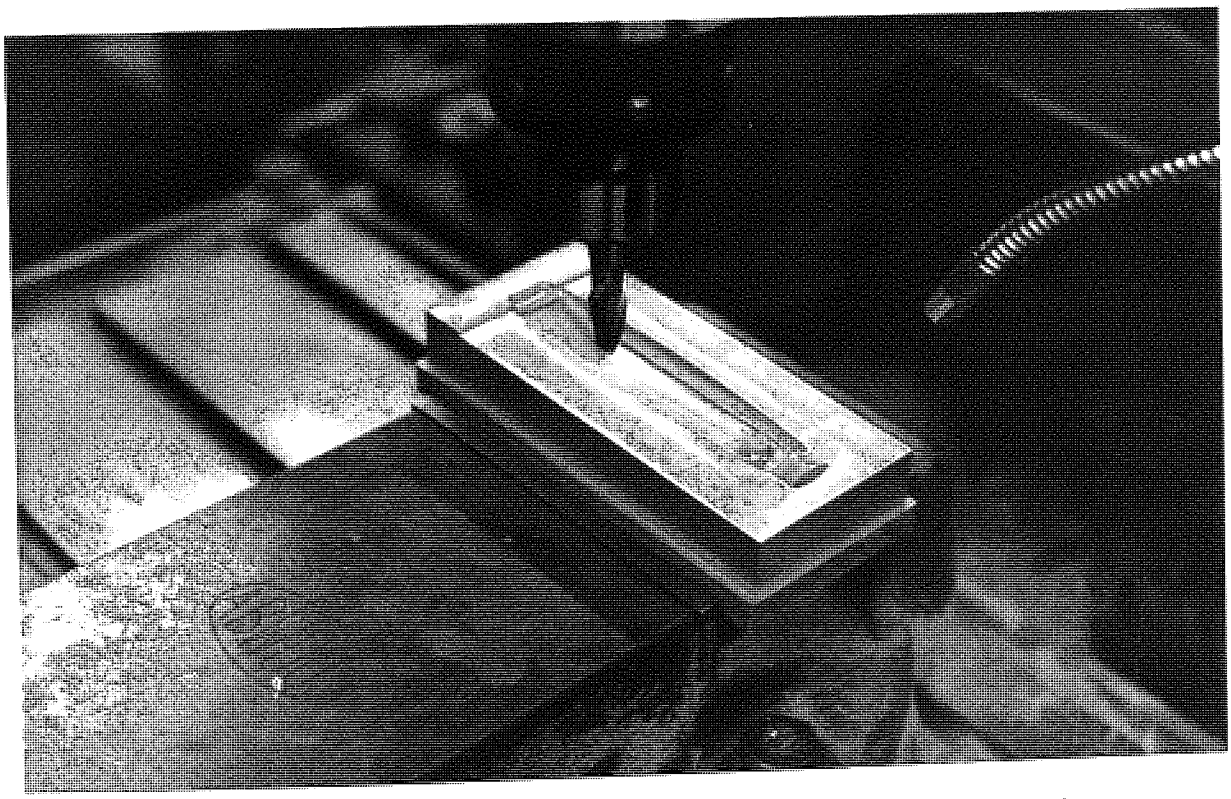


Plate 9.2 The Mould for Plastic Handle Machine

on the Bridgeport CNC Machine

```
N001 G54  X0000000   Y0000000                                    M03
N002 G55                                          R-0009000     M08
N003                                 Z0000000  F0050C
N004      X0000007   Y0000006        Z0000005  F00500
N005      X0000099   Y0000076        Z0000065
N006      X0000654   Y0000456        Z0000123
N007      X0006543   Y0005678        Z0009876
B009      X0986432   Y0575743        Z0763476
N010      X-0896542  Y-0549876       Z-0567654
N011      X-0039876  Y-0065678       Z-0098765
N012      X-0008765  Y-0008754       Z-0004567
N013      X-0000654  Y-0000045       Z-0000059
N014      X-0000056  Y-0000034       Z-0000076
N015      X-0000005  Y-0000007       Z-0000003
N016 G54                                          R0000000      M05
N017      X0000000   Y0000000                                    M30
N018 G55
```

(a) VARIABLE K = 0

```
N001 G54  X0000000   Y0000000                                    M03
N002 G55                                          R-0009000     M08
N003                                 Z0000000  F00500
N004      X0000007   Y0000006                    F00500
N005                                 Z0000005  F00500
N006      X0000099   Y0000076
N007                                 Z0000065
N008      X0000654   Y0000456
N009                                 Z0000123
N010      X0006543   Y0005678
N011                                 Z0009876
N012      X0070987   Y0038765
N013                                 Z0069876
N014      X096432    Y057543
N015                                 Z0763476
N016      X-0896542  Y-0549876
N017                                 Z-0567654
N018      X-0039876  Y-0065678
N019                                 Z-0098765
N020      X-0008765  Y-0008754
B021                                 Z-0004567
N022      X-0000654  Y-0000045
N023                                 Z-0000059
N024      X-0000056  Y-0000034
N025                                 Z-0000076
N026      X-0000005  Y-0000007
N027                                 Z-0000003
N028 G54                             R00000000                   M05
N029      X0000000   Y0000000                                    M30
N030 G55
```

(6) VARIABLE K = 1

N001	G54	X0000000	Y0000000				M03
N002	G55					R-0009000	M08
N003				Z0000000	F00500		
N004		X0000007	Y0000006		F00500		
N005		X0000099	Y0000076				
N006		X0000654	Y0000456				
N007		X0006543	Y0005678				
N008		X0070987	Y0038765				
N009		X0986432	Y0575743				
N010		X-0896542	Y-0549876				
N011		X-00398876	Y-0065678				
N012		X-0008765	Y-0008754				
N013		X-0000654	Y-0000045				
N014		X-0000056	Y-0000034				
N015		X-0000005	Y-0000007				
N016	G54					R0000000	M05
N017		X0000000	T0000000				
N018	G55						M30

(C) VARIABLE K = 2

N001	G54	X0000000	Y0000000				M03
N002	G55					R-0009000	M08
N003				Z0000000	F00500		
N004			Y0000006	Z0000005	F00500		
N005			Y0000076	Z0000065			
N006			Y0000456	Z0000123			
N007			Y0005678	Z0009876			
N008			Y0038765	Z0069876			
N009			Y057543	Z0763476			
N010			Y-0549876	Z-0567654			
N011			Y-0065678	Z-0098765			
N012			Y-0008754	Z-0004567			
N013			Y-0000045	Z-0000059			
N014			Y-0000034	Z-000076			
N015			Y-0000007	Z-0000003			
N016	G54					R0000000	M05
N017		X0000000	Y0000000				
N018	G55						M30

(d) VARIABLE K = 3

```
N001 G54  X0000000   Y0000000                                        M03
N002 G55                                                R-0009000   M08
N003                                 Z0000000  F00500
N004      X0000007                               F00500
N005                 Y0000076
N006      X0000654
N007                 Y0005678
N008      X0070987
N009                 Y0575743
N010      X-0896542
N011                 Y-0065678
N012      X-0008765
N013                 Y-0000045
N014      X-0000056
N015                 Y-0000007
N016 G54                                                R0000000    M05
N017      X0000000   Y0000000
N018      G55                                                       M30
```

(E) VARIABLE K = 4

```
N001 G54  X0000000   Y0000000                                        M03
N002 G55                                                R-0009000
N003                                 Z0000000  F00500
N004                 Y0000006                           F00500
N005      X0000099   N006                                Y0000456
N007      X0006543
N008                 Y0038765
N009      X0986432
N010                 Y-0549876
N011      X-0039876
N012                 Y-0008754
N013      X-0000654
N014                 Y-0000034
N015      X-0000005                                     R0000000    M05
N016 G54
N017      X0000000   Y0000000                                       M30
N018 G55
```

(F) VARIABLE K = 5

Fig 9.1 Outputed Formats from Post-processor Programme 'Prepare 3D'

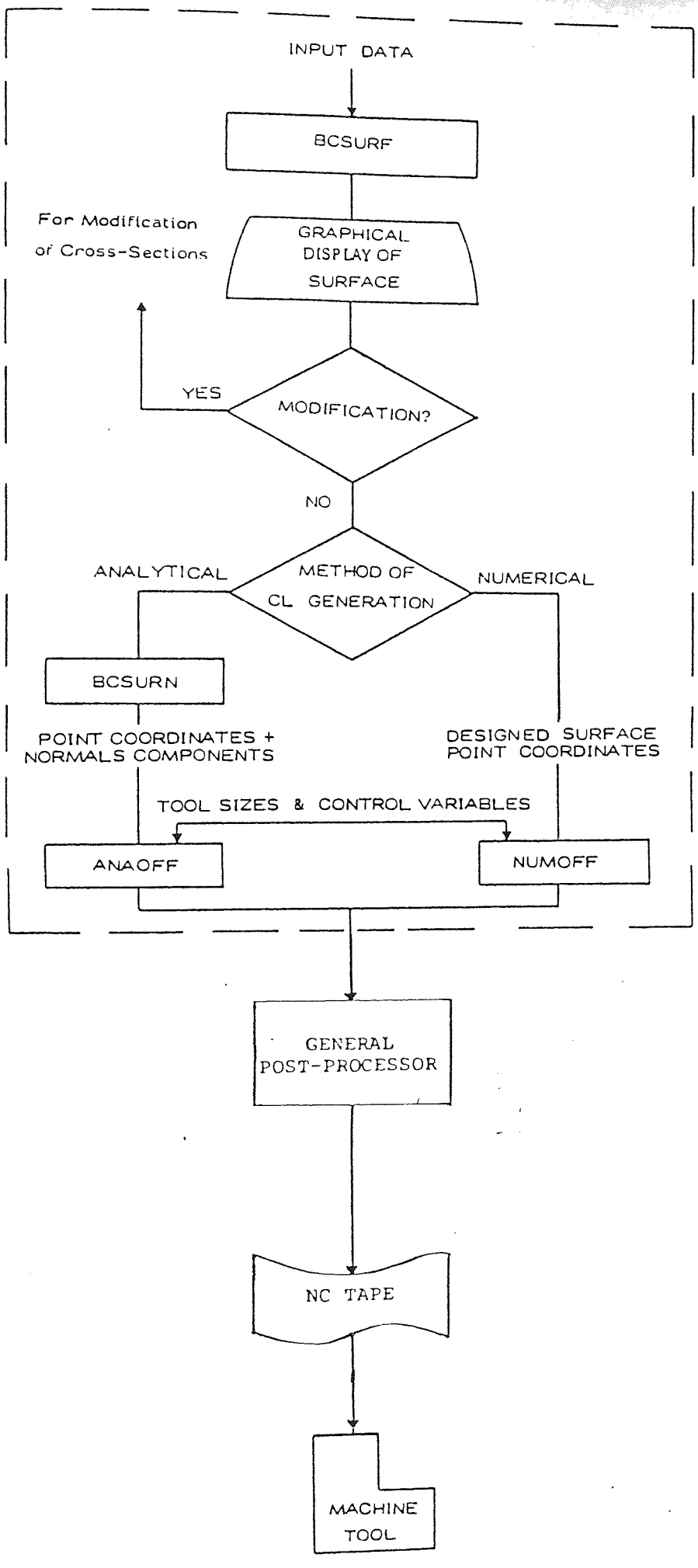


Fig. (9.2) Block Diagram of Data Flow Using 'BCSURF'

CHAPTER 10

CHAPTER 10

CONCLUSIONS AND FUTURE WORK

10.1 CONCLUSIONS AND REMARKS

A major area of interest in computer-aided geometrical design is in the representation (CAD) and manufacture (CAM) of the three-dimensional objects composed in part or in total of free-form curved surfaces.

The design and manufacture of free-form surfaces has always presented difficulties to the manufacturing engineer, who traditionally, has sought a solution to this problem by the construction and subsequent copy-milling of a manually made model. Advances in both computer technology and the necessary mathematical models capable of capturing the geometry of arbitrarily-shaped objects has allowed the development of surface generation systems aimed at providing a more economic solution to the problem of free-form surfce manufacture.

In general, this solution has taken the form of a two-tier system of surface design package (CAD) linked to the computer-aided manufacturing (CAM) capability of a numerically controlled (NC) machine tool via the medium of punched tape.

The research objectives outlined at the initial stage of this development programme have been achieved accordingly. All the computer programs have been successfully implemented on the VAX 11/750 computer system forming an interactive CAD/CAM system for designing and manufacturing free-form surfaces.

Based upon all aspects of the research programme, the following results and conclusions were obtained.

10.1.1 "IBSCURF" as a CAD/CAM Software Package

10.1.1.1 Design Philosophy

A major part of this work has been allocated to developing a prototype design and machining software package called 'IBSCURF', short for interactive B-spline cardinal spline surface. The method used is based on the so called computational geometry with particular reference to free-from surfaces. The main reason for developing computational geometry is largely due to the inadequacies of conventional techniques to handle doubly curved surfaces and the need to minimise labour costs taking advantage of modern technology. In choosing a model for computer aided geometric design, it is important to evaluate the utility of the power of a model and the control parameters. In addition, the model must be compatible with the equipment upon which it is to be implemented.

Lofting has long been established as a practical method in design, and therefore, it was only logical that the importance of this technique be considered in making a decision on adopting a design philosophy. Lofting has been used successfully in this work and proved not only able to produce a satisfactory result in the design of a wide range of shapes, but also presents no difficulty for shape representation using the existing two-dimensional cathode ray tubes, plotters etc.

10.1.1.2 Suitability of the Method for Design and Machining

The potentials of the B-spline basis in computer aided curve description could only be fully realised and appreciated when it is being applied in practice and primarily in the course of its development. In curve description, whether one is faced with the problem of design or repesentation, in other words, creating a new curve or approximating an existing one, B-spline basis provides simple alogorithms which are computationally very efficient and easy to implement. It also gives multiple control parameters providing many degrees of freedom.

The general form of the B-spline curve description procedure used in design and fitting routines of BCSURF demonstrated to be easily capable of fulfilling the requirements of surface design in a practical sense with the defining

polygon being the main control handle. Ideally it is better if the control points lie on the curve itself. This can, of course, be achieved by combining the inversion technique in the curve generating routine. In such a routine, however, any modification will require that the whole computation be repeated . The use of polygon vertices as a control parameter gave a good indication of the shape of the curve, and it can also be used effectively for subsequent modifications. Although the fourth order curve seemed best suited in most situations providing a compromise between smoothness and 'localness', varying curve orders have a role to play when being used as a handle in design.

The routine for curve design allows space curves to be designed when the appropriate control parameters are specified and three-dimensional data is input. Experience, however has shown that in the majority of cases a limit could be employed where only planar sectional curves are defined, as judging the shape of a doubly curved contour on the two-dimensional graphics devices is not very easy if actually possible at all.

It was found that the local property of B-spline for curve design is of great importance, because in using the lofting method, cross-sectional curves can have very complicated shapes and it is often requried to modify a small portion of these curves without altering the rest. Furthermore, designing such curves may need a polygon containing many sides.

Compared to the Bezier method, B-splines have the advantage that the degree of the curve is independent of the number of polygon vertices and therefore, difficulties involved with the Bezier curve do not occur here.

It was realised that in designing free-form surface, the constraints presented are not simply aesthetic and therefore, fitting procedures were investigated and two routines both based on the mathematics of B-splines were developed to accommodate these. In dealing with a large number of points, the least squares method can provide a more satisfactory solution but it might involve guessing the position of vertices, and if the problem is to approximate an existing curve, inversion technique with its more efficient performance could be employed.

The most attractive feature realised with 'IBCSURF' is the simplicity of approach to design and the case of determining the main input elements.

In addition, the mathematics of surface description used readily lends itself to providing a link between the design and manufacture thus simplifying production of designed objects on NC machine tools. Here, the analytical method of computing normals to the surface is more efficient than the numerical method, as both B-splines and cardinal spline bases allow the tangent vectors to be determined with very little extra computation in the surface design and routine. It might also be argued that by offsetting the tool directly above the points specifying the surface, the resulting machined surface will more closely approximate the designed shape than in the numerical method where the tool cuts between the points and leaves the ridges over the data points.

In 'IBSCURF' the object being defind in a bi-parametric space with B-spline sectional curve shaping of the surface in one direction and cubic cardinal splines in the other, the behaviour of interpolating cardinal splines is an important factor in the outcome of the design. The result using 'IBSCURF' has, however shown that cubic cardinal splines produce a smooth surface free from extraneous undulations, though the prediction of their behaviour betwen the control points is a matter for experience and a function of the position of sectional curves relative to each other.

Part of this work was allocated to developing the mathematical modelling of the plastic handle based on engineering drawings. These drawings were provided by HEALEY MOULDINGS LTD, a Birmingham based company. The firm's traditional method used to produce the moulds was time consuming an uneconomical. Engineering drawings were produced by the engineer for each design. These had to include all relevant information for the specialist to cut the mould. Designs were based on the engineers own ability to visualize the object and describe its aesthetic features on paper. The complexity of the objects shape sometimes led to unwanted shape compromise.

Obviously a new approach to the problem was needed. This mathematical model for the handle fulfilled the need specially of all the control parameters introduced in the mathematical modelling IREP2 proved to be the most important tool in generating the different shapes from the original model of the handle.

One intention of this research was to fully automate the manufacture of machined free-from surface by developing a post-processor for converting the cutter location data information into an NC tape program that suits the exact

requirements of the particular NC system on which the component is to be machined.

There are two types of post-processors, namely; the specific post-processor, which outputs the precise code for a particular machine tool, and the general post-processor which outputs generalised formats which need to be edited to satisfy the requirements of a particular machine tool. In this research, two different types of post-processor were used to generate NC tape for the Olivetti NC machine to machine the plastic handle.

In the course of the research, time was spent on searching for the universal programming language. As machine control manufacturers add more and more facilities to their systems, programming is forced back onto the shop floor and the problems of program transferability multiply. The need for a 'universal/ machine programming language seems to be increasingly desirable. Each day in our workshops new NC control systems are being introduced, generating new programming problems; new post processors are needed and new difficulties arise in transferring tapes from one machine to another.

The machine tool supplier wants to sell a complete machining package which means that MDI - Machine Data Input - is becoming more acceptable than the major users of NC machines would like. MDI is only suitable for the small user. Companies using many NC machines tools need to prepare post-program data off the shop floor away from the expensive machine tools.

It ought to be possible now to produce a high level NC programming language that is compatible with all machine tools (and other numerically controlled equipment) and their control systems. The trouble is that a language is not a commercial property and so perhaps the only body in the world that can do something about this problem is CAM-I. What CAM-I had to decide was whether it was best to allow individual interest groups to go their own way, or to set up a body to co-ordinate their language work.

Some companies and organisations are all for developing increased sophistication within the machine control. Others especially the users, want to see some major standardisation in programming systems. For example; the aim should be to turn machines into 'intelligent machines' so that the operator will be able to instruct a machine to do particular jobs accuractly. Why bother the programmer with details of part-geometry, tool changes and so on, when the

machine control can have the intelligence to do it. Instead of having the programmer give instructions "cut from A to B" we should be able to simply program "cut this part", thus relieving the programmer from routine details.

A company believes that the problems of programming for FMS, for example; can be simplified by building up such systems from modern stand-alone machine which are now being increasingly fitted with controls powerful enough to take care of pallet shuttle movements, tool monitoring and replacement part recognition and so on. These control functions could then easily be pre-programmed into machine controls by the supplier of a system to avoid unnecessary programming work by the user.

This approach, however, usually only works successfully where the machine tools, robots and so on are provided through one supplier who becomes responsible for the control systems. Where equipment is bought from different sources, the problem of linking up machines and programming then become very difficult.

An alternative approach is for machine and control system manufactures to agree on a standard type of programming language and standared of output format. One suggestion put forwared was to use APT-like languages outputting data in CLFile format (CL data). One result of the standards committee's work has been the development of what calls the 'node and path theory'.

"This came about from a need to approve a standard processor. But processors change and to standardise on one particular type seemed wrong. Thus, we were led to the node and path theory. Basically, it meas that you standardise on the interfaces (nodes) and let the processors (paths) go their own way. The major interface standard should be the CLFile (CL data). In this way all programming languages whether CAD, APT, Unigraphics, Sculptured surfaces, languages for robots, co-ordinate measuring machines or whatever, should all become APT-line and output data as standard CLFile.

In this work the opportunity was taken to assess the two methods of machining the mould of the plastic handle. The two methods are off-line and on-line input data.

The off-line input data uses the post processor to convert the cutter location data information into an NC tape program that suits the exact requirements of

the particular NC system on which the mould is to be machined. This was successfully machined on the Olivetti NC machine.

The second method, called on-line feeding data, was achieved using the Bridgeport CNC machine. This has an MDI (Manual Input Data) into which the cutter location data information is keyed into the computer which is connected to the controller directly.

With this necessarily limited study it would appear that for very complicated shapes, using 'IBSCURF' programme and off-line input that the Olivetti NC machine is more desirable.

For shapes of fairly conventional geometric definition the Bridgeport is easy to input the data using the 'IBSCURF' programme.

'IBSCURF' will enable the designer both at the design and the manufacturing stage to optimise, shape properties and the ease of manufacture. The use of this programme should enable faster production of the component and will better 'lead' time.

10.2 SUGGESTIONS FOR FUTURE WORK

1. At the moment an object designed using 'IBSCURF' can be machined by incrementally sinking the die in two parts. Using a roughing tape could reduce machining time and for the package to be really practical a routine for generation of roughing tape shold be included.

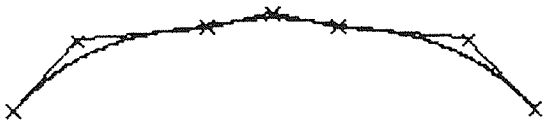
2. The conventional methods of design of free form surface cannot produce an exact description of the object because of the complexity of the shape and the presence of blended regions. It is only after the pattern has been made that the object can be said to be fully defined. Consequently, at the design stage only an approximate analysis of features such as stresses, volumes, weights areas, cross-sections etc. can be performed. Computer aided defined surfaces usually have the potential to faciiittate the analysis of the object at the design state programs could be developed in conjunction with 'IBSCURF to cater for such requirements. Routines could also be developed to aid determining the split lines when a three-dimensional object is to be cast in a two part mould.

3. The parametric method has many advantages; (as previously discussed) but presents some practical difficulties that limit its effectiveness in application. In order to overcome these problems, more investigations are needed to find a relationship between parametric values, physical values and the factors affecting this relationship. The investigation could be carried out for both-splines and cardinal splines. Finally, looking at curve fitting using the inversion technique there are many factors affecting the shape of the fitted curve. These are the position of data points, the number of vertices and the order of B-splines. Further study is needed in determining the role of these elemrnts and finding a relationship between them.

4. Concerning the N.C. part-programming for machining the component, other machining cycles, such as pack drill cycle, slot milling cycle and circular pocket cycle can be included in the software to cover all the jobs done on a CNC milling machine.

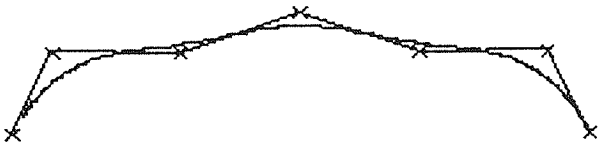
5. Elimination of papertape from the machining process and increasing the flexibility at the machine tool which could be achieved by linking the machine to the computer directly.

APPENDIX 1



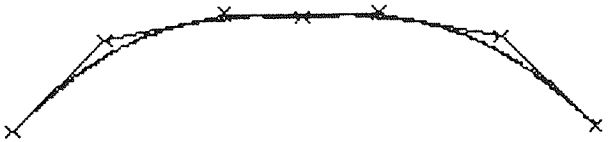
X-SCALING FACTOR = 3.0 Y-SCALING FACTOR = 3.0	NUMBER OF VERTICES = 7 THE ORDER OF THE CURVE= 4	MODEL NUMBER = 1 SECTION NUMBER =2

CROSS-SECTIONAL CURVE FOR UPPER PART OF THE MOULD.



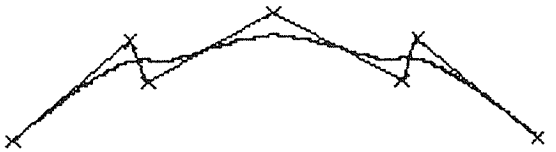
X-SCALING FACTOR = 3.0 Y-SCALING FACTOR = 3.0	NUMBER OF VERTICES = 7 THE ORDER OF THE CURVE= 4	MODEL NUMBER = 1 SECTION NUMBER =3

CROSS-SECTIONAL CURVE FOR UPPER PART OF THE MOULD.



X-SCALING FACTOR = 3.0 Y-SCALING FACTOR = 3.0	NUMBER OF VERTICES = 7 THE ORDER OF THE CURVE= 4	MODEL NUMBER = 1 SECTION NUMBER =4

CROSS-SECTIONAL CURVE FOR UPPER PART OF THE MOULD.



X-SCALING FACTOR = 3.0 Y-SCALING FACTOR = 3.0	NUMBER OF VERTICES = 7 THE ORDER OF THE CURVE= 4	MODEL NUMBER = 1 SECTION NUMBER =6

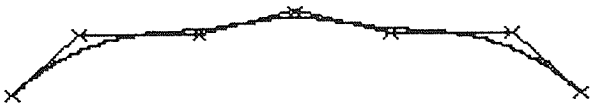
CROSS-SECTIONAL CURVE FOR UPPER PART OF THE MOULD.

X-SCALING FACTOR = 3.0 Y-SCALING FACTOR = 3.0	NUMBER OF VERTICES = 7 THE ORDER OF THE CURVE= 4	MODEL NUMBER = 1 SECTION NUMBER =7

CROSS-SECTIONAL CURVE FOR UPPER PART OF THE MOULD.

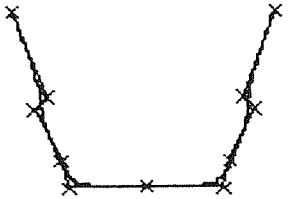
X-SCALING FACTOR = 3.0 Y-SCALING FACTOR = 3.0	NUMBER OF VERTICES = 7 THE ORDER OF THE CURVE= 4	MODEL NUMBER = 1 SECTION NUMBER =7

CROSS-SECTIONAL CURVE FOR UPPER PART OF THE MOULD.



X-SCALING FACTOR = 3.0 Y-SCALING FACTOR = 3.0	NUMBER OF VERTICES = 7 THE ORDER OF THE CURVE= 4	MODEL NUMBER = 1 SECTION NUMBER =8

CROSS-SECTIONAL CURVE FOR UPPER PART OF THE MOULD.



X-SCALING FACTOR = 3.0 Y-SCALING FACTOR = 3.0	NUMBER OF VERTICES = 7 THE ORDER OF THE CURVE= 4	MODEL NUMBER = 1 SECTION NUMBER =1

CROSS-SECTIONAL CURVE FOR LOWER PART OF THE MOULD.

X-SCALING FACTOR = 2.0 Y-SCALING FACTOR = 2.0	NUMBER OF VERTICES = 7 THE ORDER OF THE CURVE= 4	MODEL NUMBER = 1 SECTION NUMBER =2

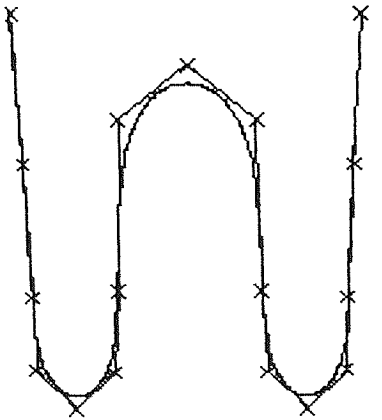
CROSS-SECTIONAL CURVE FOR LOWER PART OF THE MOULD.

X-SCALING FACTOR = 2.0 Y-SCALING FACTOR = 2.0	NUMBER OF VERTICES = 7 THE ORDER OF THE CURVE= 4	MODEL NUMBER = 1 SECTION NUMBER =4

CROSS-SECTIONAL CURVE FOR LOWER PART OF THE MOULD.

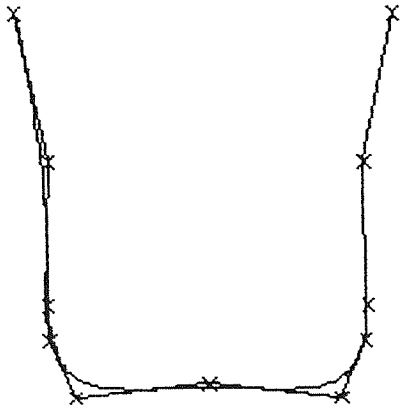
X-SCALING FACTOR = 2.0 Y-SCALING FACTOR = 2.0	NUMBER OF VERTICES = 7 THE ORDER OF THE CURVE= 4	MODEL NUMBER = 1 SECTION NUMBER =5

CROSS-SECTIONAL CURVE FOR LOWER PART OF THE MOULD.



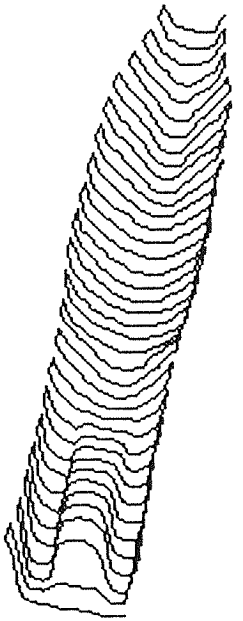
X-SCALING FACTOR = 2.0 Y-SCALING FACTOR = 2.0	NUMBER OF VERTICES = 7 THE ORDER OF THE CURVE= 4	MODEL NUMBER = 1 SECTION NUMBER =6

CROSS-SECTIONAL CURVE FOR LOWER PART OF THE MOULD.



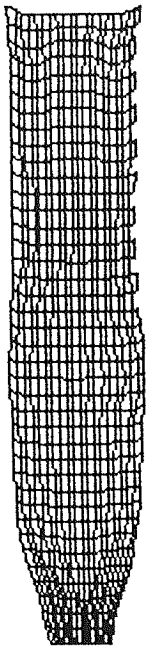
X-SCALING FACTOR = 2.0 Y-SCALING FACTOR = 2.0	NUMBER OF VERTICES = 7 THE ORDER OF THE CURVE= 4	MODEL NUMBER = 1 SECTION NUMBER =7

CROSS-SECTIONAL CURVE FOR LOWER PART OF THE MOULD.



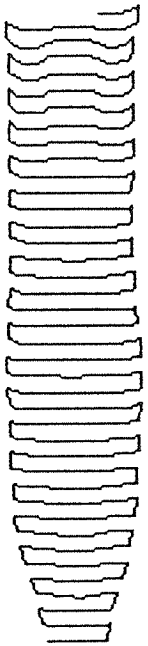
MODEL NUMBER = 1 VIEW NUMBER =1	LENGTH OF THE HANDLE =140 HIGHT OF THE HANDLE =30	SHIFTING IN X-DIR. = 250.00 SHIFTING IN Y-DIR. = 300.00
THE VIEWPOINT IN AXONOMETRIC PROJECTION IS: X-COORDINATE = 100.00 Y-COORDINATE = -1000.00 Z-COORDINATE = -500.00	X-SCALING FACTOR = 1.5 Y-SCALING FACTOR = 1.5 Z-SCALING FACTOR = 1.5	CONTROL VARIABLE =0 NO.OF POINTS ON PARAMETER U=20 NO.OF POINTS ON PARAMETER V=40

TOP&BOTTOM MOULD FOR THE PLASTIC HANDLE VIEWED FROM DIFFERENT ANGLES.



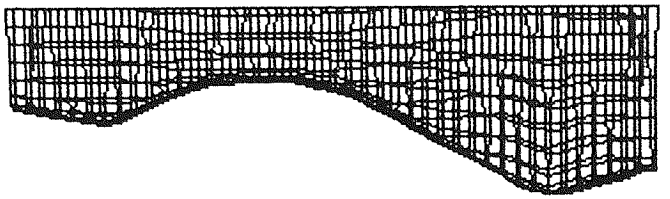
MODEL NUMBER = 1 VIEW NUMBER =2	LENGTH OF THE HANDLE =140 HIGHT OF THE HANDLE =30	SHIFTING IN X-DIR. = 250.00 SHIFTING IN Y-DIR. = 140.00
THE VIEWPOINT IN AXONOMETRIC PROJECTION IS: X-COORDINATE = 0.00 Y-COORDINATE = -1000.00 Z-COORDINATE = 60.00	X-SCALING FACTOR = 1.5 Y-SCALING FACTOR = 1.5 Z-SCALING FACTOR = 1.5	CONTROL VARIABLE =1 NO.OF POINTS ON PARAMETER U=20 NO.OF POINTS ON PARAMETER V=40

TOP&BOTTOM MOULD FOR THE PLASTIC HANDLE VIEWED FROM DIFFERENT ANGLES.

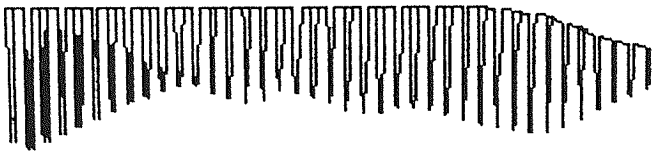


MODEL NUMBER = 1 VIEW NUMBER =2	LENGTH OF THE HANDLE =140 HIGHT OF THE HANDLE =30	SHIFTING IN X-DIR. = 250.00 SHIFTING IN Y-DIR. = 140.00
THE VIEWPOINT IN AXONOMETRIC PROJECTION IS: X-COORDINATE = 0.00 Y-COORDINATE = -1000.00 Z-COORDINATE = 60.00	X-SCALING FACTOR = 1.5 Y-SCALING FACTOR = 1.5 Z-SCALING FACTOR = 1.5	CONTROL VARIABLE =0 NO.OF POINTS ON PARAMETER U=20 NO.OF POINTS ON PARAMETER V=40

TOP&BOTTOM MOULD FOR THE PLASTIC HANDLE VIEWED FROM DIFFERENT ANGLES.

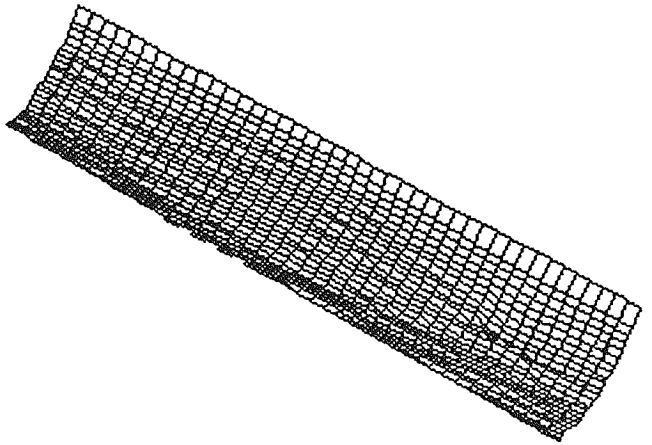


MODEL NUMBER = 1 VIEW NUMBER =3	LENGTH OF THE HANDLE =140 HIGHT OF THE HANDLE =30	SHIFTING IN X-DIR. = 350.00 SHIFTING IN Y-DIR. = 140.00
THE VIEWPOINT IN AXONOMETRIC PROJECTION IS: X-COORDINATE = 1000.00 Y-COORDINATE = 0.00 Z-COORDINATE = 60.00	X-SCALING FACTOR = 2.0 Y-SCALING FACTOR = 2.0 Z-SCALING FACTOR = 2.0	CONTROL VARIABLE =1 NO.OF POINTS ON PARAMETER U=25 NO.OF POINTS ON PARAMETER V=45
TOP&BOTTOM MOULD FOR THE PLASTIC HANDLE VIEWED FROM DIFFERENT ANGLES.		



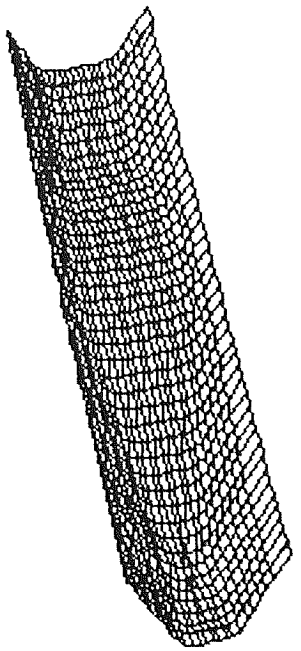
MODEL NUMBER = 1 VIEW NUMBER =3	LENGTH OF THE HANDLE =140 HIGHT OF THE HANDLE =30	SHIFTING IN X-DIR. = 350.00 SHIFTING IN Y-DIR. = 140.00
THE VIEWPOINT IN AXONOMETRIC PROJECTION IS: X-COORDINATE = 1000.00 Y-COORDINATE = 0.00 Z-COORDINATE = 60.00	X-SCALING FACTOR = 1.5 Y-SCALING FACTOR = 1.5 Z-SCALING FACTOR = 1.5	CONTROL VARIABLE =0 NO.OF POINTS ON PARAMETER U=20 NO.OF POINTS ON PARAMETER V=40

TOP&BOTTOM MOULD FOR THE PLASTIC HANDLE VIEWED FROM DIFFERENT ANGLES.



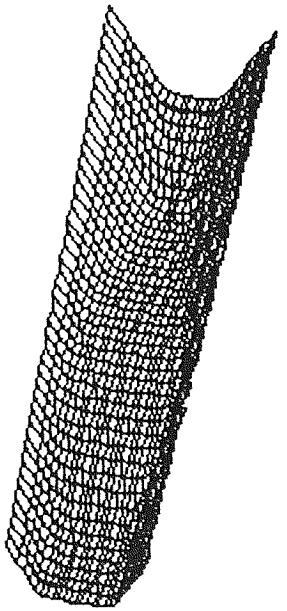
MODEL NUMBER = 1 VIEW NUMBER =3	LENGTH OF THE HANDLE =140 HIGHT OF THE HANDLE =30	SHIFTING IN X-DIR. = 350.00 SHIFTING IN Y-DIR. = 140.00
THE VIEWPOINT IN AXONOMETRIC PROJECTION IS: X-COORDINATE = 100.00 Y-COORDINATE = -1000.00 Z-COORDINATE = 60.00	X-SCALING FACTOR = 2.0 Y-SCALING FACTOR = 2.0 Z-SCALING FACTOR = 2.0	CONTROL VARIABLE =1 NO.OF POINTS ON PARAMETER U=25 NO.OF POINTS ON PARAMETER V=45

TOP&BOTTOM MOULD FOR THE PLASTIC HANDLE VIEWED FROM DIFFERENT ANGLES.



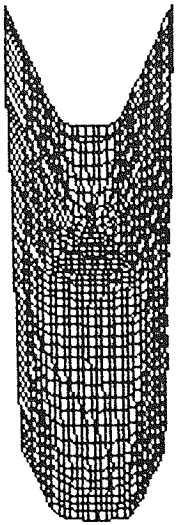
MODEL NUMBER = 1 VIEW NUMBER =4	LENGTH OF THE HANDLE =140 HIGHT OF THE HANDLE =30	SHIFTING IN X-DIR. = 250.00 SHIFTING IN Y-DIR. = 140.00
THE VIEWPOINT IN AXONOMETRIC PROJECTION IS: X-COORDINATE = 100.00 Y-COORDINATE = -1000.00 Z-COORDINATE = 500.00	X-SCALING FACTOR = 2.0 Y-SCALING FACTOR = 2.0 Z-SCALING FACTOR = 2.0	CONTROL VARIABLE =1 NO.OF POINTS ON PARAMETER U=25 NO.OF POINTS ON PARAMETER V=45

TOP&BOTTOM MOULD FOR THE PLASTIC HANDLE VIEWED FROM DIFFERENT ANGLES.



MODEL NUMBER = 1 VIEW NUMBER =5	LENGTH OF THE HANDLE =140 HIGHT OF THE HANDLE =30	SHIFTING IN X-DIR. = 250.00 SHIFTING IN Y-DIR. = 250.00
THE VIEWPOINT IN AXONOMETRIC PROJECTION IS: X-COORDINATE = 100.00 Y-COORDINATE = -1000.00 Z-COORDINATE = -500.00	X-SCALING FACTOR = 2.0 Y-SCALING FACTOR = 2.0 Z-SCALING FACTOR = 2.0	CONTROL VARIABLE =1 NO.OF POINTS ON PARAMETER U=25 NO.OF POINTS ON PARAMETER V=45

TOP&BOTTOM MOULD FOR THE PLASTIC HANDLE VIEWED FROM DIFFERENT ANGLES.



MODEL NUMBER = 1 VIEW NUMBER =6	LENGTH OF THE HANDLE =140 HIGHT OF THE HANDLE =30	SHIFTING IN X-DIR. = 250.00 SHIFTING IN Y-DIR. = 250.00
THE VIEWPOINT IN AXONOMETRIC PROJECTION IS: X-COORDINATE = 0.00 Y-COORDINATE = -500.00 Z-COORDINATE = -500.00	X-SCALING FACTOR = 2.0 Y-SCALING FACTOR = 2.0 Z-SCALING FACTOR = 2.0	CONTROL VARIABLE =1 NO.OF POINTS ON PARAMETER U=25 NO.OF POINTS ON PARAMETER V=45

TOP&BOTTOM MOULD FOR THE PLASTIC HANDLE VIEWED FROM DIFFERENT ANGLES.

APPENDIX 2

```
      PROGRAM BSLSQ
C  THIS PROGRAMME BY CALLING SUBROUTINE COEFF FINDS LEAST
SQUARES
C  COFFICIENTS TO FIT A B-SPLINE CURVE TO A SET OF GIVEN DATA,THEN
C  BY CALLING FUNCAL GENERATE THE B-SPLINE CURVE AT INTERVALS OF
'DELX'.
C  FINALLY IT PLOTS THE CURVE.
C  THE COMPUTED COEFFICIENTS ARE STORED SEPARATELY FOR FUTURE
USE.
C
C
C  M: NUMBER OF DATA POINTS
C  NCAP: NUMBER OF INTERVALS
C  K: THE ORDER OF THE CURVE
C  XSCL,YSCL: X&Y SCALING FACTORS IN CURVE PLOTTING
C  XKNOT: INTERIOR KNOTS
C  X,Y: COORDINATES OF THE DATA POINTS
C
C  TO FIND THE MINIMUM DIMENSION FOR THE ARRAYS REFER TO
C  SUBROUTINE COEFF.
C
C
      DIMENSION X(50),Y(50),XKNOT(50),XCUR(100),YCUR(100),
     &          F(50,50),FT(50,50),A(50,51),B(50),C(50),AN(10,10)
      OPEN(UNIT=1,NAME='DBSLSQ',STATUS='UNKNOWN')
      OPEN(UNIT=2,NAME='OBSLSQ2',STATUS='UNKNOWN')
      OPEN(UNIT=6,NAME='OBSLSQ6',STATUS='UNKNOWN')
      L=0
      CALL GINO
      CALL UNITS(10.0)
      CALL T4105
      CALL CHASIZ(0.3,0.3)
10    READ(1,*)M,NCAP
      IF(M.EQ.0)GO TO 60
      WRITE(2,998)M,NCAP
      READ(1,*)K,DELX
      WRITE(2,996)K,DELX
      WRITE(*,3131)
3131  FORMAT(3X,'ENTER(XSHIFT,SHIFT)')
      READ*,XSHIFT,SHIFT
      CALL SHIFT2(XSHIFT,SHIFT)
      CALL PICCLE
      WRITE(*,2121)
2121  FORMAT(3X,'ENTER XSCL,YSCL')
      READ*,XSCL,YSCL
      WRITE(2,981)XSCL,YSCL
      IF(NCAP.EQ.1)GO TO 11
      READ(1,*)XKNOT(K)
      DO 6 J=K+1,K+NCAP-2
      READ(1,*)XKNOT(J)
      IF(XKNOT(J-1).GT.XKNOT(J))GO TO 70
6     CONTINUE
      WRITE(2,994)
      WRITE(2,984)(J,XKNOT(J),J=K,K+NCAP-2)
11    READ(1,*)X(1),Y(1)
      DO 7 I=2,M
```

```
      READ(1,*)X(I),Y(I)
      IF(X(I-1).GT.X(I))GO TO 80
    7  CONTINUE
      WRITE(2,992)
      WRITE(2,982)(I,X(I),Y(I),I=1,M)
      WRITE(2,980)
C     DRAW THE AXES
      IF(L.NE.0)GO TO 12
   12  CALL SCALE2(XSCL,YSCL)
C     FIND YLOW,YHIGH
C
      YL=Y(1)
      YH=Y(1)
      DO 14 I=1,M
      IF(Y(I).GT.YH)YH=Y(I)
      IF(Y(I).LT.YL)YL=Y(I)
   14  CONTINUE
C     CALCULATE X&Y AXES LENGTH
C
      XLEN=X(M)-X(1)
      YLEN=YH-YL
      CALL AXIPOS(1,0.0,0.0,XLEN,1)
      CALL AXIPOS(1,0.0,0.0,YLEN,2)
      CALL AXISCA(3,10,X(1),X(M),1)
      CALL AXISCA(3,10,YL,YH,2)
      CALL AXIDRA(2,1,1)
      CALL AXIDRA(-2,-1,2)
      NCAP5=NCAP+2*K-3
      NCAP3=NCAP+K-1
      NCAP4=NCAP3+1
      CALL COEFF(X,Y,XKNOT,M,NCAP5,K,AN,F,FT,A,B,NCAP3,NCAP4,C)
      INTRVL=K-1
      XCUR1=XKNOT(INTRVL)-DELX
   20  I=0
   30  I=I+1
      IF(I.GT.100)GO TO 45
      XCUR(I)=XCUR1+DELX
      XCUR1=XCUR(I)
      IF(XCUR1.LT.(XKNOT(INTRVL+1)))GO TO 40
      INTRVL=INTRVL+1
   40  CALL FUNCAL(K,INTRVL,XKNOT,NCAP5,XCUR1,C,YCUR1,AN)
      YCUR(I)=YCUR1
      IF(XCUR1.EQ.X(M))GO TO 43
      IF(XCUR1.LT.(X(M)-DELX))GO TO 30
      I=I+1
      XCUR1=X(M)
      XCUR(I)=XCUR1
      GO TO 40
   43  N=I
      GO TO 50
   45  N=100
C     PRINT THE GENERATED POINTS AT THE INTERVAL OF DELX
C     AND SHIFT THE ORIGIN TO X(1),YL
C
   50  DO 55 II=1,N
      WRITE(2,982)II,XCUR(II),YCUR(II)
      XCUR(II)=XCUR(II)-X(1)
      YCUR(II)=YCUR(II)-YL
   55  CONTINUE
      DO 57 I=1,M
```

```
      X(I)=X(I)-X(1)
57    Y(I)=Y(I)-YL
      CALL LINCOL(2)
      CALL SYMTO2(X,Y,M,8)
      CALL MOVTO2(XCUR(1),YCUR(1))
      CALL LINCOL(5)
      CALL POLTO2(XCUR,YCUR,N)
      IF(XCUR1.NE.X(M))GO TO 20
      L=L+1
      CALL SCALE2(1/XSCL,1/YSCL)
      CALL SHIFT2(-XSHIFT,-SHIFT)
C SAVE THE RESULT FOR USE IN SUBSEQUENT STAGES
      NUM=NCAP+1
      WRITE(6,969)(C(J),J=1,NCAP3)
      GO TO 10
60    CALL DEVEND
      WRITE(6,968)
      STOP
70    WRITE(*,960)
      STOP
80    WRITE(*,950)
      STOP
998   FORMAT(1H1,//,5X,'LEAST SQUARES B-SPLINE FIT TO A GIVEN SET OF',
     &       /,4X,'DATA WITH INTERIOR KNOTS AT ARBITRARY INTERVALS',
     &       /,4X,'-----------------------------------------------',
     &       //' NUMBER OF DATA POINTS   M=',I3,/
     &        ' NUMBER OF INTERVALS     =',I3,/)
996   FORMAT(' THE ORDER OF THE CURVE  K=',I3,/
     &      ' SCANNING INTERVAL ALONG THE CURVE=',F7.3)
994   FORMAT(//'   J',6X,'INTERIOR KNOT',/)
984   FORMAT(1X,I3,6X,E12.5)
992   FORMAT(//'  X-Y COORDINATES OF THE DATA POINTS',
     &      //'   I',7X,'X-ORDINATE',8X,'Y-ORDINATE',/)
982   FORMAT(1X,I3,6X,E12.5,6X,E12.5)
981   FORMAT(/' X SCALING FACTOR IN CURVE PLOTTING  XSCL=',F7.3,
     &      /' Y SCALING FACTOR IN CURVE PLOTTING  YSCL=',F7.3)
980   FORMAT(//'  X-Y COORDINATES OF THE GENERATED POINTS',
     &      //'   I',7X,'X-ORDINATE',8X,'YORDINATE',/)
970   FORMAT(I3,1X,I3)
969   FORMAT(F8.3)
968   FORMAT(' 0 0')
960   FORMAT(' ERROR MESSAGE:----',/
     &' THE SUPPLIED INTERRIOR KNOTS ARE NOT IN ASCENDING ORDER')
950   FORMAT(' ERROR MESSAGE-----',/
     &' THE SUPPLIED POINT DATA ARE NOT IN ASCENDING ORDER')
      END


C SUBROUTINE TO CALCULATE COEFFICIENTS OF A LEAST SQUARE
C B-SPLINE FIT TO A SET OF POINTS WITH ARBITRARY INTERIOR KNOTS.
C
C X,Y: ARRAYS CONTAINING COORDINATES OF THE DATA POINTS
C XKNOT: ARRAY CONTAINING THE GIVEN KNOTS
C M: NUMBER OF DATA POINTS
C NCAP: NUMBER OF INTERVALS
C NCAP5: NCAP+2*K-3
C NCAP4: NCAP+K
C NCAP3: NCAP+K-1
C K: THE ORDER OF THE CURVE
C C :ARRAY CONTAINING THE COMPUTED COEFFICIENTS
```

```
C   AN,F,FT,A,B,C: WORKING SPACES


C
      SUBROUTINE
COEFF(X,Y,XKNOT,M,NCAP5,K,AN,F,FT,A,B,NCAP3,NCAP4,C)
      DIMENSION X(M),Y(M),XKNOT(NCAP5),C(NCAP5)
      REAL AN(K,K),F(M,NCAP3),FT(NCAP3,M),A(NCAP3,NCAP4),B(NCAP3)
      INTEGER R
      DO 100 J=1,K-1
100   XKNOT(J)=X(1)
      DO 110 J=(NCAP5-K+2),NCAP5
110   XKNOT(J)=X(M)
C  LEAST SQUARE FIT :
C  1-THE DIMENSION OF THE F MATRIX
      IF=M
      JF=NCAP3
C  2-GENERATE THE F MATRIX
      DO 120 I=1,M
      DO 120 J=1,NCAP3
120   F(I,J)=0.0
      IFRST=1
      ILAST=K
      R=0
      INTRVL=K-1
130   R=R+1
      IF(R.EQ.(M+1))GO TO 151
      IF(X(R).LT.XKNOT(INTRVL+1))GO TO 140
      IF(X(R).EQ.XKNOT(NCAP5-K+2))GO TO 140
C  CHECK FOR COINCIDENT KNOTS
      IF(XKNOT(INTRVL).NE.XKNOT(INTRVL+1))GO TO 131
      IF(XKNOT(INTRVL+1).EQ.XKNOT(INTRVL+2))INTRVL=INTRVL+1
131   INTRVL=INTRVL+1
      IFRST=IFRST+1
      ILAST=ILAST+1
140   CALL BASCAL(K,INTRVL,XKNOT,NCAP5,X(R),AN)
      L=0
      DO 150 J=IFRST,ILAST
      L=L+1
150   F(R,J)=AN(L,K)
      GO TO 130
C  3-GENERATE TRANSPOSE OF THE F MATRIX
151   DO 160 I=1,IF
      DO 160 J=1,JF
160   FT(J,I)=F(I,J)
C  4-DETERMINE THE COEFFICIENT MATRIX A OF SIMULTANEOUS
EQUATION SYSTEM
      CALL MATMPY(FT,F,A,JF,IF,JF)
C  5-DETERMINE THE COLUMN OF CONSTANTS FOR SIMULTANEOUS
EQUATION SYSTEM
      CALL MATMPY(FT,Y,B,JF,IF,1)
      DO 180 I=1,JF
180   A(I,JF+1)=B(I)
C  6-DETERMINE COEFFICIENT VALUES BY SOLVING SIMULTANEOUS
EQUATIONS
C    USING CHOLESKY METHOD
      JFP1=JF+1
      CALL CHLSKY(A,JF,JFP1,C)
      RETURN
      END
```

```
C SUBROUTINE TO CALCULATE K ORDER SPLINE FIT TO A SET OF
INTERVALS
C  WITH SUPPLIED KNOTS AND COEFFICIENTS
      SUBROUTINE FUNCAL(K,INTRVL,XKNOT,NCAP5,X1,C,FIT,AN)

      DIMENSION XKNOT(NCAP5),C(NCAP5),AN(K,K)
      CALL BASCAL(K,INTRVL,XKNOT,NCAP5,X1,AN)
      FIT=0
      DO 10 L=1,K
10    FIT=FIT+C(INTRVL-K+1+L)*AN(L,K)
      RETURN
      END


C SBROUTINE TO CALCULATE B-SPLINE BASIS AT A POINT X1 ON A SET OF
KNOTS
      SUBROUTINE BASCAL(K,INTRVL,XKNOT,NCAP5,X1,AN)
      DIMENSION DP(20),DM(20),AN(K,K),XKNOT(NCAP5)
      INTEGER R,S
      AN(1,1)=1.0
      DO 10 S=1,K-1
      DP(S)=XKNOT(INTRVL+S)-X1
      DM(S)=X1-XKNOT(INTRVL+1-S)
      AN(1,S+1)=0.0
      DO 11 R=1,S
      AM=AN(R,S)/(DP(R)+DM(S+1-R))
      AN(R,S+1)=AN(R,S+1)+DP(R)*AM
      AN(R+1,S+1)=DM(S+1-R)*AM
11    CONTINUE
10    CONTINUE
      RETURN
      END


C SUBROUTINE TO DETERMINE MATRIX C AS PRODUCT OF A & B MATRICES
      SUBROUTINE MATMPY(A,B,C,M,N,L)
      DIMENSION A(M,N),B(N,L),C(M,L)
      DO 10 I=1,M
      DO 10 J=1,L
      C(I,J)=0
      DO 10 K=1,N
10    C(I,J)=C(I,J)+A(I,K)*B(K,J)
      RETURN
      END


C SUBROUTINE TO SOLVE SIMULTANEOUS EQUATION SYSTEM USING
CHPLESKY METHOD
      SUBROUTINE CHLSKY(A,N,M,X)
      DIMENSION A(N,M),X(N)
C  CALCULATE FIRST ROW OF UPPER UNIT TRIANGULAR MATRIX
      DO 3 J=2,M
3     A(I,J)=A(I,J)/A(1,1)
C  CALCULATE OTHER ELEMENTS OF U AND L MATRICES
      DO 8 I=2,N
      J=I
      DO 5 II=J,N
      SUM=0.0
```

```
      JM1=J-1
      DO 4 K=1,JM1
    4  SUM=SUM+A(II,K)*A(K,J)
    5  A(II,J)=A(II,J)-SUM
      IP1=I+1
      DO 7 JJ=IP1,M
      SUM=0.0
      IM1=I-1
      DO 6 K=1,IM1
    6  SUM=SUM+A(I,K)*A(K,JJ)
    7  A(I,JJ)=(A(I,JJ)-SUM)/A(I,I)
    8  CONTINUE
C  SOLVE FOR X(I) BY BACK SUBSTITUTION
      X(N)=A(N,N+1)
      L=N-1
      DO 10 NN=1,L
      SUM=0.0
      I=N-NN
      IP1=I+1
      DO 9 J=IP1,N
    9  SUM=SUM+A(I,J)*X(J)
   10  X(I)=A(I,M)-SUM
      RETURN
      END
```

PROGRAM S_2

```
C  THIS PROGROMME BY CALLING SUBROUTINE 'INVERT' PRODUCES
VERTICES
C  OF A POLYGON,B-SPLINE APPROXIMATION OF WHICH FITS A GIVEN SET
OF
C  DATA POINTS
C
C
     OPEN(UNIT=1,NAME='RE30',STATUS='UNKNOWN')
     OPEN(UNIT=2,NAME='OINVERT',STATUS='UNKNOWN')
     DIMENSION XP(90),YP(90),ZP(90),X(90),
    &        XV(90),YV(90),ZV(90),E(90)
     REAL N(90,10),NN(90,90)
C

C  DIMENSION FOR  XP,YP,ZP,
C              XV,YV,ZV,
C              X,E,N,NN : NUM+2*M
C  READ THE INPUT DATA
C  REFER TO SUBROUTINE BSINVERT FOR FURTHER INFORMATION
C  ABOUT THE VARIABLES USED.
500   READ(1,*)NUM,M
      IF(NUM.EQ.0)GO TO 600
      WRITE(2,998)
      WRITE(2,997)NUM,M
      READ(1,*)ID,ICLOS
      IF(ICLOS.EQ.1)GO TO 510
      WRITE(2,996)
      GO TO 520
510   WRITE(2,995)
520   IF(ID.EQ.2)GO TO 530
      WRITE(2,994)
      GO TO 540
530   WRITE(2,993)
540   WRITE(2,992)
      IF(ID.EQ.3)GO TO 550
      READ(1,*)(XP(I),YP(I),I=1,NUM)
      WRITE(2,990)(I,XP(I),YP(I),I=1,NUM)
      GO TO 560
550   READ(1,*)(XP(I),YP(I),ZP(I),I=1,NUM)
      WRITE(2,988)(I,XP(I),YP(I),ZP(I),I=1,NUM)
560   NUM2M=NUM+2*M
      IFAULT=0
C  CALL SUBROUTINE BSINVERT TO PRODUCE THE VERTICES
C
      CALL BSINVERT(M,NUM,XP,YP,ZP,ID,NUM2M,ICLOS,
     &          XV,YV,ZV,IFAULT,N,X,E,NN)
      IF (IFAULT.EQ.1)GO TO 590
      WRITE(2,987)
      IF(ID.EQ.2)GO TO 570
      WRITE(2,988)(I,XV(I),YV(I),ZV(I),I=1,NUM)
      WRITE(6,983)NUM,M,ID,ICLOS
      WRITE(6,985)(XV(I),YV(I),ZV(I),I=1,NUM)
      GO TO 580
570   WRITE(2,990)(I,XV(I),YV(I),I=1,NUM)
      WRITE(6,983)NUM,M,ID,ICLOS
```

```
      WRITE(6,984)(XV(I),YV(I),I=1,NUM)
  580 GO TO 500
  590 WRITE(2,986)
      STOP
  600 WRITE(6,982)
      STOP

  998 FORMAT(1H1,5X,'B-SPLINE INVERTION METHOD TO PRODUCE THE
     VERTICES',
     & /1X,' OF A POLYGON TO APPROXIMATE A SET OF GIVEN DATA
     POINTS',
     & /1X,'
     ==========================================================')
  997 FORMAT(/' NUMBER OF POINS NUM=',I3,
     &      /' THE ORDER OF THE CURVE M=',I3)
  996 FORMAT(/' CLOSED OR OPEN CURVE? OPEN')
  995 FORMAT(/' CLOSED OR OPEN CURVE? CLOSED')
  994 FORMAT(' PLANE OR SPACE CURVE? SPACE')
  993 FORMAT(' PLANE OR SPACE CURVE? PLANE')
  992 FORMAT(//' X-Y-Z COORDINATES OF THE DATA POINTS',
     &      //' I',6X,'X-ORDINATE',6X,'Y-ORDINATE',6X,'Z-ORDINATE')
  990 FORMAT(1X,I3,7X,F8.4,8X,F8.4)
  988 FORMAT(1X,I3,7X,F8.4,8X,F8.4,8X,F8.4)
  987 FORMAT(///' X-Y-Z COORDINATES OF THE VERTICES',
     &      //' I',6X,'X-ORDINATE',6X,'Y-ORDINATE',6X,'Z-ORDINATE')
  986 FORMAT(//' ERROR RETURN M>NUM')
  985 FORMAT(1X,F8.4,1X,F8.4,1X,F8.4)
  984 FORMAT(1X,F8.4,1X,F8.4)
  983 FORMAT(1X,I3,1X,I3,/1X,I3,1X,I3)
  982 FORMAT(' 0 0')
      END
C
C
C  THIS SUBROUTINE BASED ON B-SPLINE INVERSION METHOD,FINDS
VERTICES
C  OF A POLYGON,THE B-SPLINE APPROXIMATION OF WHICH FITS A SET
C  OF GIVEN DATA POINTS.
C
C
      SUBROUTINE BSINVERT(M,NUM,XP,YP,ZP,ID,NUM2M,ICLOS,
     &            XV,YV,ZV,IFAULT,N,X,E,NN)
C
C
C  M: THE ORDER OF THE CURVE
C  NUM: NUMBER OF POINTS
C  XP,YP,ZP: COORDINATES OF THE INPUT POINTS
C  XV,YV,ZV: COORDINATES OF THE VERTICES
C  ID: CONTROL VARIABLE ;ID=2 PLANE,ID=3 SPACE CURVE
C  ICLOS: CONTROL VARIABLE ;ICLOS=1 CLOSED,ICLOS=0 OPEN CURVE
C  IFAULT: ERROR RETURN M>NUM
C
C
      DIMENSION XP(NUM2M),YP(NUM2M),ZP(NUM2M),X(NUM2M),
     &      XV(NUM2M),YV(NUM2M),ZV(NUM2M),E(NUM2M)
      REAL N(NUM2M,M),NN(NUM2M,NUM2M)
      INTEGER W
      NUM1=NUM
      DO 90 I=1,NUM2M
```

```
      DO 90 K=1,M
      N(I,K)=0.0
   90 NN(I,K)=0.0
      IF(ICLOS.NE.1)GO TO 110
      DO 100 I=1,M-1
      XP(NUM+I)=XP(I)
      YP(NUM+I)=YP(I)
  100 ZP(NUM+I)=ZP(I)
      NUM=NUM+M-1
  110 IF(NUM.LT.M)GO TO 250
      MAX=NUM-M+1
      IX=NUM+M
      IF(ICLOS.EQ.1)GO TO 120
      CALL KNOTOP(M,MAX,XP,YP,ZP,NUM,ID,X,IX)
      GO TO 130
  120 CALL KNOTCL(M,MAX,XP,YP,ZP,NUM,ID,X,IX)
C     CALCULATE THE VALUE OF THE NODES
C
  130 DO 205 I=1,NUM
      E(I)=0.0
      DO 140 I1=I+1,M+I-1
  140 E(I)=E(I)+X(I1)
      E(I)=E(I)/FLOAT(M-1)
C     CALCULATE B-SPLINE BASIS FUNCTIONS
C
      W=0
      IF(ABS(E(I)-X(NUM+1)).LT.1E-6)GO TO 154
      IF(E(I).GT.X(NUM+1))GO TO 206
  150 W=W+1
      IF(E(I).GE.X(W).AND.E(I).LT.X(W+1))GO TO 155
      GO TO 150
  154 W=NUM
  155 N(W,1)=1.0
      DO 190 K=2,M
      DO 190 I1=1,NUM
      IF(N(I1,K-1).NE.0.0)GO TO 160
      D=0.0
      GO TO 170
  160 D=(E(I)-X(I1))*N(I1,K-1)/(X(I1+K-1)-X(I1))
  170 IF(N(I1+1,K-1).NE.0.0)GO TO 180
      F=0.0
      GO TO 190
  180 F=(X(I1+K)-E(I))*N(I1+1,K-1)/(X(I1+K)-X(I1+1))
  190 N(I1,K)=D+F
C
C
      DO 200 J=1,NUM
  200 NN(I,J)=N(J,M)
      N(W,1)=0.0
  205 CONTINUE
  206 IF(ICLOS.NE.1)GO TO 308
      I2=(M-1)/2
      L=0
      I=0
  207 I=I+1
      IF(E(I).GE.0.0)GO TO 209
      L=L+1
      DO 208 K=NUM1-I2+L,NUM1
```

```
208  NN(L,K)=NN(NUM1+L,K)
     GO TO 207
209  DO 307 I1=1,I2
     I=NUM-I1+1
     DO 307 J1=1,NUM
     J=NUM-J1+1
307  NN(I,J)=NN(I1,J1)
C    INVERT MATRIX NN
C
C    MATRIX INVERSION USING GAUSS-JORDAN REDUCTION
C    PARTIAL PIVOTING IS NOT USED.
C    INVERTED MATRIX OVERLAY ORIGINAL MATRIX IN CORE
C    CALCULATE ELEMENTS OF REDUCED MATRIX
C
308  DO 230 K=1,NUM
C    CALCULATE NEW ELEMENTS OF PIVOT ROW
C
     DO 210 J=1,NUM
     IF(J.EQ.K)GO TO 210
     NN(K,J)=NN(K,J)/NN(K,K)
210  CONTINUE
C    CALCULATE ELEMENT REPLACING PIVOT ELEMENT
     NN(K,K)=1.0/NN(K,K)
C    CALCULATE ELEMENTS NOT IN PIVOT ROW OR PIVOT COLUMN
     DO 220 I=1,NUM
     IF(I.EQ.K)GO TO 220
     DO 220 J=1,NUM
     IF(J.EQ.K)GO TO 220
     NN(I,J)=NN(I,J)-NN(K,J)*NN(I,K)
220  CONTINUE
C    CALCULATE REPLACING ELEMENTS FOR PIVOT COLUMN
C    EXCEPT PIVOT ELEMENT
C
     DO 230 I=1,NUM
     IF(I.EQ.K)GO TO 230
     NN(I,K)=-NN(I,K)*NN(K,K)
230  CONTINUE
C    CALCULATE VERTICES OF THE POLYGON
C
     DO 240 I=1,NUM
     XV(I)=0.0
     YV(I)=0.0
     ZV(I)=0.0
     DO 240 J=1,NUM
     XV(I)=XV(I)+NN(I,J)*XP(J)
     YV(I)=YV(I)+NN(I,J)*YP(J)
     IF(ID.EQ.2)GO TO 240
     ZV(I)=ZV(I)+NN(I,J)*ZP(J)
240  CONTINUE
     NUM=NUM1
     RETURN
250  IFAULT=1
     RETURN
     END
C    SUBROUTINE TO GENERATE BASIS KNOT VECTOR FOR
C    OPEN POLYGON
C
     SUBROUTINE KNOTOP(M,MAX,XV,YV,ZV,NUM,ID,X,IX)
```

```
      DIMENSION XV(NUM),YV(NUM),ZV(NUM),X(IX)
      DO 50 I=1,IX
      IF(I.GT.M)GO TO 10
C     ASSIGN MULTIPLE KNOTS FOR THE BEGINNING OF SPAN
      X(I)=0.0
      GO TO 50
10    IF(I.LT.(MAX+M+1)) GO TO 30
C     ASSIGN MULTIPLE KNOTS
C
20    X(I)=X(I-1)
      GO TO 50
30    IF(ID.EQ.3)GO TO 40
C     CHECK FOR MULTIPLE KNOTS
C
      IF(XV(I-M).EQ.XV(I-M+1).AND.
     &   YV(I-M).EQ.YV(I-M+1))GO TO 20
      X(I)=X(I-1)+1.
      GO TO 50
40    IF(XV(I-M).EQ.XV(I-M+1).AND.
     &   YV(I-M).EQ.YV(I-M+1).AND.
     &   ZV(I-M).EQ.ZV(I-M+1))GO TO 20
      X(I)=X(I-1)+1.
50    CONTINUE
      RETURN
      END
C
C     SUBROUTINE TO GENERATE BASIS KNOT VECTORS FOR
C     CLOSED POLYGON
C
      SUBROUTINE KNOTCL(M,MAX,XV,YV,ZV,NUM,ID,X,IX)
      DIMENSION XV(NUM),YV(NUM),ZV(NUM),X(IX)
C     FIRST KNOT VECTOR
C
      X(1)=FLOAT(1-M)
      DO 120 I=2,IX
      IF(I.GT.M)GO TO 40
      IF(ID.EQ.3)GO TO 20
C     CHECK FOR REPEATING VERTICES
C
      IF(XV(I+MAX-M).EQ.XV(I+MAX-M+1).AND.
     &   YV(I+MAX-M).EQ.YV(I+MAX-M+1))GO TO 10
      X(I)=X(I-1)+1.
      GO TO 120
10    X(I)=X(I-1)
      GO TO 120
20    IF(XV(I+MAX-M).EQ.XV(I+MAX-M+1).AND.
     &   YV(I+MAX-M).EQ.YV(I+MAX-M+1).AND.
     &   ZV(I+MAX-M).EQ.ZV(I+MAX-M+1)) GO TO 30
      X(I)=X(I-1)+1.
      GO TO 120
30    X(I)=X(I-1)
      GO TO 120
40    IF(I.LT.(MAX+M+1))GO TO 80
      IF(ID.EQ.3)GO TO 60
      IF(XV(I-MAX-M).EQ.XV(I-MAX-M+1).AND.
     &   YV(I-MAX-M).EQ.YV(I-MAX-M+1))GO TO 50
      X(I)=X(I-1)+1.
      GO TO 120
```

```
50   X(I)=X(I-1)
     GO TO 120
60   IF(XV(I-MAX-M).EQ.XV(I-MAX-M+1).AND.
    &   YV(I-MAX-M).EQ.YV(I-MAX-M+1).AND.
    &   ZV(I-MAX-M).EQ.ZV(I-MAX-M+1))GO TO 70
     X(I)=X(I-1)+1.
     GO TO 120
70   X(I)=X(I-1)
     GO TO 120
80   IF(ID.EQ.3)GO TO 100
     IF(XV(I-M).EQ.XV(I-M+1).AND.
    &   YV(I-M).EQ.YV(I-M+1))GO TO 90
     X(I)=X(I-1)+1.
     GO TO 120
90   X(I)=X(I-1)
     GO TO 120
100  IF(XV(I-M).EQ.XV(I-M+1).AND.
    &   YV(I-M).EQ.YV(I-M+1).AND.
    &   ZV(I-M).EQ.ZV(I-M+1))GO TO 110
     X(I)=X(I-1)+1.
     GO TO 120
110  X(I)=X(I-1)
120  CONTINUE
     RETURN
     END
```

PROGRAM S_3

```
C  THIS PROGRAM CALCULATES 21 DATA POINTS ON EVERY CROSS
C  SECTION AND THE OUTPUT IS IN THE FORM REQUIRED BY THE
C  PROGRAM INVERT.THE COORDINATES OF THE CENTRES OF ALL
C  CIRCLES ARE CALCULATED AND STORED.THEY CAN BE
C  CALLED FOR FUTURE WORK BY INCLUDING IN THE
C  PROGRAM "WRITE AND FORMAT"  STATEMENTS,WHICH
C  ARE NOT INCLUDED.
C
C   THE PARAMETERS USED ARE :
C  N = NUMBER OF CROSS SECTIONS REQUIRED (MAX.100).
C  IREP2 = NUMBER OF TIMES SECTION-2 HAS TO BE
C        REPEATED (MAX.4).
C  FI,THETA,BETA,ALFA = ANGLES
C  ALFAX = ANGLE IN CROSS-SECTIONS.
C  ZH1 = Z-COORDINATE OF LINE L2,FROM DATUM O.
C  Y(1) = THICKNESS OF HANDLE AT THE  REAR END
C       ( Y-COORDINATE).
C  R1,R4,R5 = RADII OF SMALL CIRCLES.
C  RCS1,RCS2 = RADII IN CROSS SECTIONS.
C  XL = LENGTH AT REAR END OF HANDLE (X-COORDINATE)
C  XS =  "" "" FROND "" "" "" ""
C  ZT = LENGTH OF THE HANDLE(Z-COORDINATE).
C  M  = CONTROL VARIABLE FOR THE ORDER OF THE CURVE
C  ID =  ""  "" .(SEE REFERENCE 13)
C  ICLOS =  "" ""  ""
C  RA(I) = RADII OF LARGE CIRCLES.
C  RB(I) =  "" "" SMALL  ""  "" "" ""
C  DU(I) = PARAMETERS FOR POSITIONING THE CIRCLES.
C  EU(I) =  ""  ""  ""  ""  "".
C
C  FOR MORE INFORMATIONS ON THE PARAMETERS,REFER
C  TO FIGURES IN CHAPTER FIVE.
C
      DIMENSION X(100),Y(100),Z(100),ZHA(5),ZPCA(100),
     &      ZHB(5),ZPCB(100),RA(5),RB(5),XA(100,21),
     &      YA(100,21),BS(100),FS(100),YPCA(100),
     &      XQ(100),YQ(100),YPCB(100),DM(100),
     &      PM(100),HM(100),
     &      DU(5),EU(5),PF(100),PE(100),
     &      ZRA(5),YRA(5),ZPC(5),YPC(5),PET(5),
     &      YC(5),ZCM(5),YCM(5),YB(5),ZB(5),
     &      WAD(100),ZCAL(5),ZCBL(5),YCAL(5),YCBL(5),
     &      XQA(100),XQB(100),YQA(100),YQB(100),
     &      ZTR(5),YTR(5),BE(5),BF(5),ZCA(5),EB(5),ZC(5),
     &      FP(5),WA(100),WB(100),WC(100),WD(100),WAA(100),
     &      YCA(5),FB(5),ZRT(5),YRT(5),WAB(100),WAC(100),
     &      BA(100),BB(100),BC(100),BD(100)
C
C
C
      OPEN(UNIT=1,NAME='SK3',STATUS='UNKNOWN')
      OPEN(UNIT=6,NAME='RE30',STATUS='UNKNOWN')
      OPEN(UNIT=2,NAME='RE31',STATUS='UNKNOWN')
      READ(1,*) N,IREP2
      READ(1,*) FI,THETA,BETA,ALPHA
```

```
      READ(1,*) ZH1,Y(1),YHAP
      READ(1,*) R1,R4,R5
      READ(1,*) XL,XS,ZT
      READ(1,*) RCS1,RCS2,ALFAX
      READ(1,*) M,ID,ICLOS
      IF(IREP2.EQ.0)  GO TO 98
      READ(1,*) (RA(I),I=1,IREP2)
      READ(1,*) (DU(I),I=1,IREP2)
      IF(IREP2.EQ.1)  GO TO 98
      READ(1,*) (EU(I),I=2,IREP2)
      READ(1,*) (RB(I),I=1,IREP2-1)
C
98    X(N)=XS
      X(1)=XL
      Z(N)=ZT
      Z(1)=0.0
      Y(N)=0.0
C
C     CALCULATES THE X(I) & Z(I) C0ORDINATES OF ALL
C     CROSS SECTIONS.
      DO 50  I=2,N
      Z(I)=(Z(N)/FLOAT(N-1))*FLOAT(I-1)
      X(I)=X(1)+((X(N)-X(1))/Z(N))*Z(I)
50    CONTINUE
      WRITE(6,911) (I,X(I),Z(I),I=1,N)
911   FORMAT(3X,I3,2X,F10.4,2X,F10.4)
C
C     CALCULATES THE COORDINATES ZCOM2 & YCOM2
      ZR1=ZH1
      YR1=ZH1*TAN(DTR(FI))+Y(1)-(R1/COS(DTR(FI)))
      ZCOM2=R1*SIN(DTR(THETA))+ZH1
      YCOM2=R1*COS(DTR(THETA))+YR1
      DIF=(180.0-THETA)
C     CALCULATES THE COORDINATES ZCOM1 & YCOM1.
      ZCOM1=-R1*SIN(DTR(FI))+ZH1
      YCOM1=R1*COS(DTR(FI))+YR1
      IF(IREP2.EQ.0)  GO TO 97
      RB(IREP2)=R4
      YRA(1)=YHAP+RA(1)
      ZPM=((YCOM2-YHAP)/TAN(DTR(THETA)))+ZCOM2
      ZRA(1)=(ZCOM2*TAN(DTR(DIF))-YCOM2-(RA(1)/COS(DTR(THETA)))+
     &YHAP
     &-ZPM*TAN(DTR(DIF/2.0)))/(TAN(DTR(DIF))-TAN(DTR(DIF/2.0)))
      CALL CIRLIN(RA(1),ZRA(1),YRA(1),ZCOM2,YCOM2,DIF,
     &          KIND1,ZCOM31,ZCOM32,YCOM31,YCOM32)
      IF (KIND1) 51,53,51
53    ZCM(1)=AMAX1(ZCOM31,ZCOM32)
      YCM(1)=AMIN1(YCOM31,YCOM32)
      ZHB(1)=ZRA(1)+RA(1)+RB(1)-DU(1)
      ZB(1)=ZHB(1)
      CALL PLINCI(RA(1)+RB(1),ZRA(1),YRA(1),ZHB(1),YBA1,YBB1,
     &          KIND3)
      IF (KIND3) 51,55,51
55    YB(1)=AMIN1(YBA1,YBB1)
      IF (IREP2.EQ.1)  GO TO 101
C
      DO 333  I=2,IREP2
      IF(YB(I-1).LT.0.0) GO TO  51
      PET(I)=(RB(I-1)+RA(I))*(1.0-COS(ASIN((RA(I)-YB(I-1))/
```

```
&          (RB(I-1)+RA(I)))))
   ZHA(I)=ZHB(I-1)+RB(I-1)+RA(I)-PET(I)-EU(I)
   ZHB(I)=ZHA(I)+RA(I)+RB(I)-DU(I)
   ZB(I)=ZHB(I)
   ZRA(I)=ZHA(I)
   CALL PLINCI(RB(I-1)+RA(I),ZB(I-1),YB(I-1),ZHA(I),
  & YRAC,YRAD,KIND6)
   IF (KIND6)51,58,51
58  YRA(I)=AMAX1(YRAC,YRAD)
   CALL PLINCI(RA(I)+RB(I),ZRA(I),YRA(I),ZHB(I),
  & YBA3,YBB3,KIND7)
   IF (KIND7) 51,59,51
59  YB(I)=AMIN1(YBA3,YBB3)
333 CONTINUE
    DO 70 J=2,IREP2
   CALL CIRCLS(-ZRA(J),-YRA(J),-ZB(J-1),-YB(J-1),RA(J),RB(J-1)+
  &          0.1,KIN2,ZCAL(J),ZCBL(J),YCAL(J),YCBL(J))
   IF (KIN2) 51,74,51
74  ZCM(J)=ZCAL(J)
   YCM(J)=YCAL(J)
70  CONTINUE
101 DO 175 K=1,IREP2
   CALL CIRCLS(-ZB(K),-YB(K),-ZRA(K),-YRA(K),RB(K)+.01,RA(K),
  &     KIPC,ZPCA(K),ZPCB(K),YPCA(K),YPCB(K))
   WRITE(6,1999) KIPC
1999 FORMAT(1X,I2)
   IF (KIPC) 51,176,51
176 ZPC(K)=ZPCA(K)
   YPC(K)=YPCA(K)
175 CONTINUE
   ZCOM6=ZB(IREP2)+R4*SIN(DTR(BETA))
   YCOM6=R4*COS(DTR(BETA))+YB(IREP2)
C
   IF(IREP2.GT.0) GO TO 96
97  ZCOM6=ZCOM2
   YCOM6=YCOM2
96  AM=TAN(DTR(180.0-BETA))
   BM=TAN(DTR(180.0-ALPHA))
   TM=(R5/SIN(DTR(ALPHA)))-ZT
   TF=(R5/SIN(DTR(BETA)))-ZCOM6
   ZR5=(BM*TM-TF*AM-YCOM6)/(AM-BM)
   YR5=BM*(ZR5+TM)
   CALL CIRLIN(R5,ZR5,YR5,ZCOM6,YCOM6,180.0-BETA,
  &          KTY,ZCOA,ZCCB,YCOA,YCOB)
   IF (KTY) 51,79,51
79  ZCOM7=AMIN1(ZCOA,ZCOB)
   YCOM7=AMAX1(YCOA,YCOB)
   CALL CIRLIN(R5,ZR5,YR5,ZT,0.0,180.0-ALPHA,KTYN,
  &          ZCOC,ZCOD,YCOC,YCOD)
   IF (KTYN) 51,76,51
76   ZCOM8=AMAX1(ZCOC,ZCOD)
   YCOM8=AMIN1(YCOC,YCOD)
109 DO 110 I=2,N
   IF(Z(I).LE.ZCOM1)   GO TO 111
   IF(Z(I).LE.ZCOM2)   GO TO 112
   IF(IREP2.EQ.0)      GO TO 114
   IF(Z(I).LE.ZCM(1))  GO TO 113
   IF(Z(I).LE.ZPC(1))  GO TO 116
```

```
      IF(IREP2.EQ.1)      GO TO 114
      IF(Z(I).LE.ZCM(2))  GO TO 117
      IF(Z(I).LE.ZPC(2))  GO TO 118
      IF(IREP2.EQ.2)      GO TO 114
      IF(Z(I).LE.ZCM(3))  GO TO 119
      IF(Z(I).LE.ZPC(3))  GO TO 120
      IF(IREP2.EQ.3)      GO TO 114
      IF(Z(I).LE.ZCM(4))  GO TO 121
      IF(Z(I).LE.ZPC(4))  GO TO 122
      IF(IREP2.EQ.4)      GO TO 114
C
111   Y(I)=Z(I)*TAN(DTR(FI))+Y(1)
      GO TO 110
112   Y(I)=YR1+SQRT(ABS(R1*R1-(Z(I)-ZH1)**2))
      GO TO 110
113   Y(I)=YCOM2+(Z(I)-ZCOM2)*TAN(DTR(DIF))
      GO TO 110
114   IF(Z(I).LE.ZCOM6)   GO TO 123
      IF(Z(I).LE.ZCOM7)   GO TO 124
      IF(Z(I).LE.ZCOM8)   GO TO 125
      IF(Z(I).LE.ZT)      GO TO 126
C
      GO TO 110
116   Y(I)=YRA(1)-SQRT(ABS(RA(1)**2-(Z(I)-ZRA(1))**2))
      GO TO 110
117   Y(I)=YB(1)+SQRT(ABS(RB(1)**2-(Z(I)-ZB(1))**2))
      GO TO 110
118   Y(I)=YRA(2)-SQRT(ABS(RA(2)**2-(Z(I)-ZRA(2))**2))
      GO TO 110
119   Y(I)=YB(2)+SQRT(ABS(RB(2)**2-(Z(I)-ZB(2))**2))
      GO TO 110
120   Y(I)=YRA(3)-SQRT(ABS(RA(3)**2-(Z(I)-ZRA(3))**2))
      GO TO 110
121   Y(I)=YB(3)+SQRT(ABS(RB(3)**2-(Z(I)-ZB(3))**2))
      GO TO 110
122   Y(I)=YRA(4)-SQRT(ABS(RA(4)**2-(Z(I)-ZRA(4))**2))
      GO TO 110
123   IF (IREP2.EQ.0)  GO TO 130
      Y(I)=YB(IREP2)+SQRT(ABS(R4**2-(Z(I)-ZB(IREP2))**2))
      GO TO 110
130   Y(I)=ZR1+SQRT(ABS(R1**2-(Z(I)-YR1)**2))
      GO TO 110
124   Y(I)=(Z(I)-ZCOM6)*TAN(DTR(180.0-BETA))+YCOM6
      GO TO 110
125   Y(I)=YR5+SQRT(ABS(R5**2-(Z(I)-ZR5)**2))
      GO TO 110
126   Y(I)=(Z(I)-ZT)*TAN(DTR(180.0-ALPHA))
110   CONTINUE
      WRITE(6,912) (I,Y(I),I=1,N)
912   FORMAT(5X,I3,3X,F10.4)
C
      DO 200 I=1,N
      YA(I,1)=39.00
      YA(I,11)=39.00-Y(I)
      XA(I,1)=X(I)/2.0
      XA(I,11)=0.0
      AS=90.0-ALFAX
      BS(I)=-(RCS2/COS(DTR(ALFAX)))+XA(I,1)
```

```
C
    IF(RCS1.EQ.0.0)    GO TO  300
    IF(Y(I).EQ.0.0)  GO TO 301
C
    FS(I)=YA(I,11)+RCS1
    CALL CIRLIN(RCS1-RCS2,0.0,FS(I),BS(I),YA(I,1),AS,KTYP,
   &  XQA(I),XQB(I),YQA(I),YQB(I))
    IF (KTYP) 51,721,51
721  XQ(I)=AMIN1(XQA(I),XQB(I))
    YQ(I)=AMIN1(YQA(I),YQB(I))
    WRITE(6,1000)  XQ(I),YQ(I)
1000 FORMAT( 1X,F10.4,1X,F10.4)
C
    CALL CIRLIN(RCS2+0.00001,XQ(I),YQ(I),XA(I,1),YA(I,1),AS,
   & KTYP1,BA(I),BB(I),BC(I),BD(I))
    IF (KTYP1) 51,722,51
722  IF(YQ(I)-39.00) 724,724,725
724   XA(I,4)=AMAX1(BA(I),BB(I))
    YA(I,4)=AMAX1(BC(I),BD(I))
    GO TO 741
725  CALL CIRLIN(RCS1,0.0,FS(I),XA(I,1),YA(I,1),AS,
   & KTY,WAA(I),WAB(I),WAC(I),WAD(I))
    IF (KTY) 51,753,51
753  XA(I,4)=AMIN1(WAA(I),WAB(I))
    YA(I,4)=AMIN1(WAC(I),WAD(I))
    DO 733 L=5,10
    PE(L)=(FLOAT(L-4)*(XA(I,11)-XA(I,4)))/7.0
    XA(I,L)=PE(L)+XA(I,4)
    YA(I,L)=-SQRT(ABS(RCS1**2-XA(I,L)**2))+FS(I)
733  CONTINUE
    GO TO 788
741  CALL CIRCLS(0.0,-FS(I),-XQ(I),-YQ(I),RCS1,RCS2+0.1,KTYPE,
   &  WA(I),WB(I),WC(I),WD(I))
    IF (KTYPE) 51,723,51
723  XA(I,7)=AMIN1(WA(I),WB(I))
    YA(I,7)=AMIN1(WC(I),WD(I))
788  PM(I)=XA(I,4)-XA(I,1)
    XA(I,2)=+(PM(I)/3.0)+XA(I,1)
    XA(I,3)=+((2.0*PM(I))/3.0)+XA(I,1)
C
    DO  201  J=2,3
    YA(I,J)=(XA(I,J)-XA(I,1))*TAN(DTR(90.0-ALFAX))+YA(I,1)
201  CONTINUE
    IF((YQ(I)-39.00).GT.0.0) GO TO 789
C
    PF(I)=(XA(I,4)-XA(I,7))/3.0
    XA(I,5)=+2.0*PF(I)+XA(I,7)
    XA(I,6)=+PF(I)+XA(I,7)
C
    DO  202  K=5,6
    YA(I,K)=-SQRT(ABS(RCS2**2-(XA(I,K)-XQ(I))**2))+YQ(I)
202  CONTINUE
C
C
    DO  203  L=1,3
    PE(L+7)=(FLOAT(L)*(XA(I,11)-XA(I,7)))/4.0
203  CONTINUE
C
```

```
      DO  204 L=8,10
      XA(I,L)=PE(L)+XA(I,7)
      YA(I,L)=-SQRT(ABS(RCS1**2-XA(I,L)**2))+FS(I)
204   CONTINUE
789   WRITE(6,1001) XA(I,4),YA(I,4),XA(I,7),YA(I,7)
1001  FORMAT(1X,F10.4,1X,F10.4,1X,F10.4,1X,F10.4)
      GO TO 834
C
301   DO 302 J=2,10
      YA(I,J)=39.00
      XA(I,J)=(XA(I,1)/10.0)*FLOAT(-J+1)+XA(I,1)
302   CONTINUE
834   DO 205 K=12,21
      XA(I,K)=-XA(I,-K+22)
      YA(I,K)=YA(I,-K+22)
205   CONTINUE
      GO  TO  200

300   XQ(I)=(Y(I)-RCS2-BS(I)*AS)/AS
      YQ(I)=Y(I)-RCS2
      XA(I,4)=RCS2*COS(DTR(ALFAX))+XQ(I)
      YA(I,4)=RCS2*SIN(DTR(ALFAX))+YQ(I)
C
      DO  310  K=2,3
      PM(I)=(XA(I,4)-XA(I,1))/3.0
      XA(I,K)=-PM(I)*FLOAT(K-1)+XA(I,1)
      YA(I,K)=TAN(DTR(ALFAX+90.0))*(XA(I,K)-XA(I,1))
310   CONTINUE
C
      XA(I,7)=XQ(I)
      YA(I,7)=YA(I,11)
C
      DO  320  K=8,10
      YA(I,K)=YA(I,11)
320   CONTINUE
C
      PF(I)=(XA(I,7)-XA(I,11))/4.0
      XA(I,8)=-PF(I)+XA(I,7)
      XA(I,9)=-PF(I)*2.0+XA(I,7)
      XA(I,10)=-PF(I)*3.0+XA(I,7)
      DM(I)=(XA(I,4)-XA(I,7))/3.0
      XA(I,5)=-2.0*DM(I)+XA(I,7)
      XA(I,6)=-DM(I)+XA(I,7)
C
C
      DO  330  L=5,6
      YA(I,L)=SQRT(ABS(RCS2**2-XA(I,L)**2))
330   CONTINUE
C
      DO  340  J=12,21
      XA(I,J)=-XA(I,-J+22)
      YA(I,J)=YA(I,-J+22)
340   CONTINUE
200   CONTINUE
      NUM=21
C
      DO  500  L=1,N
      WRITE(2,510) NUM,M,ID,ICLOS
```

```
      WRITE(2,512) (XA(L,J),YA(L,J),J=1,21)
C
510   FORMAT(1X,I3,1X,I3,/1X,I3,1X,I3)
512   FORMAT(1X,F10.4,1X,F10.4)
C
500   CONTINUE
      INUZ=0
      WRITE(2,518) INUZ,INUZ
518   FORMAT(1X,I3,1X,I3)
      WRITE(6,900) (K,X(K),Y(K),Z(K),K=1,N)
900   FORMAT(1X,I3,1X,F10.4,1X,F10.4,1X,F10.4)
      STOP
51    WRITE(6,530)
530   FORMAT(' IMAGINARY ')
      END
      SUBROUTINE CIRCLS(G1,F1,G2,F2,R1,R2,IL,X1,X2,Y1,Y2)
C     THIS SUBROUTINE FINDS THE INTERSECTIONS POINTS
C     OF TWO CIRCLES,DEFINED BY ((-G1,-F1),R1)
C     AND ((-G2,F2),R2) RESPECTIVELY.
      C1=G1**2+F1**2-R1**2
      C2=G2**2+F2**2-R2**2
      A=+(C2-C1)/(2.0*(G1-G2))+G1
      B=-(F1-F2)/(G1-G2)
      U=B**2+1.0
      V=2.0*(A*B+F1)
      W=A**2-R1**2+F1**2
      Y=V**2-4.0*U*W
      IF (Y) 3,8,8
8     IL=0
      Y1=(-V+SQRT(Y))/(2.0*U)
      Y2=(-V-SQRT(Y))/(2.0*U)
      X1=(C1-C2-2.0*(F2-F1)*Y1)/(2.0*(G2-G1))
      X2=(C1-C2-2.0*(F2-F1)*Y2)/(2.0*(G2-G1))
      RETURN
3     IL=1
      RETURN
      END
C
      SUBROUTINE CIRLIN(R,G1,F1,G2,F2,ANG,KL,X1,X2,Y1,Y2)
C     THIS SUBROUTINE FINDS THE INTERSECTION POINTS
C     OF A CIRCLE ((G1,F1),R) AND A LINE
C     WITH SLOPE NOT EQUAL TO (PI/2).
      SL=TAN(DTR(ANG))
      A=SL**2+1.0
      B=-2.0*SL*(G2*SL-F2+F1)-2.0*G1
      C=-(R**2)+G1**2+(G2*SL-F2+F1)**2
      Y=B**2-4.0*A*C
      IF (Y) 4,5,5
5     KL=0
      X1=(-B+SQRT(Y))/(2.0*A)
      X2=(-B-SQRT(Y))/(2.0*A)
      Y1=(X1-G2)*SL+F2
      Y2=(X2-G2)*SL+F2
      RETURN
4     KL=1
      RETURN
      END
C
```

```
C
      SUBROUTINE PLINCI(RN,G1,F1,G2,Y1,Y2,KINT)
C     THIS SUBROUTINE FINDS THE INTERSECTION POINTS
C     OF A CIRCLE ((G1,F1),RN) AND A LINE WITH
C     SLOPE EQUAL TO (PI/2).
      U=RN**2-(G2-G1)**2
      IF (U) 6,7,7
7     KINT=0
      V=SQRT(U)
      Y1=F1+V
      Y2=F1-V
6     KIN=1
      RETURN
      END
C
C
C
C
      FUNCTION  DTR(ANGLE)
C
      PI=3.1416
      DTR=(ANGLE*PI)/180.0
      RETURN
      END
C
C
C
C
      FUNCTION ACCLOSE(ACCL)
      CHARACTER*64 BUFF
      WRITE(UNIT=BUFF,FMT=20)ACCL
   20 FORMAT(F9.5)
      READ(UNIT=BUFF,FMT=21)ACCL
   21 FORMAT(F8.4)
      ACCLOSE=ACCL
      END
C



C     COMMENTS:
C      POINTS ZCOM1,....,ZCOM8,YCOM1,.....YCOM8,
C      ZCM(I),YCM(I),ZPC(I),YPC(I),REFER TO THE
C      INTERSECTION POINTS P1,.....P9 IN THE
C      FIGURE 5.1.A.
C      FOR ANY ENQUIRES ON THE STRACTURE OF THE
C      PROGRAM REFER TO THE THESIS (CHAPTER 5).
C
C      INPUT:THERE IS ONE INPUT FILE THROUGH
C      CHANEL 1 (*CR0 FILENAME)
C      OUTPUT: THERE ARE TWO OUTPUT FILES THROUGH
C           CHANELS 2 & 6 (*LP0 FILENAME,*LP1 FILENAME)
C           *LP0 IS USED BY THE PROGRAM BCSRF1 AS INPUT
C           AND *LP1 SERVES AS INPUT TO THE PROGRAM
C           INVERT (REFERENCE 13).
```

PROGRAM S_4

```
C    THIS PROGRAM READS THE NUMBER OF CROSS-SECTIONS,
C    THE NUMBER OF CARDINAT B-SPLINES REQUIRED
C    AND THE NUMBER OF REQUIRED CROSS-SECTIONS
C    AS WELL AS THE OUTPUT FROM CHANEL 6,FROM
C    PROGRAM "SECT" ,TO PRODUCE A NEW FILE IN
C    THE FORM REQUIRED BY THE PROGRAMS BCSURF
C    & BCSURN.
     DIMENSION X(100),Z(100)
     OPEN(UNIT=2,NAME='OBCSURF1',STATUS='UNKNOWN')
     OPEN(UNIT=1,NAME='RE31',STATUS='UNKNOWN')
     OPEN(UNIT=5,NAME='DARFRE',STATUS='UNKNOWN')
     READ(5,*) N,NPV,NPU
     IL=0
     JL=1
     DP=0.0
     DO 110 I=1,N
     READ(1,*)IZ,X(I),Z(I)
110  CONTINUE
     WRITE(2,20) N,NPU,NPV
     WRITE(2,30) IL,JL
     DO 100 K=1,N
     WRITE(2,40) JL
     WRITE(2,50) DP,DP,Z(K),DP,DP
40   FORMAT(1X,I3)
50   FORMAT(1X,F3.1,1X,F3.1,1X,F10.4,1X,F3.1,1X,F3.1)
100  CONTINUE
     WRITE(2,60) IL,IL,IL
60   FORMAT(1X,I3,1X,I3,1X,I3)
20   FORMAT(1X,I3,1X,I3,1X,I3)
30   FORMAT(1X,I3,1X,I3)
     STOP
     END
```

```
      PROGRAM BSCURV
C     THIS PROGRAM BY CALLING SUBROUTINE 'BSPLINE'
C     APPROXIMATE A B-SPLINE CURVE TO A GIVEN POLYGON
C     AND THEN PLOTS THE RESULTED CURVE.
C
C
      DIMENSION XV(50),YV(50),ZV(50),X(50),
     &          XR(1800),YR(1800),ZR(1800)
      REAL N(50,10)
      COMMON/BL1/XSCL,YSCL,NSECT
C     DIMENSION FOR XV,YV,ZV: NUM+2*M
C           FOR XR,YR,ZR: NO.OF GENERATED PNTS.
C           FOR N: (NUM+2*M,M)
C           FOR X: NUM+2*M
      OPEN(UNIT=5,NAME='DBSCURV5',STATUS='UNKNOWN')
      OPEN(UNIT=2,NAME='OBSCURV2B',STATUS='UNKNOWN')
      L=0
C     READ THE DATA AND PRODUCE PRINTOUT OF THE INPUT
C
C
      PRINT*,'   '
      PRINT*,'   '
      PRINT*,'   '
      PRINT*,'   '
C
      PRINT 201
      PRINT 202
      PRINT 203
      PRINT 204
      PRINT 205
      PRINT 206
      PRINT 207
      PRINT 211
      PRINT 208
      PRINT 202
      PRINT 201
C
      READ(*,209)IT
C
 201  FORMAT('        ****************************************')
 202  FORMAT('        **                                    **')
 203  FORMAT('        **      TERMINAL DEFIONITION          **')
 204  FORMAT('        **      PLEASE INPUT  1,2,3,4         **')
 205  FORMAT('        **      (1) FOR T4010                 **')
 206  FORMAT('        **      (2) FOR T4107                 **')
 207  FORMAT('        **      (3) FOR T4113                 **')
 211  FORMAT('        **      (4) FOR T4105                 **')
 208  FORMAT('        **      (5) FOR VT125                 **')
 209  FORMAT(I2)
C
 90   READ(5,*)NUM,M
      IF(NUM.EQ.0)GO TO 190
      WRITE(2,999)
      WRITE(2,997)NUM,M
      READ(5,*)ID,ICLOS
      IF(ICLOS.EQ.0)GO TO 100
```

```
      WRITE(2,996)
      GO TO 110
  100 WRITE(2,995)
  110 IF(ID.EQ.2)GO TO 120
      WRITE(2,994)
      GO TO 130
  120 WRITE(2,993)
  130 READ(5,*)NP
      WRITE(2,992)NP
      WRITE(*,3131)
 3131 FORMAT(3X,'ENTER(XSHIFT,SHIFT)')
      READ*,XSHIFT,SHIFT
      CALL PICCLE
      WRITE(*,2121)
 2121 FORMAT(3X,'ENTER XSCL,YSCL')
      READ*,XSCL,YSCL
      WRITE(2,990)XSCL,YSCL
      WRITE(*,4444)
 4444 FORMAT(3X,'ENTER NSECT')
      READ*,NSECT
      CALL GINO
      IF(IT.EQ.1)CALL T4010
      IF(IT.EQ.2)CALL T4107
      IF(IT.EQ.3)CALL T4113
      IF(IT.EQ.4)CALL T4105
      IF(IT.EQ.5)CALL VT125
      PRINT*,' '
      PRINT*,' '
      PRINT*,' '
      PRINT*,' '
      IF(IT.EQ.1)CALL UNITS(1.0)
      IF(IT.EQ.2)CALL UNITS(1.0)
      IF(IT.EQ.3)CALL UNITS(1.55)
      IF(IT.EQ.4)CALL UNITS(1.0)
      IF(IT.EQ.5)CALL UNITS(1.0)
      CALL FRAME
      CALL NAME
      CALL YNAME
      CALL SHIFT2(XSHIFT,SHIFT)
      CALL SCALE2(XSCL,YSCL)
      IF(ID.EQ.2)GO TO 133
      READ(5,*)((XV(I),YV(I),ZV(I)),I=1,NUM)
      GO TO 134
  133 READ(5,*)((XV(I),YV(I)),I=1,NUM)
  134 WRITE(2,989)
      IF(ID.EQ.2)GO TO 135
      WRITE(2,988)((I,XV(I),YV(I),ZV(I)),I=1,NUM)
      GO TO 136
  135 WRITE(2,985)((I,XV(I),YV(I)),I=1,NUM)
  136 NUM2M=NUM+2*M
      IFAULT=0
C     CALL SUBROUTINE BSPLINE TO APPROXIMATE THE CURVE
C
      CALL
BSPLINE(M,NUM,XV,YV,ZV,ID,NUM2M,ICLOS,NP,XR,YR,ZR,IFAULT,N,X)
      IF(IFAULT.EQ.0)GO TO 140
      WRITE(2,986)
      GO TO 200
  140 WRITE(2,987)
C     PRINTOUT THE GENERATED POINTS ALONG THE CURVE.
```

```
C
      IF(ID .EQ. 2)GO TO 145
      WRITE(2,988)((I,XR(I),YR(I),ZR(I)),I=1,NP)
      GO TO 146
145   WRITE(2,985)((I,XR(I),YR(I)),I=1,NP)
146   IF(L.NE.0)GO TO 150
C  PLOT THE CURVE GIVEN THE POINTS PRODUCED BY 'BSPLINE'
C
150   CALL MOVTO2(XV(1),YV(1))
      CALL LINCOL(5)
      CALL POLTO2(XV,YV,NUM)
      IF(ICLOS.EQ.1)CALL LINTO2(XV(1),YV(1))
      CALL LINCOL(6)
      CALL SYMTO2(XV,YV,NUM,4)
      CALL LINCOL(2)
      CALL MOVTO2(XR(1),YR(1))
      CALL POLTO2(XR,YR,NP)
      CALL SCALE2(1/XSCL,1/YSCL)
      CALL SHIFT2(-XSHIFT,-SHIFT)
      L=L+1
      READ*,IR
      IF(IR.EQ.3)CALL DEVEND
      GO TO 90
190   CALL DEVEND
200   STOP
C
C
999   FORMAT(1H1,//'  B-SPLINE APPROXIMATION TO A GIVEN POLYGON',
     &          /'  ----------------------------------------')
997   FORMAT(//' NUMBER OF VERTICES   NUM=',I3,
     &          /' THE ORDER OF THE CURVE  M=',I3)
996   FORMAT(/' OPEN OR CLOSED POLYGON?  CLOSED')
995   FORMAT(/' OPEN OR CLOSED POLYGON?  OPEN')
994   FORMAT(' PLANE OR SPACE CURVE?    SPACE')
993   FORMAT(' PLANE OR SPACE CURVE?    PLANE')
992   FORMAT(/' NO.OF GENERATED POINTS ALONG THE CURVE=',I6)
990   FORMAT(//' X-SCALING FACTOR IN CURVE PLOTTING=',F6.3,
     &       /' Y-SCALING FACTOR IN CURVE PLOTTING=',F6.3)
989   FORMAT(//' X-Y-Z COORDINATES OF THE VERTICES.',
     &      //'  I',6X,'X-ORDINATE',6X,'Y-ORDINATE',6X,'Z-ORDINATE')
988   FORMAT(1X,I3,7X,F8.4,8X,F8.4,8X,F8.4)
987   FORMAT(//' X-Y-Z COORDINATES OF THE GENERATED POINTS.',
     &      //'  I',6X,'X-ORDINATE',6X,'Y-ORDINATE',6X,'Z-ORDINATE')
986   FORMAT(//' ERROR RETURN --M>NUM--')
985   FORMAT(1X,I6,7X,F8.4,8X,F8.4)
      END
C  THIS SUBROUTINE GENERATES B-SPLINE BASIS KNOT VECTORS
C AND B-SPLINE CURVE OF VARIOUS ORDER TO APPROXIMATE A
C  GIVEN POLYGON.
C
C
      SUBROUTINE BSPLINE(M,NUM,XV,YV,ZV,ID,NUM2M,
     &                   ICLOS,NP,XR,YR,ZR,IFAULT,N,X)
C  M: THE ORDER OF B-SPLINE BASIS
C  NUM: NUMBER OF POLYGON VERTICES
C  XV: X-ORDINATES OF DEFINING POLYGON VERTICES
C  YV: Y-ORDINATES OF DEFINING POLYGON VERTICES
C  ZV: Z-ORDINATES OF DEFINING POLYGON VERTICES
C  ID: CONTROL VARIABLE, ID=2 FOR 2D, ID=3FOR 3D CURVE
C  NUM2M: NUM+2*M INTEGER NUMBER FOR DEFINING VARIABLE
```

```
DIMENSIONS
C  ICLOS: CONTROL VARIABLE, ICLOS=1 FOR CLOSED POLYGON,
C                  ICLOS=0 FOR OPEN POLYGON.
C  NP: THE NUMBER OF POINTS GENERATED ALONG THE CURVE

C  XR,YR,ZR: ARRAYS CONTAINING COORDINATES OF THE GENERATED
POINTS
C  IFAULT: ERROR RETURN IF M>NUM
C
    DIMENSION XV(NUM2M),YV(NUM2M),ZV(NUM2M),X(NUM2M),
   &       XR(NP),YR(NP),ZR(NP)
    REAL N(NUM2M,M)
    INTEGER W
    L=0
    G=0.0
    H=0.0
    S=0.0
    IF(ICLOS.NE.1)GO TO 110
C  ADD M-1 EXTRA VERTICES TO THE DEFINING ARRAY OF VERTICES
C  IF CLOSED CURVE
C
    DO 100 J=1,M-1
    XV(NUM+J)=XV(J)
    YV(NUM+J)=YV(J)
100  ZV(NUM+J)=ZV(J)
    NUM=NUM+M-1
110  NUM1=NUM-1
    IF(M.GT.NUM)GO TO 260
    MAX=NUM1-M+2
C  FILL THE WEIGHING FUNCTIONS ARRAY WITH ZERO
C
    DO 120 I=1,NUM+1
    DO 120 K=1,M
120  N(I,K)=0.0
    IX=NUM+M
    IF(ICLOS.NE.1)GO TO 130
C  DEFINE THE KNOT VECTORS
C
    CALL KNOTCL(M,MAX,XV,YV,ZV,NUM,ID,X,IX)
    GO TO 135
130  CALL KNOTOP(M,MAX,XV,YV,ZV,NUM,ID,X,IX)
135  STEP=(X(NUM+1)-X(M))/FLOAT(NP-1)
    W=M-1
C  INCREMENT KNOT VECTOR SUBSCRIPT
140  W=W+1
    IF(X(W).EQ.X(W+1))GO TO 140
    N(W,1)=1.
    T=X(W)
150  L=L+1
C  CALCULATE VALUES OF N(I,K) WEIGHING FUNCTIONS
    DO 210 K=2,M
    DO 200 I=1,NUM
    IF(N(I,K-1).NE.0.0)GO TO 160
    E=0.0
    GO TO 170
160  E=(T-X(I))*N(I,K-1)/(X(I+K-1)-X(I))
170  IF(N(I+1,K-1).NE.0.0)GO TO 180
    F=0.0
    GO TO 190
180  F=(X(I+K)-T)*N(I+1,K-1)/(X(I+K)-X(I+1))
```

```
190  N(I,K)=E+F
     G=XV(I)*N(I,K)+G
     H=YV(I)*N(I,K)+H
     IF(ID.EQ.2)GO TO 200
     S=ZV(I)*N(I,K)+S
200  CONTINUE
     IF(K.EQ.M)GO TO 220
     G=0.0
     H=0.0
     S=0.0
210  CONTINUE
220  XR(L)=G
     YR(L)=H
     IF(ID.EQ.3)ZR(L)=S
C  INCREMENT PARAMETER T
C
     T=T+STEP
     IF(L.EQ.(NP-1))GO TO 240
     IF(T.GE.X(W).AND.T.LT.X(W+1))GO TO 150
     N(W,1)=0.0
230  W=W+1
     IF(X(W).EQ.X(W+1))GO TO 230
     N(W,1)=1.0
     GO TO 150
C  DEFINE THE LAST POINT ON THE CURVE
C
240  L=L+1
     IF(ICLOS.EQ.1)GO TO 250
     XR(L)=XV(NUM)
     YR(L)=YV(NUM)
     IF(ID.EQ.3)ZR(L)=ZV(NUM)
     NP=L
     RETURN
250  XR(L)=XR(1)
     YR(L)=YR(1)
     IF(ID.EQ.3)ZR(L)=ZR(1)
     NP=L
     NUM=NUM-M+1
     RETURN
260  IFAULT=1
     RETURN
     END
C  SUBROUTINE TO GENERATE BASIS KNOT VECTOR FOR
C  OPEN POLYGON
C
     SUBROUTINE KNOTOP(M,MAX,XV,YV,ZV,NUM,ID,X,IX)
     DIMENSION XV(NUM),YV(NUM),ZV(NUM),X(IX)
     DO 50 I=1,IX
     IF(I.GT.M)GO TO 10
C  ASSIGN MULTIPLE KNOTS FOR THE BEGINNING OF SPAN
     X(I)=0.0
     GO TO 50
10   IF(I.LT.(MAX+M+1)) GO TO 30
C    ASSIGN MULTIPLE KNOTS
C
20   X(I)=X(I-1)
     GO TO 50
30   IF(ID.EQ.3)GO TO 40
C    CHECK FOR MULTIPLE KNOTS
C
```

```
      IF(XV(I-M).EQ.XV(I-M+1).AND.
     &  YV(I-M).EQ.YV(I-M+1))GO TO 20
      X(I)=X(I-1)+1.
      GO TO 50
40    IF(XV(I-M).EQ.XV(I-M+1).AND.
     &  YV(I-M).EQ.YV(I-M+1).AND.
     &  ZV(I-M).EQ.ZV(I-M+1))GO TO 20
      X(I)=X(I-1)+1.
50    CONTINUE
      RETURN
      END
C
C  SUBROUTINE TO GENERATE BASIS KNOT VECTORS FOR
C CLOSED POLYGON
C
      SUBROUTINE KNOTCL(M,MAX,XV,YV,ZV,NUM,ID,X,IX)
      DIMENSION XV(NUM),YV(NUM),ZV(NUM),X(IX)
C  FIRST KNOT VECTOR
C
      X(1)=FLOAT(1-M)
      DO 120 I=2,IX
      IF(I.GT.M)GO TO 40
      IF(ID.EQ.3)GO TO 20
C  CHECK FOR REPEATING VERTICES
C
      IF(XV(I+MAX-M).EQ.XV(I+MAX-M+1).AND.
     &  YV(I+MAX-M).EQ.YV(I+MAX-M+1))GO TO 10
      X(I)=X(I-1)+1.
      GO TO 120
10    X(I)=X(I-1)
      GO TO 120
20    IF(XV(I+MAX-M).EQ.XV(I+MAX-M+1).AND.
     &  YV(I+MAX-M).EQ.YV(I+MAX-M+1).AND.
     &  ZV(I+MAX-M).EQ.ZV(I+MAX-M+1)) GO TO 30
      X(I)=X(I-1)+1.
      GO TO 120
30    X(I)=X(I-1)
      GO TO 120
40    IF(I.LT.(MAX+M+1))GO TO 80
      IF(ID.EQ.3)GO TO 60
      IF(XV(I-MAX-M).EQ.XV(I-MAX-M+1).AND.
     &  YV(I-MAX-M).EQ.YV(I-MAX-M+1))GO TO 50
      X(I)=X(I-1)+1.
      GO TO 120
50    X(I)=X(I-1)
      GO TO 120
60    IF(XV(I-MAX-M).EQ.XV(I-MAX-M+1).AND.
     &  YV(I-MAX-M).EQ.YV(I-MAX-M+1).AND.
     &  ZV(I-MAX-M).EQ.ZV(I-MAX-M+1))GO TO 70
      X(I)=X(I-1)+1.
      GO TO 120
70    X(I)=X(I-1)
      GO TO 120
80    IF(ID.EQ.3)GO TO 100
      IF(XV(I-M).EQ.XV(I-M+1).AND.
     &  YV(I-M).EQ.YV(I-M+1))GO TO 90
      X(I)=X(I-1)+1.
      GO TO 120
90    X(I)=X(I-1)
      GO TO 120
```

```
100  IF(XV(I-M).EQ.XV(I-M+1).AND.
   &  YV(I-M).EQ.YV(I-M+1).AND.
   &  ZV(I-M).EQ.ZV(I-M+1))GO TO 110
     X(I)=X(I-1)+1.
     GO TO 120
110  X(I)=X(I-1)
120  CONTINUE
     RETURN
     END

     SUBROUTINE FRAME
     COMMON/BL1/XSCL,YSCL,NSECT
     CALL LINCOL(5)
     CALL MOVTO2(0.0,0.0)
     CALL LINBY2(230.0,0.0)
     CALL LINBY2(0.0,170.0)
     CALL LINBY2(-230.0,0.0)
     CALL LINBY2(0.0,-170.0)
     CALL MOVTO2(0.0,0.0)
     CALL LINBY2(0.0,40.0)
     CALL LINBY2(230.0,0.0)
     CALL MOVTO2(0.0,0.0)
     CALL MOVTO2(10.0,10.0)
     CALL CHASIZ(3.5,3.5)
     CALL CHASTR('CROSS-SECTIONAL CURVE FOR LOWER PART OF THE M' )
     CALL CHASTR('OULD.')
     CALL MOVTO2(0.0,20.0)
     CALL LINBY2(230.0,0.0)
     CALL MOVBY2(-70.0,0.0)
     CALL LINBY2(0.0,20.0)
     CALL MOVBY2(-70.0,0.0)
     CALL LINBY2(0.0,-20.0)
     CALL MOVTO2(0.0,0.0)
     CALL MOVTO2(95.0,25.0)
     CALL CHASIZ(2.5,2.5)
     CALL CHAHOL(27HTHE ORDER OF THE CURVE= 4*.)
     CALL MOVTO2(95.0,30.0)
     CALL CHAHOL(24HNUMBER OF VERTICES = 7*.)
     CALL MOVTO2(170.0,35.0)
     CALL CHAHOL(18HMODEL NUMBER = 1*.)
     CALL MOVTO2(170.0,25.0)
     CALL SECTION
     CALL MOVTO2(160.0,29.0)
     CALL LINBY2(70.0,0.0)
     RETURN
     END
C
C
C
C
     SUBROUTINE NAME
     COMMON/BL1/XSCL,YSCL,NSECT
     CALL MOVTO2(0.0,0.0)
     CALL MOVTO2(10.0,30.0)
     CALL CHASIZ(2.5,2.5)
     CALL CHAHOL(20HX-SCALING FACTOR =*.)
     CALL CHAFIX(XSCL,5,1)
     RETURN
     END
C
```

```
C
CC
CC
C
      SUBROUTINE YNAME
      COMMON/BL1/XSCL,YSCL,NSECT
      CALL MOVTO2(0.0,0.0)
      CALL MOVTO2(10.0,25.0)
      CALL CHASIZ(2.5,2.5)
      CALL CHAHOL(20HY-SCALING FACTOR =*.)
      CALL CHAFIX(YSCL,5,1)
      RETURN
      END
C
CC
CC
C
      SUBROUTINE SECTION
      COMMON/BL1/XSCL,YSCL,NSECT
      CALL CHAHOL(18HSECTION NUMBER =*.)
      CALL CHAINT(NSECT,1)
      RETURN
      END
```

```
C   THIS PROGRAMME BY CALLING SUBROUTINE 'REPAR' FIRST FITS
C  B-SPLINE CURVES TO CROSS SECTIONS OF A 3-D OBJECT,THEN BY
C  CALLING SUBROUTINE 'EQL' INTERPOLATES SELECTED POINTS ON
C  THE B-SPLINE CURVES USING CARDINAL CUBIC SPLINE.
C
C
     DIMENSION X1(70),Y1(70),Z1(70),
    &       XX(80,80),YY(80,80),ZZ(80,80),
    &       XV(80),YV(80),ZV(80),XF(80,80),YF(80,80),ZF(80,80)
     REAL N(80,10),X(80)
C   NPU: NO. OF SELECTED POINTS OVER PARAMETER "U". <CROSS
SECTIONS.>
C   NPV: NO. OF SELECTED POINTS OVER PARAMETER "V".
C   NS: NO. OF CROSS SECTIONS.
C   NUM: NO. OF VERTICES.
C   M: THE ORDER OF THE B-SPLINE CURVE.
C
C
C   DIMENSION FOR X1,Y1,Z1: MAX(NPU,NPV)
C             XV,YV,ZV: MAX(MAX(NUM),NS+3)
C             XX,YY,ZZ: (MAX(NPU,NPV),NS+3)
C             XF,YF,ZF: (NPU,NPV)
C             N: (MAX(NUM)+2*MAX(M),MAX(M))
     OPEN(UNIT=1,NAME='OBCSURF1',STATUS='UNKNOWN')
     OPEN(UNIT=5,NAME='OINVERT',STATUS='UNKNOWN')
     OPEN(UNIT=6,NAME='OBCSURF',STATUS='UNKNOWN')
     OPEN(UNIT=3,NAME='66',STATUS='UNKNOWN')
     ND=100
1000  READ(1,*)NS,NPU,NPV
     IF(NS.EQ.0)GO TO 1120
     READ(1,*)JCLOS,ISPR
     PIE=3.1415
C  READ DATA TO APPROXIMATE CROSS SECTIONS
C
     DO 1070 J=1,NS
     READ(5,*)NUM,M
     READ(5,*)ID,ICLOS
     READ(1,*)IN
     READ(1,*)XOR,YOR,ZOR,BETA,GAMA
     BETA=BETA/180*PIE
     GAMA=GAMA/180*PIE
     DO 1050 I=1,NUM
     IF(ID.EQ.2)GO TO 1046
     READ(5,*)XV(I),YV(I),ZV(I)
     GO TO 1047
1046  READ(5,*)XV(I),YV(I)
     ZV(I)=0.0
1047  SX=XOR+(YV(I)*SIN(BETA)+ZV(I)*COS(BETA))*SIN(GAMA)
    &     +XV(I)*COS(GAMA)
     SY=YOR+YV(I)*COS(BETA)-ZV(I)*SIN(BETA)
     SZ=ZOR+(YV(I)*SIN(BETA)+ZV(I)*COS(BETA))*COS(GAMA)
    &     -XV(I)*SIN(GAMA)
      XV(I)=SX
      YV(I)=SY
      ZV(I)=SZ
```

```
1050  CONTINUE

      NUM2M=NUM+2*M
      IFAULT=0
C  CALL SUBROUTINE REPAR TO APPROXIMATE THE CROSS SECTION AND
C  SELECT "NPU" POINTS AT EQUAL INTERVALS.
      CALL REPAR(M,NUM,XV,YV,ZV,IN,3,NUM2M,ICLOS,
     &          NPU,ND,X1,Y1,Z1,IFAULT,N,X)
      IF(IFAULT.EQ.1)GO TO 1110
C  STORE THE SELECTED POINTS.
      DO 1060 I=1,NPU
      XX(I,J+1)=X1(I)
      YY(I,J+1)=Y1(I)
      ZZ(I,J+1)=Z1(I)
1060  CONTINUE
1070  CONTINUE
C
C  INTERPOLATE THE SELECTED POINTS.
1066  DO 1100 I=1,NPU
      DO 1080 J1=2,NS+1
      XV(J1)=XX(I,J1)
      YV(J1)=YY(I,J1)
      ZV(J1)=ZZ(I,J1)
1080  CONTINUE
      NS1=NS+3
C  CALLSUBROUTINE EQL TO INTERPOLATE ONE SET OF POINTS AND
SELECT
C  "NPV" POINTS AT EQUAL INTERVALS OVER THE CURVE.
      CALL EQL(NS,XV,YV,ZV,3,JCLOS,NPV,ND,X1,Y1,Z1,NS1)
      DO 1090 J=1,NPV
      XF(I,J)=X1(J)
      YF(I,J)=Y1(J)
      ZF(I,J)=Z1(J)
1090  CONTINUE
1100  CONTINUE
      WRITE(6,969)NPU,NPV
      WRITE(6,972)((XF(I,J),YF(I,J),ZF(I,J),J=1,NPV),I=1,NPU)
      GO TO 1000
1120  WRITE(6,968)
1110     STOP
C
C
972   FORMAT(1X,3(1X,F10.4))
969   FORMAT(1X,I3,1X,I3)
968   FORMAT(' 0 0')
      END
C
C  THIS SUBROUTINE SELECTS NPS POINTS ALONG A GIVEN
C B-SPLINE CURVE AT EQUAL INTERVALS.
C  THE VALUE FOR PARAMETER'IN' WHICH REFERS TO THE
C DESIRED STARTING POINT ON THE CURVE IS DETERMINED
C USING THE LISTING OF GENERATED POINTS.
C
C
      SUBROUTINE REPAR(M,NUM,XV,YV,ZV,IN,ID,NUM2M,ICLOS,
     &          NPS,ND,XS,YS,ZS,IFAULT,N,X)
C
C  XS,YS,ZS: ARRAYS CONTAINING COORDINATES OF SELECTED POINTS
C  XD,YD,ZD: ARRAYS CONTAINING COORDINATES OF ND GENERATED
```

```
POINTS
C       ALONG THE CURVE.
C
C
C  NPS: NO. OF SELECTED POINTS.
C  ND: NO. OF GENERATED POINTS TO CALCULATE ARC LENGTH
C
      DIMENSION
     XV(NUM2M),YV(NUM2M),ZV(NUM2M),XS(NPS),YS(NPS),ZS(NPS)
      REAL N(NUM2M,M),X(NUM2M),XD(400),YD(400),ZD(400)
C
C  CALCULATE THE LENGTH OF THE CURVE
C
      ARC=0.0
      WRITE(3,6161)ID,NPS,ND
 6161 FORMAT(3X,I2,3X,I3,3X,I3)
      CALL
     BSPLINE(M,NUM,XV,YV,ZV,ID,NUM2M,ICLOS,ND,XD,YD,ZD,IFAULT,N,X)
      DO 500 I=1,ND-1
  500 ARC=ARC+SQRT((XD(I+1)-XD(I))**2+(ZD(I+1)-ZD(I))**2)
      ARCS=ARC/FLOAT(NPS-1)
      ARC1=0.0
      J=1
      I=IN
      LAST=ND
      XS(J)=XD(1)
      YS(J)=YD(1)
      ZS(J)=ZD(1)
  510 I=I+1
      IF(ICLOS.EQ.1.AND.I.EQ.ND)I=1
      ARCL=ARC1
      ARC1=ARC1+SQRT((XD(I+1)-XD(I))**2+(ZD(I+1)-ZD(I))**2)
      IF(ARC1.GE.(ARC-ARCS))GO TO 530
  520 IF(ARC1.LT.ARCS*J)GO TO 510
      IF((ARC1-ARCL).LT.0.000001)GO TO 510
      J=J+1
      XS(J)=XD(I)+((ARCL-ARCS*(J-1))/(ARCL-ARC1))*(XD(I+1)-XD(I))
      YS(J)=YD(I)+((ARCL-ARCS*(J-1))/(ARCL-ARC1))*(YD(I+1)-YD(I))
      ZS(J)=ZD(I)+((ARCL-ARCS*(J-1))/(ARCL-ARC1))*(ZD(I+1)-ZD(I))
      GO TO 520
  530 J=J+1
      XS(J)=XD(I)+((ARCL-ARCS*(J-1))/(ARCL-ARC1))*(XD(I+1)-XD(I))
      YS(J)=YD(I)+((ARCL-ARCS*(J-1))/(ARCL-ARC1))*(YD(I+1)-YD(I))
      ZS(J)=ZD(I)+((ARCL-ARCS*(J-1))/(ARCL-ARC1))*(ZD(I+1)-ZD(I))
      J=J+1
      IF(ICLOS.EQ.1)LAST=IN
      XS(J)=XD(LAST)
      YS(J)=YD(LAST)
      ZS(J)=ZD(LAST)
      NPS=J
      RETURN
      END
C
C
C  THIS SUBROUTINE GENERATES B-SPLINE BASIS KNOT VECTORS
C AND B-SPLINE CURVE OF VARIOUS ORDER TO APPROXIMATE A
C GIVEN POLYGON.
C
C
```

```
      SUBROUTINE BSPLINE(M,NUM,XV,YV,ZV,ID,NUM2M,
     &                   ICLOS,NP,XR,YR,ZR,IFAULT,N,X)
C     M: THE ORDER OF B-SPLINE BASIS
C     NUM: NUMBER OF POLYGON VERTICES
C     XV: X-ORDINATES OF DEFINING POLYGON VERTICES


C     YV: Y-ORDINATES OF DEFINING POLYGON VERTICES
C     ZV: Z-ORDINATES OF DEFINING POLYGON VERTICES
C     ID: CONTROL VARIABLE, ID=2 FOR 2D, ID=3FOR 3D CURVE
C     NUM2M: NUM+2*M INTEGER NUMBER FOR DEFINING VARIABLE
DIMENSIONS
C     ICLOS: CONTROL VARIABLE, ICLOS=1 FOR CLOSED POLYGON,
C                       ICLOS=0 FOR OPEN POLYGON.
C
C     NP: THE NUMBER OF POINTS GENERATED ALONG THE CURVE

C     XR,YR,ZR: ARRAYS CONTAINING COORDINATES OF THE GENERATED
POINTS
C     IFAULT: ERROR RETURN IF M>NUM
C
      DIMENSION XV(NUM2M),YV(NUM2M),ZV(NUM2M),X(NUM2M),
     &          XR(NP),YR(NP),ZR(NP)
      REAL N(NUM2M,M)
      INTEGER W
      L=0
      G=0.0
      H=0.0
      S=0.0
      IF(ICLOS.NE.1)GO TO 110
C     ADD M-1 EXTRA VERTICES TO THE DEFINING ARRAY OF VERTICES
C     IF CLOSED CURVE
C
      DO 100 J=1,M-1
      XV(NUM+J)=XV(J)
      YV(NUM+J)=YV(J)
  100 ZV(NUM+J)=ZV(J)
      NUM=NUM+M-1
  110 NUM1=NUM-1
      IF(M.GT.NUM)GO TO 260
      MAX=NUM1-M+2
C     FILL THE WEIGHING FUNCTIONS ARRAY WITH ZERO
C
      DO 120 I=1,NUM+1
      DO 120 K=1,M
  120 N(I,K)=0.0
      IX=NUM+M
      IF(ICLOS.NE.1)GO TO 130
C     DEFINE THE KNOT VECTORS
C
      CALL KNOTCL(M,MAX,XV,YV,ZV,NUM,ID,X,IX)
      GO TO 135
  130 CALL KNOTOP(M,MAX,XV,YV,ZV,NUM,ID,X,IX)
  135 STEP=(X(NUM+1)-X(M))/FLOAT(NP-1)
      W=M-1
C     INCREMENT KNOT VECTOR SUBSCRIPT
  140 W=W+1
      IF(X(W).EQ.X(W+1))GO TO 140
      N(W,1)=1.
      T=X(W)
  150 L=L+1
```

```
C  CALCULATE VALUES OF N(I,K) WEIGHING FUNCTIONS
   DO 210 K=2,M
   DO 200 I=1,NUM
   IF(N(I,K-1).NE.0.0)GO TO 160
   E=0.0
   GO TO 170
160  E=(T-X(I))*N(I,K-1)/(X(I+K-1)-X(I))


170  IF(N(I+1,K-1).NE.0.0)GO TO 180
   F=0.0
   GO TO 190
180  F=(X(I+K)-T)*N(I+1,K-1)/(X(I+K)-X(I+1))
190  N(I,K)=E+F
   G=XV(I)*N(I,K)+G
   H=YV(I)*N(I,K)+H
   IF(ID.EQ.2)GO TO 200
   S=ZV(I)*N(I,K)+S
200  CONTINUE
   IF(K.EQ.M)GO TO 220
   G=0.0
   H=0.0
   S=0.0
210  CONTINUE
220  XR(L)=G
   YR(L)=H
   IF(ID.EQ.3)ZR(L)=S
C  INCREMENT PARAMETER T
C
   T=T+STEP
   IF(L.EQ.(NP-1))GO TO 240
   IF(T.GE.X(W).AND.T.LT.X(W+1))GO TO 150
   N(W,1)=0.0
230  W=W+1
   IF(X(W).EQ.X(W+1))GO TO 230
   N(W,1)=1.0
   GO TO 150
C DEFINE THE LAST POINT ON THE CURVE
C
240  L=L+1
   IF(ICLOS.EQ.1)GO TO 250
   XR(L)=XV(NUM)
   YR(L)=YV(NUM)
   IF(ID.EQ.3)ZR(L)=ZV(NUM)
   NP=L
   RETURN
250  XR(L)=XR(1)
   YR(L)=YR(1)
   IF(ID.EQ.3)ZR(L)=ZR(1)
   NUM=NUM-M+1
   NP=L
   RETURN
260  IFAULT=1
   RETURN
   END
C  SUBROUTINE TO GENERATE BASIS KNOT VECTOR FOR
C  OPEN POLYGON
C
   SUBROUTINE KNOTOP(M,MAX,XV,YV,ZV,NUM,ID,X,IX)
   DIMENSION XV(NUM),YV(NUM),ZV(NUM),X(IX)
```

```
      DO 50 I=1,IX
      IF(I.GT.M)GO TO 10
C     ASSIGN MULTIPLE KNOTS FOR THE BEGINNING OF SPAN
      X(I)=0.0
      GO TO 50
10    IF(I.LT.(MAX+M+1)) GO TO 30
C     ASSIGN MULTIPLE KNOTS

C
20    X(I)=X(I-1)
      GO TO 50
30    IF(ID.EQ.3)GO TO 40
C     CHECK FOR MULTIPLE KNOTS
C
      IF(XV(I-M).EQ.XV(I-M+1).AND.
     &  YV(I-M).EQ.YV(I-M+1))GO TO 20
      X(I)=X(I-1)+1.
      GO TO 50
40    IF(XV(I-M).EQ.XV(I-M+1).AND.
     &  YV(I-M).EQ.YV(I-M+1).AND.
     &  ZV(I-M).EQ.ZV(I-M+1))GO TO 20
      X(I)=X(I-1)+1.
50    CONTINUE
      RETURN
      END
C
C     SUBROUTINE TO GENERATE BASIS KNOT VECTORS FOR
C    CLOSED POLYGON
C
      SUBROUTINE KNOTCL(M,MAX,XV,YV,ZV,NUM,ID,X,IX)
      DIMENSION XV(NUM),YV(NUM),ZV(NUM),X(IX)
C     FIRST KNOT VECTOR
C
      X(1)=FLOAT(1-M)
      DO 120 I=2,IX
      IF(I.GT.M)GO TO 40
      IF(ID.EQ.3)GO TO 20
C     CHECK FOR REPEATING VERTICES
C
      IF(XV(I+MAX-M).EQ.XV(I+MAX-M+1).AND.
     &  YV(I+MAX-M).EQ.YV(I+MAX-M+1))GO TO 10
      X(I)=X(I-1)+1.
      GO TO 120
10    X(I)=X(I-1)
      GO TO 120
20    IF(XV(I+MAX-M).EQ.XV(I+MAX-M+1).AND.
     &  YV(I+MAX-M).EQ.YV(I+MAX-M+1).AND.
     &  ZV(I+MAX-M).EQ.ZV(I+MAX-M+1)) GO TO 30
      X(I)=X(I-1)+1.
      GO TO 120
30    X(I)=X(I-1)
      GO TO 120
40    IF(I.LT.(MAX+M+1))GO TO 80
      IF(ID.EQ.3)GO TO 60
      IF(XV(I-MAX-M).EQ.XV(I-MAX-M+1).AND.
     &  YV(I-MAX-M).EQ.YV(I-MAX-M+1))GO TO 50
      X(I)=X(I-1)+1.
      GO TO 120
50    X(I)=X(I-1)
      GO TO 120
```

```
60  IF(XV(I-MAX-M).EQ.XV(I-MAX-M+1).AND.
   &  YV(I-MAX-M).EQ.YV(I-MAX-M+1).AND.
   &  ZV(I-MAX-M).EQ.ZV(I-MAX-M+1))GO TO 70
    X(I)=X(I-1)+1.
    GO TO 120
70  X(I)=X(I-1)


    GO TO 120
80  IF(ID.EQ.3)GO TO 100
    IF(XV(I-M).EQ.XV(I-M+1).AND.
   &  YV(I-M).EQ.YV(I-M+1))GO TO 90
    X(I)=X(I-1)+1.
    GO TO 120
90  X(I)=X(I-1)
    GO TO 120
100 IF(XV(I-M).EQ.XV(I-M+1).AND.
   &  YV(I-M).EQ.YV(I-M+1).AND.
   &  ZV(I-M).EQ.ZV(I-M+1))GO TO 110
    X(I)=X(I-1)+1.
    GO TO 120
110 X(I)=X(I-1)
120 CONTINUE
    RETURN
    END

C
C  THIS SUBROUTINE SELECTS 'NPS' POINTS ALONG A GIVEN
C CARDINAL CUBIC SPLINE CURVE AT EQUAL INTERVALS.
C
C
    SUBROUTINE EQL(NUM,XP,YP,ZP,ID,ICLOS,NPS,ND,XS,YS,ZS,NUM1)
    DIMENSION XP(NUM1),YP(NUM1),ZP(NUM1),XS(NPS),YS(NPS),ZS(NPS),
   &       XD(400),YD(400),ZD(400)
    ARC=0.0
    CALL CSPLINE(NUM,XP,YP,ZP,ID,ICLOS,ND,XD,YD,ZD,NUM1)
    DO 500 I=1,ND-1
500 ARC=ARC+SQRT((XD(I+1)-XD(I))**2+(ZD(I+1)-ZD(I))**2)
    ARCS=ARC/FLOAT(NPS-1)
    ARC1=0.0
    J=1
    I=0
    XS(J)=XD(1)
    YS(J)=YD(1)
    ZS(J)=ZD(1)
510 I=I+1
    ARCL=ARC1
    ARC1=ARC1+SQRT((XD(I+1)-XD(I))**2+(ZD(I+1)-ZD(I))**2)
    IF(ARC1.GE.(ARC-ARCS))GO TO 530
520 IF(ARC1.LT.ARCS*J)GO TO 510
    IF((ARC1-ARCL).LT.0.000001)GO TO 510
    J=J+1
    XS(J)=XD(I)+((ARCL-ARCS*(J-1))/(ARCL-ARC1))*(XD(I+1)-XD(I))
    YS(J)=YD(I)+((ARCL-ARCS*(J-1))/(ARCL-ARC1))*(YD(I+1)-YD(I))
    ZS(J)=ZD(I)+((ARCL-ARCS*(J-1))/(ARCL-ARC1))*(ZD(I+1)-ZD(I))
    GO TO 520
530 J=J+1
    XS(J)=XD(I)+((ARCL-ARCS*(J-1))/(ARCL-ARC1))*(XD(I+1)-XD(I))
    YS(J)=YD(I)+((ARCL-ARCS*(J-1))/(ARCL-ARC1))*(YD(I+1)-YD(I))
    ZS(J)=ZD(I)+((ARCL-ARCS*(J-1))/(ARCL-ARC1))*(ZD(I+1)-ZD(I))
```

```
      J=J+1
      XS(J)=XD(ND)
      YS(J)=YD(ND)
      ZS(J)=ZD(ND)
      NPS=J
      RETURN

      END
C     THIS SUBROUTINE FITS A CUBIC CARDINAL SPLINE TO
C     A SET OF GIVEN DATA.
C
      SUBROUTINE CSPLINE(NUM,XP,YP,ZP,ID,ICLOS,NP,XR,YR,ZR,NUM1)
C     NUM: NUMBER OF INPUT DATA
C     XP,YP,ZP:ARRAYS CONTAINING COORDINATES OF INPUT POINTS
C     ID: CONTROL VARIABLE, ID=2 FOR 2D, ID=3 FOR 3D CURVE
C     ICLOS: CONTROL VARIABLE, ICLOS=0  OPEN CURVE,
C                              ICLOS=1  CLOSED CURVE.
C     NP: NUMBER OF GENERATED POINTS ALONG THE CURVE
C     XR,YR,ZR: ARRAYS CONTAINING COORDINATES OF GENERATED POINTS
C
      DIMENSION XP(NUM1),YP(NUM1),ZP(NUM1),XR(NP),YR(NP),ZR(NP)
      INTEGER PWR
      IF(ICLOS.EQ.1)GO TO 80
      XP(1)=XP(2)
      YP(1)=YP(2)
      ZP(1)=ZP(2)
      XP(NUM+2)=XP(NUM+1)
      YP(NUM+2)=YP(NUM+1)
      ZP(NUM+2)=ZP(NUM+1)
      DELT=FLOAT(NUM-1)/FLOAT(NP-1)
      NUM=NUM+2
      GO TO 90
   80 XP(1)=XP(NUM+1)
      YP(1)=YP(NUM+1)
      ZP(1)=ZP(NUM+1)
      XP(NUM+2)=XP(2)
      YP(NUM+2)=YP(2)
      ZP(NUM+2)=ZP(2)
      XP(NUM+3)=XP(3)
      YP(NUM+3)=YP(3)
      ZP(NUM+3)=ZP(3)
      DELT=FLOAT(NUM)/FLOAT(NP-1)
      NUM=NUM+3
   90 A=-2.+3**(1./2.)
C     CALCULATE ZIGN=
C
      ZIGN1=0.0
      ZIGN2=0.0
      ZIGN3=0.0
      PWR=NUM-2
      DO 100 I=1,NUM-1
      ZIGN1=ZIGN1+3*XP(I)*(A**PWR)
      ZIGN2=ZIGN2+3*YP(I)*(A**PWR)
      IF(ID.EQ.2)GO TO 100
      ZIGN3=ZIGN3+3*ZP(I)*(A**PWR)
  100 PWR=PWR-1
C     CALCULATE TA1=
C
      TA11=0.0
      TA12=0.0
```

```
      TA13=0.0
      PWR=0
      DO 110 I=2,NUM
      TA11=TA11+3*XP(I)*(A**PWR)
      TA12=TA12+3*YP(I)*(A**PWR)

      IF(ID.EQ.2)GO TO 110
      TA13=TA13+3*ZP(I)*(A**PWR)
  110 PWR=PWR+1
C     CALCULATE CARDINAL SPLINE VALUES STARTING FROM
C   THE FIRST INTERVAL IN STEPS OF DELT=NUM-3/NP-1
C
      T=2.0
      K=2
      J=0
  106 J=J+1
C   CALCULATE ZIGK=
      ZIGK1=0.0
      ZIGK2=0.0
      ZIGK3=0.0
      PWR=K-1
      DO 120 I=1,K
      ZIGK1=ZIGK1+3*XP(I)*(A**PWR)
      ZIGK2=ZIGK2+3*YP(I)*(A**PWR)
      IF(ID.EQ.2)GO TO 120
      ZIGK3=ZIGK3+3*ZP(I)*(A**PWR)
  120 PWR=PWR-1
C     CALCULATE TAK=
C
      TAK1=0.0
      TAK2=0.0
      TAK3=0.0
      PWR=0
      DO 130 I=K+1,NUM
      TAK1=TAK1+3*XP(I)*(A**PWR)
      TAK2=TAK2+3*YP(I)*(A**PWR)
      IF(ID.EQ.2)GO TO 130
      TAK3=TAK3+3*ZP(I)*(A**PWR)
  130 PWR=PWR+1
  140 B=(1-A**(NUM-1))
      C=(T-K)-(A+2)*((T-K)**2)+(A+1)*((T-K)**3)
      D=(1-T+K)-(A+2)*((1-T+K)**2)+(A+1)*((1-T+K)**3)
      XR(J)=(C/B)*(ZIGN1*(A**K)+ZIGK1*B)+((1-T+K)**3)*XP(K)
     &     +(D/B)*(TA11*(A**(NUM-K))+TAK1*B)+((T-K)**3)*XP(K+1)
      YR(J)=(C/B)*(ZIGN2*(A**K)+ZIGK2*B)+((1-T+K)**3)*YP(K)
     &     +(D/B)*(TA12*(A**(NUM-K))+TAK2*B)+((T-K)**3)*YP(K+1)
      IF(ID.EQ.2)GO TO 145
      ZR(J)=(C/B)*(ZIGN3*(A**K)+ZIGK3*B)+((1-T+K)**3)*ZP(K)
     &     +(D/B)*(TA13*(A**(NUM-K))+TAK3*B)+((T-K)**3)*ZP(K+1)
C     INCREMENT T
C
  145 T=T+DELT
      IF(T.GT.(NUM-1-DELT))GO TO 160
      IF(T.GE.K.AND.T.LT.(K+1))GO TO 150
      K=K+1
      GO TO 106
  150 J=J+1
      GO TO 140
  160 J=J+1
      XR(J)=XP(NUM-1)
```

```
YR(J)=YP(NUM-1)
IF(ID.EQ.3)ZR(J)=ZP(NUM-1)
NP=J
NUM=NUM-2
IF(ICLOS.EQ.1)NUM=NUM-1
RETURN
END
```

PROGRAM S_7

```
C  SURFACE COORDINATE DATA INPUT FROM CHANNEL 5 "*CR2"
C  CONTROL VARIABLE DATA INPUT FROM CHANNEL 1 "*CR0"
C
C THIS PROGRAMME PRODUCES AXONOMETRIC PROJECTION OF THE
SURFACE USING
C THE DATA POINTS ON THE SURFACE VIEWED FROM XVU,YVU,ZVU.THE
SCALE IS
C SET BY XSCL,YSCL,ZSCL.
C
    DIMENSION XF(200,200),YF(200,200),ZF(200,200),

   &        X1(200),Y1(200),Z1(200)

C
C  NPU :NO.OF SELECTED POINTS ON THE CROSS SECTIONS
"PARAMETER U"
C  NPV := =    =      =   =  = INTERPOLATED POINTS "PARAMETER V"
C  IDRCTN :CONTROL VARIABLE (IDRCTN=1 PLOTS LINES ALONG
PARAMETER V)
C
C  DIMENSION FOR XF,YF,ZF :(NPU,NPV)
C            X1,Y1,Z1 :MAX(NPU,NPV)
C
     COMMON/BL1/XVU,YVU,ZVU
     COMMON/BL2/XSCL,YSCL,ZSCL
     COMMON/BL3/IDRCTN,NVE
     COMMON/BL4/NPU,NPV
     COMMON/BL5/XSHIFT,SHIFT
     OPEN(UNIT=5,NAME='LLVG4',STATUS='UNKNOWN')
     OPEN(UNIT=2,NAME='OVIEW65W',STATUS='UNKNOWN')
C
    PRINT*,'   '
    PRINT*,'   '
    PRINT*,'   '
    PRINT*,'   '


C
    PRINT 201
    PRINT 202
    PRINT 203
    PRINT 204
    PRINT 205
    PRINT 206
    PRINT 207
    PRINT 211
    PRINT 208
    PRINT 202
    PRINT 201
C
    READ(*,209)IT
C
 201 FORMAT('        ************************************')
 202 FORMAT('        **                                **')
```

```
203 FORMAT('        **      TERMINAL DEFINITION     **')
204 FORMAT('        **      PLEASE INPUT 1,2,3,4    **')
205 FORMAT('        **      (1) FOR T4010           **')
206 FORMAT('        **      (2) FOR T4107           **')
207 FORMAT('        **      (3) FOR T4113           **')
211 FORMAT('        **      (4) FOR T4105           **')
208 FORMAT('        **      (5) FOR VT125           **')
209 FORMAT(I2)
C
100 WRITE(*,631)
631 FORMAT(3X'ENTER NPU,NPV')
   READ*,NPU,NPV
   IF(NPU.EQ.0)GO TO 1130
   READ(5,*)((XF(I,J),YF(I,J),ZF(I,J),J=1,NPV),I=1,NPU)
   WRITE(2,97)
110  WRITE(*,50)
50  FORMAT(3X' ENTER XSCL,YSCL,ZSCL')
   READ*,XSCL,YSCL,ZSCL
   WRITE(*,51)
51  FORMAT(3X'ENTER XVU,YVU,ZVU')


READ*,XVU,YVU,ZVU
   WRITE(*,52)
52  FORMAT(3X'ENTER IDRCTN')
   READ*,IDRCTN
   IF(XSCL.EQ.0.0)GO TO 100
   WRITE(2,96)XSCL,YSCL,ZSCL
   WRITE(2,95)XVU,YVU,ZVU
   WRITE(*,731)
731 FORMAT(3X,'ENTER NVE')
   READ*,NVE
   WRITE(*,56)
56  FORMAT(3X'ENTER(XSHIFT,SHIFT)')
   READ*,XSHIFT,SHIFT
   CALL GINO
   IF(IT.EQ.1)CALL T4010
   IF(IT.EQ.2)CALL T4107
   IF(IT.EQ.3)CALL T4113
   IF(IT.EQ.4)CALL T4105
   IF(IT.EQ.5)CALL VT125
   PRINT*,'  '
   PRINT*,'  '
   PRINT*,'  '
   PRINT*,'  '
   IF(IT.EQ.3)CALL UNITS(0.75)
   IF(IT.EQ.1)CALL UNITS(0.5)
   IF(IT.EQ.2)CALL UNITS(0.5)
   IF(IT.EQ.4)CALL UNITS(0.5)
   IF(IT.EQ.5)CALL UNITS(0.45)
C
   CALL PICCLE
   CALL FRAME
   CALL SHIFT2(XSHIFT,SHIFT)
   CALL LINCOL(5)
   CALL AXON3(XVU,YVU,ZVU)
   CALL SCALE3(XSCL,YSCL,ZSCL)
   CALL MOVTO3(XF(1,1),YF(1,1),ZF(1,1))
```

```
      DO 1109 J=1,NPV
      K=MOD(J,2)


      IF(K.EQ.0)GO TO 1111
      DO 1108 I=1,NPU
      X1(I)=XF(I,J)
      Y1(I)=YF(I,J)
1108  Z1(I)=ZF(I,J)
      CALL POLTO3(X1,Y1,Z1,NPU)
      IF(J.NE.NPV)CALL LINTO3(XF(NPU,J+1),YF(NPU,J+1),ZF(NPU,J+1))
      GO TO 1109
1111  DO 1112 I=1,NPU
      K=NPU-I+1
      X1(I)=XF(K,J)
      Y1(I)=YF(K,J)
1112  Z1(I)=ZF(K,J)
      CALL POLTO3(X1,Y1,Z1,NPU)
      IF(J.NE.NPV)CALL LINTO3(XF(1,J+1),YF(1,J+1),ZF(1,J+1))
1109  CONTINUE
      IF(IDRCTN.NE.1)GO TO 1125
      CALL LINCOL(6)
      DO 1125 I=1,NPU
      DO 1120 J=1,NPV
      X1(J)=XF(I,J)
      Y1(J)=YF(I,J)
1120  Z1(J)=ZF(I,J)
      CALL MOVTO3(X1(1),Y1(1),Z1(1))
      CALL POLTO3(X1,Y1,Z1,NPV)
1125  CONTINUE
      READ*,IR
      IF(IR.EQ.3)CALL DEVEND
      GO TO 110
1130  CONTINUE
      STOP
97    FORMAT(1H1)
96    FORMAT(///////' X SCALING FACTOR IN SURFACE DRAWING=',F9.2,
     &          /' Y SCALING FACTOR IN SURFACE DRAWING=',F9.2,
     &          /' Z SCALING FACTOR =   =    =   =',F9.2)
95    FORMAT(//' X VIEWPOINT IN AXONOMETRIC PROJECTION=',F9.2,
     &      /' Y VIEWPOINT =    =         =   =',F9.2,


     &      /' Z VIEWPOINT =    =         =   =',F9.2)
      END
      SUBROUTINE FRAME
      COMMON/BL2/XSCL,YSCL,ZSCL
      COMMON/BL1/XVU,YVU,ZVU
      COMMON/BL3/IDRCTN,NVE
      COMMON/BL4/NPU,NPV
      COMMON/BL5/XSHIFT,SHIFT
      CALL LINCOL(5)
      CALL MOVTO2(0.0,0.0)
      CALL LINBY2(475.0,0.0)
      CALL LINBY2(0.0,350.0)
      CALL LINBY2(-475.0,0.0)
      CALL LINBY2(0.0,-350.0)
      CALL MOVTO2(0.0,0.0)
```

```
      CALL LINBY2(0.0,30.0)
      CALL LINBY2(475.0,0.0)
      CALL MOVTO2(0.0,0.0)
      CALL MOVTO2(60.0,15.0)
      CALL CHASIZ(5.0,5.0)
      CALL CHASTR('TOP&BOTTOM MOULD FOR THE PLASTIC HANDLE
     VIEWED')
      CALL CHASTR(' FROM DIFFERENT ANGLES.')
      CALL MOVTO2(0.0,0.0)
      CALL LINBY2(0.0,100.0)
      CALL LINBY2(475.0,0.0)
      CALL LINBY2(-163.0,0.0)
      CALL LINBY2(0.0,-70.0)
      CALL LINBY2(-158.0,0.0)
      CALL LINBY2(0.0,70.0)
      CALL MOVTO2(0.0,0.0)
      CALL MOVTO2(0.0,75.0)
      CALL LINBY2(475.0,0.0)
      CALL MOVTO2(0.0,0.0)
      CALL MOVTO2(10.0,90.0)
      CALL CHAHOL(18HMODEL NUMBER = 1*.)
      CALL AXONOM
      CALL XNAME


      CALL YNAME
      CALL ZNAME
      CALL TROL
      CALL AXONOM1
      CALL AXONOM2
      CALL NSECT
      CALL NSECT1
      CALL NVEIW
      CALL ASHIFT
      CALL DIMEN
      RETURN
      END
C
C
C
      SUBROUTINE XNAME
C
C
      COMMON/BL2/XSCL,YSCL,ZSCL
      CALL MOVTO2(0.0,0.0)
      CALL MOVTO2(170.0,65.0)
      CALL CHASIZ(5.0,5.0)
      CALL CHAHOL(20HX-SCALING FACTOR =*.)
      CALL CHAFIX(XSCL,5,1)
      RETURN
      END
C
C
C
C
      SUBROUTINE YNAME
C
C
```

```
      COMMON/BL2/XSCL,YSCL,ZSCL
      CALL MOVTO2(0.0,0.0)
      CALL MOVTO2(170.0,55.0)
      CALL CHASIZ(5.0,5.0)
      CALL CHAHOL(20HY-SCALING FACTOR =*.)


      CALL CHAFIX(YSCL,5,1)
      RETURN
      END
C
C
C
      SUBROUTINE ZNAME
C
C
      COMMON/BL2/XSCL,YSCL,ZSCL
      CALL MOVTO2(0.0,0.0)
      CALL MOVTO2(170.0,45.0)
      CALL CHASIZ(5.0,5.0)
      CALL CHAHOL(20HZ-SCALING FACTOR =*.)
      CALL CHAFIX(ZSCL,5,1)
      RETURN
      END
C
C
C
      SUBROUTINE NVEIW
C
C
      COMMON/BL3/IDRCTN,NVE
      CALL MOVTO2(0.0,0.0)
      CALL MOVTO2(10.0,80.0)
      CALL CHASIZ(5.0,5.0)
      CALL CHAHOL(15HVIEW NUMBER =*.)
      CALL CHAINT(NVE,1)
      RETURN
      END

      SUBROUTINE TROL
C
C
      COMMON/BL3/IDRCTN,NVE
      CALL MOVTO2(0.0,0.0)
      CALL MOVTO2(327.0,68.0)


      CALL CHASIZ(5.0,5.0)
      CALL CHAHOL(20HCONTROL VARIABLE =*.)
      CALL CHAINT(IDRCTN,1)
      RETURN
      END
C
C
      SUBROUTINE AXONOM
C
C
      COMMON/BL1/XVU,YVU,ZVU
```

```
      CALL MOVTO2(0.0,0.0)
      CALL MOVTO2(10.0,68.0)
      CALL CHASIZ(5.0,5.0)
      CALL CHAHOL(18HTHE VIEWPOINT IN*.)
      CALL MOVTO2(10.0,60.0)
      CALL CHAHOL(28HAXONOMETRIC PROJECTION IS:*.)
      CALL MOVTO2(10.0,51.0)
      CALL CHAHOL(17HX-COORDINATE = *.)
      CALL CHAFIX(XVU,10,2)
      RETURN
      END
C
C
C
      SUBROUTINE AXONOM1
C
C
      COMMON/BL1/XVU,YVU,ZVU
      CALL MOVTO2(0.0,0.0)
      CALL MOVTO2(10.0,42.0)
      CALL CHASIZ(5.0,5.0)
      CALL CHAHOL(17HY-COORDINATE = *.)
      CALL CHAFIX(YVU,10,2)
      RETURN
      END
C
C
      SUBROUTINE AXONOM2
C
C
      COMMON/BL1/XVU,YVU,ZVU
      CALL MOVTO2(0.0,0.0)
      CALL MOVTO2(10.0,35.0)
      CALL CHASIZ(5.0,5.0)
      CALL CHAHOL(17HZ-COORDINATE = *.)
      CALL CHAFIX(ZVU,10,2)
      RETURN
      END
C
C
C
C
      SUBROUTINE NSECT
      COMMON/BL4/NPU,NPV
      CALL MOVTO2(0.0,0.0)
      CALL MOVTO2(322.0,55.0)
      CALL CHASIZ(5.0,5.0)
      CALL CHAHOL(30HNO.OF POINTS ON PARAMETER U=*.)
      CALL CHAINT(NPU,2)
      RETURN
      END
C
```

```
C
C
      SUBROUTINE NSECT1
C
C
      COMMON/BL4/NPU,NPV
      CALL MOVTO2(0.0,0.0)
      CALL MOVTO2(322.0,46.0)
      CALL CHASIZ(5.0,5.0)


      CALL CHAHOL(30HNO.OF POINTS ON PARAMETER V=*.)
      CALL CHAINT(NPV,2)
      RETURN
      END
C
C
C
C
      SUBROUTINE ASHIFT
C
C
      COMMON/BL5/XSHIFT,SHIFT
C
C
      CALL MOVTO2(0.0,0.0)
      CALL MOVTO2(325.0,90.0)
      CALL CHASIZ(5.0,5.0)
      CALL CHAHOL(22HSHIFTING IN X-DIR. =*.)
      CALL CHAFIX(XSHIFT,10,2)
      CALL MOVTO2(325.0,80.0)
      CALL CHAHOL(22HSHIFTING IN Y-DIR. =*.)
      CALL CHAFIX(SHIFT,10,2)
      RETURN
      END
      SUBROUTINE  DIMEN
      CALL MOVTO2(0.0,0.0)
      CALL MOVTO2(170.0,90.0)
      CALL CHASIZ(5.0,5.0)
      CALL CHAHOL(27HLENGTH OF THE HANDLE =140*.)
      CALL MOVTO2(0.0,0.0)
      CALL MOVTO2(170.0,80.0)
      CALL CHAHOL(25HHIGHT OF THE HANDLE =30*.)
      RETURN
      END
```

```
C
C   THIS PROGRAMME BY CALLING SUBROUTINE 'REPART' FIRST FITS
C  B-SPLINE CURVES TO CROSS SECTIONS OF A 3-D OBJECT,THEN BY
C  CALLING SUBROUTINE 'EQLT' INTERPOLATES SELECTED POINTS ON
C  THE B-SPLINE CURVES USING CARDINAL CUBIC SPLINE. FINALLY IT
C  CACULATES DIRECTIN COSINES OF THE NORMALS TO THE SURFACE.
C
      DIMENSION X1(300),Y1(300),Z1(300),
     &          XX(50,20),YY(50,20),ZZ(50,20),
     &          XV(50),YV(50),ZV(50),XF(50,50),YF(50,50),ZF(50,50),
     &          XT1(300),YT1(300),ZT1(300),
     &          XXT(50,50),YYT(50,50),ZZT(50,50),
     &          XFN(50,50),YFN(50,50),ZFN(50,50),
     &          NPT(50),NPT2(50,50)
      REAL N(50,10),X(50)
C   NPU: NO. OF SELECTED POINTS OVER PARAMETER "U". <CROSS
SECTIONS.>
C   NPV: NO. OF SELECTED POINTS OVER PARAMETER "V".
C   NS: NO. OF CROSS SECTIONS.
C   NUM: NO. OF VERTICES.
C   M: THE ORDER OF THE B-SPLINE CURVE.
C
C
C   DIMENSION FOR X1,Y1,Z1: MAX(NPU,NPV,ND)
C             XV,YV,ZV: MAX(MAX(NUM),NS+3)
C             XX,YY,ZZ: (MAX(NPU,NPV),NS+3)
C             XF,YF,ZF: (NPU,NPV)
C             XT1,YT1,ZT1: MAX(MAX(NUM),NS+3,ND)
C             XXT,YYT,ZZT: (MAX(NPU,NPV),NS+3)
C             XFN,YFN,ZFN: (NPU,NPV)
C             N: (MAX(NUM)+2*MAX(M),MAX(M))
C             NPT: (NPV)
C             NPT2: (NPU,NPV)
      OPEN(UNIT=1,NAME='OBCSRF1',STATUS='UNKNOWN')
      OPEN(UNIT=5,NAME='OINVERT',STATUS='UNKNOWN')
      OPEN(UNIT=2,NAME='OBCSURN30',STATUS='UNKNOWN')
      OPEN(UNIT=6,NAME='OBCSURN31',STATUS='UNKNOWN')
      OPEN(UNIT=3,NAME='IFAU',STATUS='UNKNOWN')
      ND=200
1000  READ(1,*)NS,NPU,NPV
      IF(NS.EQ.0)GO TO 1120
      READ(1,*)JCLOS,ISPR
        PIE=3.1415926
C   READ DATA TO APPROXIMATE CROSS SECTIONS
      DO 1070 J=1,NS
      READ(5,*)NUM,M
      READ(5,*)ID,ICLOS
      READ(1,*)IN
      READ(1,*)XOR,YOR,ZOR,BETA,GAMA
      BETA=BETA/180*PIE
      GAMA=GAMA/180*PIE
      DO 1050 I=1,NUM
      IF(ID.EQ.2)GO TO 1046
      READ(5,*)XV(I),YV(I),ZV(I)
```

```
      GO TO 1047
1046  READ(5,*)XV(I),YV(I)
      ZV(I)=0.0
1047  SX=XOR+(YV(I)*SIN(BETA)+ZV(I)*COS(BETA))*SIN(GAMA)
     &     +XV(I)*COS(GAMA)
      SY=YOR+YV(I)*COS(BETA)-ZV(I)*SIN(BETA)
      SZ=ZOR+(YV(I)*SIN(BETA)+ZV(I)*COS(BETA))*COS(GAMA)
     &     -XV(I)*SIN(GAMA)
      XV(I)=SX
      YV(I)=SY
      ZV(I)=SZ
1050  CONTINUE
      NUM2M=NUM+2*M
      IFAULT=0
C  CALL SUBROUTINE REPART TO APPROXIMATE THE CROSS SECTION AND
C  SELECT "NPU" POINTS AT EQUAL INTERVALS.
      CALL REPART(M,NUM,XV,YV,ZV,IN,3,NUM2M,ICLOS,
     &          NPU,ND,X1,Y1,Z1,XT1,YT1,ZT1,IFAULT,N,X)
C  STORE THE SELECTED POINTS.
      WRITE(3,12121)IFAULT
12121 FORMAT(1X,I3)
      DO 1060 I=1,NPU
      XX(I,J+1)=X1(I)
      YY(I,J+1)=Y1(I)
      ZZ(I,J+1)=Z1(I)
      XXT(I,J+1)=XT1(I)
      YYT(I,J+1)=YT1(I)
      ZZT(I,J+1)=ZT1(I)
1060  CONTINUE
1070  CONTINUE
C  INTERPOLATE THE SELECTED POINTS.
1066  DO 1100 I=1,NPU
      DO 1080 J1=2,NS+1
      XV(J1)=XX(I,J1)
      YV(J1)=YY(I,J1)
      ZV(J1)=ZZ(I,J1)
1080  CONTINUE
      NS1=NS+3
C  CALLSUBROUTINE EQLT TO INTERPOLATE ONE SET OF POINTS AND
SELECT
C  "NPV" POINTS AT EQUAL INTERVALS OVER THE CURVE.
      CALL EQLT(NS,XV,YV,ZV,3,JCLOS,NPV,ND,X1,Y1,Z1,
     &          XT1,YT1,ZT1,NS1,NPT)
      DO 1090 J=1,NPV
      XF(I,J)=X1(J)
      YF(I,J)=Y1(J)
      ZF(I,J)=Z1(J)
      XFN(I,J)=XT1(J)
      YFN(I,J)=YT1(J)
      ZFN(I,J)=ZT1(J)
      NPT2(I,J)=NPT(J)
1090  CONTINUE
1100  CONTINUE
      DO 1104 I=1,NPU
      DO 1102 J1=2,NS+1
      XV(J1)=XXT(I,J1)
      YV(J1)=YYT(I,J1)
      ZV(J1)=ZZT(I,J1)
```

```
1102 CONTINUE
C
C
   CALL CSPLINE(NS,XV,YV,ZV,3,JCLOS,ND,X1,Y1,Z1,XT1,YT1,ZT1,NS1)
   DO 1103 J=1,NPV
C  SELECT THE POINTS WITH EQUAL VALUE OF PARAMETER 'V' AS BEFOR
C
   X1(J)=X1(NPT2(I,J))
   Y1(J)=Y1(NPT2(I,J))
   Z1(J)=Z1(NPT2(I,J))
C
C  CALCULATE NORMALS AT THE GENERATED POINTS ON THE SURFACE
C
   XFNS=Y1(J)*ZFN(I,J)-Z1(J)*YFN(I,J)
   YFNS=Z1(J)*XFN(I,J)-X1(J)*ZFN(I,J)
   ZFNS=X1(J)*YFN(I,J)-Y1(J)*XFN(I,J)
C  SAVE THE ELEMENTS OF DIRECTION COSINES.
C
   SUM=SQRT(XFNS**2+YFNS**2+ZFNS**2)
   XFNS=XFNS/SUM
   YFNS=YFNS/SUM
   ZFNS=ZFNS/SUM
   XFN(I,J)=XFNS
   YFN(I,J)=YFNS
   ZFN(I,J)=ZFNS
1103 CONTINUE
1104 CONTINUE
   WRITE(6,992)NPU,NPV
   DO 1105 I=1,NPU
   DO 1105 J=1,NPV
   WRITE(6,969)XF(I,J),YF(I,J),ZF(I,J),XFN(I,J),YFN(I,J),ZFN(I,J)
1105 CONTINUE
   IF(ISPR.NE.1)GO TO 1000
   GO TO 1000
1120 WRITE(6,972)
   STOP
992  FORMAT(1X,I3,1X,I3)
972  FORMAT(' 0 0')
969  FORMAT(1X,6(1X,F9.4))
   END
C
C  THIS SUBROUTINE SELECTS NPS POINTS ALONG A GIVEN
C B-SPLINE CURVE AT EQUAL INTERVALS.
C  THE VALUE FOR PARAMETER'IN' WHICH REFERS TO THE
C DESIRED STARTING POINT ON THE CURVE IS DETERMINED
C USING THE LISTING OF GENERATED POINTS.
C
C
   SUBROUTINE REPART(M,NUM,XV,YV,ZV,IN,ID,NUM2M,ICLOS,
   &          NPS,ND,XS,YS,ZS,XTS,YTS,ZTS,IFAULT,N,X)
C
C  XS,YS,ZS: ARRAYS CONTAINING COORDINATES OF SELECTED POINTS
C  XD,YD,ZD: ARRAYS CONTAINING COORDINATES OF ND GENERATED
POINTS
C       ALONG THE CURVE.
C  XTS,YTS,ZTS: ARRAYS CONTAINING COORDINATES OF VECTORS OF
C         TANGENTS AT SELECTED POINTS
C
C  NPS: NO. OF SELECTED POINTS.
```

```
C  ND: NO. OF GENERATED POINTS TO CALCULATE ARC LENGTH
C
      DIMENSION
XV(NUM2M),YV(NUM2M),ZV(NUM2M),XS(NPS),YS(NPS),ZS(NPS),
     &          XTS(NPS),YTS(NPS),ZTS(NPS)
      REAL N(NUM2M,M),X(NUM2M),XD(300),YD(300),ZD(300),
     &                    XT(300),YT(300),ZT(300)

C
C  CALCULATE THE LENGTH OF THE CURVE
C
      ARC=0.0
      CALL
BSPLINE(M,NUM,XV,YV,ZV,ID,NUM2M,ICLOS,ND,XD,YD,ZD,IFAULT,N,X,
     &           XT,YT,ZT)
      DO 500 I=1,ND-1
500   ARC=ARC+SQRT((XD(I+1)-XD(I))**2+(ZD(I+1)-ZD(I))**2)
      ARCS=ARC/FLOAT(NPS-1)
      J=0
      I=IN-1
      ARC1=0.0
      IF(ICLOS.EQ.1)GO TO 560
510   J=J+1
      I=I+1
      IF(I.EQ.ND)GO TO 530
      XS(J)=XD(I)
      YS(J)=YD(I)
      ZS(J)=ZD(I)
      XTS(J)=XT(I)
      YTS(J)=YT(I)
      ZTS(J)=ZT(I)
520   ARCD=SQRT((XD(I+1)-XD(I))**2+(ZD(I+1)-ZD(I))**2)
      ARC1=ARC1+ARCD
      IF(ARC1.GE.(ARC-ARCS))GO TO 540
      IF(ARC1.GE.ARCS*J)GO TO 510
      I=I+1
      GO TO 520
530   J=J+1
      XS(J)=XD(ND)
      YS(J)=YD(ND)
      ZS(J)=ZD(ND)
      XTS(J)=XT(ND)
      YTS(J)=YT(ND)
      ZTS(J)=ZT(ND)
      GO TO 590
540   J=J+1
      I=I+1
      XS(J)=XD(I)
      YS(J)=YD(I)
      ZS(J)=ZD(I)
      XTS(J)=XT(I)
      YTS(J)=YT(I)
      ZTS(J)=ZT(I)
      J=J+1
      XS(J)=XD(ND)
      YS(J)=YD(ND)
      ZS(J)=ZD(ND)
      XTS(J)=XT(ND)
      YTS(J)=YT(ND)
```

```
      ZTS(J)=ZT(ND)
      GO TO 590
  560 J=J+1
      I=I+1
      IF(I.EQ.ND)I=1
      XS(J)=XD(I)
      YS(J)=YD(I)
      ZS(J)=ZD(I)
      XTS(J)=XT(I)
      YTS(J)=YT(I)
      ZTS(J)=ZT(I)
  570 ARC1=ARC1+SQRT((XD(I+1)-XD(I))**2+(ZD(I+1)-ZD(I))**2)
      IF(ARC1.GE.(ARC-ARCS))GO TO 580
      IF(ARC1.GE.ARCS*J)GO TO 560
      I=I+1
      IF(I.EQ.ND)I=1
      GO TO 570
  580 J=J+1
      I=I+1
      XS(J)=XD(I)
      YS(J)=YD(I)
      ZS(J)=ZD(I)
      XTS(J)=XT(I)
      YTS(J)=YT(I)
      ZTS(J)=ZT(I)
      J=J+1
      XS(J)=XD(IN)
      YS(J)=YD(IN)
      ZS(J)=ZD(IN)
      XTS(J)=XT(IN)
      YTS(J)=YT(IN)
      ZTS(J)=ZT(IN)
  590 NPS=J
      RETURN
      END
C
C
C  THIS SUBROUTINE GENERATES B-SPLINE BASIS KNOT VECTORS
C AND B-SPLINE CURVE OF VARIOUS ORDER TO APPROXIMATE A
C GIVEN POLYGON.
C
C
      SUBROUTINE BSPLINE(M,NUM,XV,YV,ZV,ID,NUM2M,
     &                   ICLOS,NP,XR,YR,ZR,IFAULT,N,X,XT,YT,ZT)
C  M: THE ORDER OF B-SPLINE BASIS
C  NUM: NUMBER OF POLYGON VERTICES
C  XV: X-ORDINATES OF DEFINING POLYGON VERTICES
C  YV: Y-ORDINATES OF DEFINING POLYGON VERTICES
C  ZV: Z-ORDINATES OF DEFINING POLYGON VERTICES
C  ID: CONTROL VARIABLE, ID=2 FOR 2D, ID=3FOR 3D CURVE
C  NUM2M: NUM+2*M INTEGER NUMBER FOR DEFINING VARIABLE
DIMENSIONS
C  ICLOS: CONTROL VARIABLE, ICLOS=1 FOR CLOSED POLYGON,
C                   ICLOS=0 FOR OPEN POLYGON.
C  NP: THE NUMBER OF POINTS GENERATED ALONG THE CURVE

C  XR,YR,ZR: ARRAYS CONTAINING COORDINATES OF THE GENERATED
POINTS
C  XT,YT,ZT: ARRAYS CONTAINING COORDINATES OF VECTORS OF
C        TANGENTS AT GENERATED POINTS
```

```
C  IFAULT: ERROR RETURN IF M>NUM
C
   DIMENSION XV(NUM2M),YV(NUM2M),ZV(NUM2M),X(NUM2M),
  &       XR(NP),YR(NP),ZR(NP),XT(NP),YT(NP),ZT(NP)
   REAL N(NUM2M,M)
   INTEGER W
   L=0
   G=0.0
   H=0.0
   S=0.0
   IF(ICLOS.NE.1)GO TO 110
C  ADD M-1 EXTRA VERTICES TO THE DEFINING ARRAY OF VERTICES
C  IF CLOSED CURVE
C
   DO 100 J=1,M-1
   XV(NUM+J)=XV(J)
   YV(NUM+J)=YV(J)
100  ZV(NUM+J)=ZV(J)
   NUM=NUM+M-1
110  NUM1=NUM-1
   IF(M.GT.NUM)GO TO 260
   MAX=NUM1-M+2
C  FILL THE WEIGHTING FUNCTIONS ARRAY WITH ZERO
C
   DO 120 I=1,NUM+1
   DO 120 K=1,M
120  N(I,K)=0.0
   IX=NUM+M
   IF(ICLOS.NE.1)GO TO 130
C  DEFINE THE KNOT VECTORS
C
   CALL KNOTCL(M,MAX,XV,YV,ZV,NUM,ID,X,IX)
   GO TO 135
130  CALL KNOTOP(M,MAX,XV,YV,ZV,NUM,ID,X,IX)
135  STEP=(X(NUM+1)-X(M))/FLOAT(NP-1)
   W=M-1
C  INCREMENT KNOT VECTOR SUBSCRIPT
140  W=W+1
   IF(X(W).EQ.X(W+1))GO TO 140
   N(W,1)=1.
   T=X(W)
150  L=L+1
C  CALCULATE VALUES OF N(I,K) WEIGHTING FUNCTIONS
   XT(L)=0.0
   YT(L)=0.0
   ZT(L)=0.0
   DO 210 K=2,M
   DO 200 I=1,NUM
   IF(N(I,K-1).NE.0.0)GO TO 160
   E=0.0
   GO TO 170
160  E=(T-X(I))*N(I,K-1)/(X(I+K-1)-X(I))
170  IF(N(I+1,K-1).NE.0.0)GO TO 180
   F=0.0
   GO TO 190
180  F=(X(I+K)-T)*N(I+1,K-1)/(X(I+K)-X(I+1))
190  N(I,K)=E+F
   G=XV(I)*N(I,K)+G
   H=YV(I)*N(I,K)+H
```

```
      IF(ID.EQ.2)GO TO 200
      S=ZV(I)*N(I,K)+S
  200 CONTINUE
      IF(K.EQ.M)GO TO 220
      G=0.0
      H=0.0
      S=0.0
  210 CONTINUE
  220 XR(L)=G
      YR(L)=H
      IF(ID.EQ.3)ZR(L)=S
C  CALCULATE TANGENTS AT GENERATED POINTS
C
      DO 225 II=2,NUM
      IF(X(II+M-1).EQ.X(II))GO TO 223
      XXV=(XV(II)-XV(II-1))/(X(II+M-1)-X(II))
      YYV=(YV(II)-YV(II-1))/(X(II+M-1)-X(II))
      ZZV=(ZV(II)-ZV(II-1))/(X(II+M-1)-X(II))
      GO TO 224
  223 XXV=0.0
      YYV=0.0
      ZZV=0.0
  224 XT(L)=XT(L)+(M-1)*XXV*N(II,M-1)
      YT(L)=YT(L)+(M-1)*YYV*N(II,M-1)
  225 ZT(L)=ZT(L)+(M-1)*ZZV*N(II,M-1)
C  INCREMENT PARAMETER T
C
      T=T+STEP
      IF(L.EQ.NP)GO TO 250
      IF(L.EQ.(NP-1))GO TO 240
      IF(T.GE.X(W).AND.T.LT.X(W+1))GO TO 150
      N(W,1)=0.0
  230 W=W+1
      IF(X(W).EQ.X(W+1))GO TO 230
      N(W,1)=1.0
      GO TO 150
C DEFINE THE LAST POINT ON THE CURVE
C
  240 T=X(NUM+1)
      GO TO 150
  250 NUM=NUM-M+1
      NP=L
      RETURN
  260 IFAULT=1
      RETURN
      END
C  SUBROUTINE TO GENERATE BASIS KNOT VECTOR FOR
C  OPEN POLYGON
C
      SUBROUTINE KNOTOP(M,MAX,XV,YV,ZV,NUM,ID,X,IX)
      DIMENSION XV(NUM),YV(NUM),ZV(NUM),X(IX)
      DO 50 I=1,IX
      IF(I.GT.M)GO TO 10
C  ASSIGN MULTIPLE KNOTS FOR THE BEGINNING OF SPAN
      X(I)=0.0
      GO TO 50
   10 IF(I.LT.(MAX+M+1)) GO TO 30
C    ASSIGN MULTIPLE KNOTS
C
```

```
   X(I)=X(I-1)
20 GO TO 50
30 IF(ID.EQ.3)GO TO 40
C  CHECK FOR MULTIPLE KNOTS
C
   IF(XV(I-M).EQ.XV(I-M+1).AND.
  &  YV(I-M).EQ.YV(I-M+1))GO TO 20
   X(I)=X(I-1)+1.
   GO TO 50
40 IF(XV(I-M).EQ.XV(I-M+1).AND.
  &  YV(I-M).EQ.YV(I-M+1).AND.
  &  ZV(I-M).EQ.ZV(I-M+1))GO TO 20
   X(I)=X(I-1)+1.
50 CONTINUE
   RETURN
   END
C
C  SUBROUTINE TO GENERATE BASIS KNOT VECTORS FOR
C CLOSED POLYGON
C
   SUBROUTINE KNOTCL(M,MAX,XV,YV,ZV,NUM,ID,X,IX)
   DIMENSION XV(NUM),YV(NUM),ZV(NUM),X(IX)
C  FIRST KNOT VECTOR
C
   X(1)=FLOAT(1-M)
   DO 120 I=2,IX
   IF(I.GT.M)GO TO 40
   IF(ID.EQ.3)GO TO 20
C  CHECK FOR REPEATING VERTICES
C
   IF(XV(I+MAX-M).EQ.XV(I+MAX-M+1).AND.
  &  YV(I+MAX-M).EQ.YV(I+MAX-M+1))GO TO 10
   X(I)=X(I-1)+1.
   GO TO 120
10 X(I)=X(I-1)
   GO TO 120
20 IF(XV(I+MAX-M).EQ.XV(I+MAX-M+1).AND.
  &  YV(I+MAX-M).EQ.YV(I+MAX-M+1).AND.
  &  ZV(I+MAX-M).EQ.ZV(I+MAX-M+1)) GO TO 30
   X(I)=X(I-1)+1.
   GO TO 120
30 X(I)=X(I-1)
   GO TO 120
40 IF(I.LT.(MAX+M+1))GO TO 80
   IF(ID.EQ.3)GO TO 60
   IF(XV(I-MAX-M).EQ.XV(I-MAX-M+1).AND.
  &  YV(I-MAX-M).EQ.YV(I-MAX-M+1))GO TO 50
   X(I)=X(I-1)+1.
   GO TO 120
50 X(I)=X(I-1)
   GO TO 120
60 IF(XV(I-MAX-M).EQ.XV(I-MAX-M+1).AND.
  &  YV(I-MAX-M).EQ.YV(I-MAX-M+1).AND.
  &  ZV(I-MAX-M).EQ.ZV(I-MAX-M+1))GO TO 70
   X(I)=X(I-1)+1.
   GO TO 120
70 X(I)=X(I-1)
   GO TO 120
80 IF(ID.EQ.3)GO TO 100
   IF(XV(I-M).EQ.XV(I-M+1).AND.
```

```
     &  YV(I-M).EQ.YV(I-M+1))GO TO 90
      X(I)=X(I-1)+1.
      GO TO 120
   90  X(I)=X(I-1)
      GO TO 120
  100  IF(XV(I-M).EQ.XV(I-M+1).AND.
     &  YV(I-M).EQ.YV(I-M+1).AND.
     &  ZV(I-M).EQ.ZV(I-M+1))GO TO 110
      X(I)=X(I-1)+1.
      GO TO 120
  110  X(I)=X(I-1)
  120  CONTINUE
      RETURN
      END

C
C  THIS SUBROUTINE SELECTS 'NPS' POINTS ALONG A GIVEN
C CARDINAL CUBIC SPLINE CURVE AT EQUAL INTERVALS.
C
C
      SUBROUTINE EQLT(NUM,XP,YP,ZP,ID,ICLOS,NPS,ND,XS,YS,ZS,
     &           XTS,YTS,ZTS,NUM1,NPT)
      DIMENSION XP(NUM1),YP(NUM1),ZP(NUM1),XS(NPS),YS(NPS),ZS(NPS),
     &       XTS(NPS),YTS(NPS),ZTS(NPS),NPT(NPS),
     &       XD(300),YD(300),ZD(300),XT(300),YT(300),ZT(300)
      ARC=0.0
      CALL CSPLINE(NUM,XP,YP,ZP,ID,ICLOS,ND,XD,YD,ZD,XT,YT,ZT,NUM1)
      DO 500 I=1,ND-1
  500  ARC=ARC+SQRT((XD(I+1)-XD(I))**2+(ZD(I+1)-ZD(I))**2)
      ARCS=ARC/FLOAT(NPS-1)
      ARC1=0.0
      J=0
      I=0
  510  J=J+1
      I=I+1
      XS(J)=XD(I)
      YS(J)=YD(I)
      ZS(J)=ZD(I)
      XTS(J)=XT(I)
      YTS(J)=YT(I)
      ZTS(J)=ZT(I)
C  STORE THE NUMBER OF SELECTED POINTS
C
      NPT(J)=I
  520  ARC1=ARC1+SQRT((XD(I+1)-XD(I))**2+(ZD(I+1)-ZD(I))**2)
      IF(ARC1.GE.(ARC-ARCS))GO TO 530
      IF(ARC1.GE.ARCS*J)GO TO 510
      I=I+1
      GO TO 520
  530  J=J+1
      I=I+1
      XS(J)=XD(I)
      YS(J)=YD(I)
      ZS(J)=ZD(I)
      XTS(J)=XT(I)
      YTS(J)=YT(I)
      ZTS(J)=ZT(I)
C  STORE THE NUMBER OF SELECTED POINTS
C
      NPT(J)=I
```

```
      J=J+1
      XS(J)=XD(ND)
      YS(J)=YD(ND)
      ZS(J)=ZD(ND)
      XTS(J)=XT(ND)
      YTS(J)=YT(ND)
      ZTS(J)=ZT(ND)
      NPT(J)=ND
      NPS=J
      RETURN
      END
C  THIS SUBROUTINE FITS A CUBIC CARDINAL SPLINE TO
C A SET OF GIVEN DATA.
C
      SUBROUTINE CSPLINE(NUM,XP,YP,ZP,ID,ICLOS,NP,XR,YR,ZR,
     &                XT,YT,ZT,NUM1)
C  NUM: NUMBER OF INPUT DATA
C  XP,YP,ZP:ARRAYS CONTAINING COORDINATES OF INPUT POINTS
C  ID: CONTROL VARIABLE, ID=2 FOR 2D, ID=3 FOR 3D CURVE
C  ICLOS: CONTROL VARIABLE, ICLOS=0  OPEN CURVE,
C                    ICLOS=1  CLOSED CURVE.
C  NP: NUMBER OF GENERATED POINTS ALONG THE CURVE
C  XR,YR,ZR: ARRAYS CONTAINING COORDINATES OF GENERATED POINTS
C  XT,YT,ZT: ARRAYS CONTAINING COORDINATES OF VECTORS OF
C        TANGENTS AT GENERATED POINTS
C
      DIMENSION XP(NUM1),YP(NUM1),ZP(NUM1),XR(NP),YR(NP),ZR(NP),
     &       XT(NP),YT(NP),ZT(NP)

      INTEGER PWR
      IF(ICLOS.EQ.1)GO TO 80
      XP(1)=XP(2)
      YP(1)=YP(2)
      ZP(1)=ZP(2)
      XP(NUM+2)=XP(NUM+1)
      YP(NUM+2)=YP(NUM+1)
      ZP(NUM+2)=ZP(NUM+1)
      DELT=FLOAT(NUM-1)/FLOAT(NP-1)
      NUM=NUM+2
      GO TO 90
   80  XP(1)=XP(NUM+1)
      YP(1)=YP(NUM+1)
      ZP(1)=ZP(NUM+1)
      XP(NUM+2)=XP(2)
      YP(NUM+2)=YP(2)
      ZP(NUM+2)=ZP(2)
      XP(NUM+3)=XP(3)
      YP(NUM+3)=YP(3)
      ZP(NUM+3)=ZP(3)
      DELT=FLOAT(NUM)/FLOAT(NP-1)
      NUM=NUM+3
   90  A=-2.+3**(1./2.)
C  CALCULATE ZIGN=
C
      ZIGN1=0.0
      ZIGN2=0.0
      ZIGN3=0.0
      PWR=NUM-2
      DO 100 I=1,NUM-1
      ZIGN1=ZIGN1+3*XP(I)*(A**PWR)
```

```
      ZIGN2=ZIGN2+3*YP(I)*(A**PWR)
      IF(ID.EQ.2)GO TO 100
      ZIGN3=ZIGN3+3*ZP(I)*(A**PWR)
  100 PWR=PWR-1
C     CALCULATE TA1=
C
      TA11=0.0
      TA12=0.0
      TA13=0.0
      PWR=0
      DO 110 I=2,NUM
      TA11=TA11+3*XP(I)*(A**PWR)
      TA12=TA12+3*YP(I)*(A**PWR)
      IF(ID.EQ.2)GO TO 110
      TA13=TA13+3*ZP(I)*(A**PWR)
  110 PWR=PWR+1
C     CALCULATE CARDINAL SPLINE VALUES STARTING FROM
C     THE FIRST INTERVAL IN STEPS OF DELT=NUM-1/NP-1
C
      T=2.0
      K=2
      J=0
  106 J=J+1
C     CALCULATE ZIGK=
      ZIGK1=0.0
      ZIGK2=0.0
      ZIGK3=0.0
      PWR=K-1
      DO 120 I=1,K
      ZIGK1=ZIGK1+3*XP(I)*(A**PWR)
      ZIGK2=ZIGK2+3*YP(I)*(A**PWR)
      IF(ID.EQ.2)GO TO 120
      ZIGK3=ZIGK3+3*ZP(I)*(A**PWR)
  120 PWR=PWR-1
C     CALCULATE TAK=
C
      TAK1=0.0
      TAK2=0.0
      TAK3=0.0
      PWR=0
      DO 130 I=K+1,NUM
      TAK1=TAK1+3*XP(I)*(A**PWR)
      TAK2=TAK2+3*YP(I)*(A**PWR)
      IF(ID.EQ.2)GO TO 130
      TAK3=TAK3+3*ZP(I)*(A**PWR)
  130 PWR=PWR+1
  140 B=(1-A**(NUM-1))
      C=(T-K)-(A+2)*((T-K)**2)+(A+1)*((T-K)**3)
      D=(1-T+K)-(A+2)*((1-T+K)**2)+(A+1)*((1-T+K)**3)
      XR(J)=(C/B)*(ZIGN1*(A**K)+ZIGK1*B)+((1-T+K)**3)*XP(K)
     &     +(D/B)*(TA11*(A**(NUM-K))+TAK1*B)+((T-K)**3)*XP(K+1)
      YR(J)=(C/B)*(ZIGN2*(A**K)+ZIGK2*B)+((1-T+K)**3)*YP(K)
     &     +(D/B)*(TA12*(A**(NUM-K))+TAK2*B)+((T-K)**3)*YP(K+1)
      IF(ID.EQ.2)GO TO 144
      ZR(J)=(C/B)*(ZIGN3*(A**K)+ZIGK3*B)+((1-T+K)**3)*ZP(K)
     &     +(D/B)*(TA13*(A**(NUM-K))+TAK3*B)+((T-K)**3)*ZP(K+1)
C     CALCULATE TANGENTS AT THE GENERATED POINTS
C
  144 CC=1-2*(A+2)*(T-K)+3*(A+1)*((T-K)**2)
      DD=-1+2*(A+2)*(1-T+K)-3*(A+1)*((1-T+K)**2)
```

```
      XT(J)=(CC/B)*(ZIGN1*(A**K)+ZIGK1*B)-3*((1-T+K)**2)*XP(K)
     &    +(DD/B)*(TA11*(A**(NUM-K))+TAK1*B)+3*((T-K)**2)*XP(K+1)
      YT(J)=(CC/B)*(ZIGN2*(A**K)+ZIGK2*B)-3*((1-T+K)**2)*YP(K)
     &    +(DD/B)*(TA12*(A**(NUM-K))+TAK2*B)+3*((T-K)**2)*YP(K+1)
      IF(ID.EQ.2)GO TO 145
      ZT(J)=(CC/B)*(ZIGN3*(A**K)+ZIGK3*B)-3*((1-T+K)**2)*ZP(K)
     &    +(DD/B)*(TA13*(A**(NUM-K))+TAK3*B)+3*((T-K)**2)*ZP(K+1)
C  INCREMENT T
C
  145 T=T+DELT
      IF(J.EQ.NP)GO TO 170
      IF(J.EQ.(NP-1))GO TO 160
      IF(T.GE.K.AND.T.LT.(K+1))GO TO 150
      K=K+1
      GO TO 106
  150 J=J+1
      GO TO 140
  160 J=J+1
      T=NUM-1
      GO TO 140
  170 NUM=NUM-2
      IF(ICLOS.EQ.1)NUM=NUM-1
      NP=J
      RETURN
      END
```

```
C
C  THIS PROGRAM BY READING DATA POINTS ON A SURFACE AND
DIRECTION COSINE
C  AT THOSE POINTS,PRODUCES CUTTER CENTRE-LINE PATH DATA ALONG
PARAMETER
C  U,AND,OR V. BY SETTING "INTCHCK" SWITCH CUTTER INTERFERENCE
CHECK CAN ALSO
C  BE CARRIED OUT. IN THIS PROCESS THE SIZE OF THE TOOL USED IS
SUCCESSIVELY
C  REDUCED TO AVOID EXCESSIVE REMOVAL OF ADJACENT AREA.
C
C  SURFACE COORDINATE,AND DIRECTION COSINE DATA INPUT FROM
CHANNEL 5 "*CR1"
C  CONTROL VARIABLES INPUT FROM CHANNEL 1 "*CR0"
C  CUTTER CENTRE-LINE PATH DATA OUTPUT FROM CHANNEL 6 "*LP1"
      DIMENSION XF(50,50),YF(50,50),ZF(50,50),
     &          XFF(50,50),YFF(50,50),ZFF(50,50),
     &          XDC(50,50),YDC(50,50),ZDC(50,50),NUM(2500,2),
     &          R(8),D(8),ORIGIN(3)
C
C
C  DIMENSION FOR XF,YF,ZF :(NPU,NPV)
C            XFF,YFF,ZFF :(NPU,NPV)
C            XDC,YDC,ZDC :(NPU,NPV)
C            NUM :(NPU*NPV)
C            R,D :(NTOOL)
C
C  IUPATH :CONTROL VARIABLE ;(IUPATH=1 OUTPUTS CUTTER CENTRE-LINE
PATH DATA
C       ALONG PARAMETER U.)
C  IVPATH :CONTROL VARIABLE ;(IVPATH=1 OUTPUTS CUTTER CENTRE-LINE
PATH DATA
C       ALONG PARAMETER V.)
C  ICOMP  :CONTROL VARIABLE ;(ICOMP=1 CUTTER CENTRE-LINE IS
COMPENSATED BY
C       THE RADIUS OF THE CUTTER ALONG PARAMETER V WHEN CUTTING A
GROOVE.)
C  IMROR  :CONTROL VARIABLE ;(IMROR=1 MIRROR IMAGE IN Z OF THE DATA
INPUT IS
C       PRODUCED.)
C  ISCF :MM TO INCHES CONVERSION FACTOR;(ISCF=1 CUTTER PATH DATA IS
C     DIVIDED BY 25.4)
C  XDC,YDC,ZDC :ELEMENTS OF DIRECTION COSINES AT POINTS XF,YF,ZF.
C  ORIGIN :CUTTER SET POINT IN THE DESIGN COORDINATE SYSTEM.
C  D :DIAMETER OF THE CUTTER
C
      OPEN(UNIT=1,NAME='DAANRE',STATUS='UNKNOWN')
      OPEN(UNIT=2,NAME='OANAOFF1',STATUS='UNKNOWN')
      OPEN(UNIT=5,NAME='OBCSURN31',STATUS='UNKNOWN')
      OPEN(UNIT=6,NAME='OANAOFF2',STATUS='UNKNOWN')
      ZHIGH=-10000
      ZLOW=10000
      READ(1,*)NTOOL,(D(I),I=1,NTOOL)
      READ(1,*)(ORIGIN(I),I=1,3)
      READ(1,*)TLRNCE
      READ(1,*)ICOMP,INTCHK
```

```
      READ(1,*)IUPATH,IVPATH
      READ(1,*)INOUT,IMROR


      READ(1,*)ISCF
      READ(5,*)NPU,NPV
C     READ THE DATA POINTS
      DO 120 I=1,NPU
      DO 120 J=1,NPV
      READ(5,*)XF(I,J),ZF(I,J),YF(I,J),XDC(I,J),ZDC(I,J),YDC(I,J)
      IF(ISCF.NE.1)GO TO 110
      XF(I,J)=XF(I,J)/25.4
      YF(I,J)=YF(I,J)/25.4
      ZF(I,J)=ZF(I,J)/25.4
  110 IF(ZF(I,J).LT.ZLOW)ZLOW=ZF(I,J)
      IF(ZF(I,J).GT.ZHIGH)ZHIGH=ZF(I,J)
  120 CONTINUE
C     FIND MIRROR IMAGE IF REQUIRED
C
      IF(IMROR.NE.1)GO TO 127
      DO 125 I=1,NPU
      DO 125 J=1,NPV
  125 ZF(I,J)=2*ZLOW-ZF(I,J)
C     CUTTER SET POINT IN Z SHOULD BE ABOVE THE HIGHEST POINT ON THE
SURFACE.
C
  127 IF(ORIGIN(3).LT.ZHIGH)ORIGIN(3)=ZHIGH+1
      IF(IMROR.EQ.1)ORIGIN(3)=ZLOW+1
      WRITE(2,996)NTOOL,(I,D(I),I=1,NTOOL)
      WRITE(2,995)(ORIGIN(I),I=1,3)
      WRITE(2,994)TLRNCE
      IF(ICOMP.EQ.1)WRITE(2,993)
      IF(INTCHK.EQ.1)WRITE(2,992)
      IF(IUPATH.EQ.1)WRITE(2,991)
      IF(IVPATH.EQ.1)WRITE(2,990)
      IF(IMROR.EQ.1)WRITE(2,989)
      DO 130 I=1,NTOOL
  130 R(I)=D(I)/2.0
C
C     INTERFERENCE CHECK.
C
      KN=0
C     TAKE THE LARGEST TOOL ;TOOL NUMBER 1
      ITOOL=1
C     SAVE THE TAGS OF THE POINTS TO BE CHECKED FOR INTERFERENCE
      DO 140 I=1,NPU
      DO 140 J=1,NPV
      KN=KN+1
      NUM(KN,1)=I
      NUM(KN,2)=J
  140 CONTINUE
  150 KKN=KN
      KN=0
C     COMPUTE THE CUTTER OFFSETS
      KSGN=1
      IF(IMROR.EQ.1)KSGN=-1
      DO 160 I=1,NPU
      DO 160 J=1,NPV
      XFF(I,J)=XF(I,J)-INOUT*XDC(I,J)*R(ITOOL)
      YFF(I,J)=YF(I,J)-INOUT*YDC(I,J)*R(ITOOL)
```

```
      ZFF(I,J)=ZF(I,J)-KSGN*INOUT*ZDC(I,J)*R(ITOOL)
160   CONTINUE

      IF(INTCHK.EQ.0)GO TO 190
170   DO 190 K=1,KKN
      INT=0
C     RETRIEVE THE TAGS OF THE POINTS TO BE CHECKED FOR INTERFERENCE
      II=NUM(K,1)
      JJ=NUM(K,2)
      DO 180 I=1,NPU
      DO 180 J=1,NPV
      XDIS=(XFF(II,JJ)-XF(I,J))**2
      YDIS=(YFF(II,JJ)-YF(I,J))**2
      ZDIS=(ZFF(II,JJ)-ZF(I,J))**2
      DIS=SQRT(XDIS+YDIS+ZDIS)
C     CHECK THE TOOL POSITION FOR INTERFERNCE WITH NEIGHBOURING
POINTS.
      IF(DIS.GE.R(ITOOL)-TLRNCE)GO TO 180
C     FIND THE HEIGHT TO AVOID INTERFERENCE
      ZFF(II,JJ)=SQRT(ABS(R(ITOOL)**2-(XDIS+YDIS)))+ZF(I,J)
      INT=1
180   CONTINUE
      IF(INT.EQ.0)GO TO 190
      KN=KN+1
C     SAVE THE TAGS OF THE POINTS TO BE RECHECKED
      NUM(KN,1)=NUM(K,1)
      NUM(KN,2)=NUM(K,2)
190   CONTINUE
      IF(ICOMP.EQ.0)GO TO 235
C     TOOL DIAMETER COMPENSATION
      DO 215 I=1,NPU
195   J=0
200   J=J+1
      IF(J.EQ.NPV)GO TO 310
      DIS=(XF(I,1)-XFF(I,J))**2+(YF(I,1)-YFF(I,J))**2
      IF(DIS.GT.R(ITOOL)**2)GO TO 210
      GO TO 200
210   IF(J.LE.2)GO TO 215
      DO 212 J1=1,J-1
      XFF(I,J1)=XFF(I,J)
      YFF(I,J1)=YFF(I,J)
212   ZFF(I,J1)=ZFF(I,J)
215   CONTINUE
      DO 235 I=1,NPU
216   J=NPV+1
220   J=J-1
      IF(J.EQ.1)GO TO 310
      DIS=(XFF(I,J)-XF(I,NPV))**2+(YFF(I,J)-YF(I,NPV))**2
      IF(DIS.GT.R(ITOOL)**2)GO TO 230
      GO TO 220
230   IF(J.GE.NPV-1)GO TO 235
      DO 226 J1=J+1,NPV
      XFF(I,J1)=XFF(I,J)
      YFF(I,J1)=YFF(I,J)
226   ZFF(I,J1)=ZFF(I,J)
235   CONTINUE
C     PART PROGRAM OUTPUT
      N=0
      WRITE(6,988)ITOOL
C     CUTTER CENTRE-LINE PATH DATA ALONG PARAMETER U
```

```
      IF(IUPATH.EQ.0)GO TO 255
      K1=0


      DO 250 J=1,NPV
      K1=MOD(J,2)
      IF(K1.EQ.0)GO TO 245
      DO 240 I=1,NPU
C     CALCULATE DATA IN RELATION TO MACHINE ORIGIN
      X=XFF(I,J)-ORIGIN(1)
      Y=YFF(I,J)-ORIGIN(2)
      Z=ZFF(I,J)-ORIGIN(3)
      N=N+1
      IF(N.GT.999)N=1
      WRITE(6,987)N,X,Y,Z
  240 CONTINUE
      GO TO 250
  245 DO 250 I=1,NPU
      K=NPU-I+1
      X=XFF(K,J)-ORIGIN(1)
      Y=YFF(K,J)-ORIGIN(2)
      Z=ZFF(K,J)-ORIGIN(3)
      N=N+1
      IF(N.GT.999)N=1
      WRITE(6,987)N,X,Y,Z
  250 CONTINUE
C     CUTTER CENTRE-LINE PATH DATA ALONG PARAMETER V
  255 IF(IVPATH.EQ.0)GO TO 300
      IF(K1.EQ.0)GO TO 260
      LU=NPU
      ISGN=-1
      GO TO 270
  260 LU=1
      ISGN=1
  270 DO 280 J=1,NPV
      X=XFF(LU,NPV-J+1)-ORIGIN(1)
      Y=YFF(LU,NPV-J+1)-ORIGIN(2)
      Z=ZFF(LU,NPV-J+1)-ORIGIN(3)
      N=N+1
      IF(N.GT.999)N=1
  280 WRITE(6,987)N,X,Y,Z
      LU=LU+ISGN

      IF(LU.EQ.0.OR.LU.EQ.NPU+1)GO TO 300
      DO 290 J=1,NPV
      X=XFF(LU,J)-ORIGIN(1)
      Y=YFF(LU,J)-ORIGIN(2)
      Z=ZFF(LU,J)-ORIGIN(3)
      N=N+1
      IF(N.EQ.999)N=1
  290 WRITE(6,987)N,X,Y,Z
      LU=LU+ISGN
      IF(LU.EQ.0.OR.LU.EQ.NPU+1)GO TO 300
      GO TO 270
  300 IF(INTCHK.EQ.0)STOP
      IF(KN.EQ.0)STOP
      ITOOL=ITOOL+1
      IF(ITOOL.LE.NTOOL)GO TO 150
      WRITE(2,986)
      STOP
```

```
310  WRITE(2,985)
     STOP

C
C
996  FORMAT(1H1,///' NUMBER OF TOOLS USED=',I2,
     &      10(/' D(',I2,')=',F8.4))
995  FORMAT(' TOOL SET POINT IN RELATION TO DESIGN',
     &      ' COORDINATE SYSTEM',
     &      /' ORIGIN(1)=',F7.3,
     &      /' ORIGIN(2)=',F7.3,
     &      /' ORIGIN(3)=',F7.3)
994  FORMAT(/' DEFINED TOLERANCE IN CUTTER INTERFERENCE=',F8.6)
993  FORMAT(//' CUTTER DIAMETER COMPENSATION SWITCH IS SET')
992  FORMAT(//' INTERFERENCE CHECK SWITCH IS SET')
991  FORMAT(//' TOOL CENTRE-LINE PATH ALONG PARAMETER U IS
GENERATED')
990  FORMAT(/' TOOL CENTRE-LINE PATH ALONG PARAMETER V IS
GENERATED')
989  FORMAT(//' Z MIRROR IMAGE SWITCH IS SET')
988  FORMAT(' T',I2)
987  FORMAT(1X,I3,2X,F8.4,2X,F8.4,2X,F8.4)
986  FORMAT(///'****TO AVOID INTERFERENCE SMALLER TOOL THAN THE',
     &       ' ONES USED ARE NEEDED***')
985  FORMAT(' TOOL SIZE TOO LARGE TO ALLOW DIAMETER COMPENSATION')
     END
$
```

```
      PROGRAM NUMOFF
C THIS PROGRAM READS X-Y-Z COORDINATES OF A GRID PROJECTED
C  ON A SURFACE,TAKING THE LARGEST POSSIBLE TOOL,FINDS THE
C  CENTRE OF THE TOOL TO TOUCH THE SURFACE.IF INTERFERENCE
C  OCCURS CHANGES THE TOOL TO A SMALLER SIZE.
C
      COMMON/ALL/VERT(9,1000),A(1000),B(1000),C(1000),D(1000),
     &     X(50,50),Y(50,50),Z(50,50),SX(1000),SY(1000),
     &     SZ(1000),ORIGIN(3),RAD(8),TLRNCE,NUMINT(1000),
     &     INTVAR,IMTRX,JMTRX,NUMS(1000),NUM
      OPEN(UNIT=1,NAME=WEE,STATUS='UNKNOWN')
      OPEN(UNIT=5,NAME=MEE,STATUS='UNKNOWN')
      OPEN(UNIT=2,NAME=ONUMOFF,STATUS='UNKNOWN')
      READ(1,*)ISECTN
      READ(1,*)(ORIGIN(J),J=1,3)
      ORIGIN3=ORIGIN(3)
      READ(1,*)(RAD(J),J=1,8)
      READ(1,*)INTCHK,NUMPAS,TLRNCE
 1000 READ(5,*)IMTRX,JMTRX
      READ(5,*)((X(I,J),Y(I,J),Z(I,J),J=1,JMTRX),I=1,IMTRX)
      ITRNGL=1
 1010 ITOOL=1
      ORIGIN(3)=ORIGIN3+RAD(ITOOL)
      INTVAR=0
      K=0
      CALL MATRIX(ITRNGL,K)
      NUM=K
      K1=K
      CALL TLCENT(K1,ITOOL)
      IF(INTCHK.EQ.0)GO TO 1050
      DO 1020 I=1,NUM
 1020 NUMINT(I)=I
 1030 DO 1040 I=1,NUM
 1040 NUMS(I)=NUMINT(I)
      CALL INTRFR(ITRNGL,ITOOL)
 1050 WRITE(2,995)ITOOL
      IF(ITOOL.NE.1)GO TO 1055
      II=0
      DO 1055 I=1,IMTRX-1
      MD=MOD(I+1,2)
      IF(MD.EQ.0)GO TO 1051
      ISGN=-1
      GO TO 1052
 1051 ISGN=1
 1052 II=II+MD*(JMTRX-1)*2
      DO 1055 J=1,2*(JMTRX-1)
      II=II+ISGN
      XO=SX(II)-ORIGIN(1)
      YO=SY(II)-ORIGIN(2)
      ZO=-(SZ(II)-ORIGIN(3))
      WRITE(2,994)XO,YO,ZO
 1055 CONTINUE
      GO TO 1060
      DO 1060 I=1,NUM
      XO=SX(I)-ORIGIN(1)
      YO=SY(I)-ORIGIN(2)
C POSITIVE Z DIRECTION FOR DATA IS OPPOSITE TO
```

```
C  MACHINE POSITIVE Z DIRECTION
   ZO=-(SZ(I)-ORIGIN(3))
   WRITE(2,994)XO,YO,ZO
1060  CONTINUE
   IF(INTVAR.EQ.0)GO TO 1070
   ITOOL=ITOOL+1
   IF(ITOOL.GT.8)GO TO 1080
   NUM=INTVAR
   CALL TLCENT(K1,ITOOL)
   GO TO 1030
1070  IF(NUMPAS.EQ.1)GO TO 1071
   IF(ITRNGL.NE.1)GO TO 1071
   ITRNGL=2
   GO TO 1010
1071  ISECTN=ISECTN-1
   IF(ISECTN.EQ.0)STOP
   GO TO 1000
1080  WRITE(2,993)RAD(8)
995   FORMAT('0T',I1)
994   FORMAT(' X',F8.4,'  Y',F8.4,'  Z',F8.4)
993   FORMAT('0TO AVOID INTERFERENCE THE SMALLEST TOOL SHOULD',
   &      'BE CHANGED TO A SMALLER SIZE.',/' RAD(8)=',F6.3)
   STOP
   END
C
C
C
C
   SUBROUTINE MATRIX(ITRNGL,K)
   COMMON/ALL/VERT(9,1000),A(1000),B(1000),C(1000),D(1000),
   &     X(50,50),Y(50,50),Z(50,50),SX(1000),SY(1000),
   &     SZ(1000),ORIGIN(3),RAD(8),TLRNCE,NUMINT(1000),
   &     INTVAR,IMTRX,JMTRX,NUMS(1000),NUM
   DIMENSION XX(3),YY(3),ZZ(3)
   ASSIGN 300 TO JMP
   IF(ITRNGL.EQ.1)GO TO 100
   ASSIGN 200 TO JMP
 100  IMXM1=IMTRX-1
   DO 800 I=1,IMXM1
   DO 800 L=1,JMTRX
   KK=I/2
   XKK=FLOAT(KK)
   J=L
   YKK=FLOAT(I)/2.
   IF(XKK.EQ.YKK)J=JMTRX-L+1
   XX(1)=X(I,J)
   YY(1)=Y(I,J)
   ZZ(1)=Z(I,J)
   I1=I+1
   I2=I
   GO TO JMP,(200,300)
 200  I1=I
   I2=I+1
 300  IF(J.EQ.JMTRX)GO TO 400
   IF(J.EQ.1)GO TO 600
   IF(XKK.EQ.YKK)GO TO 600
 400  XX(2)=X(I+1,J)
   YY(2)=Y(I+1,J)
   ZZ(2)=Z(I+1,J)
   XX(3)=X(I1,J-1)
```

```
      YY(3)=Y(I1,J-1)
      ZZ(3)=Z(I1,J-1)
      K=K+1
      CALL COEFF(XX,YY,ZZ,K)
      N=0
      DO 500 LL=1,7,3
      N=N+1
C STORETHE VERTICES OF THE TRIANGLE.
      VERT(LL,K)=XX(N)
      VERT(LL+1,K)=YY(N)
      VERT(LL+2,K)=ZZ(N)
500   CONTINUE
      IF(J.EQ.JMTRX)GO TO 800
      IF(XKK.EQ.YKK)GO TO 800
600   XX(2)=X(I2,J+1)
      YY(2)=Y(I2,J+1)
      ZZ(2)=Z(I2,J+1)
      XX(3)=X(I+1,J)
      YY(3)=Y(I+1,J)
      ZZ(3)=Z(I+1,J)
      K=K+1
      CALL COEFF(XX,YY,ZZ,K)
      N=0
      DO 700 LL=1,7,3
      N=N+1
C STORETHE VERTICES OF THE TRIANGLE.
      VERT(LL,K)=XX(N)
      VERT(LL+1,K)=YY(N)
      VERT(LL+2,K)=ZZ(N)
700   CONTINUE
      IF(J.EQ.1)GO TO 800
      IF(XKK.EQ.YKK)GO TO 400
800   CONTINUE
      RETURN
      END
C SUBROUTINE COEFF CALCULATES THE COEFFICIENTS A,B,C & D IN
C  AX+BY+CZ+D=0 WHICH REPRESENTS A PLANE
C
      SUBROUTINE COEFF(XX,YY,ZZ,K)
      COMMON/ALL/VERT(9,1000),A(1000),B(1000),C(1000),D(1000),
     &     X(50,50),Y(50,50),Z(50,50),SX(1000),SY(1000),
     &     SZ(1000),ORIGIN(3),RAD(8),TLRNCE,NUMINT(1000),
     &     INTVAR,IMTRX,JMTRX,NUMS(1000),NUM
      DIMENSION XX(3),YY(3),ZZ(3),UNT(3)
      UNT(1)=1
      UNT(2)=1
      UNT(3)=1
      A(K)=DETER(UNT,YY,ZZ)
      B(K)=DETER(XX,UNT,ZZ)
      C(K)=DETER(XX,YY,UNT)
      D(K)=-DETER(XX,YY,ZZ)
      SUMSQ=SQRT(A(K)*A(K)+B(K)*B(K)+C(K)*C(K))
      IF(C(K).LT.0.0)SUMSQ=-SUMSQ
C
C COEFFICIENTS OF AX+BY+CZ+D=0
C
      A(K)=A(K)/SUMSQ
      B(K)=B(K)/SUMSQ
      C(K)=C(K)/SUMSQ
      D(K)=D(K)/SUMSQ
```

```
C
C TO DETECT VERTICAL PLANES
C
      IF(C(K).GE.0.00001)RETURN
      WRITE(2,99)K
   99  FORMAT('0THERE IS A VERTICAL PLANE   PLANE',I4)
      STOP 999
      END
      FUNCTION DETER(XX,YY,ZZ)
      DIMENSION XX(3),YY(3),ZZ(3)
      DETER=XX(1)*YY(2)*ZZ(3)+YY(1)*ZZ(2)*XX(3)+ZZ(1)*XX(2)*YY(3)
     &    -ZZ(1)*YY(2)*XX(3)-XX(1)*ZZ(2)*YY(3)-YY(1)*XX(2)*ZZ(3)
      RETURN
      END
C
C SUBROUTINE INTERFER
C
C
      SUBROUTINE INTRFR(ITRNGL,ITOOL)
      COMMON/ALL/VERT(9,1000),A(1000),B(1000),C(1000),D(1000),
     &     X(50,50),Y(50,50),Z(50,50),SX(1000),SY(1000),
     &     SZ(1000),ORIGIN(3),RAD(8),TLRNCE,NUMINT(1000),
     &     INTVAR,IMTRX,JMTRX,NUMS(1000),NUM
      DIMENSION NBRPL(500)
      LOGICAL SWITCH
      INTVAR=0
      DO 600 JJ=1,NUM
      J=NUMS(JJ)
      IADD=0
      CALL RADIST(IMTRX,JMTRX,ITRNGL,J,NBRPL,NNBRPL,ITOOL)
      IF(NNBRPL.EQ.0)GO TO 600
      ZZ=SZ(J)
      DO 500 I1=1,NNBRPL
      LIST=NBRPL(I1)
      TLDIS=A(J)*SX(J)+B(J)*SY(J)+C(J)*SZ(J)+D(J)
      I2=1
      SWITCH=.FALSE.
      DO 100 I3=1,3
      VRTDIS=A(J)*VERT(I3,LIST)+B(J)*VERT(I3+1,LIST)
     &     +C(J)*VERT(I3+2,LIST)+D(J)
      VRTDIS=VRTDIS/TLDIS
      IF(VRTDIS.LT..0001)GO TO 100
      SWITCH=.TRUE.
      GO TO 200
  100 I2=I2+3
  200 IF(.NOT.SWITCH)GO TO 300
      XDIS=ABS(A(LIST)*SX(J)+B(LIST)*SY(J)+C(LIST)*SZ(J)
     &   +D(LIST))
      DIS=RAD(ITOOL)-XDIS
      IF(DIS.LT.TLRNCE)GO TO 300
      CALL INTRIN(XDIS,J,LIST,SWITCH,ITOOL)
      IF(SWITCH)GO TO 400
  300 Z1=SZ(J)
      GO TO 500
  400 Z1=(RAD(ITOOL)-A(LIST)*SX(J)-B(LIST)*SY(J)
     & -D(LIST))/C(LIST)
  410 DIFF=Z1-TLRNCE
      IF(ZZ.GE.DIFF)GO TO 500
      ZZ=Z1
      IADD=IADD+1
```

```
500  CONTINUE
     IF(IADD.EQ.0)GO TO 600
     SZ(J)=ZZ
C
C
     INTVAR=INTVAR+1
     NUMINT(INTVAR)=J
600  CONTINUE
     RETURN
     END
C
C
C
C
     SUBROUTINE RADIST(IMTRX,JMTRX,ITRNGL,NOW,NBRPL,NNBRPL,
    &         ITOOL)
     DIMENSION NBRPL(500)
     MD=2*(JMTRX-1)
     LR=(NOW-MOD(NOW,MD))/MD+1
     IF(MOD(NOW,MD).EQ.0)LR=LR-1
C
C
     NNBRPL=0
     ICOUNT=1
     NEXT=NOW
     ISAVE=NNBRPL
     DO 50 I1=1,1000
     J1=LR*MD
     JP1=NEXT+1
     IF(JP1.GT.J1)GO TO 20
C
     DO 10 JDO=JP1,J1
     DIS=DIST(NOW,JDO,ITOOL)
     IF(DIS.GT..001)GO TO 20
     NNBRPL=NNBRPL+1
     NBRPL(NNBRPL)=JDO
10   CONTINUE
20   J1=(LR-1)*MD+1
     JM1=NEXT-1
     IF(JM1.LT.J1)GO TO 40
     KOUNT=0
     DO 30 JDO=J1,JM1
     KOUNT=KOUNT+1
     NJDO=NEXT-KOUNT
     DIS=DIST(NOW,NJDO,ITOOL)
     IF(DIS.GT..001)GO TO 40
     NNBRPL=NNBRPL+1
     NBRPL(NNBRPL)=NJDO
30   CONTINUE
C
C
40   IF(ISAVE.EQ.NNBRPL)GO TO 60
     INOW=NEXT
     CALL UPDOWN(ITRNGL,ICOUNT,NEXT,INOW,LR,MD,IMTRX,JMTRX)
     IF(NEXT.EQ.0)GO TO 60
     ISAVE=NNBRPL
     DIS=DIST(NOW,NEXT,ITOOL)
     IF(DIS.GT.001)GO TO 50
     NNBRPL=NNBRPL+1
     NBRPL(NNBRPL)=NEXT
```

```
50    CONTINUE
60    IF(NNBRPL.GT.300)GO TO 131
      ICOUNT=-1
      INOW=NOW
      LR=(NOW-MOD(NOW,MD))/MD+1
      IF(MOD(NOW,MD).EQ.0)LR=LR-1
      CALL UPDOWN(ITRNGL,ICOUNT,NEXT,INOW,LR,MD,IMTRX,JMTRX)
      IF(NEXT.EQ.0)GO TO 130
      ISAVE=NNBRPL
      DIS=DIST(NOW,NEXT,ITOOL)
      IF(DIS.GT..001)GO TO 70
      NNBRPL=NNBRPL+1
      NBRPL(NNBRPL)=NEXT
70    DO 120 I1=1,1000
      J1=LR*MD
      JP1=NEXT+1
      IF(JP1.GT.J1)GO TO 90
      DO 80 JDO=JP1,J1
      DIS=DIST(NOW,JDO,ITOOL)
      IF(DIS.GT..001)GO TO 90
      NNBRPL=NNBRPL+1
      NBRPL(NNBRPL)=JDO
80    CONTINUE
90    J1=(LR-1)*MD+1
      JM1=NEXT-1
      IF(JM1.LT.J1)GO TO 110
      KOUNT=0
      DO 100 JDO=J1,JM1
      KOUNT=KOUNT+1
      NJDO=NEXT-KOUNT
      DIS=DIST(NOW,NJDO,ITOOL)
      IF(DIS.GT..001)GO TO 110
      NNBRPL=NNBRPL+1
      NBRPL(NNBRPL)=NJDO
100   CONTINUE
110   IF(ISAVE.EQ.NNBRPL)GO TO 130
      INOW=NEXT
      CALL UPDOWN(ITRNGL,ICOUNT,NEXT,INOW,LR,MD,IMTRX,JMTRX)
      IF(NEXT.EQ.0)GO TO 130
      ISAVE=NNBRPL
      DIS=DIST(NOW,NEXT,ITOOL)
      IF(DIS.GT..001)GO TO 120
      NNBRPL=NNBRPL+1
      NBRPL(NNBRPL)=NEXT
120   CONTINUE
130   IF(NNBRPL.LE.500)GO TO 140
131   WRITE(2,999)
999   FORMAT('0ERROR. DIMENSION OF NBRPL EXTENDED.')
      STOP 999
140   RETURN
      END
C
C
C
C
      SUBROUTINE
UPDOWN(ITRNGL,ICOUNT,NEXT,NOW,LR,MD,IMTRX,JMTRX)
      IF(ITRNGL.EQ.2)GO TO 5
      IF(ICOUNT.LT.0)GO TO 3
1     IF(MOD(NOW,2).EQ.1)GO TO 2
```

```
      IF(MOD(LR,2).EQ.1)JMP=3
      IF(MOD(LR,2).EQ.0)JMP=4
      GO TO 6
    2 IF(MOD(LR,2).EQ.1)JMP=2
      IF(MOD(LR,2).EQ.2)JMP=1
      GO TO 6
    3 IF(MOD(NOW,2).EQ.1)GO TO 4
      IF(MOD(LR,2).EQ.1)JMP=4
      IF(MOD(LR,2).EQ.0)JMP=3
      GO TO 6
    4 IF(MOD(LR,2).EQ.1)JMP=1
      IF(MOD(LR,2).EQ.0)JMP=2
      GO TO 6
    5 IF(ICOUNT.LT.0)GO TO 1
      GO TO 3
    6 GO TO(7,8,9,10),JMP
    7 NOW=NOW+1
      CALL NXTROW(ITRNGL,LR,MD,NOW,NEXT,IMTRX,JMTRX)
      RETURN
    8 CALL NXTROW(ITRNGL,LR,MD,NOW,NEXT,IMTRX,JMTRX)
      IF(NEXT.NE.0)NEXT=NEXT+1
      RETURN
    9 NOW=NOW-1
      CALL NXTROW(ITRNGL,LR,MD,NOW,NEXT,IMTRX,JMTRX)
      RETURN
   10 CALL NXTROW(ITRNGL,LR,MD,NOW,NEXT,IMTRX,JMTRX)
      IF(NEXT.NE.0)NEXT=NEXT-1
      RETURN
      END
C
C
C
      SUBROUTINE NXTROW(ITRNGL,LR,MD,J,NBR,IMTRX,JMTRX)
      K=2*(IMTRX-1)*(JMTRX-1)
      LRR=LR
      IF(ITRNGL.EQ.1)GO TO 1
      I1=-1
      I2=-1
      GO TO 2
    1 I1=1
      I2=0
    2 IF(MOD(J,2).EQ.0.AND.MOD(LR,2).EQ.0)LRR=LRR-I1
      IF(MOD(J,2).EQ.1.AND.MOD(LR,2).EQ.1)LRR=LRR-I1
      IF(MOD(LRR,2).EQ.0)GO TO 3
      NBR=2*(LRR+I2)*MD+2-J
      GO TO 4
    3 NBR=2*(LRR+I2)*MD-J
    4 IF(NBR.LT.0)NBR=0
      IF(NBR.GT.K)NBR=0
      LR=(NBR-MOD(NBR,MD))/MD+1
      IF(MOD(NBR,MD).EQ.0)LR=LR-1
      RETURN
      END
C
C
C
C
      SUBROUTINE INTRIN(XX,J,LIST,SWITCH,ITOOL)
      COMMON/ALL/VERT(9,1000),A(1000),B(1000),C(1000),D(1000),
     &     X(50,50),Y(50,50),Z(50,50),SX(1000),SY(1000),
```

```
     &     SZ(1000),ORIGIN(3),RAD(8),TLRNCE,NUMINT(1000),
     &     INTVAR,IMTRX,JMTRX,NUMS(1000),NUM
      DIMENSION PNT1(3),PNT2(3),PNT3(3),PNTI(3)
C
      LOGICAL SWITCH
      SWITCH=.TRUE.
      DIF=DBLE(RAD(ITOOL))**2-DBLE(XX)**2
      IF(DIF.GT.0.0D0)GO TO 1
      SWITCH=.FALSE.
      RETURN
    1 DIF=DSQRT(DIF)
      DO 10 I=1,3
      PNT1(I)=DBLE(VERT(I,LIST))
      PNT2(I)=DBLE(VERT(I+3,LIST))
   10 PNT3(I)=DBLE(VERT(I+6,LIST))
      CALL SIDLEN(PNT1,PNT2,S3)
      CALL SIDLEN(PNT2,PNT3,S1)
      CALL SIDLEN(PNT3,PNT1,S2)
      CALL AREA(S1,S2,S3,AR)
      DA=DBLE(A(LIST))
      DB=DBLE(B(LIST))
      DC=DBLE(C(LIST))
      DD=DBLE(D(LIST))
      DSX=DBLE(SX(J))
      DSY=DBLE(SY(J))
      DSZ=DBLE(SZ(J))
      IF(DA.NE.0.D0)GO TO 20
      PNTI(1)=DSX
      GO TO 30
   20 DCON=(DB**2+DC**2)/DA
      PNTI(1)=(DCON*DSX-DB*DSY-DC*DSZ-DD)/(DA+DCON)
   30 IF(DB.NE.0.D0)GO TO 40
      PNTI(2)=DSY
      GO TO 50
   40 DCON=(DA*DA+DC*DC)/DB
      PNTI(2)=(DCON*DSY-DA*DSX-DC*DSZ-DD)/(DB+DCON)
   50 IF(DC.NE.0.D0)GO TO 60
      PNTI(3)=DSZ
      GO TO 61
   60 DCON=(DB*DB+DA*DA)/DC
      PNTI(3)=(DCON*DSZ-DA*DSX-DB*DSY-DD)/(DC+DCON)
   61 CALL SIDLEN(PNT1,PNTI,SI1)
      CALL SIDLEN(PNT2,PNTI,SI2)
      CALL SIDLEN(PNT3,PNTI,SI3)
C
      CALL AREA(S2,SI1,SI3,ARI1)
      CALL AREA(S3,SI1,SI2,ARI2)
      CALL AREA(S1,SI3,SI2,ARI3)
C
      SUMAR=ARI1+ARI2+ARI3
      COMP=DABS(1.0D0-SUMAR/   (AR))
      IF(COMP.LE..00001D0)GO TO 140
      COMP=SUMAR-ARI1*2.
      COMP=DABS(1.0D0-   (AR)/COMP)
      IF(COMP.LE..00001D0)GO TO 70
      COMP=SUMAR-ARI2*2.
      COMP=DABS(1.0D0-   (AR)/COMP)
      IF(COMP.LE..00001D0)GO TO 80
      COMP=SUMAR-ARI3*2.
      COMP=DABS(1.0D0-   (AR)/COMP)
```

```
      IF(COMP.LE..00001D0)GO TO 90
      COMP=SUMAR-2.*(ARI1+ARI2)
      COMP=DABS(1.0D0-    (AR)/COMP)
      IF(COMP.LE..00001D0)GO TO 110
      COMP=SUMAR-2.*(ARI1+ARI3)
      COMP=DABS(1.0D0-    (AR)/COMP)
      IF(COMP.LE..00001D0)GO TO 120
      COMP=SUMAR-2.*(ARI2+ARI3)
      COMP=DABS(1.0D0-    (AR)/COMP)
      IF(COMP.LE..00001D0)GO TO 130
      GO TO 140
   70 RECT=S2
      ARE=ARI1
      GO TO 100
   80 RECT=S3
      ARE=ARI2
      GO TO 100
   90 RECT=S1
      ARE=ARI3
  100 RECT=RECT*DIF/2.
      IF(ARE.GE.RECT)SWITCH=.FALSE.
      GO TO 140
  110 IF(SI1.GT.DIF)SWITCH=.FALSE.
      GO TO 140
  120 IF(SI3.GT.DIF)SWITCH=.FALSE.
      GO TO 140
  130 IF(SI2.GT.DIF)SWITCH=.FALSE.
  140 RETURN
      END
C
C
C
      FUNCTION DIST(NOW,JDO,ITOOL)
      COMMON/ALL/VERT(9,1000),A(1000),B(1000),C(1000),D(1000),
     &     X(50,50),Y(50,50),Z(50,50),SX(1000),SY(1000),
     &     SZ(1000),ORIGIN(3),RAD(8),TLRNCE,NUMINT(1000),
     &     INTVAR,IMTRX,JMTRX,NUMS(1000),NUM
      DIMENSION E(3)
      L=0
      DO 1 J=1,7,3
      L=L+1
    1 E(L)=(SY(NOW)-VERT(J+1,JDO))**2+(SX(NOW)-VERT(J,JDO))**2
      DIST=MIN1(E(1),E(2),E(3))-RAD(ITOOL)**2
      RETURN
      END
C
C SIDLEN CALCULATES SIDE OF A TRIANGLE GIVEN THE TWO VERTICES
C
      SUBROUTINE SIDLEN(PNT1,PNT2,DIS)
      REAL PNT1(3),DIS,PNT2(3)
      DIS=0.D0
      DO 10 I=1,3
   10 DIS=DIS+(PNT1(I)-PNT2(I))**2
      DIS=DSQRT(DIS)
      RETURN
      END
C
C AREA CALCULATES THE AREA OF A TRIANGLE GIVEN 3 SIDES
C
      SUBROUTINE AREA(S1,S2,S3,AR)
```

```
      S=(S1+S2+S3)/2.
      AR=S*(S-S1)*(S-S2)*(S-S3)
      IF(AR.GT.0.D0)GO TO 1
      AR=0.D0
      GO TO 2
    1 AR=DSQRT(AR)
    2 RETURN
      END
C
C
C
C
      SUBROUTINE TLCENT(K,ITOOL)
      COMMON/ALL/VERT(9,1000),A(1000),B(1000),C(1000),D(1000),
     &    X(50,50),Y(50,50),Z(50,50),SX(1000),SY(1000),
     &    SZ(1000),ORIGIN(3),RAD(8),TLRNCE,NUMINT(1000),
     &    NUMVAR,IMTRX,JMTRX,NUMS(1000),NUM
      DO 100 L=1,K
      XT=(VERT(1,L)+VERT(4,L)+VERT(7,L))/3.0
      YT=(VERT(2,L)+VERT(5,L)+VERT(8,L))/3.0
      ZT=(VERT(3,L)+VERT(6,L)+VERT(9,L))/3.0
      SX(L)=XT+RAD(ITOOL)*A(L)
      SY(L)=YT+RAD(ITOOL)*B(L)
      SZ(L)=ZT+RAD(ITOOL)*C(L)
  100 CONTINUE
      RETURN
      END
```

```
S//  PROGRAM PREPARE 3-D

THIS PROGRAM ACTS AS A POST-PROCESSOR TO
PRODUCE PART PROGRAM IN WHICH THE OUTPUT
DATA POINTS FROM THE VAX11\750 SUPER-MINICOMPUTER
IS ACCEPIABLE BY THE OLIVETTI NC
MACHINE.

A CONTROL VARIABLE 'K' IS USED TO GENERATE
THE DIFFERENT FORMATS.

K=0,GENERATE PART PROGRAM FOR 3-D MILLING.
K=1,GENERATE PART PROGRAM FOR 2&1/2-D
  MILLING.
K=2,GENERATE PART PROGRAM FOR POINT TO
  POINT MILLING IN X-Y PLANE.
K=3,GENERATE PART PROGRAM FOR POINT TO
  POINT MILLING IN Y-Z PLANE.
K=4,GENERATE PART PROGRAM FOR PLAIN AND
  POCKING MILLING IN X DIRECTION.
K=5,GENERATE PART PROGRAM FOR PLAIN AND
  POCKING MILLING IN Y DIRECTION.


  OPEN(UNIT=5,NAME='DOCUMENT',STATUS='UNKNOWN')
  OPEN(UNIT=1,NAME='DCOMP',STATUS='UNKNOWN')
  OPEN(UNIT=2,NAME='DPRE',STATUS='UNKNOWN')
  COMMON NT
  NT=0


  READ(5,*)J,IF,K
  DO 20 I=1,J
  READ(5,*)N,X,Y,Z
  CALL REMOVE(I,N,X,Y,Z,IF,K)
0 CONTINUE

  READ(1,*)L
  DO 30 I=1,L
  READ(1,*)N,IG
  CALL COMPLETE(N,IG)
0 CONTINUE


  STOP
  END




  SUBROUTINE REMOVE(I,N,X,Y,Z,IF,K)

  COMMON NT
  NT=NT+1
  IF(NT.GT.999)NT=1
  IF(NT.GT.0.AND.NT.LT.10)GO TO 380
  IF(NT.GE.10.AND.NT.LT.100)GO TO 381
  IF(NT.GE.100.AND.NT.LT.1000)GO TO 382
```

```
80  WRITE(2,480)NT
    GO TO 996
81  WRITE(2,481)NT
    GO TO 996
82  WRITE(2,482)NT
    GO TO 996
80  FORMAT(1X,'N00',I1)
81  FORMAT(1X,'N0',I2)
82  FORMAT(1X,'N',I3)

96  IF(I.EQ.1)GO TO 993
    GO TO 992
93  WRITE(2,200)
00  FORMAT('+',7X,'G54',60X,'M03')
    GO TO 992

92  IX=X*(10**4)
    IY=Y*(10**4)
    IZ=Z*(10**4)

    IF(I.EQ.1)GO TO 987
    IF(K.EQ.3)GO TO 999
    IF(K.EQ.4.AND.MOD(I,2).EQ.0)GO TO 999
    IF(K.EQ.5.AND.MOD(I,2).NE.0)GO TO 999
87  IF(IX.GE.0.AND.IX.LT.10)GO TO 300
    IF(IX.GE.10.AND.IX.LT.100)GO TO 301
    IF(IX.GE.100.AND.IX.LT.1000)GO TO 302
    IF(IX.GE.1000.AND.IX.LT.10000)GO TO 303
    IF(IX.GE.10000.AND.IX.LT.100000)GO TO 304
    IF(IX.GE.100000.AND.IX.LT.1000000)GO TO 305
    IX=IX*(-1)
    IF(IX.GT.0.AND.IX.LT.10)GO TO 350
    IF(IX.GE.10.AND.IX.LT.100)GO TO 351
    IF(IX.GE.100.AND.IX.LT.1000)GO TO 352
    IF(IX.GE.1000.AND.IX.LT.10000)GO TO 353
    IF(IX.GE.10000.AND.IX.LT.100000)GO TO 354
    IF(IX.GE.100000.AND.IX.LT.1000000)GO TO 355



00  WRITE(2,400)IX
    GO TO 999
01  WRITE(2,401)IX
    GO TO 999
02  WRITE(2,402)IX
    GO TO 999
03  WRITE(2,403)IX
    GO TO 999
04  WRITE(2,404)IX
    GO TO 999
05  WRITE(2,405)IX
    GO TO 999
50  WRITE(2,450)IX
    GO TO 999
51  WRITE(2,451)IX
    GO TO 999
52  WRITE(2,452)IX
    GO TO 999
53  WRITE(2,453)IX
    GO TO 999
```

```
54  WRITE(2,454)IX
    GO TO 999
55  WRITE(2,455)IX
    GO TO 999



00  FORMAT('+',12X,'X000000',I1)
01  FORMAT('+',12X,'X00000',I2)
02  FORMAT('+',12X,'X0000',I3)
03  FORMAT('+',12X,'X000',I4)
04  FORMAT('+',12X,'X00',I5)
05  FORMAT('+',12X,'X0',I6)
50  FORMAT('+',12X,'X-000000',I1)
51  FORMAT('+',12X,'X-00000',I2)
52  FORMAT('+',12X,'X-0000',I3)
53  FORMAT('+',12X,'X-000',I4)
54  FORMAT('+',12X,'X-00',I5)
55  FORMAT('+',12X,'X-0',I6)

99  IF(I.EQ.1)GO TO 986
    IF(K.EQ.4.AND.MOD(I,2).EQ.1)GO TO 997
    IF(K.EQ.5.AND.MOD(I,2).EQ.0)GO TO 997
86  IF(IY.GE.0.AND.IY.LT.10)GO TO 310
    IF(IY.GE.10.AND.IY.LT.100)GO TO 311
    IF(IY.GE.100.AND.IY.LT.1000)GO TO 312
    IF(IY.GE.1000.AND.IY.LT.10000)GO TO 313
    IF(IY.GE.10000.AND.IY.LT.100000)GO TO 314
    IF(IY.GE.100000.AND.IY.LT.1000000)GO TO 315
    IY=IY*(-1)
    IF(IY.GT.0.AND.IY.LT.10)GO TO 360
    IF(IY.GE.10.AND.IY.LT.100)GO TO 361
    IF(IY.GE.100.AND.IY.LT.1000)GO TO 362
    IF(IY.GE.1000.AND.IY.LT.10000)GO TO 363
    IF(IY.GE.10000.AND.IY.LT.100000)GO TO 364
    IF(IY.GE.100000.AND.IY.LT.1000000)GO TO 365

10  WRITE(2,410)IY
    IF(I.GT.1)GO TO 995
    GO TO 998
11  WRITE(2,411)IY
    IF(I.GT.1)GO TO 995
    GO TO 998
12  WRITE(2,412)IY
    IF(I.GT.1)GO TO 995
    GO TO 998

13  WRITE(2,413)IY
    IF(I.GT.1)GO TO 995
    GO TO 998
14  WRITE(2,414)IY
    IF(I.GT.1)GO TO 995
    GO TO 998
15  WRITE(2,415)IY
    IF(I.GT.1)GO TO 995
    GO TO 998
60  WRITE(2,460)IY
    IF(I.GT.1)GO TO 995
    GO TO 998
61  WRITE(2,461)IY
```

```
      IF(I.GT.1)GO TO 995
      GO TO 998
   62 WRITE(2,462)IY
      IF(I.GT.1)GO TO 995
      GO TO 998
   63 WRITE(2,463)IY
      IF(I.GT.1)GO TO 995
      GO TO 998
   64 WRITE(2,464)IY
      IF(I.GT.1)GO TO 995
      GO TO 998
   65 WRITE(2,465)IY
      IF(I.GT.1)GO TO 995
      GO TO 998
   10 FORMAT('+',22X,'Y000000',I1)
   11 FORMAT('+',22X,'Y00000',I2)
   12 FORMAT('+',22X,'Y0000',I3)
   13 FORMAT('+',22X,'Y000',I4)
   14 FORMAT('+',22X,'Y00',I5)
   15 FORMAT('+',22X,'Y0',I6)
   60 FORMAT('+',22X,'Y-000000',I1)
   61 FORMAT('+',22X,'Y-00000',I2)
   62 FORMAT('+',22X,'Y-0000',I3)
   63 FORMAT('+',22X,'Y-000',I4)
   64 FORMAT('+',22X,'Y-00',I5)
   65 FORMAT('+',22X,'Y-0',I6)

   98 IF(I.EQ.1)GO TO 991
      GO TO 990
   91 NT=NT+1
      WRITE(2,210)NT
  210 FORMAT(1X,'N00',I1,3X,'G55',42X,'R-0009000',
     &     9X,'M08')
      GO TO 990

   90 NT=NT+1
      IF(NT.GT.999)NT=1
      IF(NT.GT.0.AND.NT.LT.10)GO TO 385
      IF(NT.GE.10.AND.NT.LT.100)GO TO 386
      IF(NT.GE.100.AND.NT.LT.1000)GO TO 387
   85 WRITE(2,485)NT
      GO TO 988
   86 WRITE(2,486)NT
      GO TO 988
   87 WRITE(2,487)NT
      GO TO 988
   85 FORMAT(1X,'N00',I1)
   86 FORMAT(1X,'N0',I2)
   87 FORMAT(1X,'N',I3)

   95 IF(K.EQ.1.AND.I.EQ.2)GO TO 391
      IF(K.EQ.1)GO TO 990
      IF(K.EQ.2)GO TO 997
      IF(K.EQ.4)GO TO 997
      IF(K.EQ.5)GO TO 997
   88 IF(IZ.GE.0.AND.IZ.LT.10)GO TO 320
      IF(IZ.GE.10.AND.IZ.LT.100)GO TO 321
      IF(IZ.GE.100.AND.IZ.LT.1000)GO TO 322
      IF(IZ.GE.1000.AND.IZ.LT.10000)GO TO 323
      IF(IZ.GE.10000.AND.IZ.LT.100000)GO TO 324
```

```
      IF(IZ.GE.100000.AND.IZ.LT.1000000)GO TO 325
      IZ=IZ*(-1)
      IF(IZ.GT.0.AND.IZ.LT.10)GO TO 370
      IF(IZ.GE.10.AND.IZ.LT.100)GO TO 371
      IF(IZ.GE.100.AND.IZ.LT.1000)GO TO 372
      IF(IZ.GE.1000.AND.IZ.LT.10000)GO TO 373
      IF(IZ.GE.10000.AND.IZ.LT.100000)GO TO 374
      IF(IZ.GE.100000.AND.IZ.LT.1000000)GO TO 375

   20 WRITE(2,420)IZ
      GO TO 997
   21 WRITE(2,421)IZ
      GO TO 997
   22 WRITE(2,422)IZ
      GO TO 997
   23 WRITE(2,423)IZ
      GO TO 997
   24 WRITE(2,424)IZ
      GO TO 997
   25 WRITE(2,425)IZ
      GO TO 997
   70 WRITE(2,470)IZ
      GO TO 997
   71 WRITE(2,471)IZ
      GO TO 997
   72 WRITE(2,472)IZ
      GO TO 997
   73 WRITE(2,473)IZ
      GO TO 997
   74 WRITE(2,474)IZ
      GO TO 997
   75 WRITE(2,475)IZ
      GO TO 997

   20 FORMAT('+',32X,'Z000000',I1)
   21 FORMAT('+',32X,'Z00000',I2)
   22 FORMAT('+',32X,'Z0000',I3)
   23 FORMAT('+',32X,'Z000',I4)
   24 FORMAT('+',32X,'Z00',I5)
   25 FORMAT('+',32X,'Z0',I6)
   70 FORMAT('+',32X,'Z-000000',I1)
   71 FORMAT('+',32X,'Z-00000',I2)
   72 FORMAT('+',32X,'Z-0000',I3)
   73 FORMAT('+',32X,'Z-000',I4)
   74 FORMAT('+',32X,'Z-00',I5)
   75 FORMAT('+',32X,'Z-0',I6)

   97 IF(I.EQ.1.OR.I.EQ.2)GO TO 390
      GO TO 989
   90 IF(I.EQ.1)GO TO 391
      IF(I.EQ.2)GO TO 392
   91 WRITE(2,491)IF
      IF(K.EQ.1.AND.I.EQ.2)GO TO 990
      GO TO 989
   92 WRITE(2,492)IF
      GO TO 989
   91 FORMAT('+',43X,'F00',I3)
   92 FORMAT('+',43X,'F00',I3)

   89 RETURN
```

PROGRAM S_{12} GENERAL POST-PROCESSOR

```
C
C   THIS PR ROGRAM ACTS AS A GENERAL POST-PROCESSOR APPLICABLE
C    TO ANY CNC MILLING M/C. THE INPUT FOR THIS PROGRAM
C     SHOULD BE THE CUTTER LOCATION DATA FILE.
C
C   A CONTROL VARIABLE 'K' IS USED TO GENERATE
C   THE DIFFERENT FORMATS.
C
C   K=0,GENERATE PART PROGRAM FOR 3-D MILLING.
C   K=1,GENERATE PART PROGRAM FOR 2&1/2-D
C      MILLING.
C   K=2,GENERATE PART PROGRAM FOR POINT TO
C      POINT MILLING IN X-Y PLANE.
C   K=3,GENERATE PART PROGRAM FOR POINT TO
C      POINT MILLING IN Y-Z PLANE.
C   K=4,GENERATE PART PROGRAM FOR PLAIN AND
C      POCKING MILLING IN X DIRECTION.
C   K=5,GENERATE PART PROGRAM FOR PLAIN AND
C      POCKING MILLING IN Y DIRECTION.
    COMMON NT
    PRINT*,'   '
    PRINT*,'   '
    PRINT*,'   '
    WRITE(*,3)
3   FORMAT(//////18X,'-----------------------------------------',
   %       /18X,'|                                       |',
   %       /18X,'| ** WELCOME TO THE GENERAL POSTPROCESSOR ** |',
   %       /18X,'|        FOR CNC MILLING MACHINES       |',
   %       /18X,'|                                       |',
   %       /18X,'|  This program is divided into two parts: |',
   %       /18X,'|  1. Creation of a new postprocessor    |',
   %       /18X,'|     for a CNC MILLING.                |',
   %       /18X,'|  2. Running the program with an       |',
   %       /18X,'|     EXISTing machine data file.       |',
   %       /18X,'|                                       |',
   %       /18X,'|  For Access to part 1 please type  CNP  |',
   %       /18X,'|                2 please tupe  EDF  |',
   %       /18X,'|                                       |',
   %       /18X,'-----------------------------------------')
C
4   PRINT*,'type code =?'
    READ(*,5) PH
5   FORMAT(A3)
    IF(PH.EQ.'CNP') THEN
    CALL CREATE
    ELSE IF(PH.EQ.'EDF') THEN
    CALL EXIST
    ELSE
    WRITE(*,6)
6   FORMAT(////1X,'@@@ INVALID SELECTION PLEASE RETYPE @@@',////)
    GOTO 4
    END IF
C
```

```
C
      NT=0
C
C
      OPEN(UNIT=1,NAME='DCOMP',STATUS='UNKNOWN')
      OPEN(UNIT=5,NAME='DOCUMENT',STATUS='UNKNOWN')
      OPEN(UNIT=2,NAME='OUTPRE',STATUS='UNKNOWN')
      READ(5,*)J,IF,K
      DO 20 I=1,J
      READ(5,*)N,X,Y,Z
      CALL REMOVE(I,N,X,Y,Z,IF,K)
20    CONTINUE
C
      READ(1,*)L
      DO 30 I=1,L
      READ(1,*)N,IG
      CALL COMPLETE(N,IG)
30    CONTINUE
C
C
      STOP
      END
C
C
C
C
C
      SUBROUTINE REMOVE(I,N,X,Y,Z,IF,K)
C
      COMMON NT
      NT=NT+1
      IF(NT.GT.999)NT=1
      IF(NT.GT.0.AND.NT.LT.10)GO TO 380
      IF(NT.GE.10.AND.NT.LT.100)GO TO 381
      IF(NT.GE.100.AND.NT.LT.1000)GO TO 382
380   WRITE(2,480)NT
      GO TO 996
381   WRITE(2,481)NT
      GO TO 996
382   WRITE(2,482)NT
      GO TO 996
480   FORMAT(1X,'N00',I1)
481   FORMAT(1X,'N0',I2)
482   FORMAT(1X,'N',I3)
C
996   IF(I.EQ.1)GO TO 993
      GO TO 992
993   WRITE(2,200)
200   FORMAT('+',7X,'G54',60X,'M03')
      GO TO 992
C
992   IX=X*(10**4)
      IY=Y*(10**4)
      IZ=Z*(10**4)
C
      IF(I.EQ.1)GO TO 987
      IF(K.EQ.3)GO TO 999
      IF(K.EQ.4.AND.MOD(I,2).EQ.0)GO TO 999
      IF(K.EQ.5.AND.MOD(I,2).NE.0)GO TO 999
987   IF(IX.GE.0.AND.IX.LT.10)GO TO 300
```

```
      IF(IX.GE.10.AND.IX.LT.100)GO TO 301
      IF(IX.GE.100.AND.IX.LT.1000)GO TO 302
      IF(IX.GE.1000.AND.IX.LT.10000)GO TO 303
      IF(IX.GE.10000.AND.IX.LT.100000)GO TO 304
      IF(IX.GE.100000.AND.IX.LT.1000000)GO TO 305
      IX=IX*(-1)
      IF(IX.GT.0.AND.IX.LT.10)GO TO 350
      IF(IX.GE.10.AND.IX.LT.100)GO TO 351
      IF(IX.GE.100.AND.IX.LT.1000)GO TO 352
      IF(IX.GE.1000.AND.IX.LT.10000)GO TO 353
      IF(IX.GE.10000.AND.IX.LT.100000)GO TO 354
      IF(IX.GE.100000.AND.IX.LT.1000000)GO TO 355
C
C
C
300   WRITE(2,400)IX
      GO TO 999
301   WRITE(2,401)IX
      GO TO 999
302   WRITE(2,402)IX
      GO TO 999
303   WRITE(2,403)IX
      GO TO 999
304   WRITE(2,404)IX
      GO TO 999
305   WRITE(2,405)IX
      GO TO 999
350   WRITE(2,450)IX
      GO TO 999
351   WRITE(2,451)IX
      GO TO 999
352   WRITE(2,452)IX
      GO TO 999
353   WRITE(2,453)IX
      GO TO 999
354   WRITE(2,454)IX
      GO TO 999
355   WRITE(2,455)IX
      GO TO 999
C
C
C
400   FORMAT('+',12X,'X000000',I1)
401   FORMAT('+',12X,'X00000',I2)
402   FORMAT('+',12X,'X0000',I3)
403   FORMAT('+',12X,'X000',I4)
404   FORMAT('+',12X,'X00',I5)
405   FORMAT('+',12X,'X0',I6)
450   FORMAT('+',12X,'X-000000',I1)
451   FORMAT('+',12X,'X-00000',I2)
452   FORMAT('+',12X,'X-0000',I3)
453   FORMAT('+',12X,'X-000',I4)
454   FORMAT('+',12X,'X-00',I5)
455   FORMAT('+',12X,'X-0',I6)
C
999   IF(I.EQ.1)GO TO 986
      IF(K.EQ.4.AND.MOD(I,2).EQ.1)GO TO 997
      IF(K.EQ.5.AND.MOD(I,2).EQ.0)GO TO 997
986   IF(IY.GE.0.AND.IY.LT.10)GO TO 310
      IF(IY.GE.10.AND.IY.LT.100)GO TO 311
```

```
      IF(IY.GE.100.AND.IY.LT.1000)GO TO 312
      IF(IY.GE.1000.AND.IY.LT.10000)GO TO 313
      IF(IY.GE.10000.AND.IY.LT.100000)GO TO 314
      IF(IY.GE.100000.AND.IY.LT.1000000)GO TO 315
      IY=IY*(-1)
      IF(IY.GT.0.AND.IY.LT.10)GO TO 360
      IF(IY.GE.10.AND.IY.LT.100)GO TO 361
      IF(IY.GE.100.AND.IY.LT.1000)GO TO 362
      IF(IY.GE.1000.AND.IY.LT.10000)GO TO 363
      IF(IY.GE.10000.AND.IY.LT.100000)GO TO 364
      IF(IY.GE.100000.AND.IY.LT.1000000)GO TO 365
C
310   WRITE(2,410)IY
      IF(I.GT.1)GO TO 995
      GO TO 998
311   WRITE(2,411)IY
      IF(I.GT.1)GO TO 995
      GO TO 998
312   WRITE(2,412)IY
      IF(I.GT.1)GO TO 995
      GO TO 998
C
313   WRITE(2,413)IY
      IF(I.GT.1)GO TO 995
      GO TO 998
314   WRITE(2,414)IY
      IF(I.GT.1)GO TO 995
      GO TO 998
315   WRITE(2,415)IY
      IF(I.GT.1)GO TO 995
      GO TO 998
360   WRITE(2,460)IY
      IF(I.GT.1)GO TO 995
      GO TO 998
361   WRITE(2,461)IY
      IF(I.GT.1)GO TO 995
      GO TO 998
362   WRITE(2,462)IY
      IF(I.GT.1)GO TO 995
      GO TO 998
363   WRITE(2,463)IY
      IF(I.GT.1)GO TO 995
      GO TO 998
364   WRITE(2,464)IY
      IF(I.GT.1)GO TO 995
      GO TO 998
365   WRITE(2,465)IY
      IF(I.GT.1)GO TO 995
      GO TO 998
410   FORMAT('+',22X,'Y000000',I1)
411   FORMAT('+',22X,'Y00000',I2)
412   FORMAT('+',22X,'Y0000',I3)
413   FORMAT('+',22X,'Y000',I4)
414   FORMAT('+',22X,'Y00',I5)
415   FORMAT('+',22X,'Y0',I6)
460   FORMAT('+',22X,'Y-000000',I1)
461   FORMAT('+',22X,'Y-00000',I2)
462   FORMAT('+',22X,'Y-0000',I3)
463   FORMAT('+',22X,'Y-000',I4)
464   FORMAT('+',22X,'Y-00',I5)
```

```
C
994   IF(IG.EQ.54)GO TO 500
      IF(IG.EQ.0)GO TO 510
      IF(IG.EQ.55)GO TO 520
C
500   WRITE(2,600)
      GO TO 1100
510   WRITE(2,610)
      GO TO 1100
520   WRITE(2,620)
      GO TO 1100
C
600   FORMAT('+',7X,'G54',41X,'R0000000',12X,'M05')
610   FORMAT('+',12X,'X0000000',2X,'Y0000000')
620   FORMAT('+',7X,'G55',61X,'M30')
1100  RETURN
      END
C
      SUBROUTINE EXIST
C  THIS SUBROUTINE READS THE INPUT OF MACHINE DATA
C  (I.E. G-CODES, M-CODES, AND THE FORMAT FOR NC PROGRAMMING).
C  THE INPUT IS THE MACHINE'S NAME DATA FILE, AND WHICH SHOULD
C  BE READ THROUGH CHANNEL NO.6
C
      REAL COOR(4),OCOOR(4),FEED,OFEED
C
      CHARACTER NEWMAC*30
      INTEGER INCODE,MECODE,NACODE,NRCODE
      INTEGER GCODEA,GCODEB,GCODEC,GCODED
      INTEGER GCODEE,GCODEF,GCODEG,GCODEH
      INTEGER GCODEI,GCODEJ,GCODEK,GCODEL
      INTEGER MCODEA,MCODEB,MCODEC,MCODED,MCODEE,MCODEF
      INTEGER MCODEG,MCODEH,MCODEI,MCODEJ
      INTEGER MCODEK,MCODEL,MCODEM,MCODEN
      CHARACTER*1 KCODEA,KCODEB
      CHARACTER*3 JCODEA,JCODEB
      CHARACTER LCODEA*8,LCOD(8)*1,LCOUT*8
C
      COMMON /BK1/ INCODE,MECODE,NACODE,NRCODE
      COMMON /BK2/ GCODEA,GCODEB,GCODEC,GCODED
      COMMON /BK3/ GCODEE,GCODEF,GCODEG,GCODEH
      COMMON /BK4/ GCODEI,GCODEJ,GCODEK,GCODEL
      COMMON /BK5/ MCODEA,MCODEB,MCODEC,MCODED,MCODEE,MCODEF
      COMMON /BK6/ MCODEG,MCODEH,MCODEI,MCODEJ
      COMMON /BK7/ MCODEK,MCODEL,MCODEM,MCODEN
      COMMON /BK8/ KCODEA,KCODEB
      COMMON /BK9/ JCODEA,JCODEB
      COMMON/ BK10/ LCODEA,LCOD,LCOUT
C
      WRITE(*,200)
200   FORMAT(//////18X,'-----------------------------------------',
     %       /18X,'|                                         |',
     %       /18X,'|  **  RUNNING THE POSTPROCESSOR USING  ** |',
     %       /18X,'|  **      CREATED CNC MILLING FILE     ** |',
     %       /18X,'|                                         |',
     %       /18X,'| This section reads the input of machine |',
     %       /18X,'| data (i.e. G-codes, M-codes, and the    |',
     %       /18X,'| format for NC programming). The input is |',
     %       /18X,'| the machine data file and it is read    |',
     %       /18X,'| through channel No.6                     |',
```

```
   %        /18X,'|                                    |',
   %        /18X,'-------------------------------------------')
C
    PRINT*,'       '
    PRINT*,'       '
    PRINT*,'  ** PLEASE INPUT THE NAME OF THE CNC MILLING **'
    PRINT*,'  ** DATA FILE'
    PRINT*,'  machine name data file = ?'
    READ(*,201) NEWMAC
201  FORMAT(A)
    OPEN(UNIT=6,NAME=NEWMAC,STATUS='UNKNOWN')
C
    READ(6,80) INCODE,MECODE,NACODE,NRCODE
    READ(6,81) GCODEA,GCODEB,GCODEC,GCODED
    READ(6,82) GCODEE,GCODEF,GCODEG,GCODEH
    READ(6,83) GCODEI,GCODEJ,GCODEK,GCODEL
    READ(6,84) MCODEA,MCODEB,MCODEC,MCODED,MOCDEE,MCODEF
    READ(6,85) MCODEG,MCODEH,MCODEI,MCODEJ
    READ(6,86) MCODEK,MCODEL,MCODEM,MCODEN
    READ(6,87) KCODEA,KCODEB
    READ(6,88) JCODEA,JCODEB
    READ(6,89) LCODEA
C
 89   FORMAT(1X,A8)
 88   FORMAT(1X,A1,3X,A1)
 87   FORMAT(1X,A1,3X,A1)
 86   FORMAT(1X,I2,3X,I2,3X,I2,3X,I2)
 85   FORMAT(1X,I2,3X,I2,3X,I2,3X,I2)
 84   FORMAT(1X,I2,3X,I2,3X,I2,3X,I2,3X,I2,3X,I2)
 83   FORMAT(1X,I2,3X,I2,3X,I2,3X,I2)
 82   FORMAT(1X,I2,3X,I2,3X,I2,3X,I2)
 81   FORMAT(1X,I2,3X,I2,3X,I2,3X,I2)
 80   FORMAT(1X,I2,3X,I2,3X,I2,3X,I2)
    RETURN
    END
C
C
    SUBROUTINE CREATE
C  THIS SUBROUTINE READS THE INPUT OF MACHINE DATA(I.E. G-CODES,
C  M-CODES, AND THE FORMAT FOR NC PROGRAMMING).
C  THE INPUT IS CARRIED OUT INTERRACTIVELY VIA THE TERMINAL, AND
C  ALL DATA INPUT IS STORED IN A FILE FOR LATER EXECUTION.
C
C
    REAL COOR(4),OCOOR(4),FEED,OFEED
C
    CHARACTER MACHIN*30
    INTEGER INCODE,MECODE,NACODE,NRCODE
    INTEGER GCODEA,GCODEB,GCODEC,GCODED
    INTEGER GCODEE,GCODEF,GCODEG,GCODEH
    INTEGER GCODEI,GCODEJ,GCODEK,GCODEL
    INTEGER MCODEA,MCODEB,MCODEC,MCODED,MCODEE,MCODEF
    INTEGER MCODEG,MCODEH,MCODEI,MCODEJ
    INTEGER MCODEK,MCODEL,MCODEM,MCODEN
    CHARACTER*1 KCODEA,KCODEB
    CHARACTER*3 JCODEA,JCODEB
    CHARACTER LCODEA*8,LCOD(8)*1,LCOUT*8
C
    COMMON /BK1/ INCODE,MECODE,NACODE,NRCODE
    COMMON /BK2/ GCODEA,GCODEB,GCODEC,GCODED
```

```
      COMMON /BK3/ GCODEE,GCODEF,GCODEG,GCODEH
      COMMON /BK4/ GCODEI,GCODEJ,GCODEK,GCODEL
      COMMON /BK5/ MCODEA,MCODEB,MCODEC,MCODED,MCODEE,MCODEF
      COMMON /BK6/ MCODEG,MCODEH,MCODEI,MCODEJ
      COMMON /BK7/ MCODEK,MCODEL,MCODEM,MCODEN
      COMMON /BK8/ KCODEA,KCODEB
      COMMON /BK9/ JCODEA,JCODEB
      COMMON /BK10/ LCODEA,LCOD,LCOUT
C
      WRITE(*,100)
100   FORMAT(//////18X,'--------------------------------------------',
     %        /18X,'|                                          |',
     %        /18X,'|  ** CREATION OF A NEW POSTPROCESSOR  **  |',
     %        /18X,'|                                          |',
     %        /18X,'|  This section reads the input of machine |',
     %        /18X,'|  data (i.e. G-codes,M-codes,and the format |',
     %        /18X,'|  for NC programming) interactively via   |',
     %        /18X,'|  the terminal. All data input is stored  |',
     %        /18X,'|  in a file for any further execution.    |',
     %        /18X,'|                                          |',
     %        /18X,'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~')
C
      PRINT*,'      '
      PRINT*,'      '
      PRINT*,' ** PLEASE INPUT THE NANE OF THE CNC MILLING **'
      PRINT*,' ** FOR WHICH NC PROGRAMMING IS REQUIRED    **'
      PRINT*,' machine name = ? '
      READ(*,101) MACHIN
101   FORMAT(A)
      OPEN(UNIT=4,NAME=MACHIN,STATUS='UNKNOWN')
C
C
      WRITE(*,102)
102   FORMAT(//////8X,'PLEASE INPUT THE G-CODES FOR MODE',
     %1X,'OF PROGRAMMING :'/
     %10X,'INCH MODE',12X,'METRIC MODE',5X,'PURE ABSOLUTE',
     %4X,'REFERENCE'/
     %6X,'(inch programming)',4X,'(mm programming)',4X,
     %'PROGRAMMING',4X,'POINT SEARCH'/)
C
C  INPUT THE RELEVANT G-CODES FOR MODE OF PROGRAMMING.
C  INCH MODE  - INCODE
C  METRIC MODE  - MECODE
C  PURE ABSOLUTE PROGRAMMING  - NACODE
C  REFERENCE POINT SEARCH  - NRCODE
C
      READ*,INCODE,MECODE,NACODE,NRCODE
      WRITE(4,103) INCODE,MECODE,NACODE,NRCODE
103   FORMAT(1X,4(I2,3X))
C
C
      WRITE(*,104)
104   FORMAT(//////8X,'PLEASE INPUT THE G-CODES FOR LINEAR & ',
     %'CIRCULAR INTERPOLATION :'/
     %10X,'RAPID LINEAR',8X,'LINEAR',9X,'CIRCULAR CW',5X,'CIRCULAR
CCW'/
     %10X,'INTERPOLATION',4X,'INTERPOLATION',4X,'INTERPOLATION',
     %4X,'INTERPOLATION'/)
C
C  INPUT THE RELEVANT G-CODES FOR LINEAR & CIRCULAR
```

```
INTERPOLATION.
C  RAPID LINEAR INTERPOLATION  - GCODEA
C  LINEAR INTERPOLATION  - GCODEB
C  CIRCULAR INTERPOLATION (CW)  - GCODEC
C  CIRCULAR INTERPOLATION (CCW)  - GCODED
C
     READ*,GCODEA,GCODEB,GCODEC,GCODED
     WRITE(4,105) GCODEA,GCODEB,GCODEC,GCODED
 105   FORMAT(1X,4(I2,3X))
C
C
     WRITE(*,106)
 106   FORMAT(//////8X,'PLEASE INPUT THE G-CODES FOR THE FOLLOWING ',
    %'CUTTING CYCLES :'/
    %10X,'SPINDLE CUTTING',4X,'SPINDLE',4X,'DWELL',                /
    %11X,'SURFACE SPEED',6X,'SPEED',5X,'CYCLE',8X,'CYCLE'/)
C
C  INPUT THE G-CODES FOR THE FOLLOWING CUTTING CYCLES.
C  SPINDLE CUTTING SURFACE SPEED  - GCODEE
C  SPINDLE SPEED  - GCODEF
C  DWELL CYCLE  - GCODEG
C
     READ*,GCODEE,GCODEG,GCODEH
     WRITE(4,107) GCODEE,GCODEF,GCODEG,GCODEH
 107   FORMAT(1X,4(I2,3X))
C
C
     WRITE(*,108)
 108   FORMAT(//////8X,'PLEASE INPUT THE G-CODES FOR FEED &',
    %' RADIUS COMPENSATION :'/
    %10X,'FEED PER',7X,'FEED PER',7X,'RADIUS COMPENSATION',
    %3X,'RADIUS COMPENSATION'/
C
C  INPUT THE RELEVANT G-CODES FOR FEED & RADIUS COMPENSATION.
C  FEED PER TIME  - GCODEI
C  FEED PER SPINDLE ROTATION  - GCODEJ
C  RADIUS COMPENSATION (TOOL TO THE LEFT)  - GCODEK
C  RADIUS COMPENSATION (TOOL TO THE RIGHT)  - GCODEL
C
     READ*,GCODEI,GCODEJ,GCODEK,GCODEL
     WRITE(4,109) GCODEI,GCODEJ,GCODEK,GCODEL
 109   FORMAT(1X,4(I2,3X))
C
C
     WRITE(*,110)
 110   FORMAT(//////8X,'PLEASE INPUT THE M-CODES FOR SPINDLE &',
    %' COOLANT ON/OFF :'/
    %10X,'SPINDLE',4X,'SPINDLE',4X,'COOLANT',3X,'SPINDLE',3X,
    %'SPINDLE FORWARD',3X,'SPINDLE OFF'/
    %10X,'FORWARD',4X,'REVERSE',6X,'ON',7X,'STOP',5X,'& COOLANT ON',
    %3X,'& COOLANT OFF'/)
C
C  INPUT THE RELEVANT M-CODES FOR SPINDLE & COOLANT ON/OFF.
C  SPINDLE FORWARD  - MCODEA
C  SPINDLE REVERSE  - MCODEB
C  COOLANT ON  - MCODEC
C  SPINDLE STOP  - MCODED
C  SPINDLE FORWARD & COOLANT ON  - MCODEE
C  SPINDLE OFF & COOLANT OFF  - MCODEF
C
```

```
      READ*,MCODEA,MCODEB,MCODEC,MCODED,MCODEE,MCODEF
      WRITE(4,111) MCODEA,MCODEB,MCODEC,MCODED,MCODEE,MCODEF
  111 FORMAT(1X,6(I2,3X))
C
C
      WRITE(*,112)
  112 FORMAT(//////8X,'PLEASE INPUT THE M-CODES FOR PROGRAM STOP/END
     :'/
     %10X,'PROGRAM',4X,'OPTIONAL',4X,'END OF',5X,'END OF PROGRAM'/
     %11X,'STOP',8X,'STOP',6X,'PROGRAM',5X,'WITH REWIND'/)
C
C INPUT THE RELEVANT M-CODES FOR PROGRAM STOP/END.
C PROGRAM STOP  - MCODEG
C OPTIONAL STOP  - MCODEH
C END OF PROGRAM  - MCODEI
C END OF PROGRAM WITH REWIND  -MCODEJ
C
      READ*,MCODEG,MCODEH,MCODEI,MCODEJ
      WRITE(4,113) MCODEG,MCODEH,MCODEI,MCODEJ
  113 FORMAT(1X,4(I2,3X))
C
C
      WRITE(*,114)
  114 FORMAT(//////8X,'PLEASE INPUT THE M-CODES FOR THE SPINDLE
     SPEED',
     %' RANGE :'/
     %14X,'-LOW SPEED RANGE-',22X,'-HIGH SPEED RANGE-'/
     %8X,'(Lower range)'4X,'(Higher range)',10X,'(Lower range)',
     %4X,'(Higher range)'/)
C
C INPUT THE M-CODES FOR THE SPINDLE SPEED RANGE.
C LOW SPEED RANGE   - Lower range - MCODEK
C                   - Higher range - MCODEL
C HIGH SPEED RANGE  - Lower range - MCODEM
C                   - Higher range - MCODEN
C
      READ*,MCODEK,MCODEL,MCODEM,MCODEN
      WRITE(4,115) MCODEK,MCODEL,MCODEM,MCODEN
  115 FORMAT(1X,4(I2,3X))
C
C
      WRITE(*,116)
  116 FORMAT(//////8X,'PLEASE INPUT THE CODES WHICH REPRESENT THE'/
     %8X,'X & Z COORDINATES OF THE CENTRE POINT OF AN ARC :'/
     %15X,'X-COORDINATE CODE',10X,'Z-COORDINATE CODE'/)
C
C INPUT THE CODES WHICH REPRESENT THE X & Z COORDINATES OF
C THE CENTRE POINT OF AN ARC.
C X-COORDINATE CODE  - KCODEA
C Z-COORDINATE CODE  - KCODEB
C
      READ*,KCODEA,KCODEB
      WRITE(4,117) KCODEA,KCODEB
  117 FORMAT(1X,2(A1,3X))
C
C
      WRITE(*,118)
  118 FORMAT(//////8X,'PLEASE INPUT THE CODES FOR CALLING THE'/
     %8X,'TOOL OFFSET AND COMPENSATION NUMBER :'/
     %15X,'TOOL NUMBER',15X,'COMPENSATION NUMBER'/)
```

```
C
C  INPUT THE CODES FOR THE TOOL OFFSET AND COMPENSATION NUMBER.
C  TOOL NUMBER  - JCODEA
C  COMPENSATION NUMBER  - JCODEB
C
     READ*,JCODEA,JCODEB
     WRITE(4,119) JCODEA,JCODEB
 119  FORMAT(1X,2(A1,3X))
C
C
     WRITE(*,120)
 120  FORMAT(//////8X,'PLEASE INPUT THE CODE FOR CANCELLING THE',
    %' TOOL OFFSET :'/
    %8X,'(Only 1 entry)'/)
C
C  INPUT THE CODE FOR CANCELLING THE TOOL OFFSET.
C  CODE  - LCODEA
C
     READ*,LCODEA
     WRITE(4,121) LCODEA
 121  FORMAT(1X,A8)
C
C
     RETURN
     END
```

REFERENCES

REFERENCES

1. Rathmill, K., Sackett, P.J. "Manufacturing Plant for 1985 - Developments and Justification", Proc. Instn. Mech Eng., Vol. 196, No.17, 1982, pp 265 - 280.

2. Marshall, P. "The Requirements of an Integrated CAD/CAM System", International Conference on Computer Aided Engineering, University of Warwick, Dec 1984, pp 42 to 47.

3. Hannan, R.G., Plummer, J.C.S. "Capturing Production Engineering Practice within a within a CAD/CAM System", Int. J. Prod. Res., Vol. 22, No. 2, 1984, pp 267 to 280.

4. Potts, D. "Integration the Key Success", Machinery and Production Engineering, June 1984.

5. Acard "Computer Aided Design and Manufacture", Advisory Council for Applied Research and Development, Cabinet Office, HMSO, U.K., 1980.

6. Bezier P., "Numerical Control", Mathematics and Applications, John Wiley and Sons Ltd., 1982.

7. Braid I., "Designing with Volumes", Ph.D. Thesis, Cambridge university, U.K. 1973.

8. Braid I., "New Direction in Geometric Modelling", CAD Group Document No.98, Computer Laboratory, University of Cambridge, U.K. March 1978.

9. "PADL System Documentation", Production Automation Product, University of Rochester, New York. 1977.

10. Okino N. et al., "TIPS-1", Institute of Precision Engineering, Hokkaido University, Sapporo, Japan 1978.

11. CAM-I, "An Interface Between Geometric Modelling and Application Programs", CAM-I Report R-80-GM-04, DEC. 1980.

12. Ferguson, N.C. "A History of Numerical Control. British Numerical Control Society Vol. 10, No. 1, Feb. 1979, pp 18 to 19.

13. Oleston, O.N. "Numerical Control" Wiley Interscience Pub., 1970.

14. Simon, W. "The Numerical Control of Machine Tools", Edward Arnold Pub. 1973.

15. Hollingum, J. "Microprocessor Takes Over". The Engineer, Vol. 243, Oct. 1976, pp 30 to 32.

16. Wightman, E.J. "Computer Control System for Machine Tools". Chartered Mechanical Engineer, March 1974.

17. Martin, S.J. "Numerical Control of Machine Tools", English University Press Ltd, 1970.

18. Plas, J. Blommaert, J. "A Stepping Motor Drive Assembly Especially Designed for DNC Systems. 13th International Machine Tool Design and Research Conference, 1972, pp 183 to 189.

19. Poulos, D.G. "Some Aspects of On-line Computer Control of a Lathe", Ph.D Thesis, Bristol Polytechnic, Dec. 1979.

20. Puckle, O.S. Arrowsmith, J.R. "An Introduction to NC Machine Tools", Chapman and Hall Pub, 1968.

21. Garcia, G. "Developments in Numerical Control", Engineers's Digest, Vol. 41, No.11 Nov 1980, pp 25 to 29.

22. Barnett, C. Davies, C. "A review of Developments" Syposium on Computer Controlled Machine Tools, Birnie Hill Institute, NEL, East Kilbride, Sept. 1971.

23. Milner, D.A Oliver, A.F. "Computer Involvement in Numerical Control of Machine Tools. The Prod. Eng., Aug/July 1974, pp 240 to 247.

24. Savage, R. "The Influence of Microprocessors on the Evolution of Numerical Control Systems". 14th Int. M.T.D.R. Conf. 1973, p143 to 147.

25. Carter, W.A. "Computer Numerical Control Software. 14th Int. M.T.D.R. Conf., 1973, pp 359 to 365.

26. MTIRA Report "Getting the Most Out of CNC. The Prod. Eng., Oct 1977, pp 39 to 40.

27. Walters, S. "Superautomation - A View from the Factory Floor", Mechanical Engineering, Jan 1985, pp 44 to 52.

28. Milner, D.A. Rackic, M. "Software Program Implementation in DNC System", Autumn Control Conf. 15th Proc. University of Texas, Austin, June 1974, pp 675 to 686.

29. Barden, W. Jr. "The Z-80 Microcomputer Handbook", Howard W. Sams & Co. Inc., 1978.

30. Bolinger, J. "The Role of Microprocessors in Future of CNC Systems". Journal of the CIRP Vol.25, No. 1. 1976 pp 323 to 328.

31. Kean, C.G. Savage, R. "Microprocessor in Machine Operation". The Prod. Eng., Vol. 56, No.4, April 1977, pp 32 - 36.

32. Van Brussel, H. Simons. J. "Microprocessor in Hierarchical Control Systems", Annals of CIRP, Vol. 271, 1978, pp 265 to 269.

33. Pritty, D.W. "A Continuous Path Microprocessor Numerical Control System. Dept. of Computer Science, University of Strathclyde, Livingstone Tower, Glasgow, 1979.

34. Tipton, H. "Numerical Control ", Frontiers of Technology, Engineer's Digest, 1979, pp 69 to 74.

35. Partridge, J., Clark, J., Mucci, P. "The Integration of Design and Manufacture through CAE Systems Theory and Practice", International Conference on Computer Aided Engineering, University of Warwick, Dec 1984, pp 124 to 127.

36. Gindy, N.N.Z., Vasiliou, V.C. "A Case Study for the Implementation of an FMS for Medium Size Manufacturing Employing Conventional Machinery", 4th Polytechnics Symposium on Manufacturing Engineering, May 1984.

37. Duncan, J.P., Mair, S.G. "Polyhedral NC Program Documentation Report, Dept. of Mechanical Eng., University of British Columbia, Vancouver, Canada, 1975.

38. Allan III, J.J. "Foundation of the Many Manifestation of Computer Augmented Design. IFIP Working Conference on Principles of Computer Aided Design. Eindhoven, The Netherlands, Oct. 1972, pp 29 to 58.

39. Newman, W.M. "Principles of Interactive Graphics", McGrawhill Book McGrawhill Book Company, New York, 1973.

40. Ferguson, J.C. "Multivariable Curve Interpolation, Journal ACM, 11,2, 1964, pp 221-228.

41. Curry, H.B. Schoenberg, I.J. "On Polya Frequency Functions IV, The Fundamental Spline Functions and Their Limits, J. Analy, Math., 17, 1966, pp 71 - 107.

42. Coons, S.A. "Surfaces for Computer Aided Design of Space Forms", Report MAC-TR-41, Project MAC, M.I.T., 1967.

43. Bezier, P. "Example of an Existing System in Motor Industry, The UNISURF System, Proc. Royal Society London, A321, 1971, p 207 to 218.

44. Gordon, W.J. Riesenfeld, R.F. "B-splines Curves and Surfaces", Computer Aided Geometric Design", Academic Press, 1974, pp 95 to 123.

45. Schwelkert, D.G. "An Interpolation Curve Using a Spline in Tension. J. Math. & Phys. 45, 1966, pp 312 to 317.

46. Mehlum, E. "Curve and Surface Fitting Based on a Variational Criterion for Smoothness. Central Institute of Industrial Research, Oslo, 1969.

47. Sabin, M.A. "An Existing System in the Aircraft Industry. The British Aircraft Corporation", Numerical Master Geometry System, PROC, Roy. Soc. Lond., A321, 1971, pp 197 to 205.

48. Flutter, A.G. Rolph, R.N. "POLYSURF: An Interactive System for the Computer - Aided Design and Manufacture of Components. CAD '76 Proceedings, 1976, pp 150 to 158.

49. Bloor, Susan.M., De Pennington, A., Swift, J.S., Tan, S.T. "Interactive Design and Manufacture of Sculptured Surfaces Using the B-spline Approach. "Proceedings of the Twentieth International Machine Tool Design and Research Conference, Sept. 1979, pp 3 to 15.

50. Kishi, H., Oyake, I., Shimura, Y., Nagal, K., Hatta, T. "The Oklsurf System", Proc. 11th Ann. Meeting of Numerical Control Soc. Toronto Canada, March 1974.

51. Forrest, A.R. "Mathematical Principles for Curve and Surface Representation In Curved Surfaces in Engineering", Proc. Conference at Churchill College, Cambridge, 1972, IPC Science and Technology Press Ltd.

52. Higdon, A., Ohlsen, E., Stiles, W., Weese, J. "Mechanics of Materials", 2nd Edition John Wiley & Sons Inc., New York, 1967.

53. Schoenberg, I.J. "Contributions to the Problem of Approximation of Equidistant Data by Analytic Functions. Quartery of Applied Math., V4, 1946, pp 45.

54. Groover, M.P., Zimmers, E.W. "CAD/CAM: Computer Aided Design and Manufacture", Prentice-Hall, 1984.

55. Nutborne, A.W. "A Cubic Spline Package, Part 2 - The Mathematics", Computer Aided Design, V5, No. 1, Jan 1973, pp 7 to 13.

56. Adams, J.A. "A Comparison of Methods for Cubic Spline Curve Fitting. Computer Aided Design, Vol. 6, 1974, p 9.

57. L. Piegl "Recursive Algorithms for Representation of parametric Curves and Surfaces", CAD, Vol. 17, No.5, June 1985.

58, Forrest, A.R. "Curves and Surfaces for Computer Aided Design", Ph.D. Thesis, University of Cambridge, 1968.

59. Forrest, A.R. "Coons Surfaces Appendix to Numerical Control", by Bezier, John Wiley & Son, 1972, pp 228 to 235.

60. APT Encyclopedia Published by UNIVAC, 1963.

61. Gordon, W.J. "Spline Blended Surface Interpolation Through Curve Networks. J. Math. Mech. 18, 10, 1968. pp 931 to 952

62 "VAX/VMS-11 Fortran Language Reference Manual", Digital Equipment Corporation, Oct 1983.

63. "GINO-F (General Purpose Graphics Package)" Users Guide, CAD Centre, Issue 2.7A. Oct 1983.

64. de-Boor, C. "On Calculating with B-splines", J. Approx. Th.,6 1972, pp 50 to 62.

65. Cox, M.G. "The Numerical Evaluation of B-splines". NPL DNAC 4 Report, National Physical Laboratory, Teddington, Middlesex, Aug. 1971.

66. Wilson, E.N. "Cubic Splines on Uniform Meshes. Communication of the ACM", Vol 13, No. 4, April 1970, pp 255 - 258.

67. Hayes, J.G., Halliday, J. "The Least-Squares Fitting of Cubic Spline Surfaces to General Data Sets", J. Institute of Math, and Its Application, 14, 1974, pp 89 - 103.

68. James, M.J., Smith, G.M., Wolford, J.C. "Applied Numerical Methods for Digital Computation with FORTRAN and CSMP" Harper & Row Publishing Inc., 2nd Edition, 1977.

69. James, C. Cavendish "An Approach to automatic 3-D Finite Element Mesh Generation", Int-J. Numerical Methods in Engineering, Vol.21, pp 392-347 (1985).

70. Martin, E.N. "Man-Machine Communication in Three Dimensions. In Computer Aided Geometric Design, Academic Press, New York, 1974, pp 303-315.

71. Duncan, J.P., Mair, S.G. "Polyhedral NC Machining Report", Dept. of Mech. Eng. University of British Columbia, Vancouver, Canada, 1975.

72. Black, J., Culley, S.J. "Introducing CAE - Get the Philosophy Right First", International Conference on Computer Aided Engineering, University of Warwick, Dec 1984, pp 5 to 10.

73. Horowitz, E. "Practical Strategies for Developing Large Software Systems", Addison - Wesley Publishing Co. Inc., 1975.

74. Razavi, S.E., Milner, D.A. "Design and Manufacture of Free-form Surfaces by Cross-sectional Approach". The University of Aston in Birmingham, England, 1981, J.Man.Sys. V2. No.1, pp 69 to 77.

75. Logan, F. "Automated Process Planning - The Route to CIM", The Production Engineer, Sept, 1984, pp 44 to 46.

76 "Computer Assisted Part-Programming - Its link to CAD/CAM", Production Engineer, May 1983, pp 12 to 13.

77. Adams, K.G. French "Automation of Post-Processors to Increase Productivity", Proc, of 23rd International MTDR Conference, 1982, pp 141 to 148.