A COMPUTER AIDED MACHINING SYSTEM FOR THE MANUFACTURE OF THREE-DIMENSIONAL SURFACES

David Reuben Back, BSc (Eng), Cape Town

September 1988

Submitted to the University of Cape Town in partial fulfilment for the degree of Master of Science in Engineering.

I, David Reuben Back, submit this thesis in partial fulfilment of the requirements for the degree of Master of Science in Engineering. I affirm that this is my original work and that it has not been submitted in this or in a similar form for a degree at any University.

ABSTRACT

This thesis describes the design of a computerised system for machining three-dimensional surfaces, more specifically "sculptured surfaces", using an IBM PC and a Bridgeport 3 axis Computer Numerically Controlled (CNC) Milling Machine.

There was a twofold objective to the project:

- To design and develop a milling system
- To provide a teaching tool for undergraduate students who wish to further their studies in the concepts of Computer Aided Design and Manufacture (CAD/CAM).

Five broad areas are covered:

1. The development of a mathematical algorithm to approximate a sculptured surface represented by a set of surface datapoints in the form of XYZ coordinates. These datapoints can either be measured from an existing surface or calculated from a mathematical surface equation. The algorithm links sets of four datapoints to form a mesh of small quadrilateral elements, similar to the well-known "patch methods". On sculptured surfaces these elements usually form warped quadrilaterals and the surface of the elements is known as a hyperbolic paraboloid. The surface of each element, bounded by four straight lines joining the datapoints, is then approximated by four triangular planes which closely approximate the true surface. The author believes this to be a unique method of surface modelling and one which has proved very successful in machining a variety of complex surfaces.

2. The development of an algorithm to calculate the toolpath to machine the sculptured surface. The algorithm uses the plane equation of each triangular plane to calculate the

offset position of a hemispherical milling tool to machine that plane. Every triangular plane is processed in this way, and a list of toolpositions is produced. The algorithm then sorts the toolpositions into the correct sequence to form the toolpath describing the movement of the milling tool over the surface.

3. The development of an algorithm to display the surface and toolpath in three dimensions on the computer screen. The program allows the surface to be displayed from any viewpoint in three dimensions and has three commonly used engineering views preprogrammed.

4. The development of an algorithm (the postprocessor) which translates the toolpath into a program suitable for use with the CNC milling machine.

5. The development of a communications algorithm for the direct transmission of the CNC program from the IBM microcomputer to the memory of the CNC computer.

Each of the algorithms was implemented in a computer program using the TRUE BASIC language. A commercially available add-on module, TRUE BASIC 3D Graphics, was used for part of the viewer. Certain existing software, MIRROR, was used for the communications program.

The system is fully operational and was used to machine a variety of surfaces, a typical one being illustrated overleaf.

An indication of the accuracy of the milling system was obtained by milling a mathematically generated test surface whose shape is typical of that likely to be encountered by the milling system. The milled surface was measured by means of a stereo microscope, and an analysis of the data was performed. The analysis revealed an average undercutting

error of 0.46 mm +- 0.20 mm over the entire surface. Although this error is large in terms of conventional machining requirements, it is acceptable for the manufacture of certain biomedical components - the main application for the system.

Although the system was a success in term of the objectives of this thesis, there are areas where further work is required; the most important being the speeding-up of the data transfer from the IBM microcomputer to the CNC computer and the mathematical algorithms. In addition, more surfaces should be milled and measured, varying both the cutter and element sizes, to provide a better indication of the accuracy of the system.

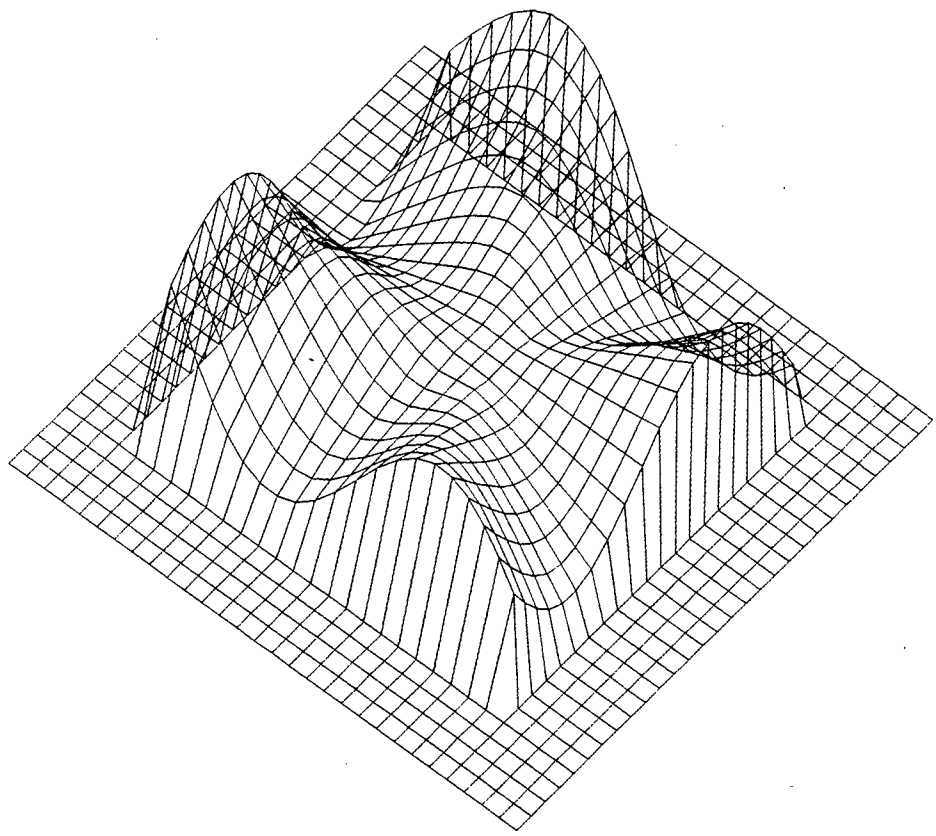


Plate 1: Mathematically generated surface which was machined by the Milling System

ACKNOWLEDGEMENTS

The author would like to gratefully acknowledge the contribution of the following:

Mr. Sydney Back, for his constant interest and support,

My supervisor, Mr. Andrew Sass, for his patience, enthusiasm and guidance, and who was always willing to give of his time,

Mr. Andrew Yates, for his help with the statistical analysis

Miss. Anne Tregdiga, who assisted with the measurements of the various surfaces

Mrs. Jill Martin, who kindly proofread the thesis,

The CSIR, for their financial assistance,

The staff of the Mechanical Engineering Department.

CONTENTS

Page

Page

APPENDICES Page

Page

LIST OF ILLUSTRATIONS

FIGURES Page

FIGURES Page

FIGURES Page

TABLES Page

CHAPTER ONE

1. INTRODUCTION

The need for the duplication of the three-dimensional (3D) shape of objects arises in many fields. For example, the duplication of prostheses with complex surfaces has applications which include shoe inserts and sockets for amputee stumps. Industry often requires products with complex shapes such as in die-making where intricate dies are used for moulding plastic products, and in the automobile industry where car body parts are pressed using dies with complicated shapes and surfaces.

Many of these surfaces can be described as "sculptured surfaces"; that is, surfaces which cannot easily be represented by mathematical equations. Duncan and Mair [1] define sculptured surfaces as "those surface shapes which cannot be continuously generated and have the arbitrary or complex character of the forms traditionally modelled by sculptors". These surfaces are typical of the anatomical surfaces commonly encountered in the Biomedical Engineering field during the manufacture of prostheses. It is important that the body part be measured accurately, but it should also cause the patient the least possible discomfort. The surface of the prosthesis should conform as closely as possible to the measured shape, and therefore the accuracy of machining is important. To this end there is a need for a system capable of measuring and machining 3D sculptured surfaces.

In 1986 Berelowitz [2] in his final year BSc thesis devised a rudimentary system able to measure and machine 3D surfaces. The system comprised three parts:

1. A digital camera called a "TopoScanner" [3] that measures the 3D topography of a surface of maximum size 600 mm x 400 mm x 200 mm.
2. An IBM PC microcomputer to perform the data capture and data processing necessary to mathematically define the surface.
3. A computer numerically controlled (CNC) milling machine for the manufacture of the surface.

The TopoScanner was previously developed in the Biomedical Engineering Department by Vaughan, Brooking, Price and Ireland [3]. It was used to measure surfaces and to create a set of X, Y and Z coordinates representing the surface. This data was fed into an IBM PC microcomputer and mathematical techniques were used to fit a surface through the data points. A second computer program determined the path and sequence a hemispherically ended cutter had to follow to visit points on the surface. These points were then translated by a third program to produce the commands required by the CNC milling machine to produce the surface. The computer was linked to a teletypewriter and the commands were then sent electronically to the teletypewriter which produced a punched tape of the commands for programming the CNC machine.

Although the system was a success from a conceptual point of view, it was not able to reproduce shapes with any great accuracy. There were fundamental problems with the TopoScanner which resulted in inaccurate surface measurements. The two-stage communication of the CNC program from the IBM PC to the CNC computer via paper tape proved to be extremely slow and tedious requiring four or five 25 metre rolls of tape to hold the data for even the simplest of surfaces, which proved very cumbersome. In addition, unacceptably large errors of greater than 2.5 mm occurred in the final milled surface owing to certain assumptions made in the surface fitting algorithms.

On the basis of the difficulties experienced by Berelowitz [2] and because the equipment is highly specialised, it was decided to divide the project into two parts: the development of an automated non-contact 3D measuring device to be undertaken by the Surveying Department, and the development of a milling system capable of machining 3D sculptured surfaces to be undertaken by the Mechanical Engineering Department.

Thus far the Surveying Department have been successful in implementing a prototype system using two digital cameras capable of measuring discrete points in space, but are not yet able to measure 3D surfaces.

The principal objective of this thesis was two-fold:

- to improve the surface fitting and milling routines developed by Berelowitz; more specifically to design a system capable of milling any single-valued 3D sculptured surface - that is a surface without undercut or overhang.

- to develop a system which can serve as an introduction to Computer Aided Manufacture for undergraduate students who wish to further their studies in this specialised field. The mathematical method should therefore be understandable to students from their second year onwards.

The requirements were :

1. A real-time 3D graphical representation of the surface and the toolpath should be provided on the computer screen for verification purposes.
2. The system should produce syntactically correct commands suitable for the Bridgeport CNC machine. The milling machine has 3D point-to-point capabilities, and data is conventionally sent to the TEXTRON controller via a teletype and/or paper tape.

3. There should be an electronic link between the IBM PC and the CNC host computer allowing direct communication of the commands without having to produce a punched paper tape was to be provided.
4. The system should be implemented on an IBM PC microcomputer.
5. A quantitative indication of the error in the machining process should be given.
6. The program should be written in such a way that it can be used to realise the long term objective, which is to develop a complete system able to measure without contact and machine 3D surfaces.

Other desirable features were :

1. The program should be user-friendly.
2. The system should operate with as little human intervention as is practically possible.
3. Any errors occurring during the operation of the program should be clearly explained by the program.
4. The resulting CNC program should be able to be stored on floppy disks.
5. The system should also be able to mill mathematically defined surfaces as well as measured surfaces.

Thesis outline

Chapter 2 of this thesis begins by reviewing a selection of currently available systems capable of milling 3D surfaces. These are mostly integrated Computer Aided Design and Manufacture (CAD/CAM) systems, some of which run on the IBM PC, but most of which use more powerful computers. There is also an account of some of the older systems which have contributed towards the development of the present 3D CAD/CAM.

This is followed by a brief history of early mathematical methods used for surface definition, otherwise known as numerical geometry, and a description of alternative methods for the mathematical treatment of the surface. The suitability of each method is assessed in terms of the objectives defined above.

Chapter 3 describes in detail the four parts of the milling system that were designed to achieve the objectives.

The accuracy of the milling system was determined by milling a mathematically defined test surface and then analysing measured surface datapoints. Chapter 4 describes the analysis of these datapoints, and gives a quantitative estimate of the errors in the milling system.

The final chapter contains the conclusions and recommendations for further improvements of the system.

CHAPTER TWO

2. LITERATURE SURVEY

2.1: A Review of Existing Three-Dimensional Milling Systems

APT

There are many existing systems that are able to mill 3D surfaces. Perhaps the best known program is Automatic Programming of Tools, commonly known as APT, which was developed at MIT in 1956 and updated in following years. The APT system refers to both a language and a computer program. The APT language is used to describe the geometry of the part and the sequence of operations to be performed by a CNC machine. The APT computer program when loaded and run allows the user to define the part using the APT language. This definition is then used to generate the necessary program commands for a CNC machine to produce the part.

POLYAPT

APT, however, is a very general system for the control of CNC machines which in recent years has become highly developed and sophisticated. APT is intended as a production tool and therefore most versions require large computing facilities. In a recent paper by Clarke [4] a version of APT, called POLYAPT, capable of machining sculptured surfaces and implemented on a PDP 11/23 microcomputer (similar power to an IBM AT) is described. In the tradition of APT, POLYAPT was designed as a CAD/CAM production tool rather than as a system for the replication of surfaces. It was rejected because it did not satisfy all the required objectives of this thesis.

CADAMP and SIPSURF

Broek and Vergeest [5] describe a system called CADAMP

(Computer Aided Design and Model Production) normally used by designers to develop shapes of industrial products. The user typically inputs a series of 3D surface datapoints and the system calculates and displays a smooth surface which approximates these points. The user can modify the surface shape interactively and when he is satisfied with the shape CADAMP can generate the toolpath to produce the surface on a CNC milling machine. The designers have interfaced this system with a photogrammetric device and the shape of a human face has been measured and replicated. The system is implemented on a PDP 11/43 computer but a version called SIPSURF [6] is available for the IBM PC with certain additional hardware.

After corresponding with the suppliers, the author was informed that they would not disclose any of the algorithms used nor would they sell SIPSURF to any South African institution. The author found similar responses from other suppliers of the latest CAD/CAM software who were also reluctant to release detailed information of the precise workings of the product they were marketing for commercial reasons. This made it difficult to complete a comprehensive survey of the state of the art in CAD/CAM. The author believes however that the mathematical techniques used by these systems are based on those available in the literature.

POLYHEDRAL NC

Another system for milling sculptures surfaces, POLYHEDRAL NC, was developed in Canada between the years 1969-1976 at the University of British Columbia. This system is described by Duncan *et al* [1].

The original objectives of this system were two-fold: firstly to avoid the complexities of analytical equations that arise when performing the surface fitting, and secondly to prevent any interference which might occur in

machining. The term interference refers to the inadvertent removal of material at one location whilst machining correctly at another [7]. This is illustrated in Figure 2.1, which shows correct milling at one point on the surface but interference occurring at another. This interference occured because the cutting tool radius is too large to machine such a sharply curved concave surface.

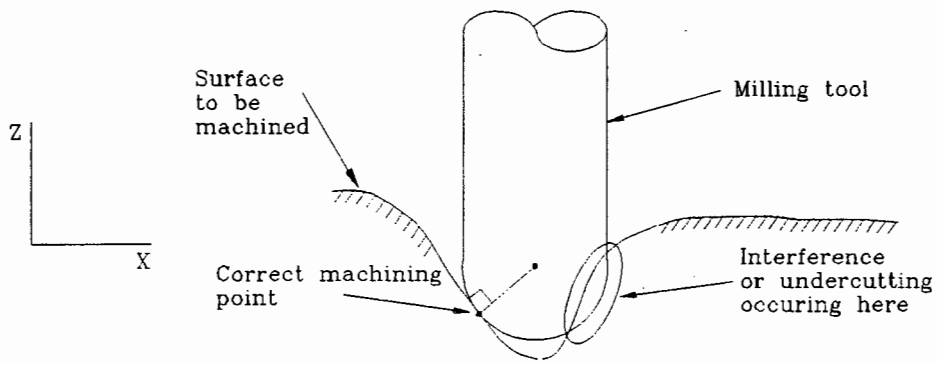


Figure 2.1: Interference during machining of a sharply curved surface

The solution chosen by Duncan *et al* [1] was to represent the required surface approximately by a polyhedron having triangular plane faces as illustrated in Figure 2.2.

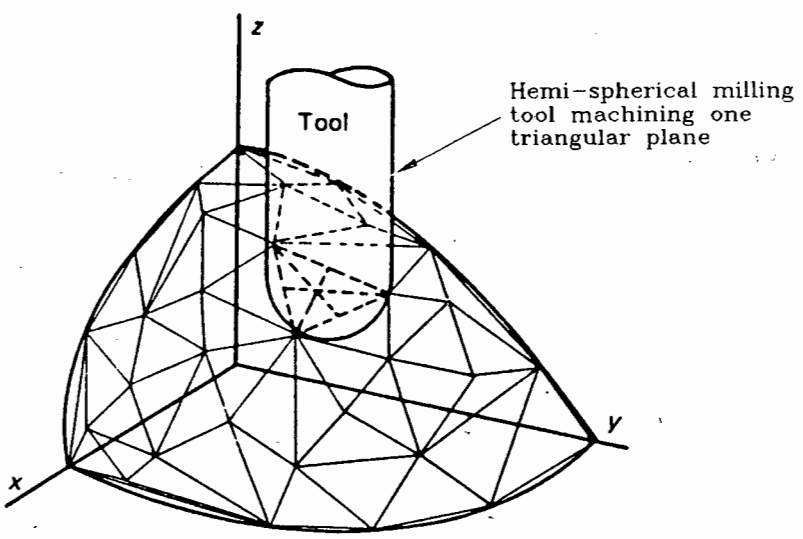


Figure 2.2: Diagram of the POLYHEDRAL representation of a surface from Duncan *et al* [1]

They describe the polyhedron as similar to a cut gem stone with the vertices being points taken on the surface, ie measured surface datapoints. The points are joined by straight line edges and each set of three edges forms a triangular plane.

During the machining of the surface, a hemispherically ended cutter moves from a point on one plane to a point on the next plane and so on. In so doing the required three-dimensional surface is produced. Duncan *et al* also describe the use of POLYHEDRAL NC in the manufacture of a variety of industrial and anatomical surfaces, and mention that interference, which is a problem with many other 3D milling systems, can be avoided by the use of POLYHEDRAL NC. A good description of the fundamental mathematics behind the method is also given in [1].

MAETRACE

Renishaw Metrology, a company based in the U.K., produces a digitising package called MAETRACE for CNC machines [8]. Its prime function is to allow the manufacture of dies and moulds from an original sample. To use the system a probe is attached to the spindle of an existing CNC machine and is moved over a sample surface in order to determine the X,Y and Z coordinates of surface datapoints. The data is then communicated to an IBM PC computer. The surface can be displayed in 3D on the screen and the user can perform data transforms, for example a male to female transform, in order to obtain the required shape of die from the original data. The computer uses the data to write a CNC program which is downloaded (sent) to a CNC milling machine, which produces the required die or mould for the manufacture of a replica of the original sample.

One of the problems of the MAETRACE system is that it relies on a probe situated in a CNC machine to actually contact the sample. This is obviously unsuitable for both

the reproduction of anatomical surfaces, and for the long term objectives set out in Chapter 1 for a non-contact measuring system.

The MAETRACE software has been developed for a range of Fanuc CNC controllers, and is not compatible with the Textron controller on the CNC milling machine in the Mechanical Engineering Department. MAETRACE also requires a more powerful IBM PC AT computer with an enhanced graphics adaptor (EGA). The price of the software alone is R16 000 without the necessary hardware. No information regarding the surface fitting methods nor the cutting methods was available from the suppliers.

MASTERCAM and PLUSCAM

MASTERCAM [9], a product of CNC Software Incorporated in the USA, is used to mill three types of surfaces: surfaces of revolution, ruled surfaces, and swept surfaces. These are illustrated in Figure 2.3.

The part to be milled must either be defined within MASTERCAM or be downloaded from another 3D CAD package. MASTERCAM enables the user to specify the necessary machining parameters such as spindle speed and feedrate. Once the part has been fully defined, MASTERCAM has the facility to write the CNC program which defines the toolpath required to produce the part. The completed CNC program is then sent directly to the CNC machine via an RS232 data link (see Appendix C for a description of data communications). MASTERCAM can write CNC programs for a variety of CNC controllers such as Textron, Fanuc, Bridgeport and others.

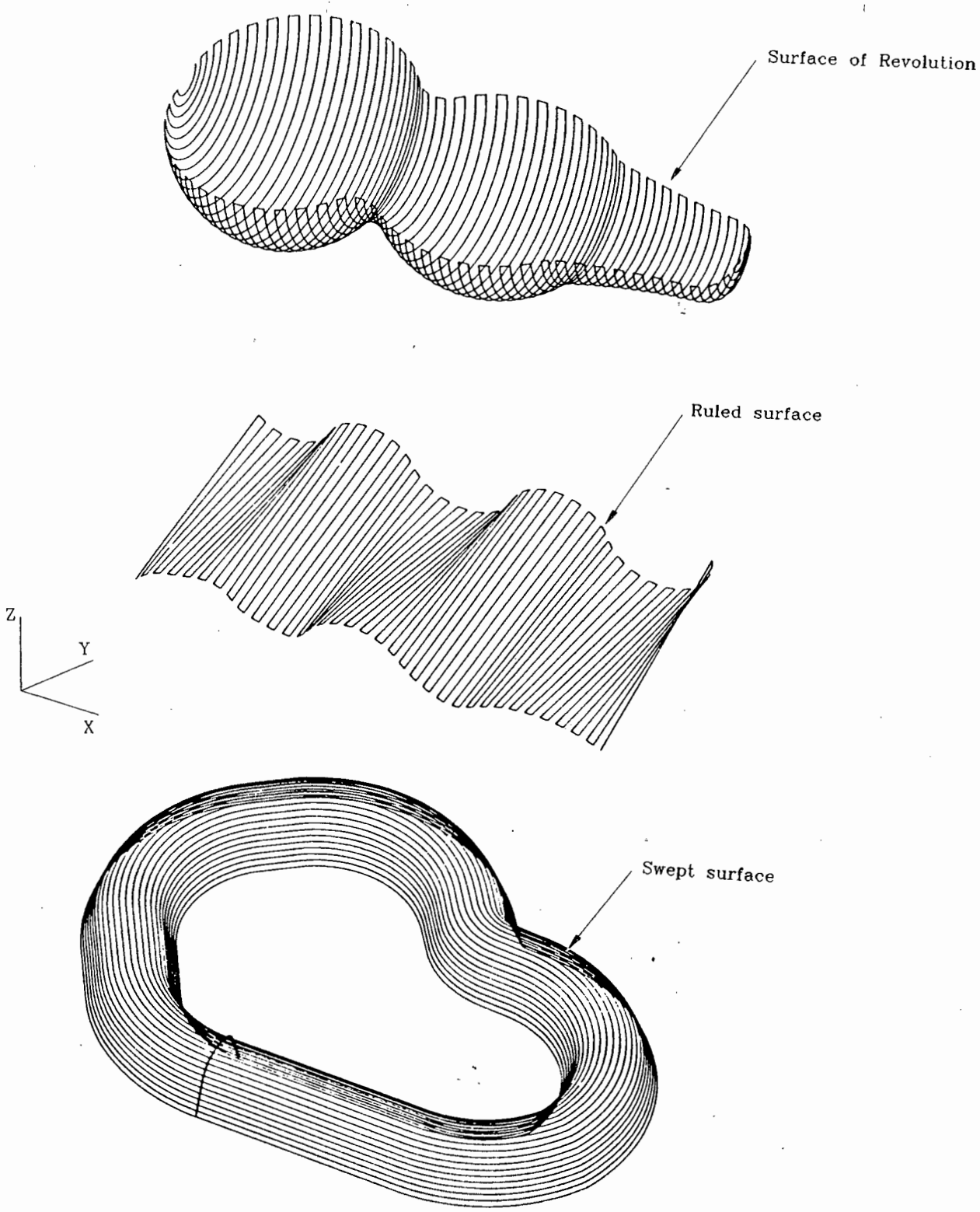


Figure 2.3: Surface of revolution, ruled surface, and swept surface as used in the MASTERCAM [9] and PLUSCAM [10] systems from MASTERCAM brochure

There is a version of MASTERCAM, called PLUSCAM [10], which is used together with CADKEY, a CAD package capable of designing 3D surfaces. CADKEY enables the user to design a part, and PLUSCAM converts the design into the appropriate CNC program defining the toolpath required to produce the part. The price of PLUSCAM alone is R8 000, which together with the necessary hardware makes it an expensive system. The product brochure does not mention any details of the mathematical procedures used.

TELL, MECANIC and AVIA

Hasler and Desjonqueres [11] report on a mould and die manufacturers experience of CAD/CAM. The CAD/CAM system consists of TELL, used to define the geometry of the mould, MECANIC, used to calculate the toolpaths to machine the mould, and AVIA which allows the part and toolpath to be displayed in 3D on the computer screen. A direct numerical control (DNC) connection is provided for the rapid transmission of the program from the CAD/CAM computer to the CNC machine. In addition, the system can measure a sample directly on the milling machine within a 5 micron reproduction tolerance and a 10 micron measuring error over a 300 mm x 300 mm x 300 mm region.

The CNC milling machine used was an ACIERA, and the CAD/CAM system was implemented on a PDP 11/23 computer. This is a highly specialised system installed in a company in France.

No details were available as to the mathematical formulation used in the system. The hardware and software are supplied by one company, the ACIERA company in Switzerland.

EUCLID

Decheletee [12] reports on an interesting application of CAD/CAM in the manufacture of dental crowns. The system was developed by a French dentist over a period of 15 years in

collaboration with universities and the French National Institute of Applied Science.

Using the CAD/CAM system, a dental crown is produced in three stages. First, a miniature optical scanner is used to examine the site inside the mouth where the new tooth is to be sited. Three to five digital images are captured and the data is fed into a computer. The data is processed using "EUCLID", a CAD/CAM and solid modelling package. The dental practitioner then selects a replacement tooth based on a model contained in a library of 32 possible tooth shapes.

It is possible to match the tooth exactly to its surroundings using EUCLID. Finally a CNC micro-milling machine is used to produce the tooth from a block of chosen material. The resulting tooth is maufactured to a tolerance of better than 80 microns and, according to the paper, looks and fits better than those using the traditional method. The most important benefit is that the tooth can be manufactured and fitted in just one visit to the dentist with a resultant time and cost saving.

There are also older 3D CAD/CAM systems, but with the advent of more powerful computers they have been superceded by faster, more versatile and powerful ones. Nevertheless, they provided the foundation upon which many of the more recent systems are based, and no study would be complete without mentioning the better known systems.

UNISURF

The UNISURF system was introduced by Bezier, [13 reported in 14], and was the first practical surface design system. The system was used for CAD/CAM of sculptured surfaces and was developed at the Renault Car Company in France for the production of car body parts. It was also used for the numerical definition of previously defined shapes [15].

NUMERICAL MASTER GEOMETRY

Numerical Master Geometry (NMG) [16] and POLYSURF [17] are another two of the early systems. The former was developed at the British Aircraft Corporation, and the latter at the Cambridge Computer Aided Design Centre.

Both use a similar mathematical approach to the surface fitting, that of bicubic patches, and were intended for the CAD/CAM of components whose surface forms were difficult to machine.

South African Systems

Although there are South African developed CAD/CAM systems which are able to generate toolpaths for CNC machines, no evidence was found of any South African developed system able to mill sculptured surfaces.

With the increase in sanctions against South Africa, this type of equipment and software may well become difficult to obtain. Expertise is therefore required in this area to meet the future needs of South African industry.

2.2: A Review and Assessment of Surface Fitting Methods

This part of the literature survey presents a brief history of numerical geometry, and then describes alternative methods for the mathematical treatment of the surface. The suitability of each method is assessed in terms of the objectives set out in Chapter 1.

2.2.1: A Brief History of Numerical Geometry

Lofting

The development of numerical geometry was stimulated by the demands for new design methods during the Second World War. There was a move away from traditional graphical methods to more computational intensive techniques.

The method of "lofting" was the customary method used to define surfaces in the shipbuilding and aircraft industry [14].

Lofting is best illustrated in a physical sense by considering the construction of a small wooden boat. Firstly, templates in the shape of the ribs of the boat are constructed. These correspond to a set of cross sections of the boat at various intervals. These templates are then laid out and thin strips of wood nailed to the (rib) template. Any humps or hollows in the wooden strips are smoothed by adjusting the shape and/or the position of the templates to ensure a smooth surface.

Once the geometry of the templates had been determined the ribs could be cut and the boat builder would be ensured of a smooth hull. In a mathematical sense each strip of wood forming the hull can be regarded as a longitudinal curve lying on the surface which blends a set of previously defined cross sections into a smooth surface. Hartley and Judd [18] give a good description of the mathematical technique used to define lofted surfaces.

Patches

Newer methods represent the surface by a set of curvilinear quadrilateral patches. Each quadrilateral patch is defined by four surface datapoints and four boundary equations which may be of any order and are usually linear, quadratic or cubic. By using appropriate values with each of the boundary equations the X,Y and Z coordinates of the surface contained in the patch can be obtained. There is usually first or second order continuity between the patches on the surface, depending on the patch formulation used. This is a significant improvement on the "lofting" method in which the surface itself was not defined but only a system of longitudinal curves lying on the surface.

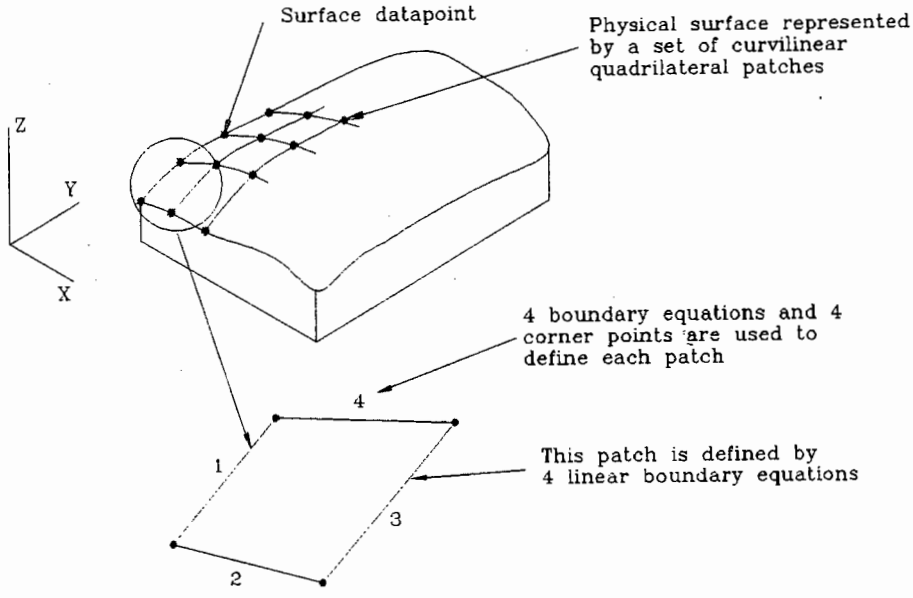


Figure 2.4: A surface as a set of curvilinear quadrilateral patches bounded by surface datapoints.

Ferguson patch

Faux and Pratt [14] report the Ferguson patch [19] as the earliest patch system. It was significant in that parametric rather than Cartesian coordinates were used for the patch boundary equations. The parametric coordinate system is a separate coordinate system in which the patch is mathematically defined. The patch is then mapped from the parametric coordinate system onto the conventional X,Y,Z Cartesian coordinate system, using a sophisticated mathematical procedure.

Bezier patch

Bezier reformulated the Ferguson patch and implemented it in the UNISURF system [15]. This enabled sections of curves and surfaces to be designed by an operator with little or no mathematical training, and was a major mathematical development. The Bezier patch, as it is called, has formed the basis for surface fitting in many CAD/CAM systems.

Coons patch

Coons [20 reported in 14], developed a generalised theory of surface patches which, although it was a significant development in the mathematics of surface fitting, was difficult for the mathematical layman to understand. It is also one of the most widely known approaches to the analytical representation of surfaces.

B-Spline patch

Other mathematical techniques were also being developed. Firstly, the spline curve method was extended and utilised. The spline is a mathematical representation similar to the physical spline used by draughtsmen.

The physical spline consists of a thin strip of material which is positioned by using weights or "ducks" at various intervals to approximate a set of points by a smooth curve. An extension of the theory of smooth spline curves is used in smooth spline surfaces.

It was shown that any spline can be expressed in terms of fundamental splines or "B-splines", [21 reported in 14], which have the advantage that local modifications can be made to a shape during the design process without the need for the numerical representation to be recomputed from the start each time [14].

Current work

Current attention in the mathematics of surface fitting is being focussed on the development of formulations which provide "fair" surfaces and which Faux *et al* [14] say is one of the central problems in surface definition.

The concept of "fairness" can be explained as the subjective opinion of whether a surface is visually appealing. They go on to say that a conventional draughtsman or loftsman would have no problem in judging

the fairness of a surface, but would find difficulty in giving reasons for their decision. In addition they say that experience in computational curve and surface defining systems has shown that continuity of gradient and curvature which is obtained with the patch systems is desirable but is not necessarily sufficient to guarantee "fairness". They explain that this is because a curve can be smooth in the mathematical sense but can still contain oscillations, such as the function SIN X which has continuous derivatives of all orders and is therefore "mathematically smooth" but still has many oscillations. Functions which have fewer oscillations and could therefore provide a better, fairer surface are the subject of current research, but unfortunately they are far more mathematically complicated. Examples of these functions are the spline in tension [22 reported in 14] and the non-linear spline [23 reported in 14].

2.2.2: An Assessment of Surface Fitting Methods in Terms of the Thesis Objectives

Before a surface can be replicated it must first be mathematically defined. This section provides a description rather than a mathematical treatment of the common surface-defining systems and assesses their suitability in terms of the objectives set out in Chapter 1.

Sculptured surfaces are impossible to define by a single mathematical equation. The technique of defining them by small patches as described earlier is a method of overcoming this problem. Woodwark [24] says that the mathematical formulation used for these patches must ensure that firstly, there are no discontinuities at the joints between patches and secondly, that there are no gaps between the patches.

Figure 2.5 illustrates two incorrectly patched surfaces; one with a gap and the other with a discontinuity between patches.

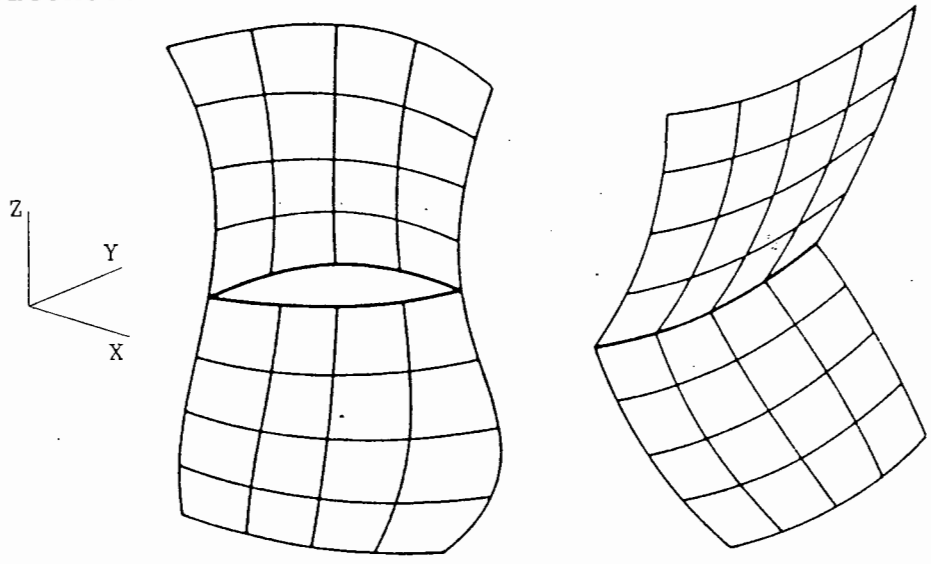


Figure 2.5: Incorrectly patched surfaces from Woodwark [24]

The fundamental difference between the various patch methods lies in the definition of the boundary equations, but without a detailed analysis the precise mathematical subtleties cannot be illustrated. The mathematics of the patch formulations is, however, well represented in the literature. Three notable texts are: Faux and Pratt [14] who give a comprehensive mathematical and theoretical treatment of the entire topic of surface fitting and patch methods, Foley and van Dam [25] who give a more practical view to the problem and also include computer algorithms for the calculation and display of patched surfaces, and Newman and Sproull [26] who also give a good basic approach to 3D surface methods.

There is, however, a common feature amongst the patch methods which is relevant to this thesis. Most of the patch formulations have been developed to allow the interactive design of surfaces, such as the shaping of a

car body part. Therefore the most common application of these patches is in CAD systems where the surface is being designed from scratch and does not already exist. The main strength of these formulations is thus their ability to serve as an original design tool rather than to accurately interpolate or approximate an existing set of surface datapoints.

In this thesis we are not concerned with designing a surface but with replicating an existing physical model for which a table of X,Y and Z coordinates of the surface has been obtained, a so-called digitised physical model. Woodwark [24] says that "patched surfaces can be created by a process of surface fitting to a digitised (measured) physical model. This is a more sophisticated form of interpolation which looks for the best rather than an exact surface through the measured data. Surface fitting is, however, not without its difficulties and there can be problems in choosing a strategy to divide the surface into different patches. It can only be done by someone with experience of the system in use and the person has to decide the orientation of the patch boundaries and also how large the patches are going to be in a given area".

There is much work being done to overcome the problems mentioned by Woodwark, in particular with regard to the B-spline patched surface used by most current systems such as SIPSURF and CADAMP.

The difficulty with the B-spline method is that the original formulation does not allow the user to specify points that lie on the surface. Instead the B-spline surface is defined by, but does not interpolate, a set of points in three-dimensional space called control points. Figure 2.6 shows a grid of control points used to define the B-spline patch.

The control points are not on the actual physical surface but the patch they define is on the physical surface.

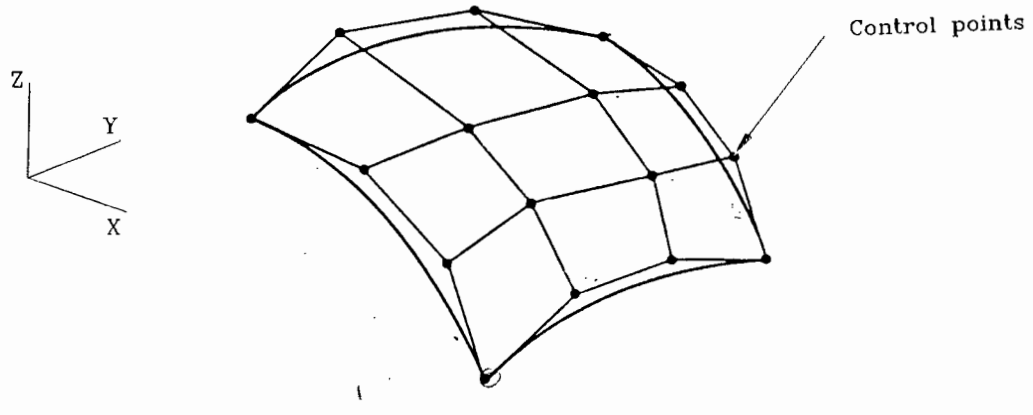


Figure 2.6: A 4 x 4 grid of control points used to define a B-spline patch

Barsky and Greenberg [27] have devised a modified method of defining the control points which does allow the B-spline formulation to interpolate a set of surface datapoints. This method generates one linear equation for every surface datapoint, and then solves the entire system of equations by specialised matrix methods to yield the control points. The control points as well as the original datapoints are required to be arranged on a rectangular grid.

However, because a surface is likely to comprise many datapoints, this method would rely on the solution of large numbers of equations and would therefore be more suited to implementation on a computer more powerful than an IBM PC.

Lord [28] developed an iterative method of fitting a B-spline surface to a set of surface datapoints. So far the method has only been implemented in a program for a 3D curve and not a 3D surface. In addition Lord says that the

efficiency of the program is such that a supermicro- or minicomputer would be required, and that the algorithms have substantial room for improvement in terms of speed of execution.

After corresponding with Lord in an attempt to obtain more information regarding the progress of the 3D surface fitting, the author was informed that the technique was still experimental at this stage and that no further information was available. Nevertheless, the technique shows much promise because of the flexibility it allows in the arrangement of the measured surface datapoints.

Berelowitz [2] used a different form of surface representation, that of a "bi-linear interpolating polynomial". The form of the equation he used was:

$$A + Bx + Cy + Dxy + z = 0$$

Each set of four points were used to determine the four constants A,B,C and D and the quadrilateral element was therefore an interpolating as opposed to an approximating element, because it was forced to pass through each of the four surface datapoints. Figure 2.7 shows the physical appearance of such an element.

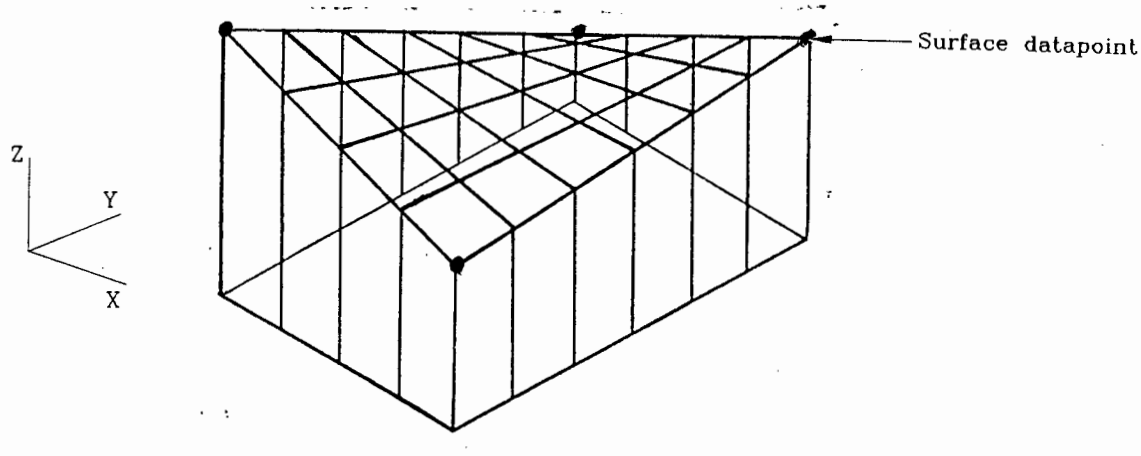


Figure 2.7: Bi-linear interpolating polynomial
from Berelowitz [2]

Another approach is to represent a surface by a series of approximating planes as shown in Figure 2.2 and described previously. Either triangular or quadrilateral planes can be used to approximate the surface. Duncan *et al* [1] used triangular interpolating planes. The form of the standard plane equation they used for each triangular plane was :

$$A + Bx + Cy + z = 0$$

Three surface datapoints were used to determine each of the constants A,B and C.

If two triangular planes are joined to form a quadrilateral element, then each set of four surface datapoints is approximated by two triangular planes.. Figure 2.8 shows this graphically.

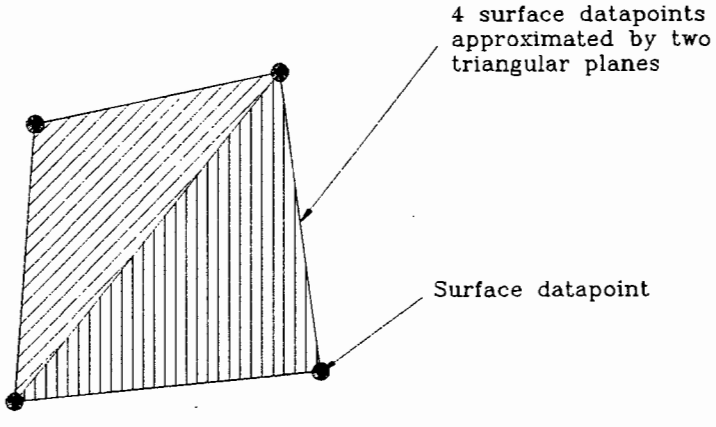


Figure 2.8: Two triangular planes used to approximate four surface datapoints in the POLYHEDRAL NC system of Duncan *et al* [1].

Using the planar approximation to a surface has a number of advantages. The most important in terms of the objectives of this thesis is that the method relies on a geometrical approach to the surface fitting which is easy to understand and visualise. The system is intended to be a teaching tool for undergraduate students and the

mathematical techniques used are familiar to students from their second year on. The patch methods on the other hand rely on sophisticated mathematics only taught at the postgraduate level and which most undergraduate students are unlikely to have experienced. In addition to being mathematically straightforward, the accuracy of the planar approximation is sufficient for the purposes of production of anatomical and other surfaces and has been comprehensively addressed by Duncan *et al* [1].

There are some drawbacks which, although they are not serious, should be pointed out. The most important is that the first and second derivatives are not continuous between the plane faces.

Whilst this can be a problem if sharply curved surfaces are approximated by elements large relative to the size of the cutter, it does overcome the problem of representing discontinuous surfaces by the patch methods.

After milling a variety of surfaces using a planar approach, the author found that the effect of the discontinuities on the final milled surface were negligible when compared to the "tracks" left by the milling cutter.

The "sagittal error" - the distance from the plane to the continuous surface - is also a consideration when using the planar approximation. Figure 2.9 shows this diagrammatically. These errors, which are largest over a sharply curved surface, can be reduced by taking smaller planes over sharply curved regions of the surface.

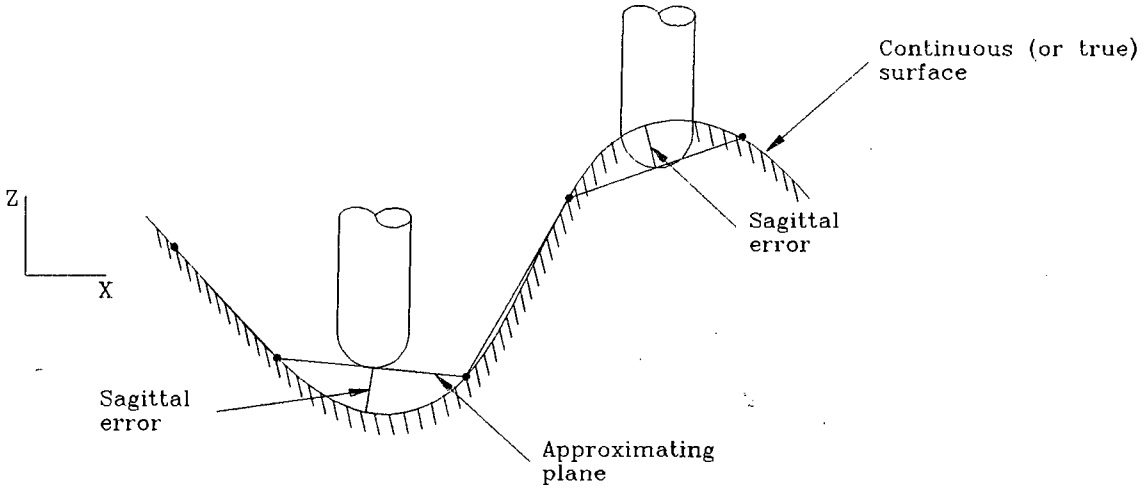


Figure 2.9: Sagittal error of POLYHEDRAL machining from Duncan *et al* [1].

In conclusion, the concept of approximating a surface by a series of small planes was found to be the most suitable approach to surface fitting and most closely met the original objectives.

2.3: Machining Considerations

After the mathematical surface fitting is completed the surface must be machined. There are two considerations with regard to machining: the choice of milling cutter to be used and the tool path to be followed by the milling cutter.

2.3.1: Choice of Milling Cutter

In preceding diagrams reference has been made to a hemi-spherically ended milling cutter, but no explanation was given as to why it was chosen. The main reason for choosing this type of cutter is that the geometry of the cutter allows for simple calculation of the offsets. By offsets is meant the perpendicular distance from the centre of the cutter to the surface, as shown in Figure 2.10.

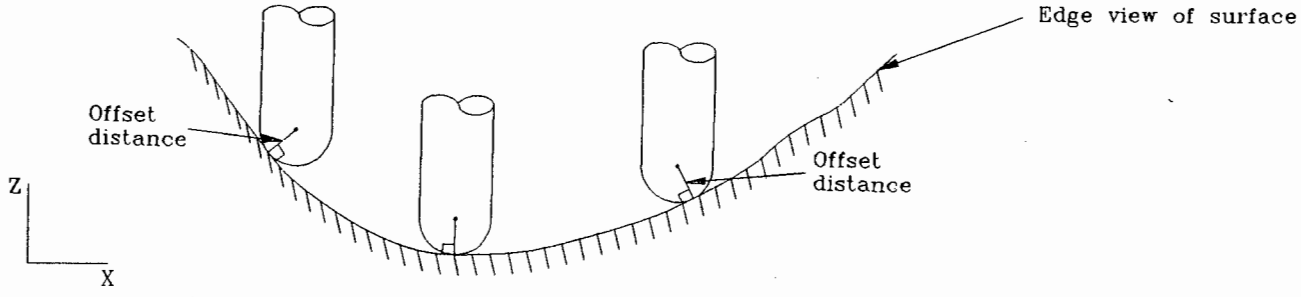


Figure 2.10: The offset distance between the centre of the hemispherical cutter and the surface.

The offsetting distance is independent of the slope of the surface or of the orientation of the cutter to the surface. To determine the offset, the normal to the surface is calculated at the point where the surface is to be machined. The centre of the cutter is then positioned at a distance equal to the cutter radius along the normal as shown in Figure 2.11.

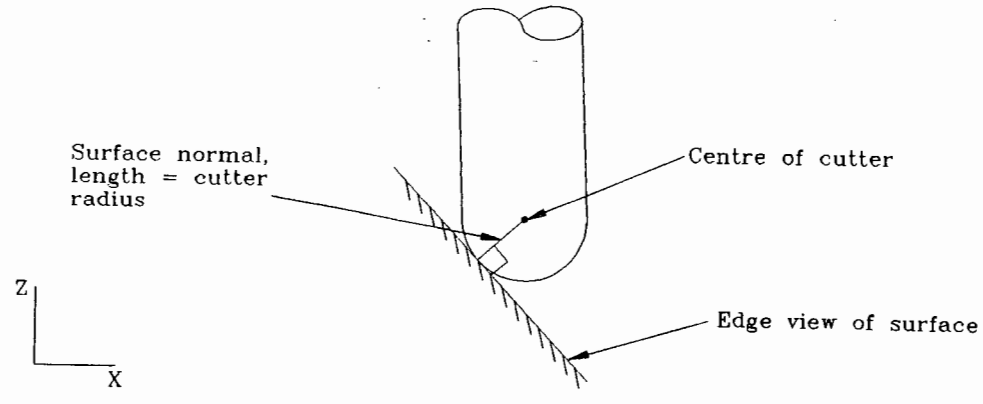


Figure 2.11: Determination of the offset for a hemispherical cutter.

One of the drawbacks of this type of cutter, however, is that when machining part of the surface that is nearly flat the cutting takes place at a small distance from the axis of rotation, as shown in Figure 2.12.

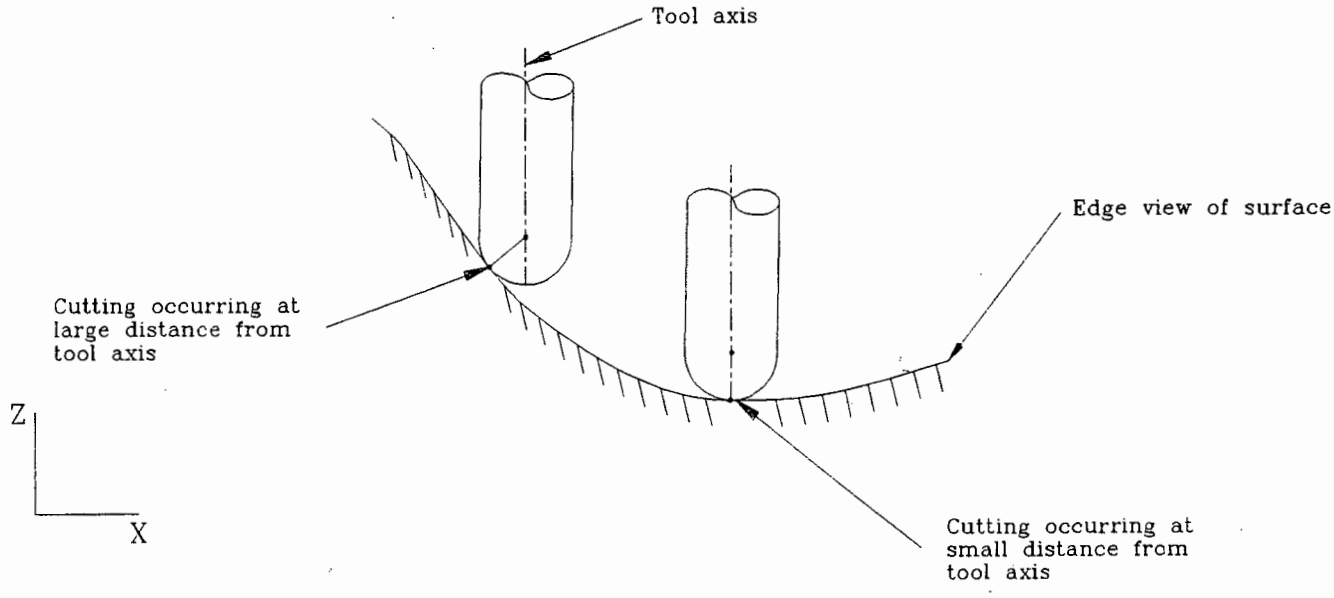


Figure 2.12: Slow cutting speed of a hemispherical cutter on flat regions of a surface

This results in a slow cutting speed and consequently the rate of material removal is slow, as well as giving a poor surface finish [24].

Other cutter geometries such as barrel or disk insert cutters (see Figure 2.13) offer far higher rates of material removal because they always cut at a large radius where cutting speeds are high.

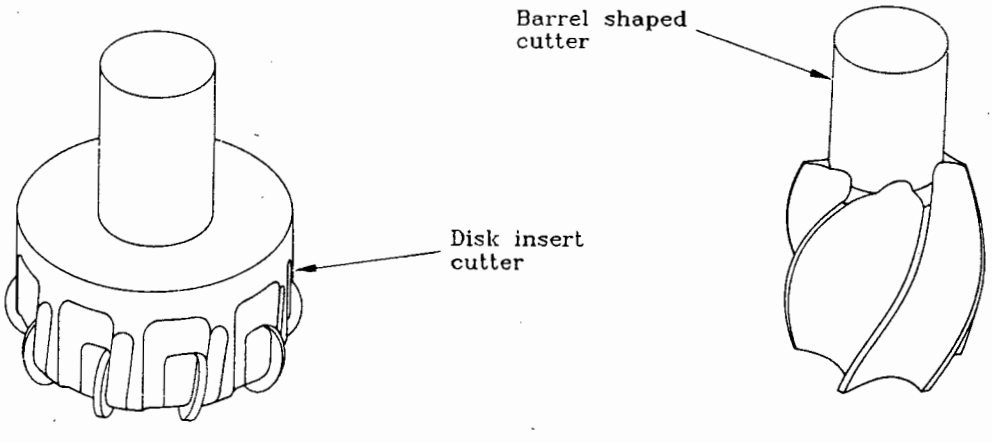


Figure 2.13: Cutter types from Woodwark [24]

The offsets for these cutter geometries are difficult to calculate, especially when the surface changes between concave and convex or vice versa as shown in Figure 2.14

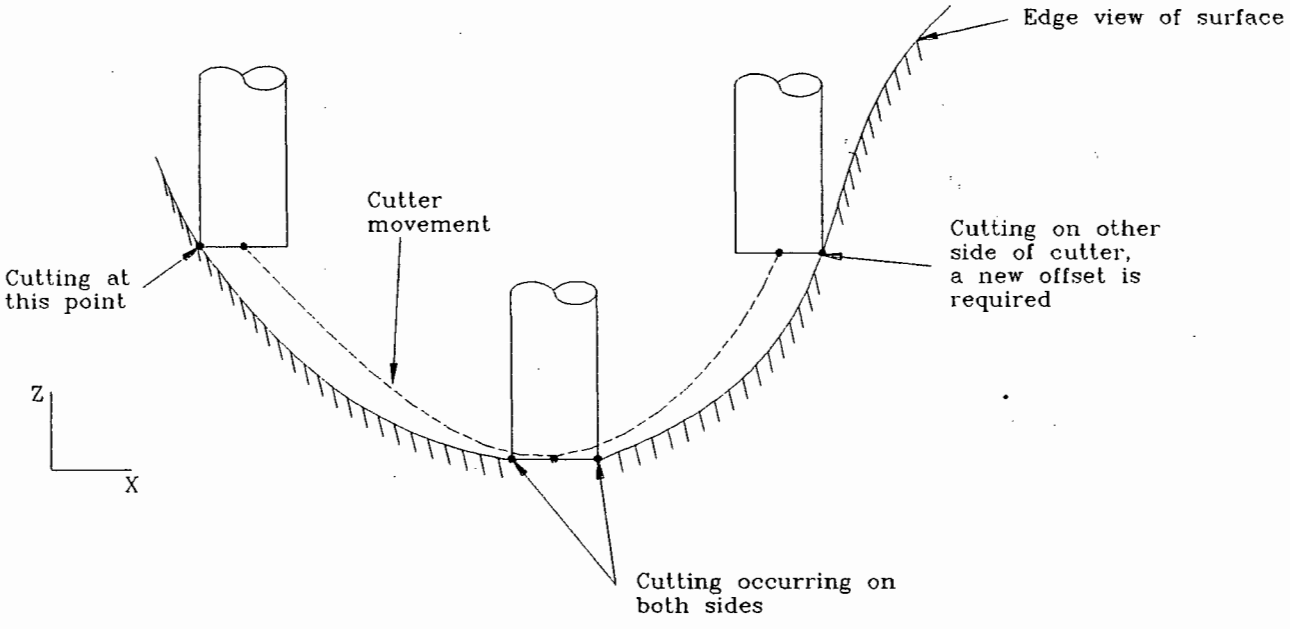


Figure 2.14: Change of cutting point of a flat bottomed cutter when milling a surface which changes from concave to convex

These cutter types could be used if the surface were to be milled by a series of flat bottomed contours or terraces, but they are usually used in machines with four or more axes which can ensure that the cutter is orientated correctly to the surface, as shown in Figure 2.15. The Bridgeport machine, however, has only three fixed orthogonal axes, and therefore it was not possible to use these cutters in this application. Also the efficiency and speed of material removal which is important in a production environment was not a priority in this thesis - the main criterion was the accuracy of the final milled surface.

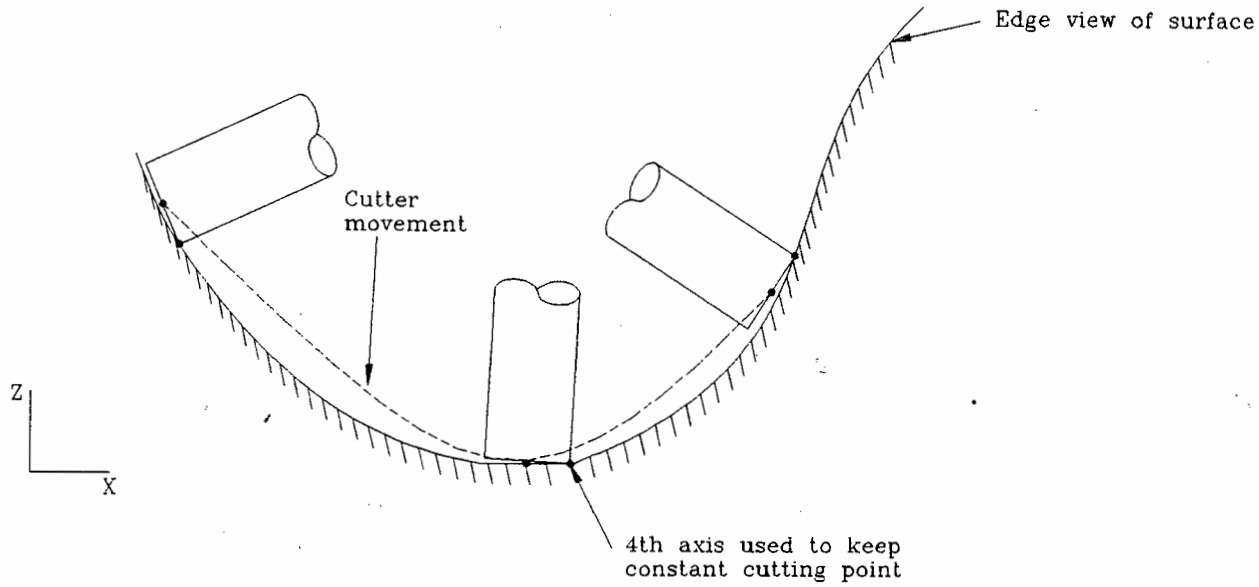


Figure 2.15: Fourth axis used to orientate the cutter correctly to the surface.

In terms of the objectives of this thesis there was no advantage in using the specialised high speed cutters, and therefore a simple hemispherical cutter was chosen.

Attention should also be given to the size of the hemispherical cutter to be used. The larger the cutter, the faster the rate of material removal. Large radii cutters, however, will cause interference if the surface contains concave regions where the radius of curvature is less than the radius of the cutter, as shown previously in Figure 2.1.

Woodwark [24] says that the minimum radius of curvature of a surface can be calculated mathematically and used to determine the maximum radius of cutter which can be used without undercutting (interference). Alternatively a cutter of the appropriate radius can be selected on the basis of the experience of the operator and verified by inspection of a model of the surface machined with the cutter.

Flutter [29] also describes a method of determining whether the cutter radius chosen to machine a given surface will cause undercutting. Firstly, the toolpositions for the desired radius of hemi-spherical cutter are calculated. Then a line is drawn which joins the centres of sequential positions of the cutter (ie the toolpath). If the cutter radius is too large a cusp or loop will occur in the toolpath in regions where undercutting will occur. This provides an easy method of checking whether undercutting will occur with a particular cutter.

2.3.2: Toolpath Selection

In a production environment two considerations regarding the toolpath are important. The first is that the waste material should be removed as quickly as possible, known as roughing, and the second is that the surface be machined as accurately as possible, known as finishing. These considerations impose different requirements on the toolpath. Woodwark [24] says that a common roughing strategy is to define a number of discrete levels, or contours, and to clear the waste material at each of these in turn. This technique allows high speed cutters as described earlier to be used.

There is an alternative method of roughing which Woodwark [24] describes as repeating the finishing cut many times, each time ensuring that the cutting depth is such that the amount of material removed will not overload the cutter. This procedure is inefficient, however, because the cutter spends much of its time milling in the air at cutting speeds which are usually slow but, as efficiency of cutting was not a priority in this thesis, this was the method chosen.

The next stage is to determine the exact path to be followed by the tool to machine the surface. Older CNC machines could only interpolate in the X-Y plane and therefore the conventional toolpath was offset from contour lines on the surface. Berelowitz [2] showed that a system of non-linear equations arise if the contouring method is used with certain surface fitting methods such as the bi-linear polynomial. In addition, the mathematics of interpolating surface datapoints into a contour representation for calculation of the toolpath is complex. Newer CNC machines can perform 3D point-to-point interpolation, which does away with the need to use the contouring approach and also simplifies the calculation of the toolpath.

Given that the 3D point to point is the most suitable choice, there is still the problem of determining which points to use. Berelowitz [2] chose two milling points on each of the bi-linear quadrilateral elements as shown in Figure 2.16. He joined each of them to form the shoelace patterned toolpath, as shown in Figure 2.17.

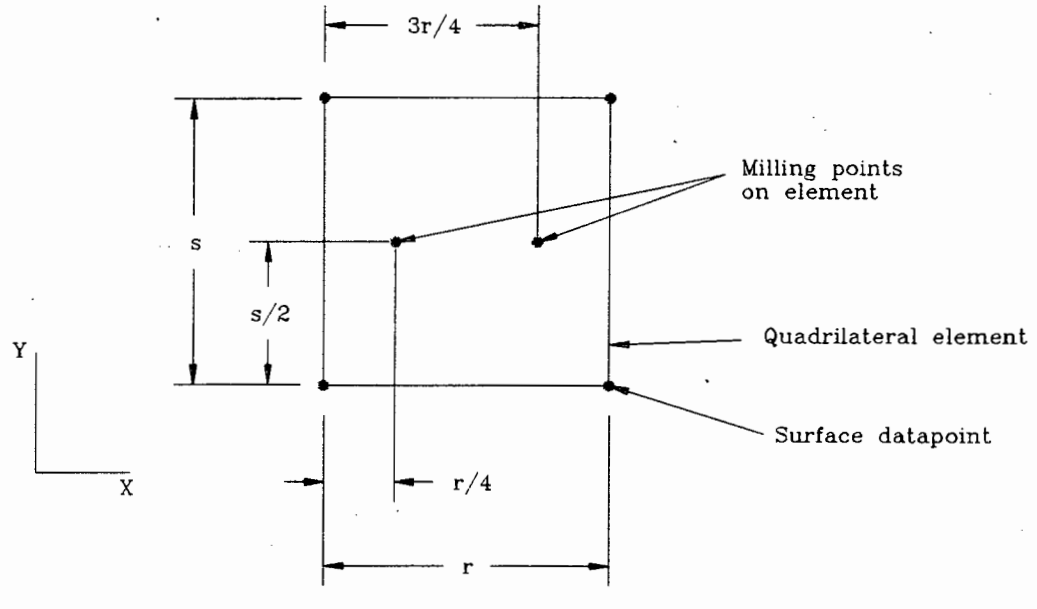


Figure 2.16: Milling points used in Berelowitz's [2] quadrilateral element

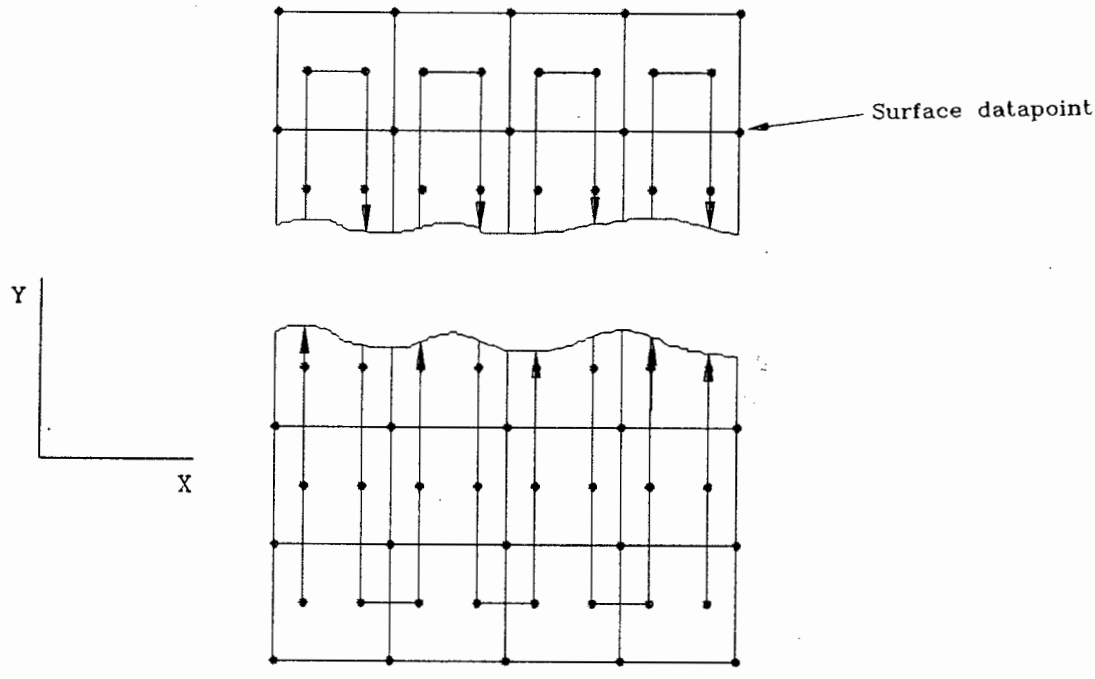


Figure 2.17: Shoelace pattern of the toolpath as used by Berelowitz [2]

The problem with this approach is that each element has only two milling points, resulting in an unacceptable surface roughness of the final milled surface. Berelowitz [2] recommended that more milling points be chosen within each element to ensure an adequate surface finish.

Duncan *et al* [1] proposed two different toolpath patterns in the POLYHEDRAL NC system. The first, which was used in early versions of the program, caused the tool to touch each triangular plane at the centroid in a zig-zag pattern, as shown in Figure 2.18.

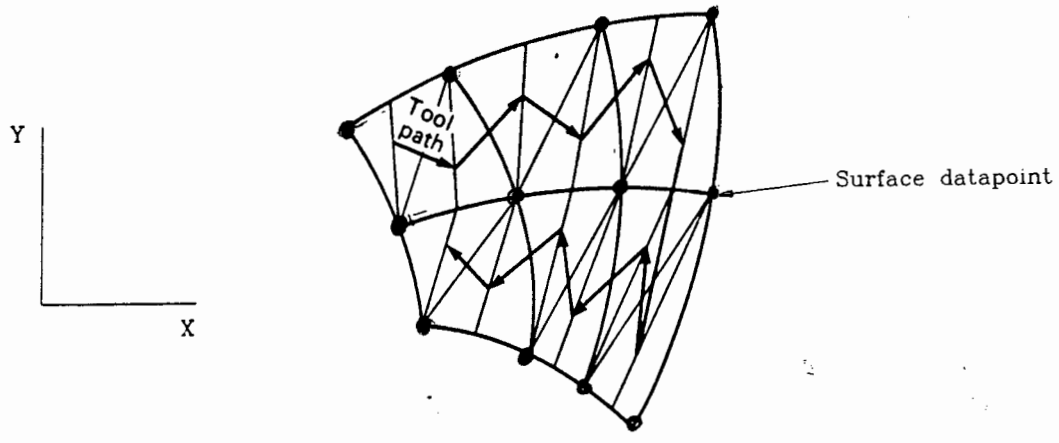


Figure 2.18: Zig-zag patterned toolpath used in early versions of POLYHEDRAL NC from Duncan *et al* [1]

The other toolpath pattern, used in later versions of the program, resembled that used by Berelowitz [2]. This pattern caused the tool to move from one centroid to the next, as shown in Figure 2.19.

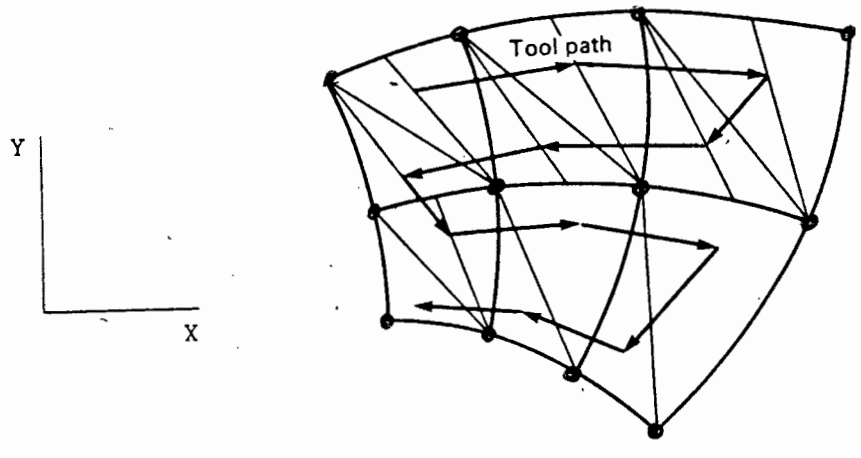


Figure 2.19: Shoelace pattern used in later versions of POLYHEDRAL NC from Duncan *et al* [1]

The shoelace pattern is the usual toolpath used to machine sculptured surfaces and the one that is used by CADAMP [5], SIPSURF [6] and MASTERCAM [9]. It was also the one chosen for this thesis.

The actual pattern of the toolpath will have little effect on the accuracy of the final milled surface. It is rather the size of the cutter used and the number of milling points which will have the greatest effect. In this regard both Duncan *et al* [1] and Berelowitz [2] had only two milling points within each quadrilateral element. Two milling points for a warped quadrilateral element are, in general, not sufficient to ensure good accuracy or surface finish, unless the elements are small relative to the cutter. Small elements require that more surface datapoints be provided, which limits the size of the surface that can be accommodated by the system. The author therefore chose to use four milling points on each quadrilateral element.

2.4: Methods for Direct Communication between an IBM PC and a CNC Computer

The weakness of most older CNC controllers is their reliance on paper tape for inputting programs. This in turn means that a teletype or similar paper puncher has to be used. The modern trend is to communicate the CNC programs directly from another computer to the CNC computer via a data link, usually an RS232 (see Appendix C). This has the advantage that the CNC program can be stored on floppy disk and not bulky inconvenient paper tape. Also the program can be input much faster than by the conventional method. For those CNC controllers without an RS232, a "black box" device is being marketed in the U.K. which is placed over the existing paper tape reader [30]. It is connected to an RS232 port of a local computer and simulates the paper tape passing over the reading head by a series of flickering lights. The cost of the device is R4 000.

Berelowitz [2] achieved a compromise in that he provided a data link from the IBM PC to the teletypewriter, which

avoided manual retyping of the CNC program. Whilst this was a significant improvement on the original system, it was still inconvenient and slow.

As the TEXTRON CNC controller has an RS232, a fast direct link from the IBM PC was required to rapidly input the CNC programs.

2.5: Three-Dimensional Graphical Display Considerations

It is always useful to be able to verify that the surface and the resulting toolpath have been computed correctly, particularly when using measured surface datapoints. In order to do this a 3D view of the surface with the resulting toolpath superimposed is the best method. This allows the user to inspect potential problem areas of the surface and to rectify them before the machining of the surface. It is preferable that different viewpoints of the surface be provided in order to give an unambiguous picture. Most modern CAD/CAM facilities such as CADAMP [5] and POLYHEDRAL NC [1] provide this feature.

One disadvantage is that a large amount of computation is required to display 3D views, and this tends to slow the system down significantly. This is the reason that most 3D CAD/CAM system have be implemented on larger computers with fast processing. In this thesis, however, the benefits of displaying 3D views outweighed the cost of the slow system and it was provided.

The next chapter deals with the system that was developed to meet the requirements as discussed in Chapter 1.

CHAPTER THREE

3. THE MILLING SYSTEM

The milling system designed in this thesis consists of four parts:

1. The surface fitting and its implementation in a computer program which creates the toolfile containing the tool path necessary to machine the surface.
2. The three-dimensional viewer which displays in 3D the surface together with the toolpath on the computer screen. It is used for verification of both the surface and the calculated tool path.
3. The postprocessor which converts the toolfile into a number of programs suitable for use with a CNC machine, using the familiar G and M codes.
4. The communications program which links an IBM compatible computer to the CNC computer for the rapid transfer of the programs to the CNC machine.

This is shown schematically in Figure 3.1 below:

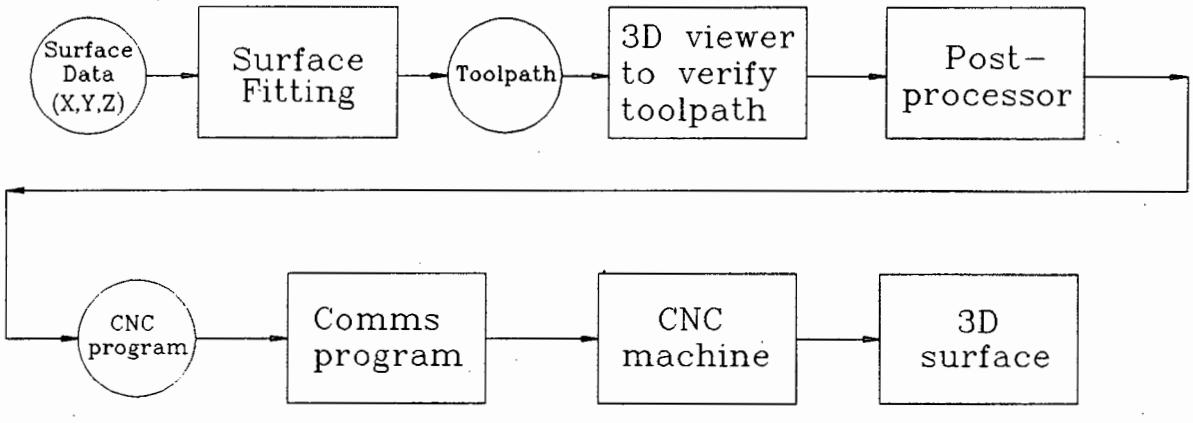


Figure 3.1: Schematic flowchart of the system

3.1: The Surface Fitting and its Implementation in a Computer Program

An overview of the mathematical approximation of the surface is described here. Only the fundamental formulae are presented. Appendix A gives the complete mathematical formulation and includes a numerical example using a small surface.

3.1.1: The Surface Fitting

Datapoints as a Representation of a Surface

The 3D surface to be machined is represented by a grid of 3D datapoints lying on the surface. Each datapoint consists of an X, Y and Z coordinate.

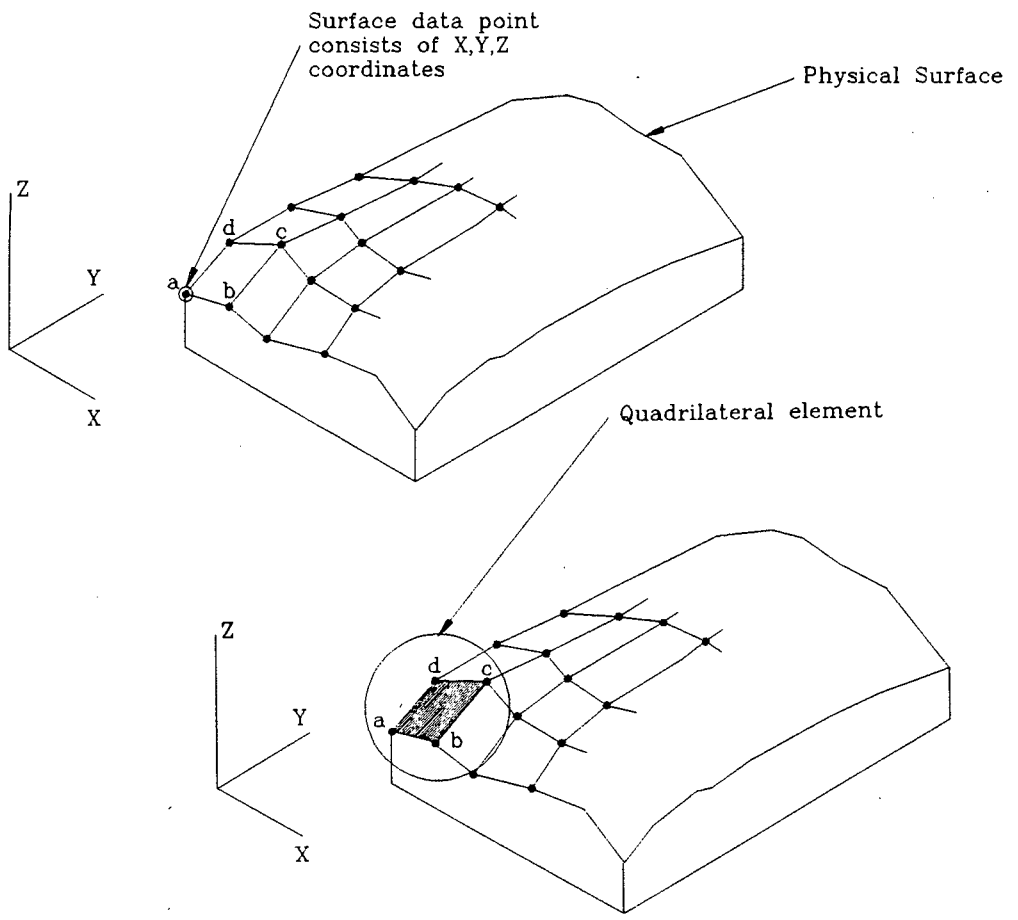


Figure 3.2: Physical surface showing datapoints and a quadrilateral element.

Figure 3.2 shows part of a grid of datapoints on the surface. Each quadrilateral subdivision of the grid is referred to as an "element". Depending on the physical position of the corner points, the element may be square, rectangular or quadrilateral in shape and if the four corner points are not co-planar a warped element will be formed. Luzadder [31] defines a warped surface as a "nondevelopable ruled surface in which consecutive straight lines are non-parallel and do not intersect". The particular type of of warped element formed in this case is called a hyperbolic paraboloid. This can best be visualised by twisting a sheet of elastic film, as shown in Figure 3.3.

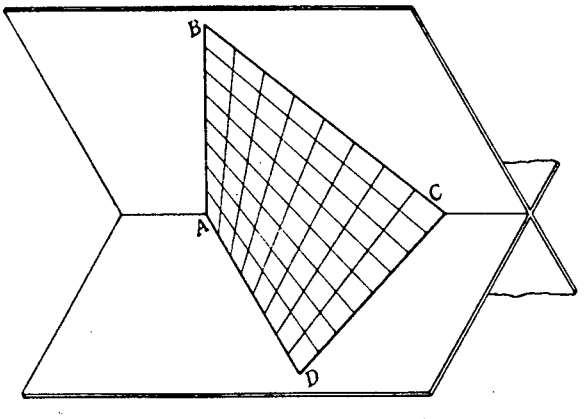


Figure 3.3: A hyperbolic paraboloid from Hoelscher *et al* [32]

Although the surface datapoints comprising the grid may be randomly spaced on the surface, they are subject to some constraints which are:

a) There must be an equal number of datapoints in every row and an equal number in every column.
b) The surface must be able to be broken down into a series of quadrilateral elements joining each set of 4 datapoints.

Figure 3.4 shows two surface representations; one which is suitable and one which is not. Although the precise

physical position of the surface datapoints is not crucial, sensible quadrilateral elements must be formed by each set of four datapoints. The author found that the best representation is a set of datapoints which conform to the nodes of an regular orthogonal grid. The more the datapoints deviated from this orthogonal grid, the more inaccurate was the final milled surface.

The datapoints also have to be fed into the computer in an ordered manner such as row by row or column by column to enable the computer to retrieve the correct datapoints to form each quadrilateral element.

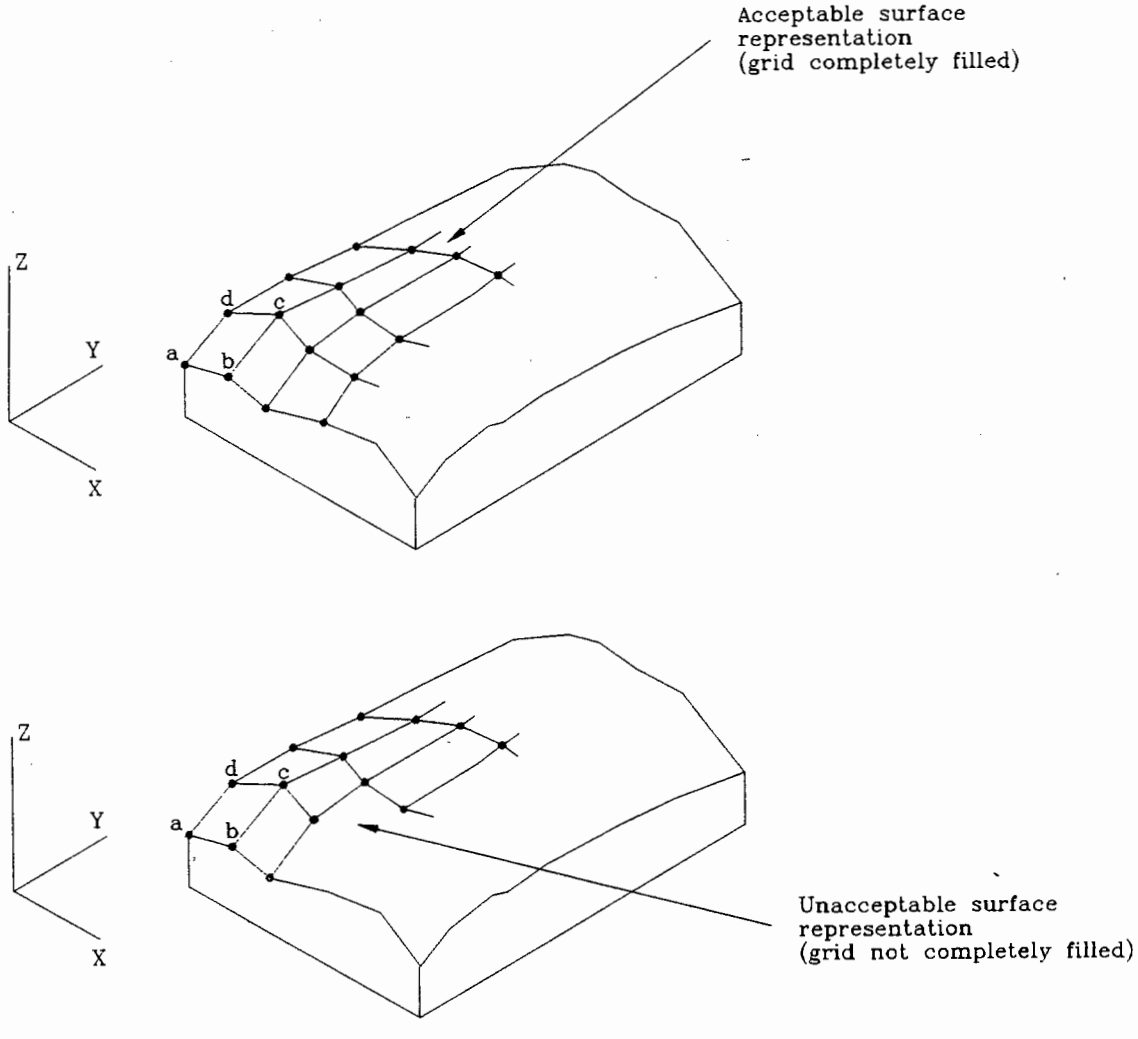


Figure 3.4: Two datapoint grids

The Storage of the Surface Data in the Computer

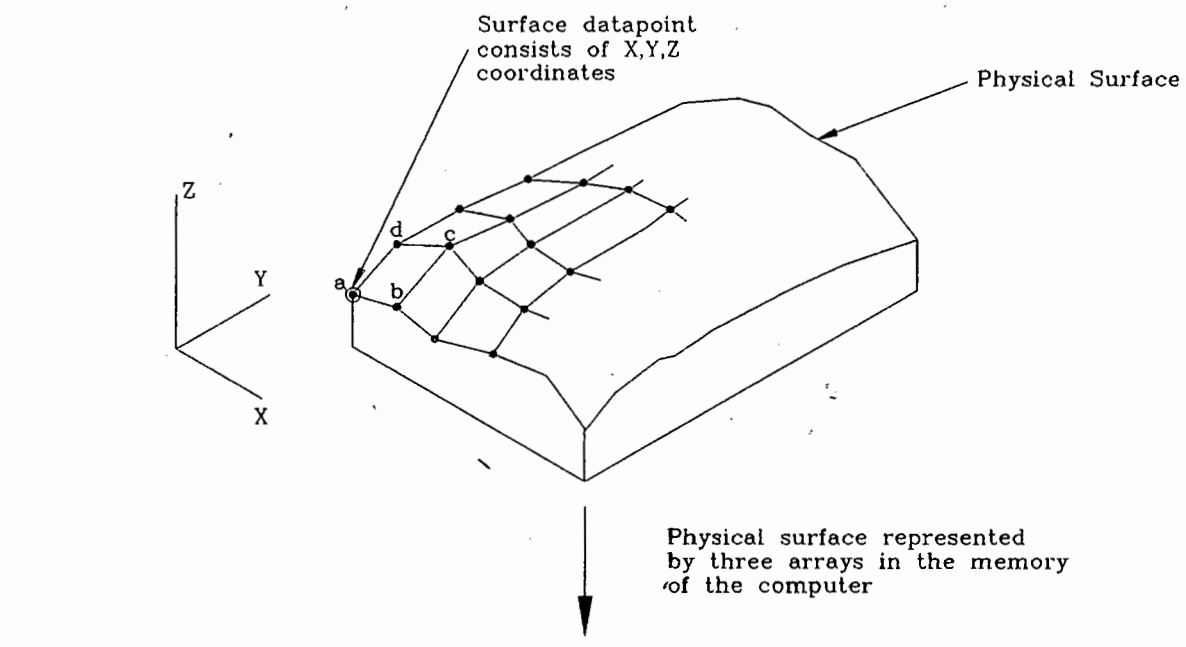
At this point a slight digression is necessary to introduce some terminology required later in the discussion of the mathematical approximation.

The terms FRAME and LEVEL were originally used by Vaughan *et al* [2] in their development of a 3D surface measuring device called a TopoScanner. The term FRAME denotes an X grid and the term LEVEL a Y grid, together forming an XY grid. The Z height of the surface is measured at the nodes of this grid. The grid is not orthogonal and the measured data results in a table of warped elements.

Berelowitz [2] developed software which used this device to measure surface datapoints, and thus the terms FRAME and LEVEL were "inherited" from earlier work.

In this thesis they are used to denote the indices of the arrays used to store the surface datapoints in the computer. In retrospect a more prudent choice would have been to replace the terms FRAME and LEVEL with the more conventional terms "i" and "j" for the indices of the storage arrays.

There are three storage arrays: an X array, a Y array and a Z array, which are used to store the X,Y and Z co-ordinates of the surface datapoints. The three coordinates of any surface point are retrieved by selecting the appropriate cell in each of the three arrays. The appropriate cell is specified by referring to the two indices; the frame and the level. Figure 3.5 shows diagrammatically a physical surface with the coordinates of the four datapoints of one quadrilateral element placed in their respective arrays.



Physical surface represented by three arrays in the memory of the computer

Level ↑

1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 Xd	2,2 Xc	3,2	4,2
1,1 Xa	2,1 Xb	3,1	4,1

X array

1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 Yd	2,2 Yc	3,2	4,2
1,1 Ya	2,1 Yb	3,1	4,1

Y array

1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 Zd	2,2 Zc	3,2	4,2
1,1 Za	2,1 Zb	3,1	4,1

Z array

Frame ⟶

To retrieve the X,Y,Z coordinates of point a for processing:

Xa = Xarray[1,1]
Ya = Yarray[1,1]
Za = Zarray[1,1]

Figure 3.5: The storage of the surface data in the computer in an X, Y and Z array.

The author found that the use of <u>three</u> arrays for the storage of the data facilitated the programming and data manipulation.

Each of the four triangular approximating planes share a common vertex which is effectively a fifthpoint within the quadrilateral element. The location of the fifthpoint is fundamentally important to the approximation of the element, and in order to determine the best position a physical model of a warped element was constructed. After experimenting with various positions of the fifthpoint the best position was decided upon. This is illustrated diagrammatically in Figure 3.7. As the fifthpoint was difficult to show in the shape of the elements in the previous diagrams, the shape of the quadrilateral element was altered slightly for this diagram.

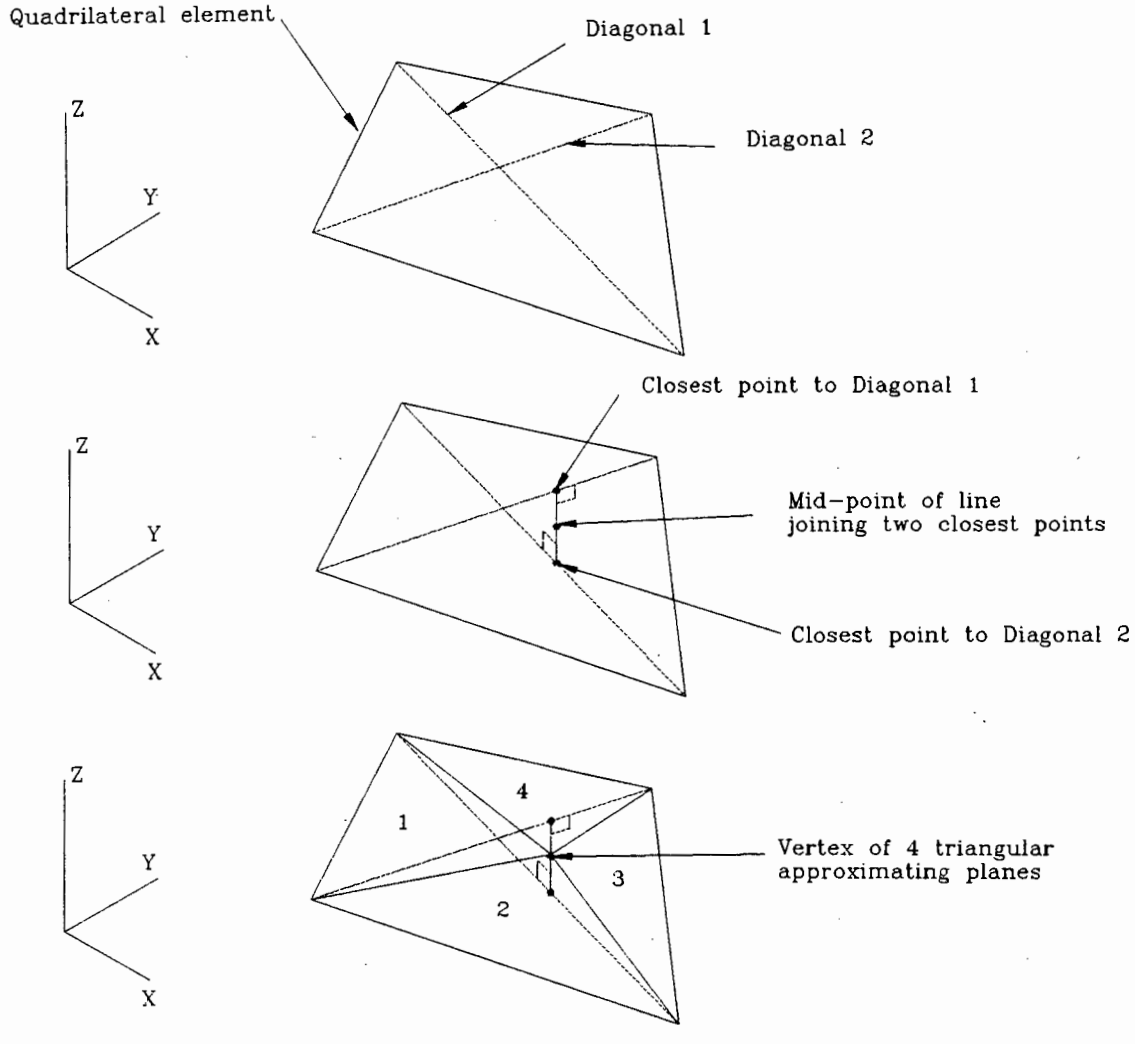


Figure 3.7: The development of the fifthpoint in the quadrilateral element.

Physically two lines which form the diagonals of the quadrilateral element are constructed. There are then two points, one on each diagonal which, if joined, will give the shortest distance between the two diagonal lines. If these points are then joined by another line, then the fifthpoint is placed at the mid-point of this line. This point is the common vertex for each of the triangular planes used to approximate the surface.

Once each element has been divided into the four triangular planes, the point of contact between the milling cutter and the plane can be determined.

The Determination of the Coordinates of the Milling Tool for Cutting the Surface Element

Each quadrilateral surface element is machined by milling with a hemispherical cutter to a point on each of its four constituent triangular planes. The centre of the hemispherical cutter is offset from each triangular plane by a distance equal to the cutter radius along the normal to the triangular plane through the plane centroid. The centroid was chosen because it was considered to be the point which best represented each triangular plane. In machining a number of different surfaces the author found that if the size of the plane was small relative to the cutter, then it was sufficient to mill to only this point on each triangular plane. Figure 3.8 shows the quadrilateral element with its four triangular planes and also the location of the cutter centres.

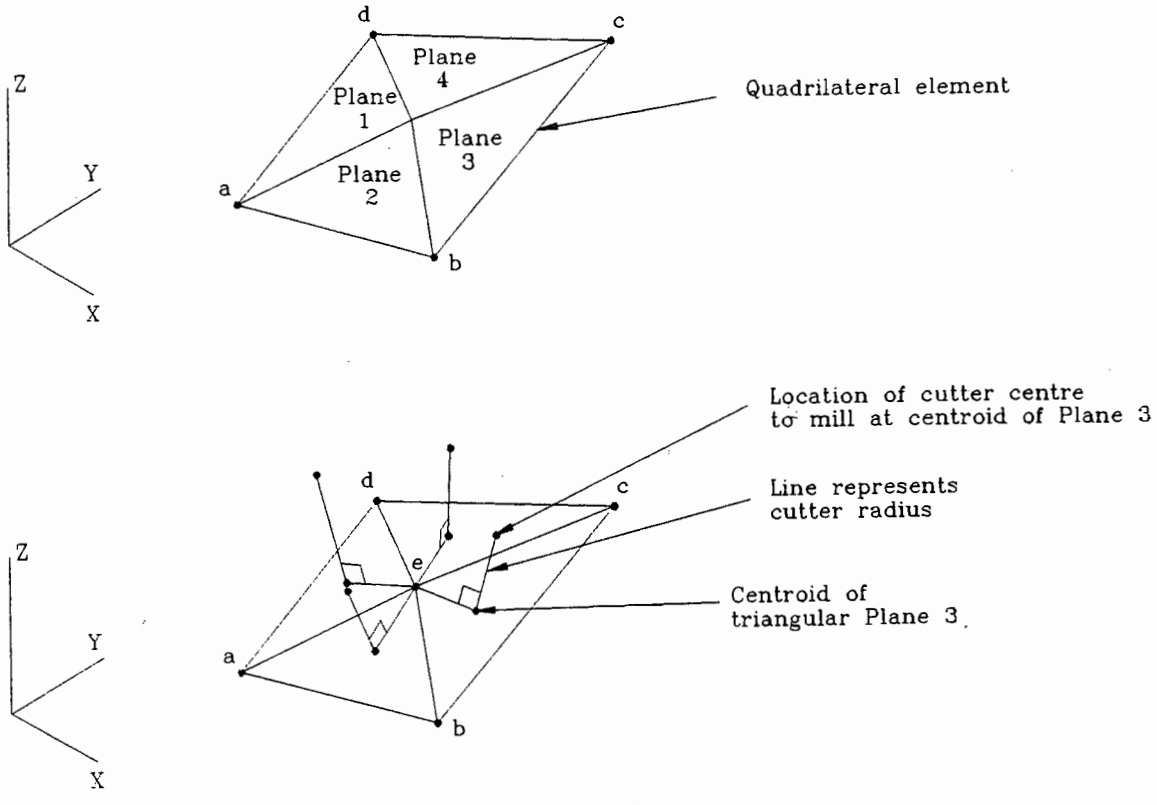


Figure 3.8: The location of the cutter centre with respect to the four triangular planes.

Although the following discussion deals with only one triangular plane, the same procedure is applied to each of the other planes.

The position of the centroid is determined from the three datapoints which define each triangular plane. The subscripts of equations 1-3 show the calculation of the centroid for triangular Plane 1 of Figure 3.8.

$$X_{centroid} = \frac{(x_a + x_e + x_d)}{3} \qquad (1)$$

$$Y_{centroid} = \frac{(Y_a + Y_e + Y_d)}{3} \qquad (2)$$

$$Z_{centroid} = \frac{(z_a + z_e + z_d)}{3} \qquad (3)$$

After the position of the centroid is determined, the coefficients A,B,C and D of the plane equation used for each triangular plane are computed, using standard algebraic methods. Equation 4 gives the form of the plane equation used.

$$Ax + By + Cz + D = 0 \qquad (4)$$

The coefficients of (4) are used to give the direction cosines ($\alpha,\beta,\varnothing$) of the normal vector to the triangular plane using (5),(6) and (7).

$$\alpha = \frac{A}{\sqrt{(A^2+B^2+C^2)}} \qquad (5)$$

$$\beta = \frac{B}{\sqrt{(A^2+B^2+C^2)}} \qquad (6)$$

$$\varnothing = \frac{C}{\sqrt{(A^2+B^2+C^2)}} \qquad (7)$$

The centre of the hemispherical end of the cutter (the toolposition) is used to position the cutter relative to

the plane, and the coordinates of the cutter centre are obtained from equations (8),(9) and (10).

$$X_{cutter} = X_{centroid} + CutterRadius * \alpha \quad (8)$$

$$Y_{cutter} = Y_{centroid} + CutterRadius * \beta \quad (9)$$

$$Z_{cutter} = Z_{centroid} + CutterRadius * \varnothing \quad (10)$$

This procedure is carried out for each of the four triangular planes in turn. Thereafter the computer selects the next quadrilateral element, and the process is repeated. The result of this processing is a list of toolpositions necessary to machine the surface.

The method used to locate the cutter centre was modified from one used by Duncan *et al* [1].

Apart from the X,Y and Z coordinates, each toolposition has a "key" associated with it. The key is used later to arrange the toolpositions in the correct sequence for milling. These sequential positions describe the movement of the tool over the surface which is known as the toolpath. The next section explains in detail the determination of the toolpath.

The Determination of the Toolpath for Milling the Sculptured Surface

As described, the computer program recalls the datapoints of the first quadrilateral element and determines the toolposition for each of its four triangular planes. Thereafter the datapoints of the next quadrilateral element are recalled and the process repeated. In this way all the quadrilateral elements of the surface are processed.

However, the milling cutter does not follow this processing sequence in the toolpath. Figure 3.9 shows the

plan view of a surface. The letters represent the sequence of calculation of the toolpositions for each triangular plane.

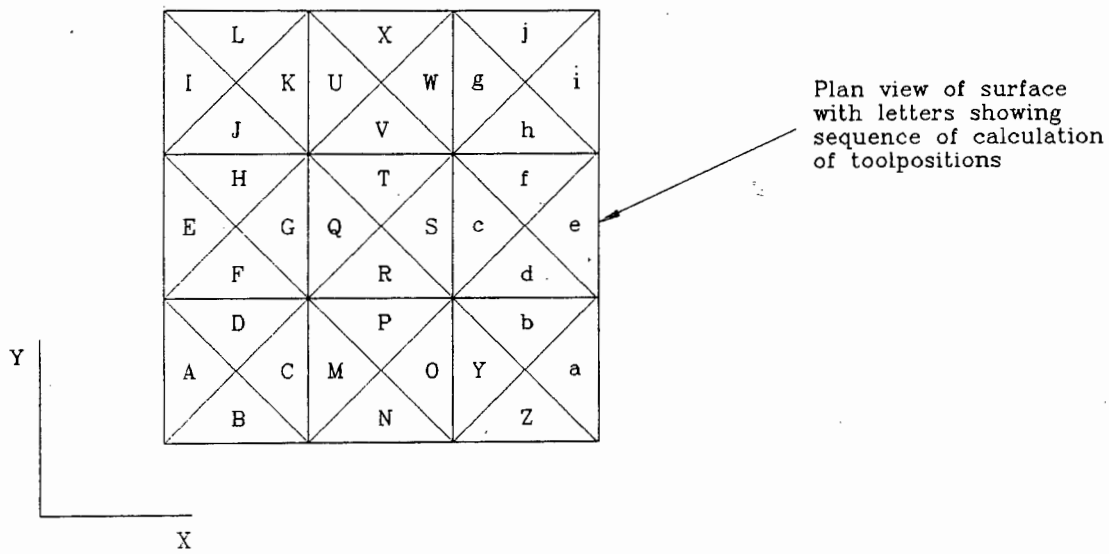


Figure 3.9: The sequence of calculation of the triangular planes on the surface

The tool, however, is required to move across the elements in a "shoelace" pattern as shown in Figure 3.10.

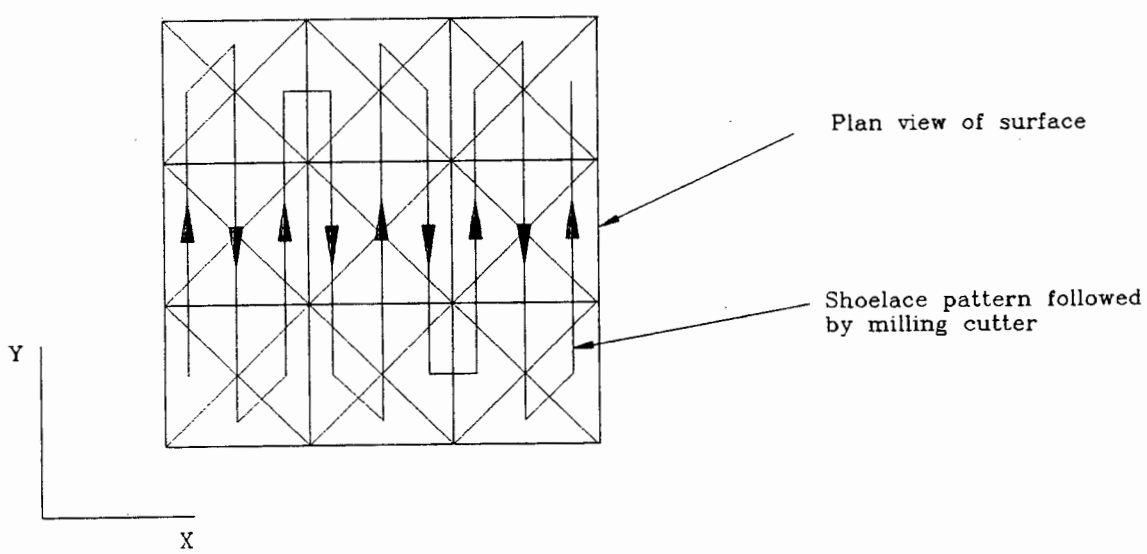


Figure 3.10: Plan view of the surface showing the shoelace type pattern of the toolpath.

When each toolposition is calculated an additional number is calculated and stored. This number, known as a "key", is used to sort the toolpositions into the correct sequence, thus creating the desired toolpath. The calculation of the keys is reasonably complex and needs to be explained.

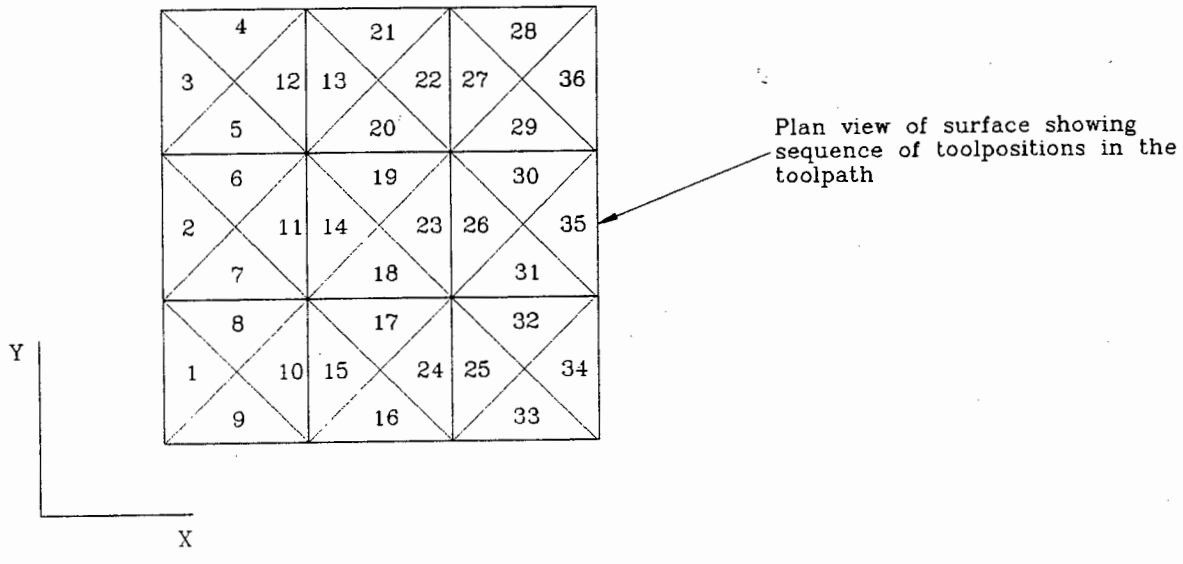


Figure 3.11: Plan view of surface showing the sequentially numbered toolpositions.

Figure 3.11 shows another plan view of the same surface, now with each toolposition numbered sequentially in the order that it is to be machined. The numbers in each triangular plane are the "keys". The four keys for each particular quadrilateral element are dependent on the location of that element within the surface. Therefore each quadrilateral has to be referenced in some way so that the computer can identify its location within the surface.

The frame and the level of the lower left datapoint are used to identify each element. This reference is used not only for the calculation of the keys, but also as an index for the retrieval of surface datapoints from the arrays in the computer.

The keys are dependent on the number of quadrilateral elements comprising each pass of the tool, which is in turn dependent on the number of levels used to define the surface. Each key can be calculated if:

a) the total number of levels of the surface and
b) the reference frame and level of the particular quadrilateral element are known.

The method is as follows:

1. Each quadrilateral element is referenced by the frame and the level of its lower left datapoint, marked A in Figure 3.12.
2. The reference frame of the quadrilateral is used to classify the element as either odd or even.
3. Depending on whether the element is odd or even four appropriate equations from a set of eight, derived by the author, are used to calculate the four keys stored with the four toolpositions of each element.

   Four of the equations are for the "odd" elements and four are for the "even" elements. Each of the eight equations is specific to the position of the triangular plane within the quadrilateral element. Figure 3.12 shows an odd and an even element in a surface. Each triangular element is numbered, and the appropriate key equations are given.

4. By using the keys the toolpositions are sorted and stored in the correct sequence for milling.

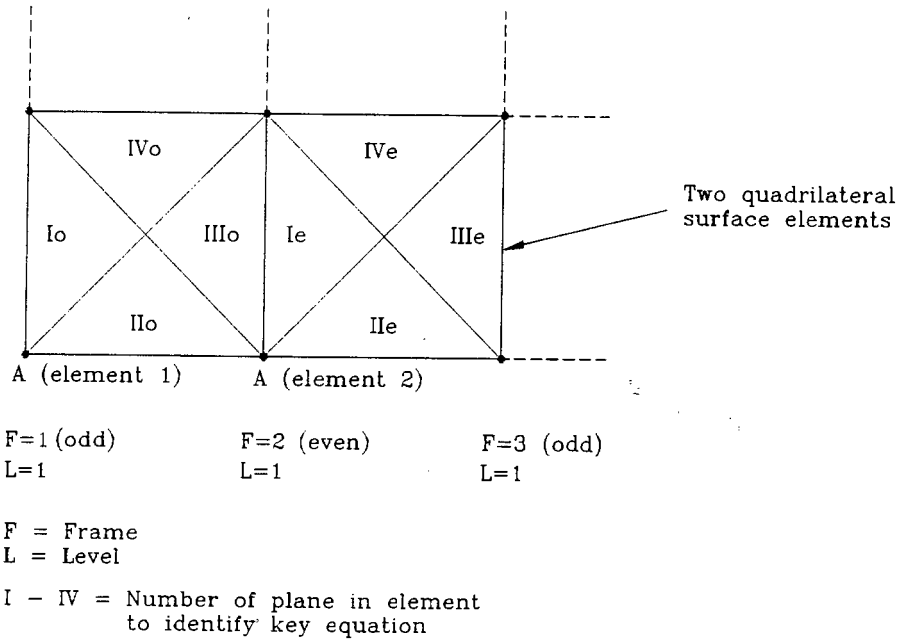


Figure 3.12: Two quadrilateral elements showing the numbering and position of the four triangular planes to select the appropriate key equation.

Below are given the 8 equations for the odd and even elements.

TL = Total Number of Levels in Surface
F = Reference Frame of Quadrilateral Element.
L = Reference Level of Quadrilateral Element.
I-IV = Number of the Triangular Plane from Figure 3.12

For the Odd Elements:

$$\text{Io} : \text{Key} = [(TL-1) \times (F-1) \times 4] + L \quad (11)$$

$$\text{IIo} : \text{Key} = [(TL-1) \times (F-1) \times 4] + (TL-1) + 2(TL-1) - 2(L-1) \quad (12)$$

$$\text{IIIo} : \text{Key} = [(TL-1) \times (F-1) \times 4] + L + 3(TL-1) \quad (13)$$

$$\text{IVo} : \text{Key} = [(TL-1) \times (F-1) \times 4] + (TL-1) + 2(TL-1) - 2(L-1) - 1 \quad (14)$$

For the Even Elements:

$$\text{Ie} : \text{Key} = [(TL-1) \times (F-1) \times 4] + (TL-1) - (L-1) \quad (15)$$

$$\text{IIe} : \text{Key} = [(TL-1) \times (F-1) \times 4] + TL + 2(L-1) \quad (16)$$

$$\text{IIIe} : \text{Key} = [(TL-1) \times (F-1) \times 4] - (L-1) + 3(TL-1) \quad (17)$$

$$\text{IVe} : \text{Key} = [(TL-1) \times (F-1) \times 4] + L + 2(TL-1) + 1 \quad (18)$$

At this point two term must be explained; the temporary toolfile and the final toolfile.

The temporary toolfile is used to store the unsorted toolcoordinates along with their associated keys. The final toolfile is used to store the sorted toolcoordinates with their associated keys. Thus the result of all the processing is the final toolfile in which the toolpositions are in the appropriate sequence for milling the surface.

In order to better illustrate the concepts a numerical example for a simple 2 x 2 element surface is presented in Appendix B.

3.1.2: The Implementation of the Surface Fitting Method in a Computer Program

A complete listing of the computer program is given in Appendices I-L.

Choice of Programming Language

Most computer languages, such as PASCAL or C, would have been able to fulfill the programming objectives of this thesis, but TRUE BASIC, which was the language chosen for the entire milling system, has certain distinct advantages. TRUE BASIC is a modern implementation of the well-known BASIC language. It is modern in the sense that it conforms to the principles of structured programming

supported by most computer scientists. One of its main advantages over other available languages is that it is easily understandable. This is important because the intention is that this system be used as a basis for further work and undergraduate teaching. All U.C.T. engineering students are familiar with BASIC, and therefore would be able to easily follow the workings of the programs.

Another significant advantage of TRUE BASIC is its ability to process matrices. The mathematical approximation described earlier relies heavily on matrix methods to calculate a variety of parameters. TRUE BASIC has full matrix capabilities built in, such as inverting, finding determinants, full matrix arithmetic and also storing and recalling entire matrices from disk. This meant that the complexity of the program could be reduced which again aids in its understanding, not to mention the ease of programming.

TRUE BASIC also has a variety of add-on modules available. The most relevant one for this thesis was the "3D Graphics Package" which was extensively used in the 3D viewer. The application of this module is explained in more detail in the next section.

TRUE BASIC programs are written using the TRUE BASIC editor. These files have the filename extension ".TRU" and are text files which can be edited on any wordprocessor. Within TRUE BASIC there is a compiler which compiles these programs into an uneditable, but quicker to run, form with the filename extension ".TRC". Whilst these programs can be run using TRUE BASIC, they cannot be used directly from the Disk Operating System (DOS).

However, quite recently the author discovered another add-on module for TRUE BASIC entitled TRUE BASIC Runtime Package. This package takes the TRUE BASIC compiled programs and converts them into an executable (.EXE) form which is independent of TRUE BASIC and can be run directly from DOS. This is the most convenient form for the programs because they do not require the TRUE BASIC package to be resident in order for them to run.

The reader will notice that the subroutine library files mentioned in the next section have the filename extension ".LIB". These files are however in the same editable form as the ".TRU" files and they were given this filename extension only to identify them as the subroutine library files. The one exception is the 3DLIB.TRC which is only in the TRUE BASIC compiled form.

The CNC milling system files have been compiled into the ".EXE" format to facilitate their operation, but all the files required for the CNC system have been included on the floppy disks accompanying this thesis in their ".TRU" or ".LIB" editable format. The only requirement is that the user have the TRUE BASIC package available to run them. The TRUE BASIC package has copyright protection and can thus only be run by licensed users.

The only disadvantage of the TRUE BASIC program is that it is slow in comparison to some other languages such as PASCAL or C. However, because this system is to be used as a first step for students interested in CAD/CAM, its ease of use, understanding and programming made this language the clear choice.

Overall Structure of the Program

The overall program structure is shown below in Figure 3.13.

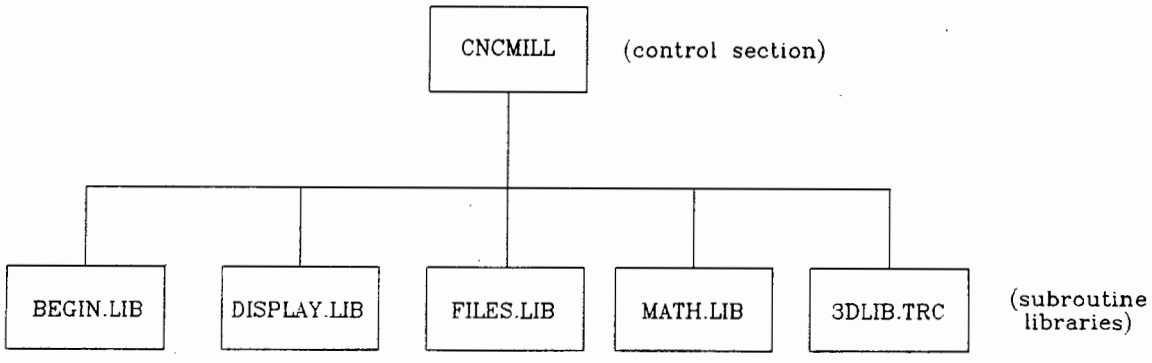


Figure 3.13: Overall program structure of the milling system

The program consists of a control program and five subroutine libraries. The control program performs the overall processing of the system and uses subroutines contained in the libraries to perform certain specialised tasks. Each subroutine library contains a collection of subroutines specific to a single purpose. For example, the DISPLAY.LIB library contains the subroutines which use the TRUE BASIC 3D Graphics module to display the surface and toolpath in 3D. All the program statements for displaying the surface are contained within the library, and only a single statement (CALL DISPLAY) is used in the control program to perform the entire 3D display.

Similarly the MATH.LIB library contains all the subroutines for the mathematical approximation of the surface and the calculation of the toolpath. Thus the single command CALL TOOLPOSITION within the control program performs all the matrix algebra to calculate the toolpositions for each element of the surface.

The library BEGIN.LIB contains subroutines which request the user to give the name of the toolfile, the diameter of the hemispherically ended cutter and the number of

frames and levels. These are used to initialise the system and to load the correct surface data file from disk into memory.

The library FILES.LIB contains the subroutine which opens and closes the disk files and sorts the toolfile into the correct sequence.

The library 3DLIB.TRC is the TRUE BASIC add-on module for displaying three-dimensional graphics. It is in the compiled TRUE BASIC form because there is no need to make any alterations to this commercially supplied program. Section 3.2 explains in more detail the application of this package to this thesis.

There are many advantages in structuring a program in this way. The most important is that the entire program is easy to read and understand. The flow of the program is logical which makes it easy to modify the system. Furthermore the fact that all the relevant instructions are contained in a single library makes the adaptation of any specific part of the system a simple matter. For example, the user may not wish to see the toolpath in the 3D display. The user immediately knows that only the subroutines in the display library need be altered to perform this modification.

Description of the Functioning of the Program

In order to describe the logic sequence of the program, a flowchart is shown in Figure 3.14. The flowchart illustrates only the main, or control, section of the system.

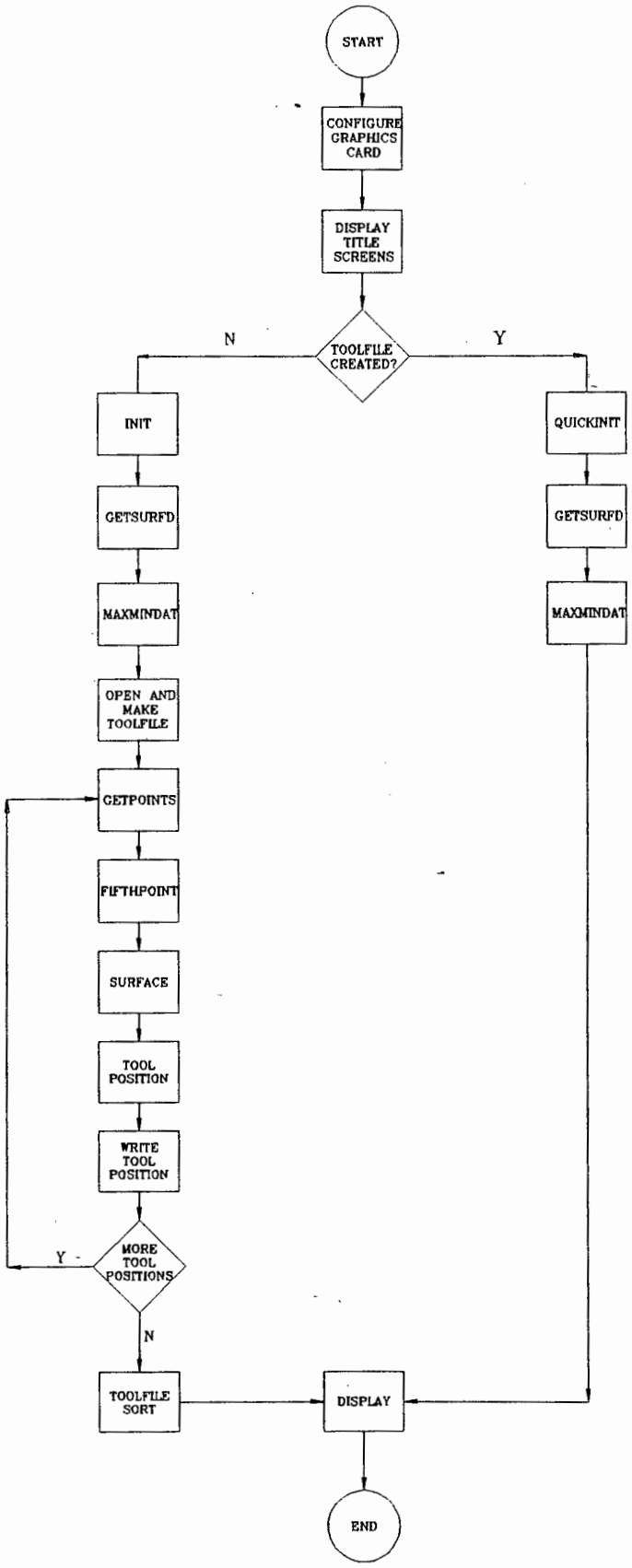


Figure 3.14: Flowchart for the milling system

The program begins by asking the user which graphics card is installed in the computer being used and then configures the graphical display accordingly. The introductory screens are then displayed and thereafter the user is asked whether a toolfile is to be created for the surface or whether a previously created toolfile and surface are to be viewed.

If the toolfile already exists the program allows the user to visually inspect the surface and toolpath to make sure that it is as anticipated. The first step, QUICKINIT, asks for the name of the toolfile and the number of frames and levels. The subroutine GETSURFDAT then loads the surface data coordinates from disk into memory, whereafter MAXMINDAT determines the maximum and minimum X,Y and Z values of the surface. The DISPLAY subroutine which follows uses these maximum and minimum values to define the viewing volume and then displays the 3D view of the surface together with the toolpath. After the user is finished viewing the program terminates.

If the toolfile does not exist then the program must perform the mathematical processing and calculation of the toolpath prior to viewing it on the screen. Subroutine INIT requests the same information from the user as QUICKINIT, but in addition requests the diameter of the hemispherically ended cutter. The same subroutines, GETSURFDAT and MAXMINDAT, that are used in the other branch are used to load and process the surface data. They have been repeated in this branch to make the flowchart as clear as possible. The subroutine OPEN AND MAKE TOOLFILE creates the temporary and final toolfiles and opens them for storing the toolposition data.

At this stage the preliminary operations are complete and the program enters the main processing loop to compute all the toolpositions. The loop begins with GETPOINTS which

contains subroutines which perform the task of projecting 3D lines onto a 2D screen. The 3D viewer written by the author for this system uses these subroutines to draw the 3D lines which make up the surface and the toolpath. The relevant terms can be explained by drawing an analogy with a conventional camera, as shown in Figure 3.15.

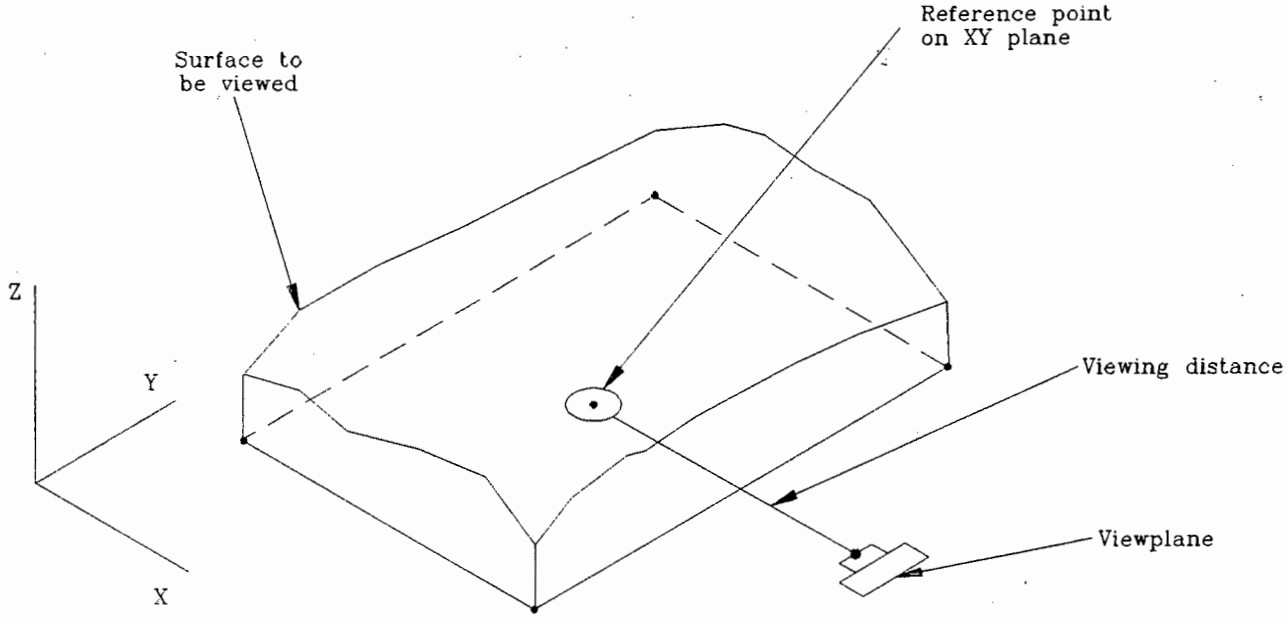


Figure 3.15: The camera analogy of the 3D graphics package

The camera is aimed at the reference point of the object to be "photographed"; the film corresponds to the viewplane; and the viewing distance is the distance between the camera and the object. In taking the picture the object is "projected" onto the view plane.

After the location of the camera and its associated variables have been specified, the type of projection must be defined. The 3D package provides both parallel and perspective projections. In the case of a parallel projection moving closer or further away from the object makes no difference to the size of the image because the projection lines never converge - they remain parallel to

infinity. This was preferable because the perspective projection tended to distort the surface and give a false idea of the shape.

The extents of the surface need to be specified in order to display the entire surface on the screen. The term "viewing window" is often used in 2D graphics to describe the portion of the total picture that is visible on the screen. This can be compared to the image from a magnifying glass representing a window of part of the total object. In 3D graphics the viewing window is given a third dimension and becomes a "viewing volume" as shown in Figure 3.16. The contents of the viewing volume will be displayed on the screen.

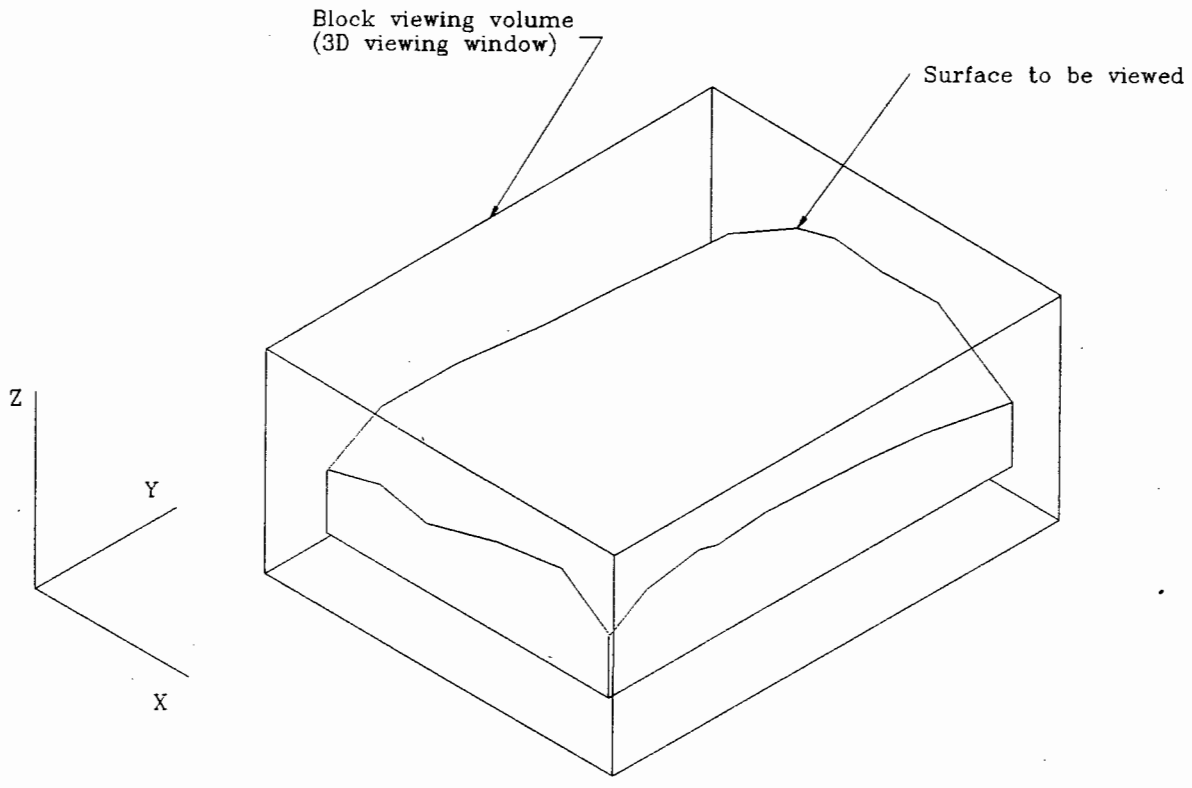


Figure 3.16: 3D viewing volume concept used by TRUE BASIC

Having outlined the pertinent terms it is now possible to explain in more detail how the 3D TRUE BASIC package was

used in this system to display the surface and the toolpath.

As previously mentioned, the extents of the surface in the X,Y and Z directions are computed by a separate subroutine (MAXMINDAT) prior to displaying the surface and toolpath. These parameters are used to define the viewing volume. The reference point at which the "camera" is aimed is set at zero in the Z direction and in the middle of the surface in the XY plane. The viewing volume and the reference point are thus fixed for that particular surface, shown graphically in Figure 3.17

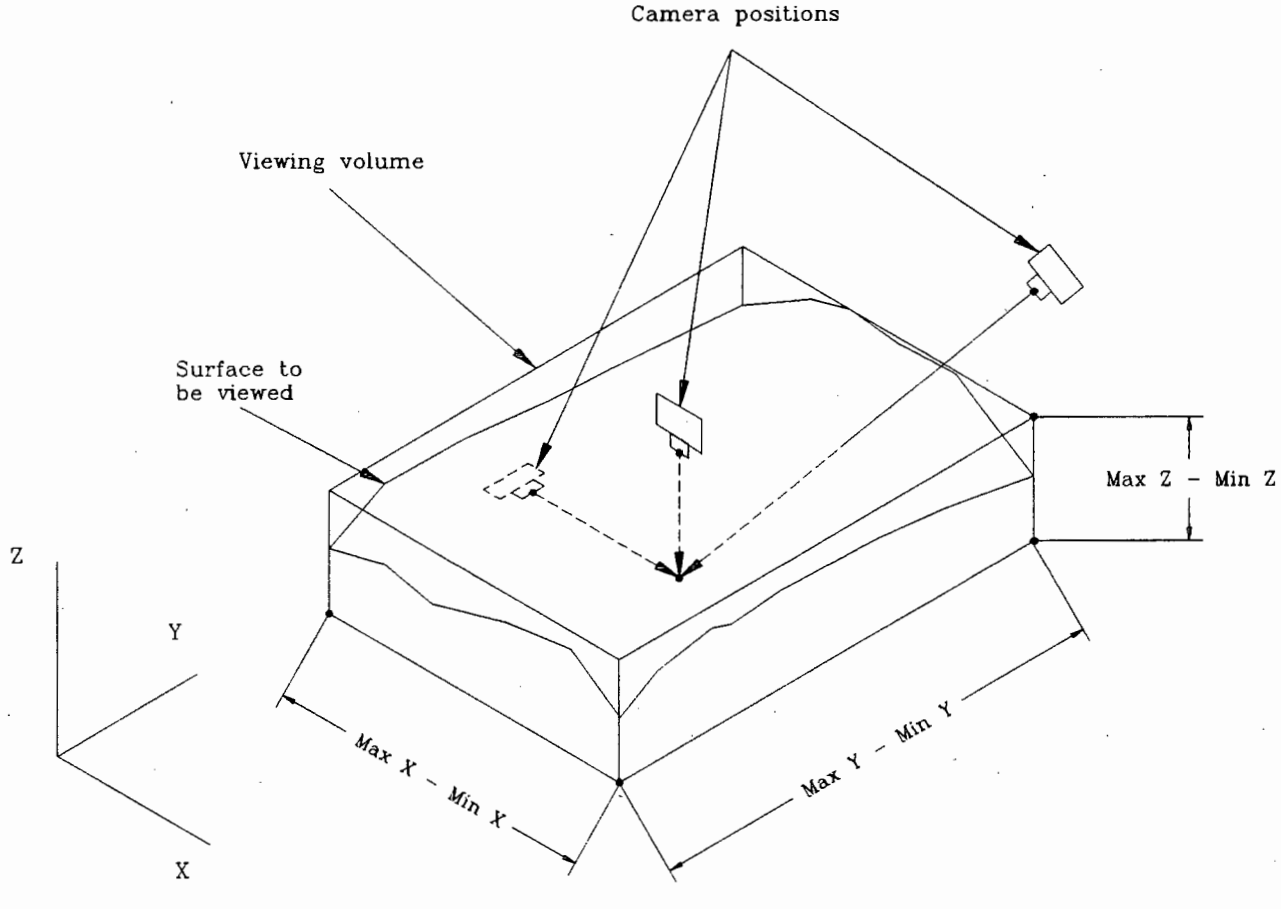


Figure 3.17: The 3 camera positions, the reference point and the viewing volume as defined in the milling system

The viewer then determines the camera positions for three commonly used views in engineering: the plan view, the side view and a 3D pictorial view. The user can select any of these and in addition he can position the camera anywhere he chooses by supplying his own set of X,Y and Z camera coordinates. The same reference point and viewing volume will still apply.

In using the system the author found that the three views provided are usually enough to give a good idea of the surface and only once, in the case of an extremely convoluted surface, was it necessary to define a different camera position. The "menu" for the viewer from which the user selects the camera position is shown in Figure 3.18.

```
                    CNC MILLING SYSTEM
              THREE DIMENSIONAL VIEW SELECTOR

Three views of the surface are available ;
   1 - 2-D view from directly above the surface , ie XY plane
   2 - 2-D view from the side of the surface , ie YZ plane
   3 - 3-D view from the X Y Z extremity
   4 - Specify your own Camera Position
   5 - End viewing session

        SELECT VIEW BY PRESSING A NUMBER FROM 1-5
```

Figure 3.18: The "menu" of the display program.

The plan view is obtained by positioning the camera directly above and aiming it at the reference point and the side view by positioning the camera half way along the surface in the Y direction and aiming it at the reference point.

The 3D view is slightly more complex. The X and Y coordinates of the camera position are taken from the X and Y coordinates of the furthest corner of the viewing volume and the Z coordinate is twice the maximum height of the surface. This is illustrated in Figure 3.17.

Once the camera position and the viewing volume have been defined, the viewer enters a loop which draws the four 3D lines comprising each quadrilateral element. This is repeated for each element until the entire surface is drawn. Each line of the quadrilateral element is drawn by retrieving two datapoints from the arrays stored in memory and then, using a TRUE BASIC 3D subroutine (PlotOn3), drawing the 3D line between the points. In this way all four lines of each quadrilateral element are drawn. Drawing the element in this way is conceptually straightforward but necessitated four separate calls to the PlotOn3 subroutine. Each subroutine "call" adds to the execution time of the program and consequently the time taken to draw a surface is lengthy. The time varies from about 32 seconds for a 10 x 10 element surface to about 5.5 minutes for a 30 x 30 element surface on a standard IBM PC.

Figure 3.19 shows a 3D view of the hemisphere on a flat plane as produced by the viewer.

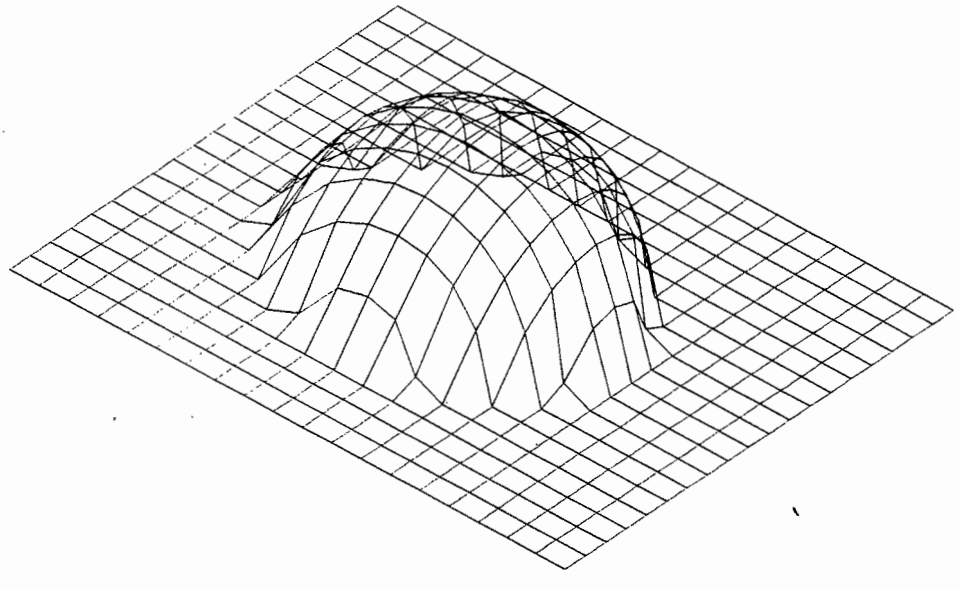


Figure 3.19: 3D view of a hemisphere on a flat plane

After the surface has been drawn the viewer uses the list of toolpositions in the final toolfile to draw the toolpath. This is a straightforward procedure which merely draws a 3D line between each successive toolposition. It would have been possible to use the same TRUE BASIC 3D subroutine, PlotOn3, as used to draw the elements, but in this case it was far quicker to use another TRUE BASIC 3D subroutine called MatLines3. This subroutine draws a series of line segments between points contained in an array. The array must contain the X,Y and Z coordinates of each successive data point in the order in which they are to be joined. The array can be considered to be a long list of points with each point consisting of its X, Y and Z coordinate. This is shown in Figure 3.20.

X	Y	Z
X_1	Y_1	Z_1
X_2	Y_2	Z_2
X_3	Y_3	Z_3
.	.	.
.	.	.
.	.	.
X_n	Y_n	Z_n

Figure 3.20: An example of a MatLines3 array

This method is faster because each call to a subroutine takes some time, and by using only one subroutine call rather than many subroutine calls to draw the toolpath the display is speeded up. The reason why this method was not used to draw the quadrilateral surface elements, which would have speeded up the drawing of the surface, is because the MatLines3 subroutine requires an <u>additional</u> array to be constructed which uses a large amount of memory. There was sufficient memory for drawing either the

toolpath OR the surface in this way but not both. The reduction in display times is greatest if the toolpath rather than the surface is drawn because the surface datapoints would have first to be sorted and stored in the correct order for the MatLines3 subroutine. By choosing the toolpath this intermediate step was not necessary and the full benefit of reduced drawing time is obtained.

In any system which uses graphics a hard copy (paper printout) of the display is desirable. One of the major drawbacks of the TRUE BASIC 3D system is its inability to generate a plot of the display. However there are some 3D CAD systems, such as AUTOCAD v9, which allow data from other programs to be "imported" into them.

The plotting facilities of the CAD packages can then be used to generate hard copy plots of the displays. Initially the author did not have access to any such package and had intended to photograph the screen to illustrate the 3D capabilities.

However in late April 1988 the AutoCad Centre in Johannesburg kindly made a copy of AutoCad v9 available. Despite having been completed at this stage the viewer in this thesis was updated to produce a disk file which could be used with AutoCad.

The format of the file, called Drawing Exchange Format (DXF), is specified by AutoCad. The file consists of three parts: the header section, the entities section and the ending section. The main part of the file is the entities section where each 3D line is represented by a start point and an end point. AutoCad requires a special coding to identify each of the X,Y and Z coordinates of the start point and the end point respectively. The format of the DXF file as created by this system for use by AutoCad is shown below in Figure 3.21.

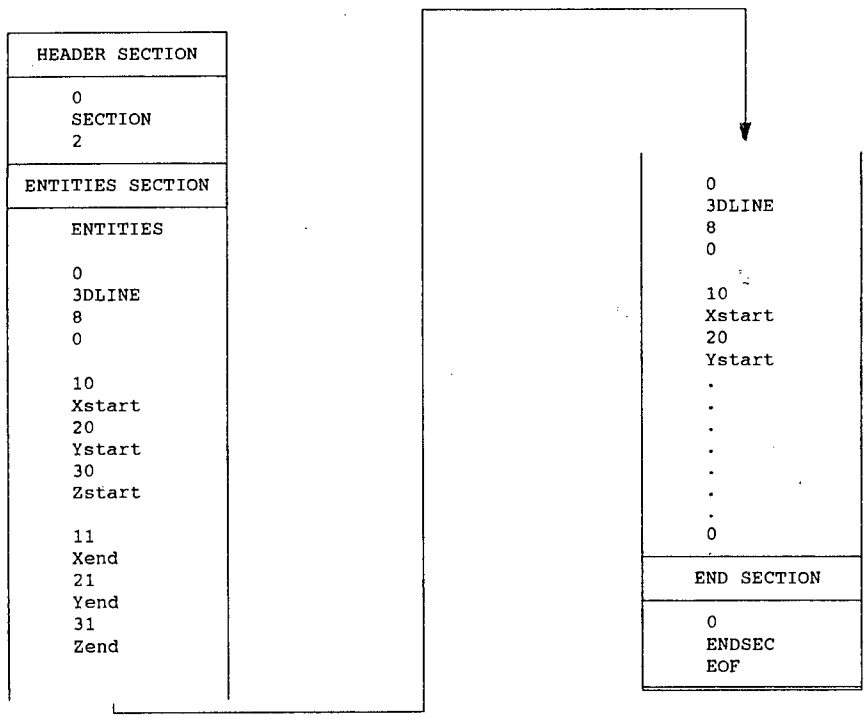


Figure 3.21: Format of the AutoCad DXF file created by the 3D viewer.

The data necessary for the AutoCad DXF file was identical to that used by the 3D display, and therefore as each data point was retrieved for use by the display subroutines it was also written to the DXF file. This system proved very useful because not only could a plot of the surface be obtained, but the user could also use AutoCad to "zoom" in to inspect potential problem areas of the surface. Although the viewer in this system can draw the 3D view from any viewpoint, AutoCad, because it is such a sophisticated program, has a significant speed advantage.

(The user should not despair if he/she does not have access to AutoCad because although it is slower the viewer provides for all the 3D display that is required. If

AutoCad is available however, the system can take full advantage of AutoCad's facility to plot and manipulate the surface. Whilst this is not essential, it does add to the overall usefulness of the system.)

The only drawback with the three-dimensional viewer is that there is a significant amount of computation involved which tends to slow the system down. However in the author's opinion the benefits of providing a three-dimensional viewer in terms of the overall productivity of the system far outweighed the extra time required to generate the 3D graphics. Also it is possible from the display to decide, albeit subjectively, whether the element size used to define the surface seems acceptable. In using the system the author found that the viewer immediately highlighted some surfaces which had been incompletely defined due to an oversight during the data capture stage. This meant that the surfaces could either be rejected or repaired without losing any time by doing further processing.

Once the surface and the toolpath have been verified, the toolpath must be converted into a CNC program for the Bridgeport 3 Axis milling machine.

3.3: The Postprocessor

(A complete program listing of the postprocessor is given in Appendix J)

The final toolfile that is created is a sequential list of the X,Y and Z coordinates of the toolpath. The postprocessor translates this list into a syntactically correct program or programs for the computer in the Bridgeport CNC milling machine. The postprocessor also has to add commands to the CNC program which tell the machine which tool to use, the feedrate, the tool change position, etc.

Unless a simple surface containing few elements is used it is not possible to create a single CNC program containing all the toolpositions required to mill a surface because it would be too big to store in the memory of the Bridgeport CNC computer. The postprocessor therefore creates a number of short independent CNC programs, or CNC modules, each of which, when loaded into the CNC machine and run, mills a part of the surface. When each CNC module has been completed it is cleared from the CNC computer memory and the next CNC module is loaded and run. In this way all the CNC modules are linked and used to mill the surface. This is explained in more detail later.

Before explaining how the postprocessor creates each CNC module the various parts of each CNC module and how they control to the tool movement must be understood. Figure 3.22 shows an actual example of a CNC module.

Apart from the START and TERMINATING sections the X,Y and Z coordinates are not important in relation to the diagrams which follow and merely illustrate the format of the CNC module.

```
G0 G90 G71 X-20.00 Y-20.00 T1 M6          - START SECTION
X 5.83  Y 7.50  Z 12.50                   - RAPID POSITIONING SECTION
G1X   5.83Y   7.50Z   2.50F800
X   5.83Y  12.50Z   2.50
X   5.83Y  17.50Z   2.50
.     .      .       .
.     .      .       .                    - TOOL FEED SECTION
.     .      .       .
X  21.96Y  47.15Z   9.09
X  22.82Y  40.48Z   5.50
X  23.69Y  36.55Z   3.10
G0 G90 G71 X0.0 Y0.0 M2                   - TERMINATING SECTION
$
```

Figure 3.22: A CNC module

The program consists of four sections: the start section, the rapid positioning section, the tool feed section and the terminating section.

The movement of the tool under the control of the CNC module will be described by referring to the CNC module above and to Figures 3.23a-d below. The same toolpath is used throughout the milling of the surface; there is no separate toolpath for roughing and finishing. The depth of each cut is controlled by raising the milling table the appropriate amount between passes, taking into account the material being milled, the cutter limitations, the feedrate, etc. The figures and the description of the tool movement show the tool in relation to the surface whilst performing the final cut.

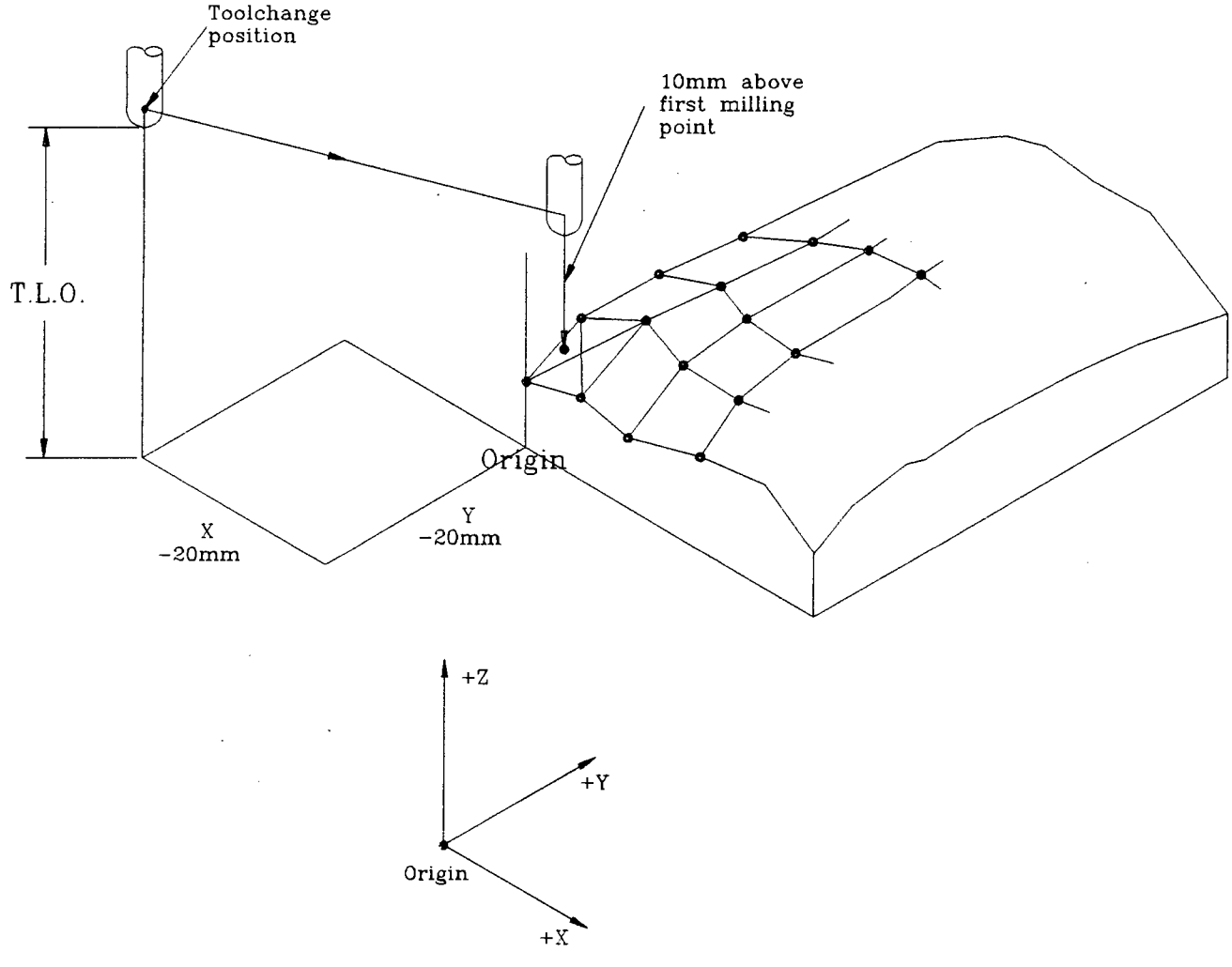


Figure 3.23a: Location of Origin, Toolchange Position and the Tool Length Offset.

Figure 3.23a shows the location of the origin, the toolchange position, the tool length offset (T.L.O.) and the surface. The toolchange position is at X=-20 mm and Y=-20 mm from the origin with the tool in the fully retracted, or home, position. The tool length offset is the distance between the tool tip when it is in the home position and the Z=0 plane. Every coordinate in the program is measured in millimeters from the origin and the physical location of the origin on the surface is chosen by the user when the workpiece and the machine are first set up. The surface to be milled is always located in the positive X, positive Y and positive Z region.

The START section is repeated below:

```
G0 G90 G71 X-20.00 Y-20.00 T1 M6
```

This is the first line executed by the machine and, irrespective of the position of the tool in space, it causes the tool to fully retract and then to move rapidly from wherever it is to the toolchange position. The TLO for Tool 1 is also loaded into the memory of the CNC computer.

The second line, or RAPID POSITIONING section:

```
X 5.83  Y 7.50  Z 12.50
```

causes the tool to move rapidly from the toolchange position to a position 10 mm above the first milling point as shown in Figure 3.23a.

The beginning of the TOOL FEED section is the first command to actually mill the surface and is shown in Figure 3.23b.

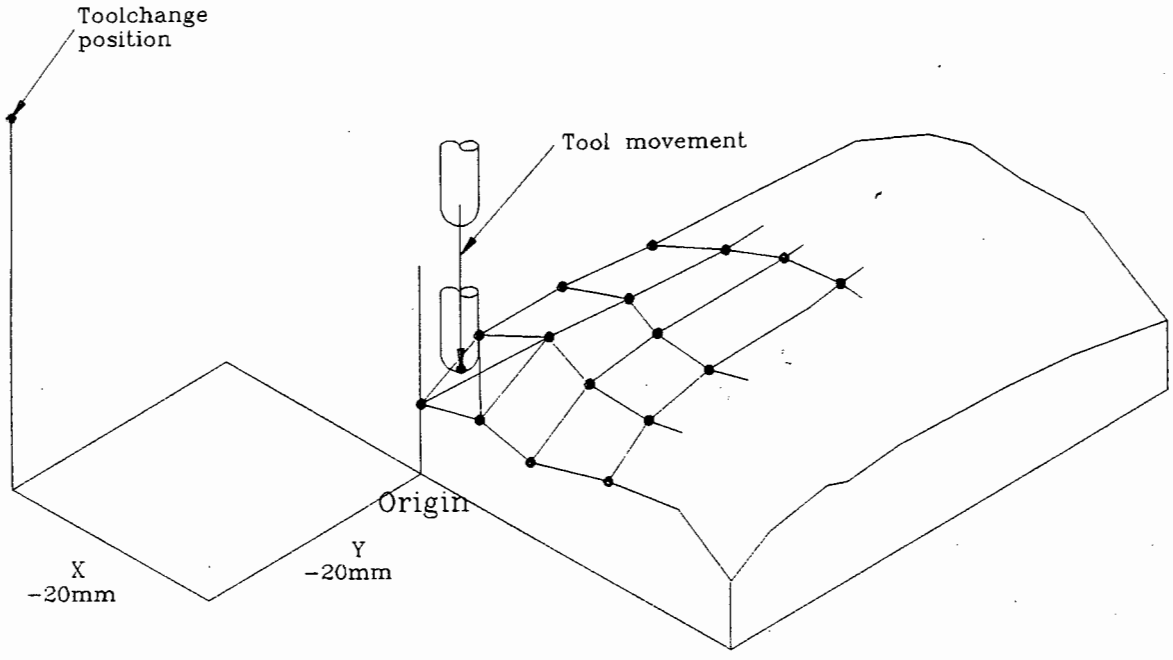


Figure 3.23b: Tool movement to the first toolposition on the surface

The CNC command which begins the tool feed section is:

G1 X 5.83 Y 7.50 Z 2.50 F 800

The machine is put into linear interpolation mode, the feedrate is set to 800 mm/min (the feedrate can be altered in the postprocessor) and the tool moves from its current position down 10 mm in the Z axis towards the surface, effectively drilling down to the first toolposition at X 5.83 Y 7.50 Z 2.50. The depth of the cut is controlled by raising the milling table the appropriate amount.

The tool continues from its current position to the toolposition:

X 5.83 Y 12.50 Z 2.50

by linear interpolation between the two toolpositions as shown in Figure 3.23c. This point-to-point linear interpolation is continued until the end of the tool feed section.

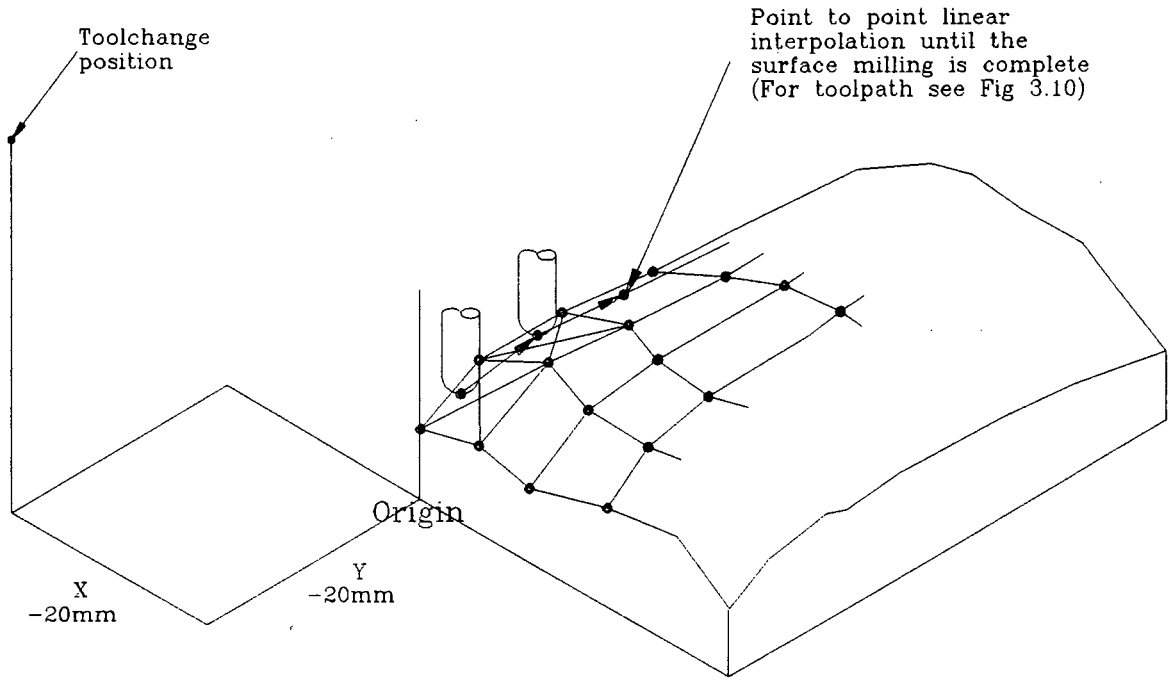


Figure 3.23c: Tool movement during the Tool Feed section

The TERMINATING section, shown in Figure 3.23d, switches the spindle off and the tool retracts to the home position. The tool then moves to the X=0 and Y=0 position. At this point the current CNC module has been completed. It is then cleared from the CNC computer's memory and the next CNC module is loaded and run.

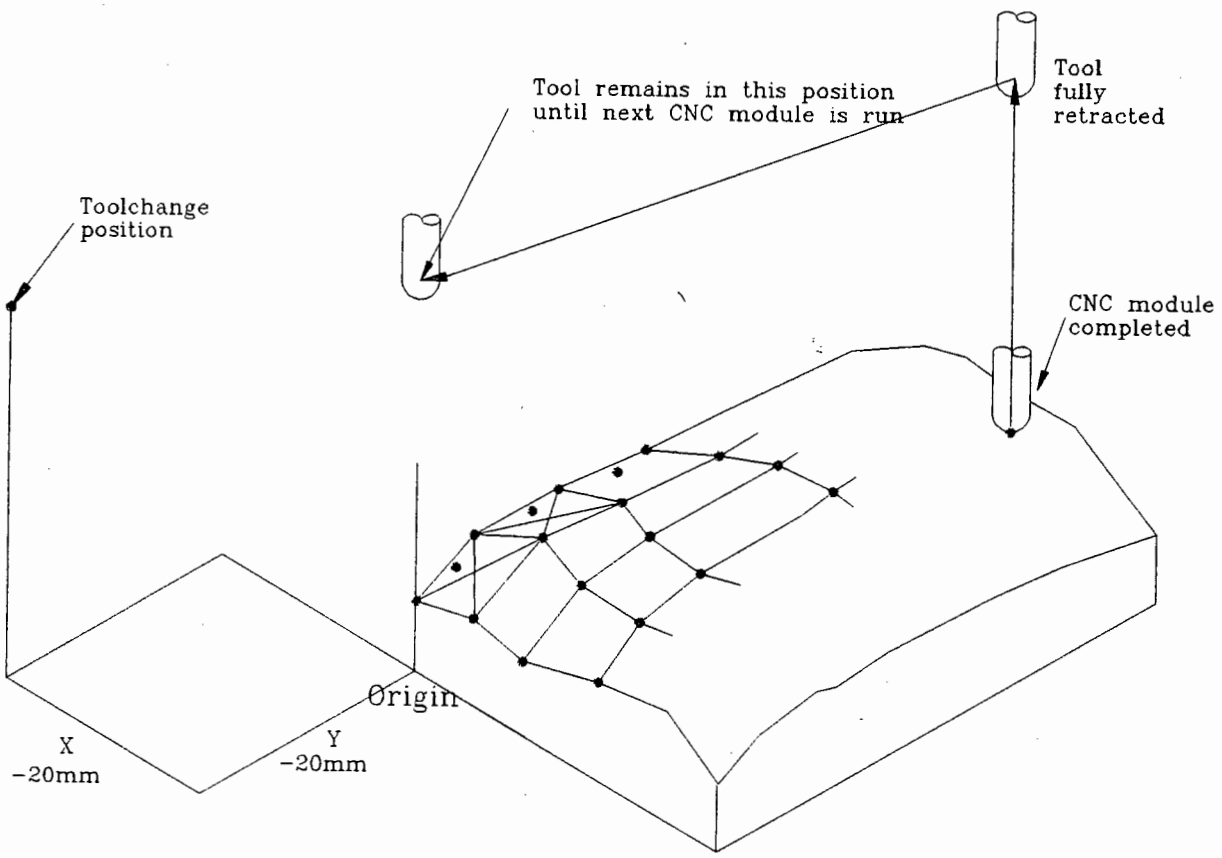


Figure 3.23d: Tool movement during the Terminating Section

The reason for leaving the tool in the X=0 Y=0 position between programs is to enable the user to turn off the machine between programs without having to reset the location of the origin on the surface.

This is very useful when machining large surfaces comprising many CNC modules which require two or more milling sessions, for example over two successive days. When the machine is restarted the physical position of the tool automatically defaults to X=0 and Y=0, and if the physical position of the tool relative to the workpiece corresponds to this position then the user does not have to reset the location of the origin. Only the tool length offset, which is simply typed into the CNC computer, needs

to be reentered. Not only is this a time-saving feature, but it also prevents any inaccuracies caused by having to reset the position of the origin. The author found this a very convenient feature when partially machining part of a large surface on one day and returning to complete it the next.

As mentioned, the amount of memory available in the CNC computer is small and is limited to 8 000 bytes. The length of a single program to mill a 20 x 20 element surface is, however, approximately 38 000 bytes long. At least five separate programs are therefore required to mill the complete surface. In translating the toolfile into the CNC module the postprocessor keeps track of the number of bytes it has accumulated, and when 7 800 bytes have been reached (this leaves some memory space free in the CNC computer) it terminates the current CNC module and begins the next one.

Obviously there must be no loss of toolpositions between programs, and the postprocessor accomplishes this by ending one program and beginning the next with the same tool coordinates to ensure continuity in the toolpositions.

The filename given to each CNC module stored on disk is numbered sequentially from the name of the CNC program file input by the user in the INITIAL section. All files are given the file extension ".PRG" to distinguish them easily on the disk. Thus if the name given by the user is "ABCD" then each CNC module will be called "ABCD1.PRG", "ABCD2.PRG", "ABCD3.PRG", etc.

Now that the output of the postprocessor has been discussed in some detail, the means by which the output is achieved can be described.

The basic functioning of the postprocessor program is quite simple. The program reads the X,Y and Z coordinates from

the toolfile and constructs a program suitable for the CNC machine. Figure 3.24 below shows the overall structure of the postprocessor.

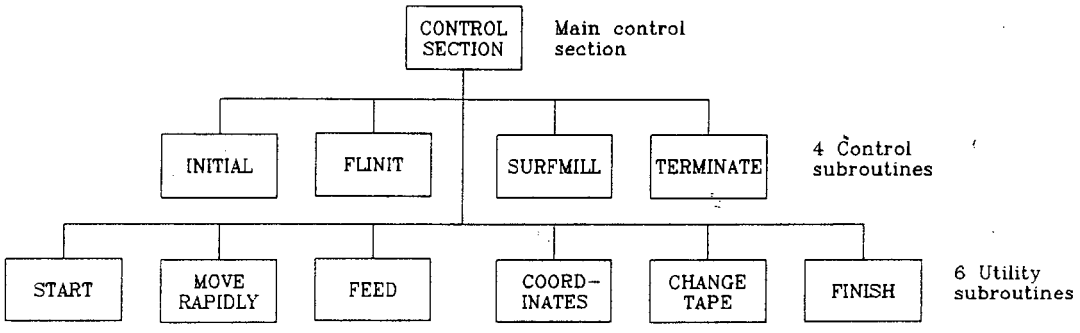


Figure 3.24: The overall structure of the postprocessor

The description of the structure and functioning of the postprocessor will be related to the structure of the CNC module file.

The postprocessor consists of three parts: the control section, the control subroutines and the utility subroutine library. The control section consists of only four lines, the function of which is to call each of the four control subroutines in the correct sequence and thus provide the program flow.

The four control subroutines: the initialisation section (INITIAL), the start and rapid positioning section (FLINIT), the tool feed section (SURFMILL) and the terminating section (TERMINATE) provide the more detailed program logic.

The utility subroutine library consists of a series of subroutines which were modified and improved from those originally written by Berelowitz [2]. These subroutines do the work of assembling each toolposition into a CNC command. Each of the four control subroutines makes use of these utility subroutines.

The purpose of the INITIAL subroutine is to gather information from the user, such as the name of the toolfile, the number of frames and levels, the names of the CNC module files, etc. It uses this information to open and create the necessary disk files and to calculate the number of toolpositions required for the surface. It does not write anything to the CNC module file.

The FLINIT subroutine reads the first toolposition from the toolfile and writes the start and rapid positioning section of each CNC module. This causes the tool to move to a position 10 mm above the first milling point and also arranges for the data to be in the correct format (two decimal places) for these lines.

The SURFMILL subroutine writes the bulk of each CNC module, the tool feed section, which contains all the toolpositions for each CNC module.

The TERMINATE subroutine uses the FINISH subroutine to write the command,"G0 G90 G71 X0.0 Y0.0 M2" which ends each CNC module. It also closes all the disk files which have been used.

As the SURFMILL subroutine is the most important part of the postprocessor it requires a more detailed explanation and therefore a flowchart is given in Figure 3.25

The SURFMILL subroutine begins by reading the X,Y and Z tool coordinates from the toolfile on disk. The coordinates are then truncated into a two decimal place format. The utility subroutine FEED is then called. This writes the tool coordinates in the appropriate form to the CNC module on disk. It also increments the counter which is used to monitor the length of the CNC module. If the length is greater than 7 800 bytes it calls another utility subroutine CHANGETAPE which ends the current CNC module and

begins the next one. This process is repeated until all the toolpositions have been used, at which time the SURFMILL subroutine returns to the control section to terminate the postprocessor.

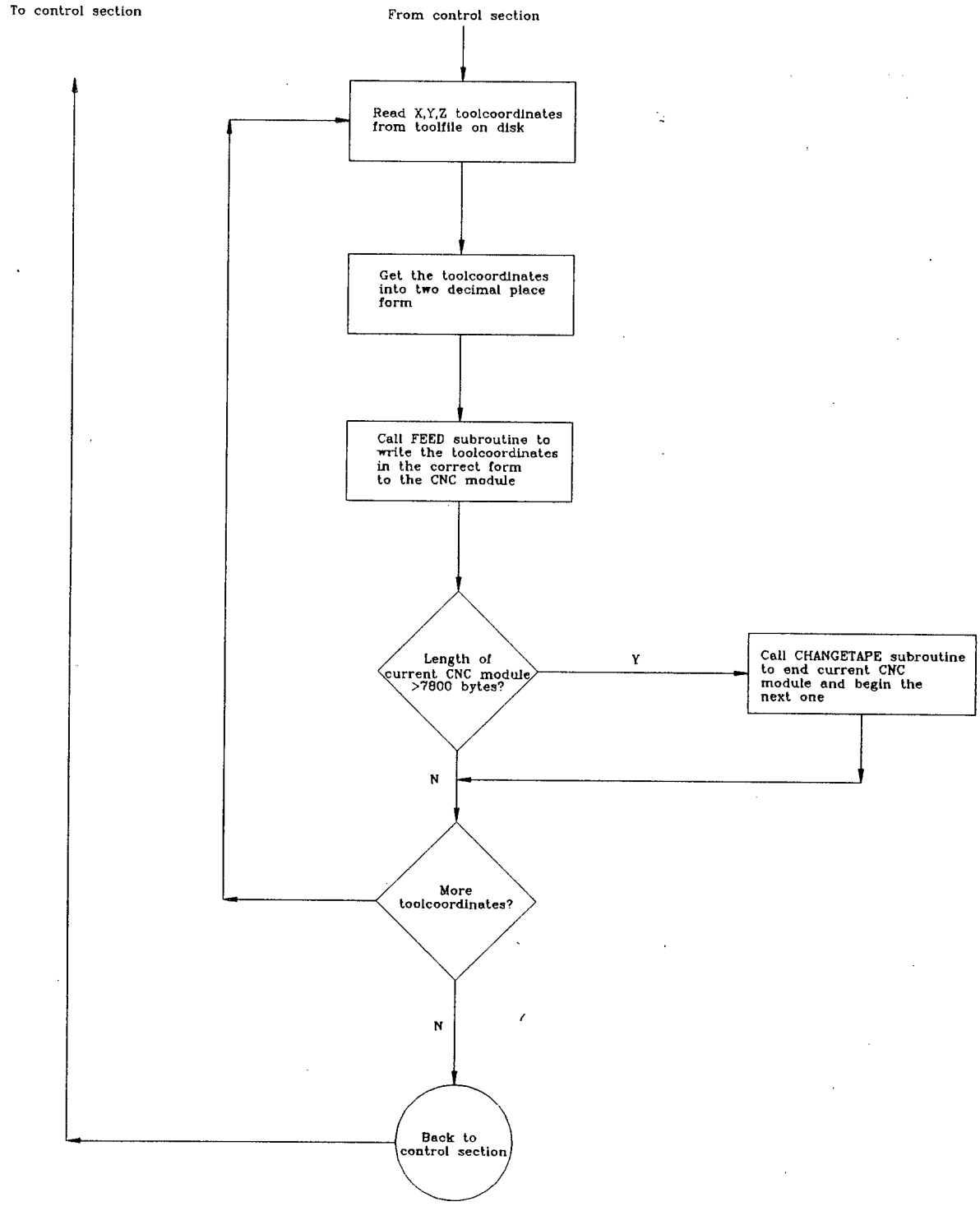


Figure 3.25: SURFMILL flowchart

The simplest utility subroutine in the library is called "START" (see Appendix J) which writes the first line of each CNC module. It is called from the FLINIT subroutine of the postprocessor by using the statement CALL START and writes the string "G0G90G71X-20.00Y-20.00T1M6", the first line of each CNC module, to the CNC module on disk. It also increments the byte counter which is used to keep a check of the length of the current CNC module.

There are a number of these utility subroutines contained in the postprocessor, such as MOVERAPIDLY, COORDINATES, FEED and others (refer to Appendix J). In addition to being called from one of the subsections of the postprocessor, subroutines can also be called by each other.

The most frequently used utility subroutine is the FEED subroutine called from the SURFMILL subroutine described earlier. It is responsible for writing each toolposition to the CNC file and also for setting the CNC machine into linear interpolation mode. The FEED subroutine uses the COORDINATES subroutine to put the toolpositions into the correct form and also to put the letter "X" "Y" or "Z" in front of the appropriate tool coordinate.

When the end of the toolfile is reached the postprocessor ends the current CNC module and displays the message:

All the CNC module files have been created.

The postprocessor then returns control of the computer to the user. The next stage is to transfer these CNC modules from the disk on which they are stored to the memory of the CNC computer.

3.4: The Communications Package

From the preceding discussion it is clear that the CNC program produced by the postprocessor is usually lengthy. It would be extremely tedious to manually retype the program into the memory of the CNC computer, and therefore a communications package was developed to facilitate the transmission of the CNC program to the CNC computer. The communications package consisted of two parts: the software and the hardware. The software that was written used the programming facilities of a commercially available data communications package called MIRROR. The hardware consisted of an IBM Asynchronous Communication adaptor and a specially designed cable.

For those readers unfamiliar with data communications a brief introduction and description of the relevant terms is given in Appendix C.

3.4.1: Facilities for Data Input on the CNC Machine

The CNC machine in the Mechanical Engineering Department at U.C.T. has a teletypewriter with a paper tape puncher and a tape reader for inputting CNC programs.

There were two conventional methods for inputting a CNC program into the CNC computer. The first was to use the teletypewriter to type the program directly into the memory of the CNC computer (via the serial interface), and the second was to use the teletypewriter to produce a punched paper tape which, via the tape reader, was then loaded into the memory of the CNC computer. Once the CNC program was in the memory of the CNC computer the teletype could be used to edit the program.

Berelowitz [2] devised a method to connect the IBM PC to the teletype. He then transmitted the CNC program from the IBM PC to the teletype and produced a paper tape which was then loaded into the tape reader in the usual manner. Whilst this eliminated having to manually retype the data, it was nevertheless slow and tedious and approximately 25 metres of paper tape were required in order to represent the 8 000 bytes of the CNC computer memory. This was also the maximum length of tape that could fit on the reel in the tape reader. As Berelowitz [2] did not mill any surface requiring more than 8 000 bytes this was not a major problem for him. However, as mentioned in the previous section, this system requires five CNC modules for a 20 x 20 element surface and, although it would be theoretically possible to produce five 25 metre reels of tape, it would be most impractical.

There were two possible alternatives to improve the data transfer. The first was to use or bypass the opto-mechanical part of the tape reader and to use it for data input. The speed of the tape reader is approximately 1200 - 1600 baud which allows rapid data transfer and is considerably faster than the teletype's 110 baud. This was not attempted however because there was no information in the CNC machine manuals about the wiring or the functioning of the tape reader. Attempts to obtain information from the supplier also proved fruitless. This made it extremely difficult to implement such a modification, and also there would be no guarantee of success. A "black box" [30], described in Chapter 2, is available for communicating CNC programs from an IBM PC to a CNC controller. It was not possible to obtain such a device because of the cost, which is approximately R4 000. Although it may have been possible to build a similar device, this was beyond the scope of this thesis.

The second alternative was to connect the serial interface of the IBM PC to the serial interface of the CNC. Since the same electronic link was used by the teletype for communicating with the CNC computer, all that was required was to make the IBM use the same signalling code as the teletype. The IBM PC could then also be used for editing programs and controlling the machine in the same way as the teletype had been previously. Although the baud rate is only 110 bits per second, which is slow, the ease and speed of data transfer would be significantly improved. Berelowitz [2] attempted to implement this but without success.

The next section describes the development of the hardware to provide the electronic link between the IBM PC and the CNC computer.

3.4.2: Hardware Development for the Serial Link

The configuration of the interfaces and a circuit diagram of the connecting cable is given in Appendix D.

The development of the serial link involved a detailed analysis of the method used by the teletype to communicate with the CNC computer. The signalling logic code used by the teletype was straightforward because it used the standard ASCII character set. The electronic link however was not so simple. The serial link and the plug connections between the teletype and the CNC computer were identified by Berelowitz [2] as a standard 20 mA current loop. The IBM Asynchronous Communications adaptor installed in the IBM PC for serial communication was configured for this type of operation (see Appendix D).

However, when the specified method for connecting two 20 mA current loop devices was used, no communication was possible. After much experimentation it was found that both devices were attempting to generate current at the same time with resultant incompatibility of currents.

After this was established it was a simple matter to correct the fault. The solution was to use an MCA 255 opto-isolator built into the cable which isolated the two current loops from each other. The opto-isolator uses a small light emitting diode (LED) to turn a switch on or off. Turning the switch on allowed current to flow, and turning it off caused the current to stop. Thus each device could communicate by means of current pulses generated and received in its own current loop.

Once the hardware for the communication link between the IBM PC and the CNC computer had been established, the software to facilitate the transmission of programs was written.

3.4.3: Software Development for the Transmission of CNC Modules from the IBM PC to the CNC Computer.

The software for the data communication used the programming facilities of MIRROR, a commercially available communications package. The MIRROR program files are referred to as "script files", and they are able to use all of the features of the MIRROR program. The communication of the CNC modules from the IBM PC to the CNC computer is a repetitive process and therefore well suited to automation by the script files. The first stage in developing the software was to configure MIRROR with the required settings.

Configuration of Mirror and the Command File

The required settings to be used by MIRROR are given below.

Baud Rate	: 110 baud
Parity	: Even
No. of Data Bits	: 8
No. of Stop Bits	: 2
No. of Start Bits	: 1
Communications Port	: COM 1, or PORT 1
Duplex	: Full
Mode	: Call

The Mode setting is specific to the MIRROR program and is only important when using a MODEM. Its default value is set to Call.

Once MIRROR had been configured manually with the correct settings, and communications established between the IBM PC and the CNC, the settings were saved in the command file (see Appendix K) for later use.

The command file, which is loaded automatically when MIRROR is run, contains the settings of the all the communications parameters required to configure the system, and also allows a script file to be called automatically.

The command file used in this system was called STD.XTK and the script file (see Appendix L) was called START.XTS.

The automated sequence of events contained in the script file to set the connection between the two computers is:

1. MIRROR is loaded and run
2. MIRROR automatically loads the command file (STD.XTK) which configures the IBM PC with the required settings.
3. The command file automatically loads the script file (START.XTS) which begins the semi-automatic transfer of the CNC modules.

The script file START.XTS developed for transferring CNC modules

The structure of the script file consists of a control program with a subroutine section as shown in Figure 3.26.

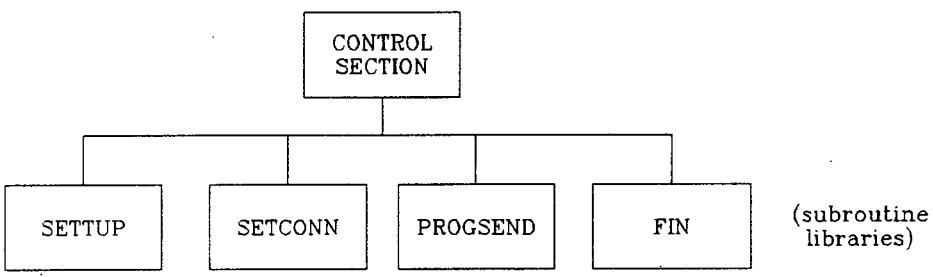


Figure 3.26: Overall structure of the START.XTS script file

The control program calls each of the subroutines in the correct sequence; it performs no other task. It was included to provide continuity between all the programs in the system.

The SETTUP subroutine asks the user whether or not he is ready to begin transferring files to the CNC computer. If he is ready the program continues. If he is not it ends.

The FIN subroutine is called when the file transferring procedure has been completed and the user wishes to disconnect the IBM PC from the CNC computer.

The most important subroutines in the script file are the SETCONN subroutine, which establishes the connection between the IBM PC and the CNC computer, and the PROGSEND subroutine, which transfers each CNC module. Figure 3.27 shows the flowchart of the SETCONN subroutine.

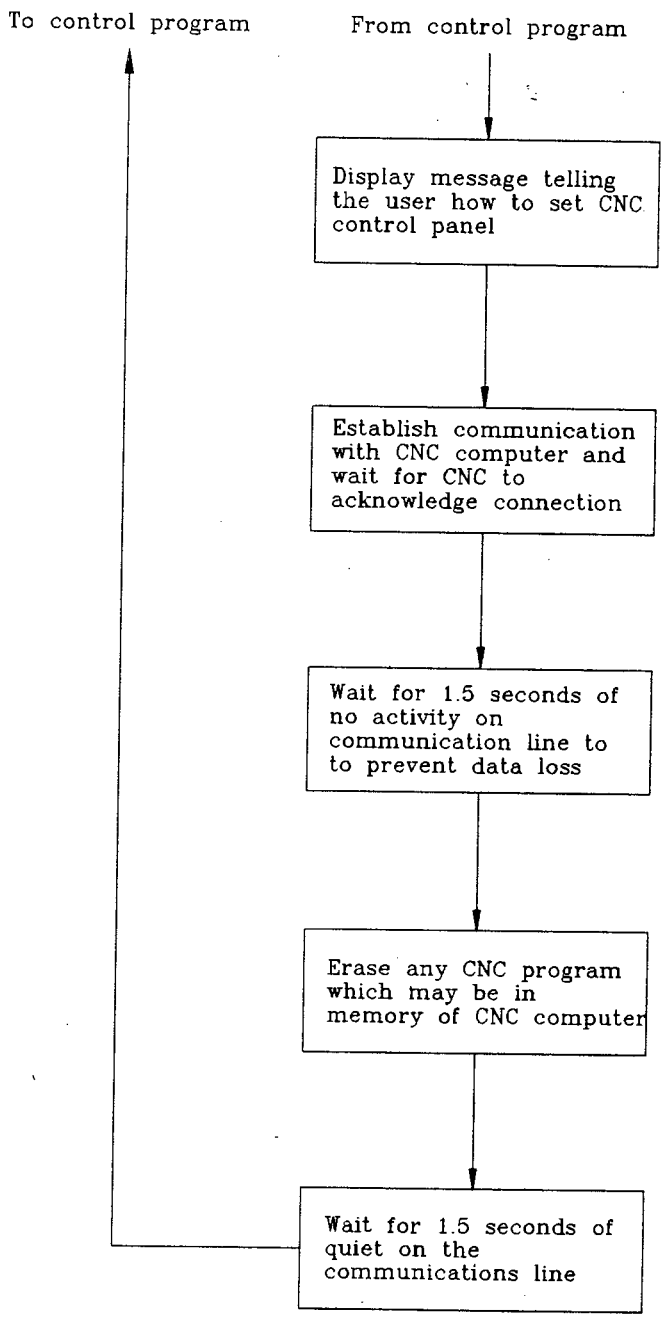


Figure 3.27: Flowchart of the SETCONN subroutine.

The subroutine begins by prompting the user to manually set the controls of the CNC machine into the correct position so that the connection between the IBM PC and the CNC computer can be established. The next step uses the statement GO LOCAL (Line 22 in the program listing in Appendix K) to actually make the connection. When the CNC computer "senses" that a connection has been made it acknowledges this by sending the string "BOSS 4.1". Therefore the program waits for this string (Line 23) to ensure that the connection has been successfully made. At this point the program waits for 1,5 seconds of no activity on the communications connection (Line 23). This statement was used in the later PROGSEND subroutine as well. Because it is crucial to ensure error free communications, it needs a slightly more detailed explanation.

Initially this statement was not included, with the result that some data was not communicated properly. The data lost did not follow any systematic pattern and only after much experimentation with the communications was it realised that, although the baud rate was correct, some characters were being transmitted by the IBM PC before the CNC computer was ready to accept them. Therefore the IBM PC was instructed to wait 1.5 seconds after each character was transmitted to the CNC machine. The one exception was during the transmission of the CNC module in the PROGSEND subroutine - it was not necessary to include any wait statement during the transmission, but a three second delay (WAIT DELAY 30) was included after the CNC module had been transmitted to allow any "slow" characters to be received by the CNC computer.

After the connection has been established the IBM PC sends a command to the CNC computer to clear any program which is currently in memory. Thereafter the IBM PC again waits 1,5 seconds and then returns to the MAIN control program.

The flowchart of the next important subroutine, PROGSEND, is shown in Figure 3.28.

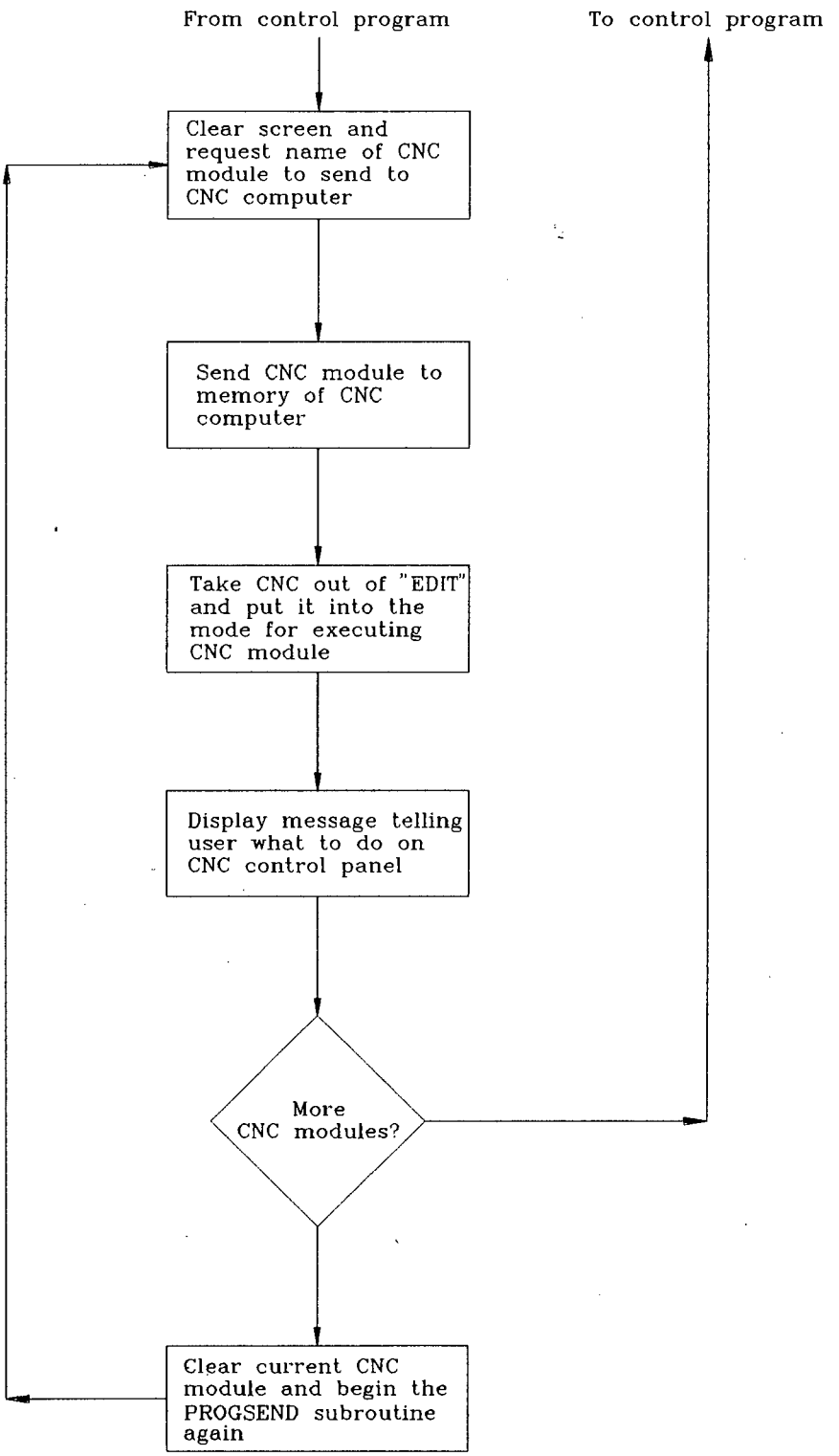


Figure 3.28: Flowchart of the PROGSEND subroutine

This subroutine begins by requesting from the user the name of the CNC module to transmit and stores the name in a variable called F10 (Line 34). The next statement (Line 35) transmits the entire CNC module specified by the user from the IBM PC to the CNC computer. After the module has been sent the IBM PC transmits the characters Control-Z to set the CNC machine into the correct mode for the execution of the CNC module. In order to help the user (Line 39) the program displays a set of instructions on the screen telling the user how to set the control panel of the CNC machine to execute the CNC module. If there are no further modules to be sent the program ends, otherwise it clears the CNC computer and begins the process of sending the next CNC module.

Once the CNC module is in the memory of the CNC computer the CNC machine is operated in the standard manner.

The script file can be used to transmit not just the CNC modules produced by the postprocessor, but any CNC program from the IBM PC to the CNC computer. This proved to be extremely popular with the normal users of the CNC machine because it allowed them to write a CNC program on their own personal computer and then, using a floppy disk, to load their program into the CNC machine. The main benefit is that the sophisticated editing facilities of a word processor can be used to edit their CNC programs rather than the simple and tedious editor on the CNC computer. Typing errors in the CNC programs, which often caused problems with the CNC machine, were also more easily avoided.

CHAPTER FOUR

4. TESTING THE ACCURACY OF THE MILLING SYSTEM

Before the milling system can be used reliably the accuracy with which a surface can be produced must be quantified. The total error of the final milled surface is the sum of four component errors:

a) the error due to the milling program
b) the error due to the milling machine
c) the error due to the measuring device
d) the error due to the mathematical technique used to calculate the error.

In order to isolate the errors in the final milled surface due to a) and b), the errors due to c) and d) must be quantified. This was done by measuring surface datapoints on a known surface and calculating the uncertainty in the measured data.

4.1: Choice of Surface and Measuring Device

The surface chosen for measurement purposes was a hemisphere on a flat plane as illustrated in Figure 3.19. This is a non undercut, doubly curved surface with a slope ranging from vertical to horizontal. The machining requirements of such a surface adequately test the capabilities of the milling system, and in addition the mathematical equations of the surface are well known.

The measuring device used was a stereo microscope in the Surveying Department at U.C.T. This precision instrument is normally used for the measurement of surface datapoints of small surfaces. The claimed accuracy of the device when

used by a skilled operator is 0.015 mm in X and Y and 0.040 in Z [34].

4.2: The Mathematical Method Used to Determine the Uncertainty in the Measured Surface Datapoints

The standard equation for a sphere with centre offset from the origin is well-known to be:

$$(x-x_0)^2 + (y-y_0)^2 + (z-z_0)^2 = R^2 \qquad (19)$$

where x_0, y_0 and z_0 are the coordinates of the centre of the sphere measured from the origin and R is the radius.

This equation can be linearised into the following form:

$$x^2+y^2+z^2 = 2x_0x+2y_0y+2z_0z - (x_0^2+y_0^2+z_0^2-R^2) \qquad (20)$$

Equation (20) can be rewritten as:

$$Y = Ax + By + Cz + D \qquad (21)$$

where
$$Y = x^2+y^2+z^2$$
$$A = 2x_0$$
$$B = 2y_0$$
$$C = 2z_0$$
$$D = -(x_0^2+y_0^2+z_0^2-R^2)$$

Now, using the measured surface datapoints, a multiple linear regression technique is used to give the values of the coefficients A, B, C and D which are then used to give the values of x_0, y_0, z_0 and R.

Using equation (19) each measured surface datapoint, together with x_0, y_0, z_0, is used to calculate a theoretical radius, Rt. The error in each surface datapoint is then:

$$\text{error} = R - R_t \qquad (22)$$

If there is no systematic error in the measuring device the errors should be random and normally distributed about a mean.

4.3: The Measuring Error and the Error in the Mathematical Method

The sum of the measuring error and the error in the mathematical technique was determined by measuring surface datapoints on a ball bearing of known radius. The diameter of the ball bearing was measured with a micrometer. The error in each datapoint measured with the stereo microscope was calculated using the technique described previously. The distribution of the error is given in Figure 4.1.

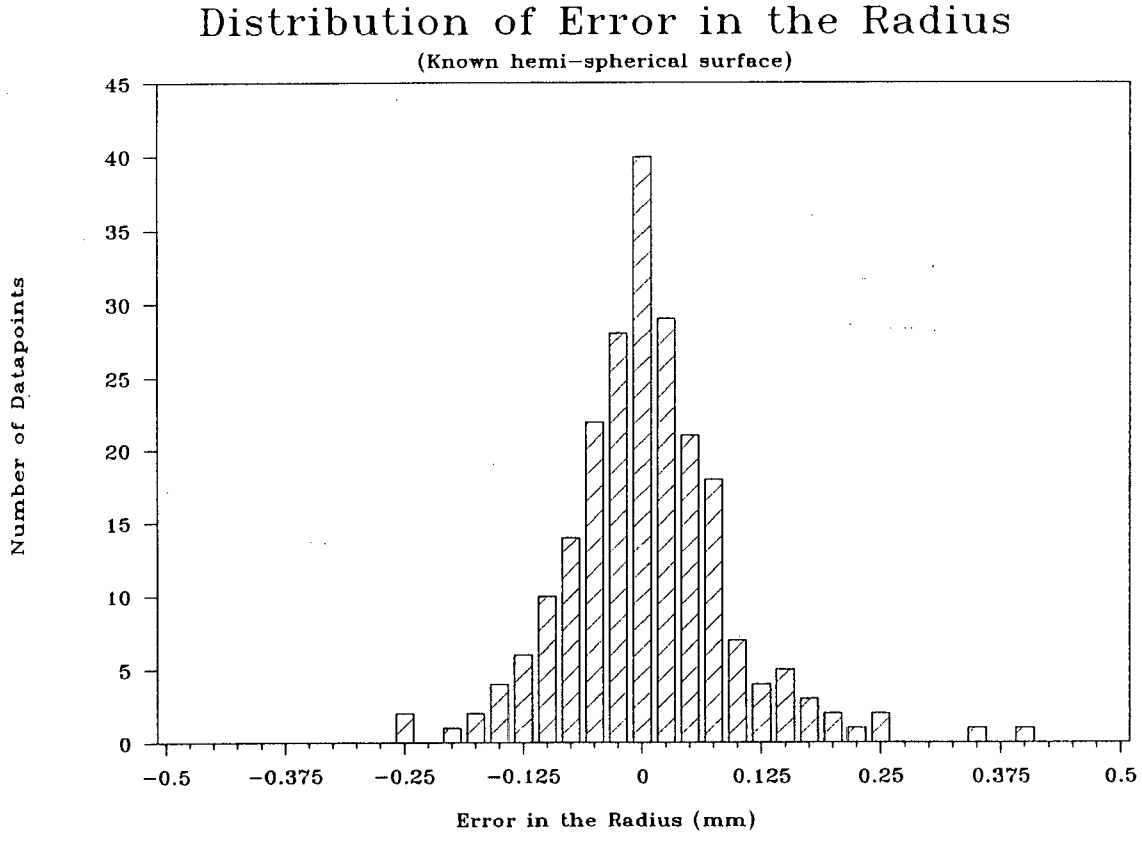


Figure 4.1: Distribution of the error of the measured datapoints of the known sphere

The frequency distribution in Figure 4.1 shows a clear bell-shaped normal distribution. It is valid, therefore, to calculate the standard deviation of the error in the radius. This is given in Table 4.1.

Radius by Multiple Linear Regression	26.990 mm
Radius by Micrometer	26.985 mm
Std Deviation of the error in the radius	0.1117 mm

Table 4.1: Radius and standard deviation of the known sphere

The calculated radius based on the measured data and the micrometer measured radius were compared and found to be within 0.005 mm. In addition, the coefficient of determination was 99.99%, indicating a good fit between the data and the calculated sphere equation. This close agreement indicated that the measurement procedure was not subject to any systematic errors. The frequency distribution curve shows that the measurement errors were random, and that the standard deviation can therefore be used as an indicator of the precision of the measurements. Based on this assumption there is a 95.5% probability that the uncertainty in the measurement of the radius of the hemisphere is:

$$\frac{+-2 \times 0.1117}{26.990} = +-0.83\%$$

4.4: Measurement and Calculation of the Machining Error

Surface datapoints for a 25 mm radius hemisphere on a flat plane were generated by a short computer program using equation (23). See Appendix M for the program listing.

$$z = \left[625 - \left[(x-50)^2 - (y-50)^2 \right] \right]^{0.5} \qquad (23)$$

for $0 > x > 100$
$0 > y > 100$

and $z = 0$ for $\left[625 - \left[(x-50)^2 - (y-50)^2 \right] \right]^{0.5} < 0$

The datapoints were then fed into the milling system and the surface milled. The milled surface was measured and the results processed by the multiple linear regression technique as previously described. The coefficient of determination of the data is 99.99%, indicating a good fit of the data to the calculated sphere equation. Figure 4.2 shows the error distribution between each measured surface datapoint and the calculated radius.

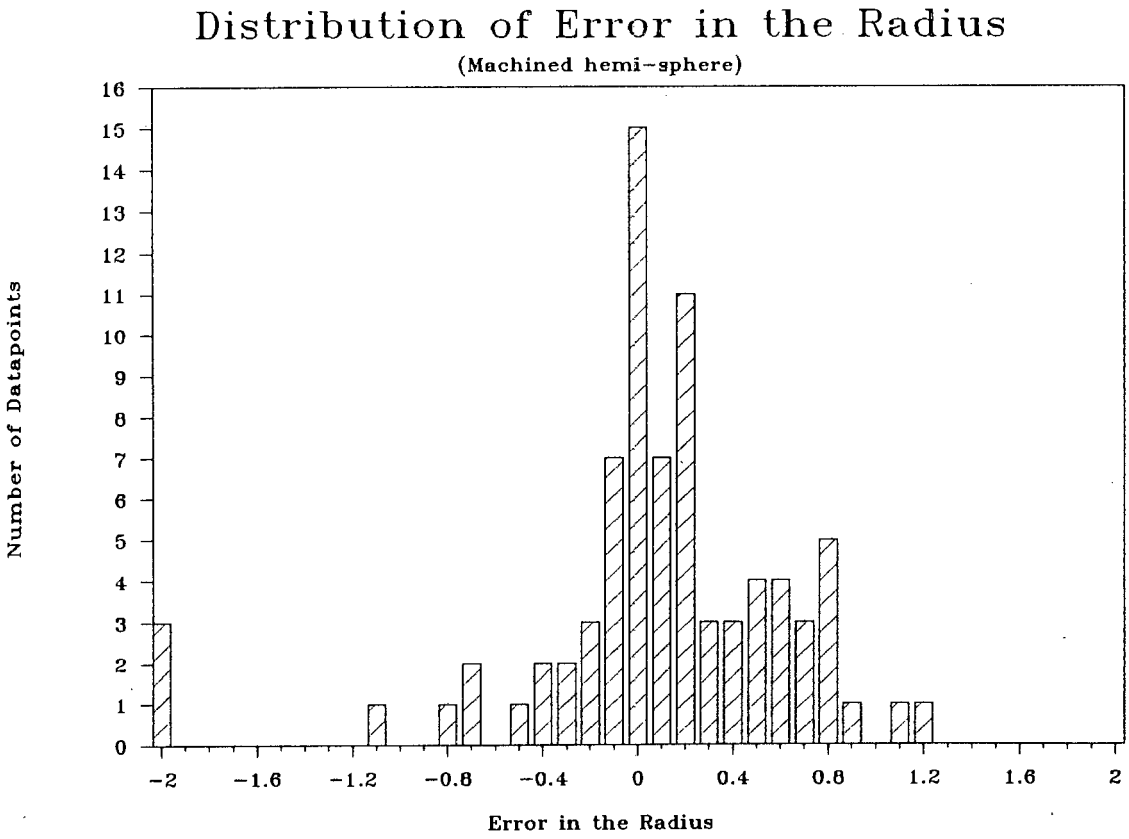


Figure 4.2 : Distribution of the error of the measured datapoints of the machined hemisphere

The shape of the frequency distribution shows two distinct peaks, one at 0 and the other at 0.5, and appears to be a superposition of two normal distributions. The shape of the frequency distribution curve was expected because of the problem of the sagittal error (see Figure 2.9) at the discontinuity between the horizontal plane and the vertical side around the base of the hemisphere. Figure 3.19 shows the planar elements straddling this discontinuity, and when the milling cutter machines points on these elements the resulting radius tends to be larger than the desired radius. The physical location on the hemisphere of the datapoints in the region of the second peak were examined and found to occur predominantly around the base, thus confirming the expectation.

Because the errors are not normally distributed, the only meaningful statistic, apart from the correlation coefficient, is the mean calculated radius of the hemisphere which was 24.54 mm. The uncertainty in the radius measurement, determined in the previous section, is +- 0.83% (or +- 0.20 mm) of the mean calculated radius. The errors due to the milling system and machining therefore give an average undercutting error of 0.46 mm +- 0.20 mm over the surface.

Possible reasons for the slight undercutting over the surface are:

a) Sagittal error - when machining a convex surface the sagittal error of the planar approximation always causes the surface to be undercut. This can, however, be minimised by taking smaller elements.

b) Inaccuracies due to machining of polystyrene - the hemispherical surface was machined in polystyrene which tends to tear away when being machined at high feedrates.

c) Inaccuracies due to difficulty in the measurement of the polystyrene surface - the texture of the surface consisted of small bubbles which were of a similar size to the pointer used in the stereo microscope. Measurements taken at the bottom of a bubble would give a smaller radius than is actually the case. A suitable alternative material for milling is PVC, which machines well and does not have the texture problem.

d) Inaccuracies due to the build-up of foam on the cutting tool - the author noticed whilst machining the surface that the waste foam tended to adhere to the cutter. This would cause the actual cutter radius to exceed the radius used in the calculation of the tool path, resulting in undercutting of the surface.

e) Inaccuracies in the milling system - although the author made every attempt to verify the milling system as it was developed it is possible that there could be an error in the calculation of the toolpath.

Although the machining requirements of the hemispherical surface are typical of those likely to be encountered by the milling system, the accuracy figure presented in this chapter cannot be generalised as being representative for all surfaces. The accuracy depends on a number of parameters such as element size, the degree of curvature (concave or convex) of the surface and the size of the cutter to name but a few.

What is required is a comprehensive accuracy analysis, using a similar technique to that described in this chapter, but for a far wider variety of mathematically defined surfaces. This was beyond the scope of this thesis but would make an excellent topic for an undergraduate project.

CHAPTER FIVE

5. CONCLUSIONS

5.1: Features of the Milling System

The milling system that has been designed has the following features:

1. The capability of mathematically approximating any single valued 3D surface on an IBM PC microcomputer for the purpose of milling.
2. A straightforward mathematical technique for the 3D surface approximation, easily understood by undergraduate engineering students from their second year onwards.
3. A real-time 3D graphical display of the approximated surface and calculated toolpath.
4. A facility to automatically convert the toolpath into suitable CNC programs for the Bridgeport CNC machine.
5. An electronic link between the IBM PC and the CNC computer to directly download the CNC program.
6. An easily understandable program structure which will simplify future modifications and facilitate its inclusion into a total system capable of measuring and replicating 3D surfaces.

5.2: Recommendations for Further Work

Although the milling system was successfully implemented, there are nevertheless areas that could be improved. These are detailed overleaf:

1. The CNC milling program was written in TRUE BASIC and structured for ease of understanding rather than for fast running. This was especially noticeable during the calculation of the toolpositions and in the generation of the 3D display. If these two sections of the program were rewritten with the emphasis on speed, the overall processing time of large surfaces would be considerably reduced.

2. The transmission of the CNC modules directly from the IBM PC to the CNC computer is a significant improvement on the previous method but is still quite tedious because the CNC computer's serial communications port can only operate at 110 baud. This means that a CNC module, which is approximately 7 800 bytes long, takes 10 minutes to download. To download the five CNC modules for a relatively small 20 x 20 element surface therefore takes 50 minutes. As the number of CNC modules rises in proportion to the number of elements used, it was found that the bulk of the machining time for large surface was spent on downloading the CNC modules rather than on the actual machining. An increase in transmission speed of approximately tenfold would be obtained if the data input facility of the paper tape reader instead of the serial port were utilised for downloading the CNC modules. The design of a system similar to that described by Dunn [30] is one way to achieve this.

3. The accuracy of the overall system was determined for a test surface (a hemisphere on a flat plane). This is, however, not necessarily representative of the accuracy for all types of surfaces. A comprehensive accuracy analysis where mathematically defined surfaces are machined using a variety of different cutter and element sizes would provide a better indication of the expected accuracy for various surface types.

The results of this analysis could be used to construct tables which could be used to aid a new user in the selection of the appropriate element and cutter size, given a maximum specified error.

4. The problem of inaccurate cutting at discontinuities and sharply concave/convex regions of a surface remains a problem. A useful modification would be for the milling system to highlight the areas where this will occur for a given cutter size, perhaps on the 3D display, and compensate in the toolpath by moving the cutting tool further away from the surface in those regions, or by choosing a smaller radius cutter.

   Another possibility is to initially select a large size cutter and then using this cutter to machine only those regions where it will not undercut. Thereafter a sequence of smaller cutters could be selected to machine more and more of the surface, each cutter machining only those regions where it will not undercut the surface. In this way the entire surface could be accurately machined. The method, however, would be fairly complex to implement but would make the entire machining process very efficient.

5. Although the mathematical algorithms can process an infinite number of elements, a maximum of 70 x 70 can be accommodated on a standard 640 K IBM PC. This is because the surface datapoints are first loaded into the memory of the computer and then used for calculation of the toolpath. A better solution would be to access blocks of surface data from disk as and when required. If this were done the only limitation on the number of elements would be the capacity of the disk drive in the machine.

5.3: Concluding Remarks

The milling system that was designed in this thesis is fully operational and has been used to accurately machine a number of different mathematically defined surfaces. The system is also capable of producing surfaces from measured data and is well suited to the manufacture of biomedical components such as prostheses for amputees and shoe inserts.

6. REFERENCES

1. DUNCAN J.P. and MAIR S.G., (1983), "Sculptured Surfaces in Engineering and Medicine", Cambridge University Press, Cambridge

2. BERELOWITZ M., (1986), "The Use of CAD/CAM in the Management of Orthopaedic Patients", BSc Mechanical Engineering Thesis

3. VAUGHAN C., BROOKING G., PRICE M. and IRELAND J., (1986), "The MicronEye Toposcanner: A Versatile Anthropometric Instrument", Cape Town Paper presented to 2nd Annual Congress, Ergonomics Society of Southern Africa

4. CLARKE A., (1982), "APT on a Micro", Chartered Mechanical Engineer, Vol 29, No.10, 10 October 1982, pp 42-47

5. VERGEEST J.S., and BROEK J.J., (1987), "Body Surface Measurement and Replication by Photogrammetry and Computer Aided Design", Engineering in Medicine, Vol 16 No.1, pp 3-8

6. SIPSURF product bulletin (1987), Delft Spline Systems, received as a result of correspondence

7. DUNCAN J.P., and LAW K.K., (1987), "Computer Aided Sculpture", unpublished work sent to the author as a result of correspondence with Prof. J.P. Duncan

8. KELLOCK B., (1987), "3-D Digitising on CNC Machines", Mach. Prod. Eng., Vol 145, No.3712, p 35

9. MASTERCAM product bulletin (1987), CNC Software Inc.

10. PLUSCAM product bulletin (1987), CNC Software Inc.

11. HASLER M.F., and DESJONQUERES R., (1987), "A Mould and Die Manufacturer's Experience of CIM", Industrial and Production Engineering, Vol 11, No.2, pp 16-22

12. DECHELETEE M.R., (1986), "A Quick Byte: CAD/CAM speeds up the Dentists Surgery", Engineering in Medicine, Vol 15, No.4, pp203

13. BEZIER P., (1971), "Example of an Existing System in the Motor Industry: The UNISURF system", Proceedings of the Royal Society London, A 321, pp207-218

14. FAUX I.D. and PRATT M.J., (1979), "Computational Geometry for Design and Manufacture", Ellis Horwood

15. BEZIER P., (1973), "UNISURF Principles, Programme, Language" Computer Languages for Numerical Control, J. Hatvany (ed), North Holland (Proceedings of the PROLAMAT 73 conference, Budapest (Hungary)),pp 417-426

16. BELL C., LANDI B. and SABIN M.A., (1973), "The Programming and use of Numerical Control to Machine Sculptured Surfaces" Proceedings of the 14th International Machine Design and Research Conference, Manchester (UK), pp233-238

17. FLUTTER A.G. and ROLPH R.N., (1976), "POLYSURF: an interactive system for the computer aided design and manufacture of components", Proceedings of the CAD 76 Conference, CAD Centre, Madingley Road, Cambridge, pp150-158

18. HARTLEY P.J. and JUDD C.J., (1980) "Curve and Surface Representations for Bezier B-spline systems," Proceedings of the CAD 80 Conference, Brighton, March 1980, pp226-234

19. FERGUSON J.C. (1963), "Multivariable Curve Interpolation, Report No. D2-22504, The Boeing Company, Seattle, Washington

20. COONS S.A. (1967), "Surfaces for Computer Aided Design of Space Forms" Report MAC-TR-41, Project MAC, MIT

21. CURRY H.B. and SCHOENBERG I.J., (1966), "On Polya Frequency Functions IV: The Fundamental Spline Functions and their Limits", J. Anal. Math., Vol 17, pp 71-107

22. SCHWEIKERT D.G., (1966), "An Interpolation curve using a Spline in Tension" J. Math. Phys., Vol 45, pp 312-317

23. MEHKUM E. (1969), "Curve and Surface Fitting based on a Variational Criterion", Central Institute of Industrial Research, Oslo.

24. WOODWARK J., (1986), "Computing Shape", Butterworths, London

25. FOLEY J.D. and VAN DAM A., (1983) "Fundamentals of Computer Graphics", Addison Wesley

26. NEWMAN W.M. and SPROULL R.F., (1979) "Principles of Interactive Computer Graphics." 2nd Edition, McGraw Hill

27. BARSKY B.A. and GREENBERG D.P., (1980), "Determining a Set of B-spline Control Vertices to Generate an Interpolating Surface", Computer Graphics and Image Processing Vol 14, pp 203-226

28. LORD M., (1987) "Curve and Surface Representation by Iterative B-spline Fit to a Data Point Set", Engineering in Medicine, Vol 16, No.1, pp 29-35

29. FLUTTER A.G., (1973), "The POLYSURF System", Computer Languages for Numerical Control, J. Hatvany (ed), North Holland (Proceedings of the PROLAMAT 73 Conference Budapest (Hungary)), pp 403-415

30. DUNN J., (1987), "Black Box System does away with Paper Tape Instructions." The Engineer, Vol 265, No.6856/7, p40

31. LUZADDER W., (1962), "Basic Graphics", Prentice-Hall Inc. Englewood Cliffs, New Jersey

32. HOELSCHER R.P., SPRINGER C.H. and DOBROVOLNY J.S., (1968), "Graphics for Engineers, Visualization, Communication, and Design", John Wiley and Sons, New York, p415

33. MORTENSEN M.E., (1986), "Geometric Modelling", John Wiley and Sons

34. RUTHER H., (1988), Associate Professor,Department of Land Surveying, U.C.T., Personal Interview

35. SEYER, M.D., (1982), "RS 232 Made Easy", Prentice Hall, Englewood Cliffs

36. Technical Manual (1974), "33 Teletypewrite Set", Teletype Corporation, Skokie, Illinois

37. IBM Personal Computer Hardware Reference Library: Technical Reference Manual, (1983), 1st edition, IBM Corporation

APPENDIX A

MATHEMATICAL FORMULATION USED IN THE MILLING SYSTEM

The surface to be milled is represented by a series of points lying on that surface. These points each consist of an X,Y and Z coordinate and are measured from some fixed origin. In this formulation and in the subsequent machining it is easiest to have all the points lying in the positive X, positive Y and positive Z region as shown in Figure A.1. Figure A.1 also shows each set of four data points joined to form quadrilateral elements.

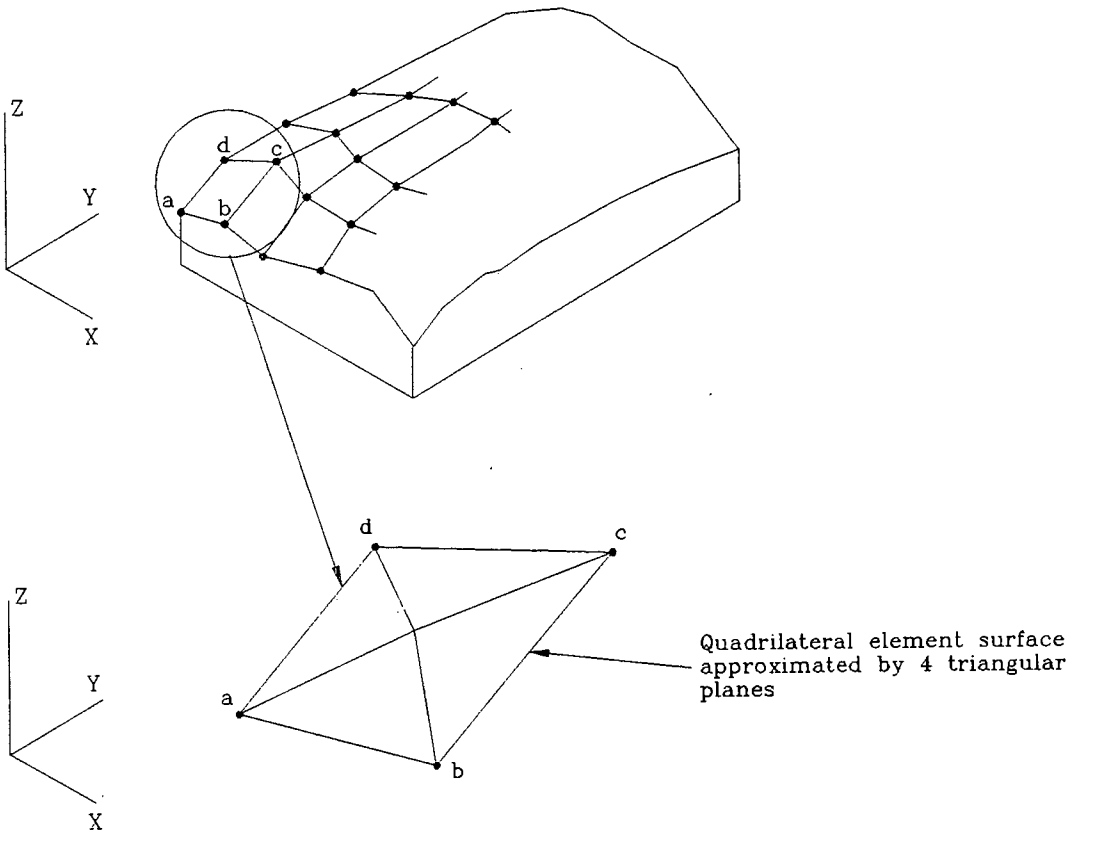


Figure A.1: Surface represented by datapoints joined to form quadrilateral elements

The mathematical approximation in this section is confined to one quadrilateral element, but the identical formulation is used for each element in the surface. Figure A.2 shows diagrammatically one quadrilateral element with the corners labelled from a to d and two diagonals constructed, l_1 which joins point a to point c, and l_2 which joins point b to point d.

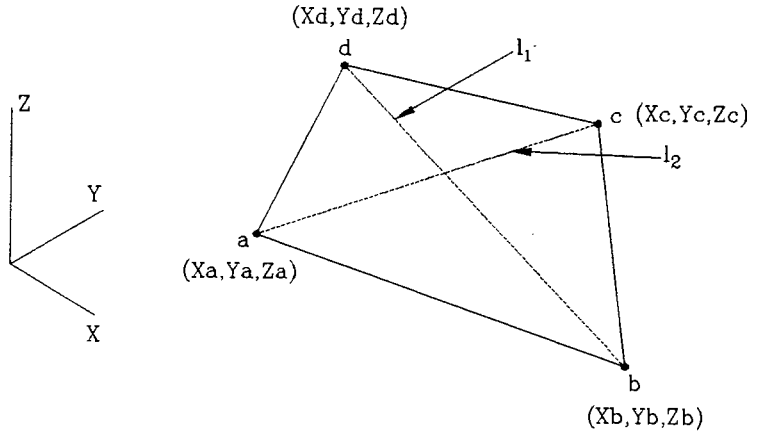


Figure A.2: Diagrammatic representation of a quadrilateral element showing the diagonals and the labelling of the corner points.

The diagonals can be expressed in vector notation as:

$$l_1 : (x_a y_a z_a) + \tau(x_c - x_a \; ; \; y_c - y_a \; ; \; z_c - z_a) \qquad (1)$$

$$l_2 : (x_b y_b z_b) + \mu(x_d - x_b \; ; \; y_d - y_b \; ; \; z_d - z_b) \qquad (2)$$

where τ and μ are the scalar multipliers. In order to find the two closest points, one on each diagonal, the following procedure, developed by the author, is used.

Let the vector equation of the line joining any two points, one on each diagonal, be:

$$v = (x_b - x_a \; ; \; y_b - y_a ; \; z_b - z_a) + \mu(x_d - x_b \; ; \; y_d - y_b \; ; \; z_d - z_b) - \tau(x_c - x_a \; ; \; y_c - y_a \; ; \; z_c - z_a) \qquad (3)$$

The magnitude (in this case the length) of the line is:

$$\|v\|^2 = \left[(x_b-x_a) + \mu(x_d-x_b) - \tau(x_c-x_a)\right]^2 + \left[(y_b-y_a) + \mu(y_d-y_b) - \tau(y_c-y_a)\right]^2 + \left[(z_b-z_a) + \mu(z_d-z_b) - \tau(z_c-z_a)\right]^2 \quad (4)$$

Now to find τ and μ for the shortest distance use:

$$\frac{\delta\|v\|^2}{\delta\mu} = 0 \quad (5) \qquad \frac{\delta\|v\|^2}{\delta\tau} = 0 \quad (6)$$

Now:

$$\frac{\delta\|v\|^2}{\delta\mu} = 2\left[(x_b-x_a) + \mu(x_d-x_b) - \tau(x_c-x_a)\right](x_d-x_b) + 2\left[(y_b-y_a) + \mu(y_d-y_b) - \tau(y_c-y_a)\right](y_d-y_b) + 2\left[(z_b-z_a) + \mu(z_d-z_b) - \tau(z_c-z_a)\right](z_d-z_b) = 0 \quad (7)$$

$$\frac{\delta\|v\|^2}{\delta\tau} = 2\left[(x_b-x_a) + \mu(x_d-x_b) - \tau(x_c-x_a)\right](x_a-x_c) + 2\left[(y_b-y_a) + \mu(y_d-y_b) - \tau(y_c-y_a)\right](y_a-y_c) + 2\left[(z_b-z_a) + \mu(z_d-z_b) - \tau(z_c-z_a)\right](z_a-z_c) = 0 \quad (8)$$

Multiplying (7) out gives:

$$\begin{aligned}
&(x_b-x_a)(x_d-x_b) + \mu(x_d-x_b)^2 - \tau(x_c-x_a)(x_d-x_b) \\
&+ (x_b-x_a)(x_d-x_b) + \mu(x_d-x_b)^2 - \tau(x_c-x_a)(x_d-x_b) \\
&+ (x_b-x_a)(x_d-x_b) + \mu(x_d-x_b)^2 - \tau(x_c-x_a)(x_d-x_b) \\
&= 0 \qquad (9)
\end{aligned}$$

Rearranging (9):

$$\begin{aligned}
&\left[(x_d-x_b)^2 + (y_d-y_b)^2 + (z_d-z_b)^2\right]\mu \\
&- \left[(x_c-x_a)(x_d-x_b) + (y_c-y_a)(y_d-y_b) + (z_c-z_a)(z_d-z_b)\right]\tau \\
&= (x_b-x_a)(x_d-x_b) + (y_b-y_a)(y_d-y_b) + (z_b-z_a)(z_d-z_b) \qquad (10)
\end{aligned}$$

Similarly multiplying (8) gives:

$$\begin{aligned}
&\left[(x_d-x_b)(x_a-x_c) + (y_d-y_b)(y_a-y_c) + (z_d-z_b)(z_c-z_a)\right]\mu \\
&+ \left[(x_c-x_a)^2 + (y_c-y_a)^2 + (z_c-z_a)^2\right]\tau \\
&= - \left[(x_b-x_a)(x_a-x_c) + (y_b-y_a)(y_a-y_c) + (z_b-z_a)(z_a-z_c)\right] \qquad (11)
\end{aligned}$$

There are now two equations, (10) and (11), in two unknowns, τ and μ, which can be solved using matrix methods. Equations (10) and (11) can be represented in matrix form as:

$$\begin{bmatrix} (x_d-x_b)^2 & -(x_c-x_a)(x_d-x_b) \\ +(y_d-y_b)^2 & -(y_c-y_a)(y_d-y_b) \\ +(z_d-z_b)^2 & -(z_c-z_a)(z_d-z_b) \\ \\ (x_d-x_b)(x_a-x_c) & (x_c-x_a)^2 \\ +(y_d-y_b)(y_a-y_c) & +(y_c-y_a)^2 \\ +(z_d-z_b)(z_c-z_a) & +(z_c-z_a)^2 \end{bmatrix} \begin{bmatrix} \mu \\ \\ \\ \\ \tau \\ \\ \end{bmatrix} =$$

$$\begin{bmatrix} (x_b-x_a)(x_d-x_b) \\ +(y_b-y_a)(y_d-y_b) \\ +(z_b-z_a)(z_d-z_b) \\ \\ -(x_b-x_a)(x_a-x_c) \\ -(y_b-y_a)(y_a-y_c) \\ -(z_b-z_a)(z_a-z_c) \end{bmatrix} \qquad (12)$$

Or :

$$\mathbf{A}\,\underline{x} = \underline{b} \qquad (13)$$

TRUE BASIC has the ability to invert matrices directly and therefore τ and μ can be found directly using:

$$\underline{x} = \mathbf{A}^{-1}\,\underline{b} \qquad (14)$$

τ and μ can then be substituted back into equations (1) and (2) to find the closest points on each line. The closest points are x_{11}, y_{11}, z_{11} on diagonal l_1 and x_{12}, y_{12}, z_{12} on diagonal l_2.

The fifthpoint is then halfway between the two points and is determined by:

$$x_e = \frac{(x_{11}+x_{12})}{2} \quad (15) \qquad y_e = \frac{(y_{11}+y_{12})}{2} \quad (16) \qquad z_e = \frac{(z_{11}+z_{12})}{2} \quad (17)$$

After the coordinates of the fifthpoint are known, the quadrilateral element can be approximated by its four constituent triangular planes as shown in Figure A.3.

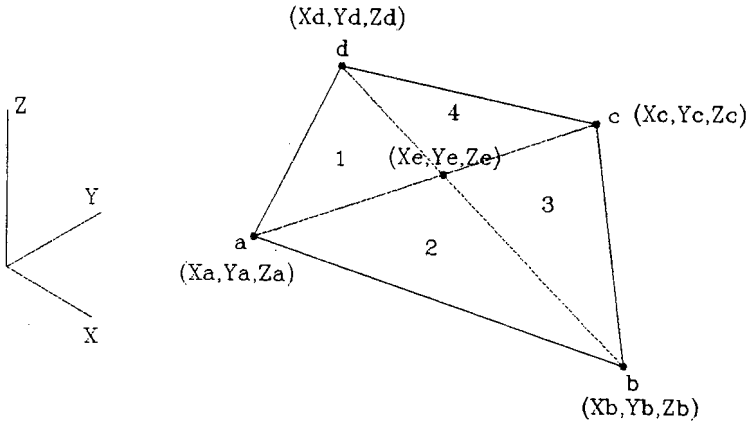


Figure A.3: Quadrilateral element approximated by four triangular planes

For the remainder of the formulation we consider only one triangular plane, Plane 1, although the identical procedure is applied to each of the four planes within the element.

For each triangular plane the position of the centroid is determined from the three surface datapoints (the fifthpoint and the two corner points) defining the plane.

$$X_{centroid} = \frac{(x_a + x_e + x_d)}{3} \tag{18}$$

$$Y_{centroid} = \frac{(y_a + y_e + y_d)}{3} \tag{19}$$

$$Z_{centroid} = \frac{(z_a + z_e + z_d)}{3} \tag{20}$$

The plane equation used for each triangular plane is:

$$Ax + By + Cz + D = 0 \tag{21}$$

Given three points on the plane, in this case the fifthpoint and two corner points, the coefficients A,B,C and D can be solved using the determinant method:

$$\begin{vmatrix} x & y & z & 1 \\ x_a & y_a & z_a & 1 \\ x_e & y_e & z_e & 1 \\ x_d & y_d & z_d & 1 \end{vmatrix} = 0 \tag{22}$$

From which:

$$A = \begin{vmatrix} y_a & z_a & 1 \\ y_e & z_e & 1 \\ y_d & z_d & 1 \end{vmatrix} \tag{23}$$

$$B = -\begin{vmatrix} x_a & z_a & 1 \\ x_e & z_e & 1 \\ x_d & z_d & 1 \end{vmatrix} \tag{24}$$

$$C = \begin{vmatrix} x_a & y_a & 1 \\ x_e & y_e & 1 \\ x_d & y_d & 1 \end{vmatrix} \tag{25}$$

$$D = -\begin{vmatrix} x_a & y_a & z_a \\ x_e & y_e & z_e \\ x_d & y_d & z_d \end{vmatrix} \tag{26}$$

The coefficients A,B,C and D of (21) can be used to give the direction cosines ($\alpha,\beta,\varnothing$) of the normal vector to the triangular plane using (27),(28) and (29).

$$\alpha = \frac{A}{\sqrt{(A^2+B^2+C^2)}} \qquad (27)$$

$$\beta = \frac{B}{\sqrt{(A^2+B^2+C^2)}} \qquad (28)$$

$$\varnothing = \frac{C}{\sqrt{(A^2+B^2+C^2)}} \qquad (29)$$

The normal vector is positioned at the centroid of the triangular plane as shown in Figure A.4.

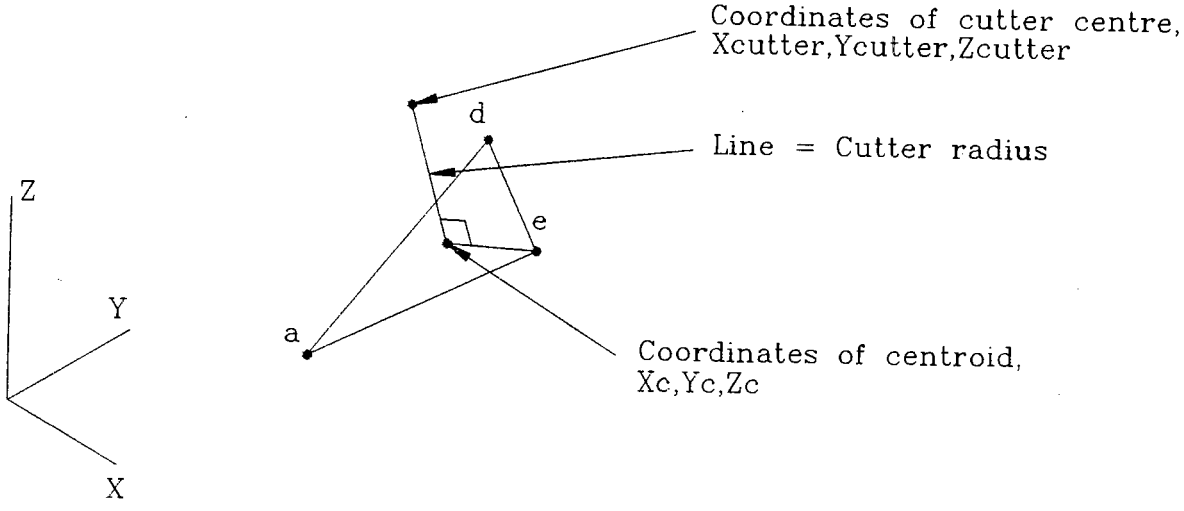


Figure A.4: The location of the normal vector and the cutter centre on the triangular plane.

The centre of the hemispherical milling cutter is located at the end of the normal vector through the centroid with magnitude equal to the cutter radius.

The coordinates of the centre of the cutter can be obtained using the following equations:

$$X_{cutter} = X_{centroid} + CutterRadius \times \alpha \quad (30)$$

$$Y_{cutter} = Y_{centroid} + CutterRadius \times \beta \quad (31)$$

$$Z_{cutter} = Z_{centroid} + CutterRadius \times \varnothing \quad (32)$$

This procedure is carried out for each of the four triangular planes in turn. The computer then selects the next quadrilateral element and the process is repeated. The result of this processing is a list of toolpositions necessary to machine the surface. Each toolposition has a "key" associated with it. The keys are used to arrange the toolpositions in the correct sequence for milling, and are described in detail towards the end of Section 3.1.1 of the main text. A numerical example using the derived key equations is given in Appendix B. The equations to calculate the keys are:

TL = Total Number of Levels in Surface
F = Reference Frame of Quadrilateral Element
L = Reference Level of Quadrilateral Element
I-IV = Number of the Triangular plane, refer to Figure 3.12 in the main text

<u>For the Odd Elements</u>:

$$Io: Key = [(TL-1) \times (F-1) \times 4] + L \quad (33)$$

$$IIo: Key = [(TL-1) \times (F-1) \times 4] + (TL-1) + 2(TL-1) - 2(L-1) \quad (34)$$

$$IIIo: Key = [(TL-1) \times (F-1) \times 4] + L + 3(TL-1) \quad (35)$$

$$IVo: Key = [(TL-1) \times (F-1) \times 4] + (TL-1) + 2(TL-1) - 2(L-1) - 1 \quad (36)$$

For the Even Elements:

$$Ie: Key = [(TL-1) \times (F-1) \times 4] + (TL-1) - (L-1) \quad (37)$$

$$IIe: Key = [(TL-1) \times (F-1) \times 4] + TL + 2(L-1 \quad (38)$$

$$IIIe: Key = [(TL-1) \times (F-1) \times 4] - (L-1) + 3(TL-1) \quad (39)$$

$$IVe: Key = [(TL-1) \times (F-1) \times 4] + L + 2(TL-1) + 1 \quad (40)$$

The determination of the keys completes the calculation of the toolpositions. In order to make the mathematical procedure as clear as possible a complete numerical example of a simple 2 x 2 element surface is given.

NUMERICAL EXAMPLE:

THE CALCULATION OF TOOLPOSITIONS AND THE TOOLPATH FOR A 2 x 2 ELEMENT SURFACE

The sample surface with the Frames and Levels numbered is shown diagrammatically in Figure A.5.

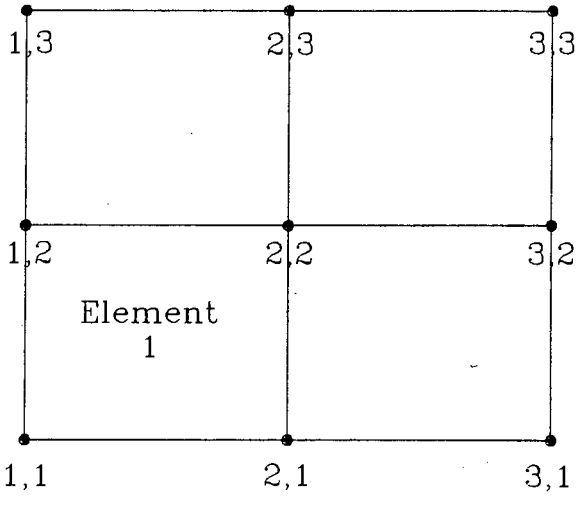


Figure A.5: 2 x 2 element surface shown diagrammatically

The nine surface datapoints are numbered according to their Frame and Level and are listed below in Table A.1. A hemispherical cutter of radius 5mm is to be used to mill the surface.

Frame	Level	X	Y	Z
1	1	0	0	0
1	2	0	5	2
1	3	1	9	3
2	1	5	0	0
2	2	3	4	1
2	3	5	10	2
3	1	10	0	0
3	2	10	5	1
3	3	10	10	2

Table A.1: X,Y and Z coordinates of the 9 surface points of sample surface

Element 1 is extracted and is shown diagrammatically in Figure A.6 below.

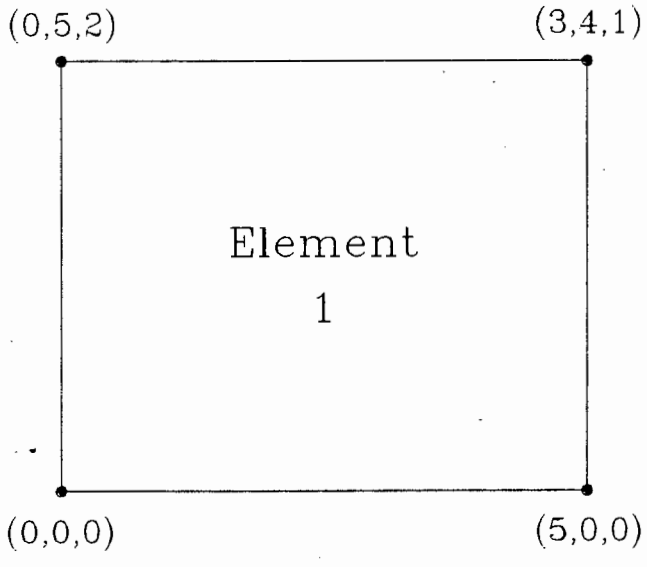


Figure A.6: Element 1 extracted from the surface and showing the X,Y,Z corner coordinates

Substituting the coordinates into equations (1) and (2), the vector equations of the diagonals of this quadrilateral element are obtained:

$$l_1 : (0,0,0) + \tau(3 - 0 \; ; \; 4 - 0 \; ; \; 1 - 0) \qquad (41)$$

$$l_2 : (5,0,0) + \mu(0 - 5 \; ; \; 5 - 0 \; ; \; 2 - 0) \qquad (42)$$

which gives

$$l_1 : (0,0,0) + \tau(3;4;1)$$

$$l_2 : (5,0,0) + \mu(-5;5;2)$$

Substituting into the matrix equation (12) gives:

$$\begin{bmatrix} (0-5)^2 & -(3-0)(0-5) \\ +(5-0)^2 & -(4-0)(5-0) \\ +(2-0)^2 & -(1-0)(2-0) \\ (0-5)(0-3) & (3-0)^2 \\ +(5-0)(0-4) & +(4-0)^2 \\ +(2-0)(0-1) & +(1-0)^2 \end{bmatrix} \begin{bmatrix} \mu \\ \tau \end{bmatrix} = \begin{bmatrix} (5-0)(0-5) \\ +(0-0)(5-0) \\ +(0-0)(2-0) \\ -(5-0)(0-3) \\ -(0-0)(0-4) \\ -(0-0)(0-1) \end{bmatrix}$$

which simplifies to:

$$\begin{bmatrix} 54 & -7 \\ -7 & 26 \end{bmatrix} \begin{bmatrix} \mu \\ \tau \end{bmatrix} = \begin{bmatrix} -25 \\ 15 \end{bmatrix}$$

solving the above system of equations gives μ and τ.

$$\tau = 0.557$$

$$\mu = 0.726$$

Substituting these values back into equations (41) and (42) gives the closest two points; one on each diagonal.

l_1 : (0,0,0) + 0.726 (3;4;1)

l_2 : (5,0,0) + 0.557 (-5;5;2)

Point	X	Y	Z
l_1	2.178	2.904	0.726
l_2	2.215	2.785	1.114

The position of the fifthpoint within the quadrilateral element can be found using equations (15),(16) and (17):

$$x_e = \frac{(2.178 + 2.215)}{2} = 2.196$$

$$y_e = \frac{(2.904 + 2.785)}{2} = 2.844$$

$$z_e = \frac{(0.726 + 1.114)}{2} = 0.920$$

Using the fifthpoint, the quadrilateral can be divided into its four constituent triangular planes. Calculations for two triangular planes are given, but the identical procedure is used for every triangular plane of every quadrilateral element.

Triangular plane I from element 1 is shown below.

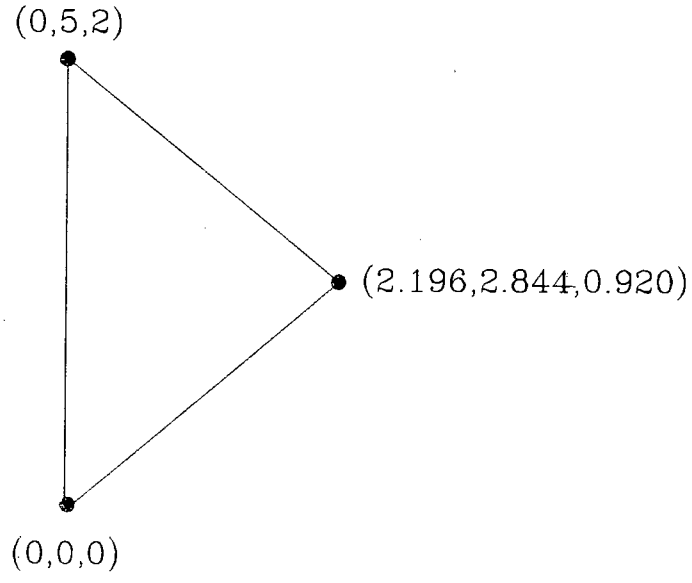


Figure A.7: Triangular plane from the element in Figure A.6 with corner coordinates shown

The centroid of the triangular plane can be found using equations (18), (19) and (20).

$$X_{centroid} = \frac{(0 + 2.196 + 0)}{3} = 0.732$$

$$Y_{centroid} = \frac{(0 + 2.844 + 5)}{3} = 2.614$$

$$Z_{centroid} = \frac{(0 + 0.920 + 2)}{3} = 0.973$$

Substituting the three corner points of the triangular plane into equation (22) and solving the resultant determinant gives the coefficients of the plane equation (21).

This gives:

$$\begin{vmatrix} x & y & z & 1 \\ 0 & 0 & 0 & 1 \\ 2.196 & 2.284 & 0.920 & 1 \\ 0 & 5 & 2 & 1 \end{vmatrix} = 0$$

Solving the determinants in the standard manner gives:

$$A = \begin{vmatrix} 0 & 0 & 1 \\ 2.844 & 0.920 & 1 \\ 5 & 2 & 1 \end{vmatrix} = 1.088$$

$$B = -\begin{vmatrix} 0 & 0 & 1 \\ 2.196 & 0.920 & 1 \\ 0 & 2 & 1 \end{vmatrix} = -4.392$$

$$C = \begin{vmatrix} 0 & 0 & 1 \\ 2.196 & 2.844 & 1 \\ 0 & 5 & 1 \end{vmatrix} = 10.980$$

$$D = -\begin{vmatrix} 0 & 0 & 0 \\ 2.196 & 2.844 & 0.920 \\ 0 & 5 & 2 \end{vmatrix} = 0.000$$

Substituting the coefficients A,B,C and D into equations (27),(28) and (29) gives the direction cosines of the normal vector to the triangular plane.

$$\alpha = \frac{1.088}{\sqrt{(1.088^2 + -4.392^2 + 10.980^2)}} = 0.091$$

$$\beta = \frac{-4.392}{\sqrt{(1.088^2 + -4.392^2 + 10.980^2)}} = -0.369$$

$$\varnothing = \frac{10.980}{\sqrt{(1.088^2 + -4.392^2 + 10.980^2)}} = 0.924$$

The coordinates of the centre of the hemispherical cutter are determined from equation (30),(31),(32):

$$X_{cutter} = 0.732 + 5 \times 0.091 = 1.187$$

$$Y_{cutter} = 2.614 + 5 \times -0.369 = 0.769$$

$$Z_{cutter} = 0.973 + 5 \times 0.924 = 5.593$$

The frame of the lower left corner of this element (the reference frame) is 1, which classifies this element as odd. The triangular plane is in position I (refer to Figure 3.12 in the main text) within the quadrilateral element, and therefore equation (33) is used to calculate the key.

TL = 3
F = 1
L = 1

Key = [(3-1) x (1-1) x 4] + 1
= 1

The next triangular plane, plane II from element 1, is shown below.

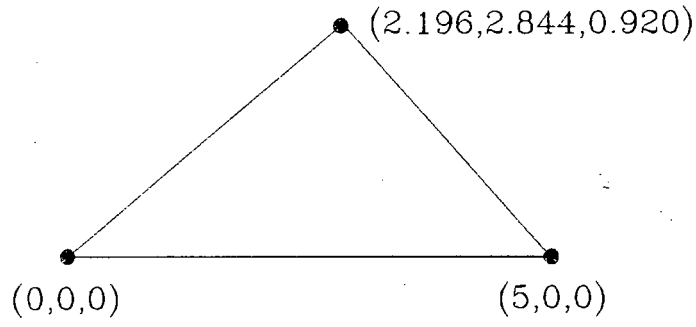


Figure A.8: Second triangular plane II from the element 1 in Figure A.6 with corner coordinates shown

The fifthpoint is already known and therefore the centroid for the next triangular plane can be calculated from:

$$X_{centroid} = \frac{(0 + 5 + 2.196)}{3} = 2.398$$

$$Y_{centroid} = \frac{(0 + 0 + 2.844)}{3} = 0.948$$

$$Z_{centroid} = \frac{(0 + 0 + 0.920)}{3} = 0.306$$

The coefficients of the plane equation are found from:

$$\begin{vmatrix} x & y & z & 1 \\ 0 & 0 & 0 & 1 \\ 5 & 0 & 0 & 1 \\ 2.196 & 2.844 & 0.920 & 1 \end{vmatrix} = 0$$

giving:

$$A = \begin{vmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 2.844 & 0.920 & 1 \end{vmatrix} = 0.000$$

$$B = -\begin{vmatrix} 0 & 0 & 1 \\ 5 & 0 & 1 \\ 2.196 & 0.920 & 1 \end{vmatrix} = -4.600$$

$$C = \begin{vmatrix} 0 & 0 & 1 \\ 5 & 0 & 1 \\ 2.196 & 2.844 & 1 \end{vmatrix} = 14.220$$

$$D = -\begin{vmatrix} 0 & 0 & 0 \\ 5 & 0 & 0 \\ 2.196 & 2.844 & 0.920 \end{vmatrix} = 0.000$$

The direction cosines are:

$$\alpha = \frac{0.000}{\sqrt{(0^2 + -4.600^2 + 14.220^2)}} = 0.000$$

$$\beta = \frac{-4.600}{\sqrt{(0^2 + -4.600^2 + 14.220^2)}} = -0.307$$

$$\varnothing = \frac{14.220}{\sqrt{(0^2 + -4.600^2 + 14.220^2)}} = 0.951$$

The coordinates of the centre of the cutter are:

$$X_{cutter} = 2.398 + 5 \times 0.000 = 2.398$$

$$Y_{cutter} = 0.948 + 5 \times -0.307 = -0.587$$

$$Z_{cutter} = 0.306 + 5 \times 0.951 = 5.061$$

The Frame of the lower left corner of the element is 1, classifying it as odd.

The triangular plane is in position II in the quadrilateral element (refer to Figure 3.12 in the main text) and therefore equation (34) is used to calculate the key.

TL = 3
F = 1
L = 1

$$\text{Key} = [(3-1) \times (1-1) \times 4] + (3-1) + 2(3-1) - 2(1-1)$$
$$= 6$$

The same procedure is used for the remaining two triangular planes within this quadrilateral element and also for the triangular planes within the rest of the quadrilateral elements of the surface. The results of the calculations are tabulated below in Table A.2.

Element Frame	Level	Key	X_{cutter}	Y_{cutter}	Z_{cutter}
1	1	1	1.191	0.766	5.596
		6	2.398	-0.587	5.064
		7	4.137	1.433	5.512
		5	2.921	2.797	6.024
1	2	2	2.246	5.468	6.953
		3	2.748	3.967	6.379
		4	4.610	5.722	6.352
		8	4.107	7.225	6.922
2	1	10	5.024	0.765	5.329
		11	7.276	-0.490	5.014
		16	9.013	1.292	5.372
		12	6.761	2.548	5.687
2	2	9	5.161	6.389	6.453
		13	6.898	4.643	6.105
		15	9.170	6.583	6.428
		14	7.433	8.329	6.777

Table A.2: List of toolpositions for the sample surface

The next phase is to sort these toolpositions into ascending order of keys. Table A.3 shows the final list of toolpositions sorted in the correct order.

Milling Sequence	X_{cutter}	Y_{cutter}	Z_{cutter}
1	1.191	0.766	5.596
2	2.246	5.468	6.953
3	2.748	3.967	6.379
4	4.610	5.722	6.352
5	2.921	2.797	6.024
6	2.398	-0.589	5.064
7	4.137	1.433	5.512
8	4.107	7.225	6.922
9	5.161	6.389	6.453
10	5.024	0.765	5.329
11	7.276	-0.490	5.014
12	6.761	2.548	5.687
13	6.898	4.643	6.105
14	7.433	8.329	6.777
15	9.170	6.583	6.428
16	9.013	1.292	5.372

Table A.3: Sequential list of toolpositions

The toolpositions are stored in this format on the final toolfile and are converted by the postprocessor into a syntactically correct program for the Bridgeport CNC machine. Figure A.9 shows the sample surface and the resultant toolpath as produced by the milling program.

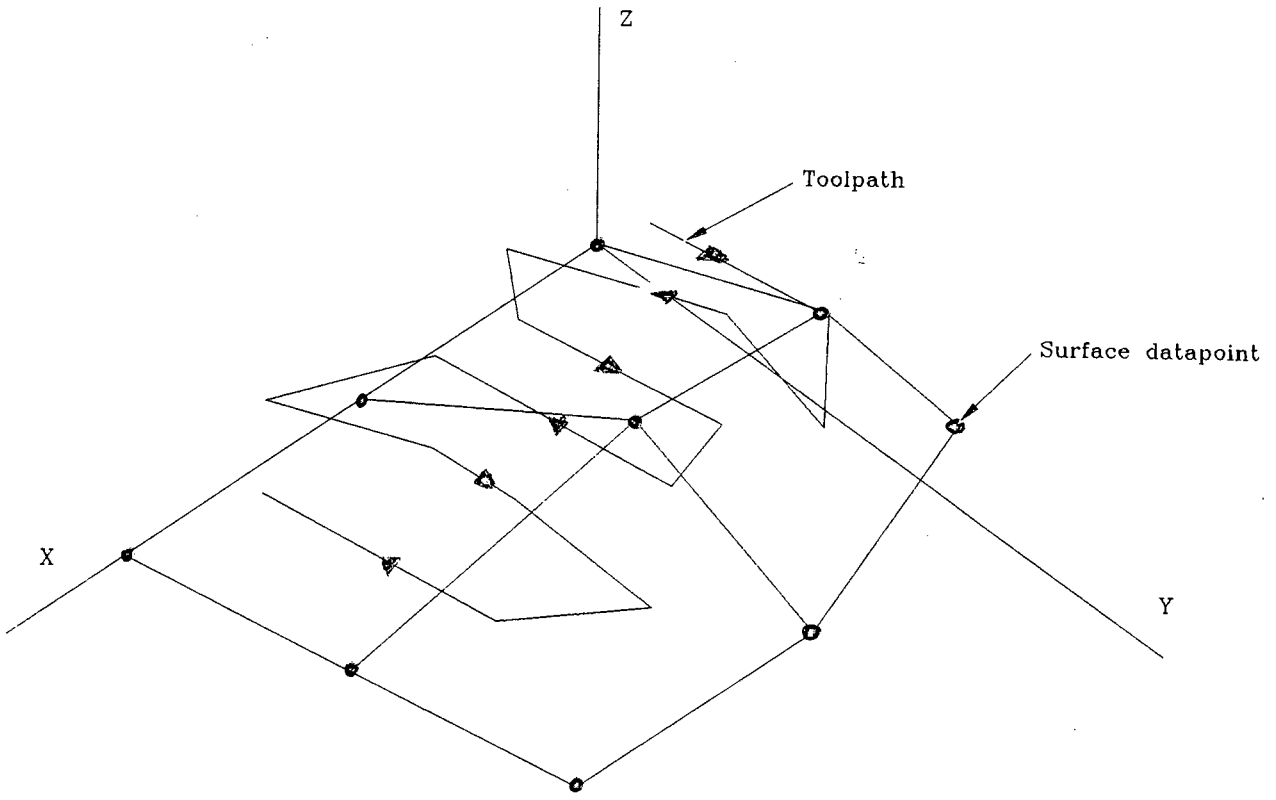


Figure A.9: The sample surface with the toolpath shown

APPENDIX B

A SAMPLE CALCULATION USING THE KEYS

As it is important to understand the use of the keys to indicate the sequence of toolpositions, a numerical example is given below.

A plan view of a 2 x 2 element surface with the frames and levels marked is shown below.

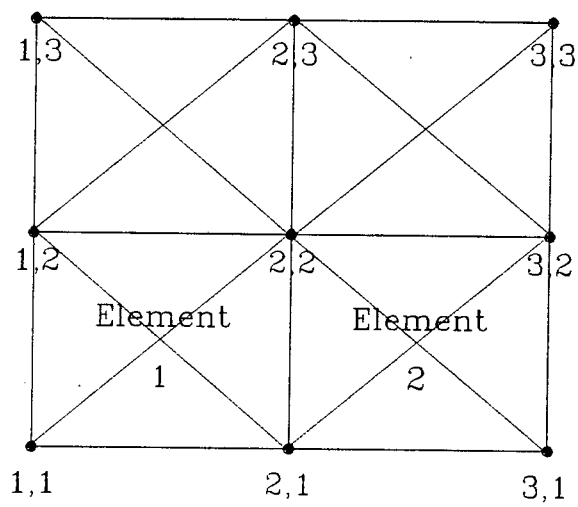


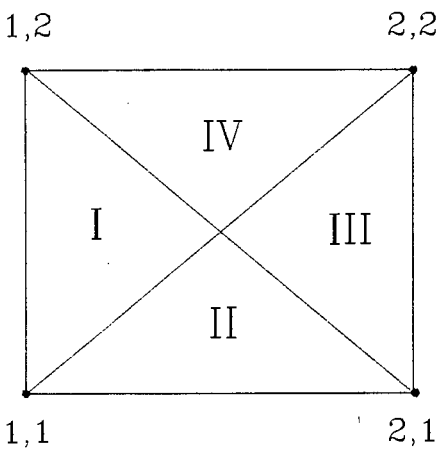
Figure B.1: A plan view of a 2 x 2 element surface with each element divided into its 4 triangular planes.

The following abbreviations are used in the key equations:

TL = Total Number of Levels in Surface
F = Reference Frame of Quadrilateral Element.
L = Reference Level of Quadrilateral Element.

The frame of the lower left coordinate of the quadrilateral element is known as the Reference Frame (see also Section 3.1.1 of the main text).

Quadrilateral Element i (an odd element because the Reference Frame is odd) has been extracted from the surface and is shown below in Figure B.2.



F = 1

L = 1

Figure B.2: Element 1 (odd) extracted from the surface in Figure B.1

The Reference Frame is odd so use equations (11) - (14) from section 3.1.1 in the main text.

Triangular Plane I for Odd Element:

$$\begin{aligned} Key &= [(TL-1) \times (F-1) \times 4] + L \\ &= [(3-1) \times (1-1) \times 4] + 1 \\ &= 1 \end{aligned} \qquad (11)$$

Triangular Plane II for Odd Element:

$$\begin{aligned} Key &= [(TL-1) \times (F-1) \times 4] + (TL-1) + 2(TL-1) - 2(L-1) \\ &= [(3-1) \times (1-1) \times 4] + (3-1) + 2(3-1) - 2(1-1) \\ &= 6 \end{aligned} \qquad (12)$$

Triangular Plane III for Odd Element:

$$\begin{aligned} \text{Key} &= [(TL-1) \times (F-1) \times 4] + L + 3(TL-1) \\ &= [(3-1) \times (1-1) \times 4] + 1 + 3(3-1) \\ &= 7 \end{aligned} \qquad (13)$$

Triangular Plane IV for Odd Element:

$$\begin{aligned} \text{Key} &= [(TL-1) \times (F-1) \times 4] + (TL-1) + 2(TL-1) - 2(L-1) - 1 \\ &= [(3-1) \times (1-1) \times 4] + (3-1) + 2(3-1) - 2(1-1) - 1 \\ &= 5 \end{aligned} \qquad (14)$$

The keys for element 1 (used to sequentially sort the toolpositions) are numbered below:

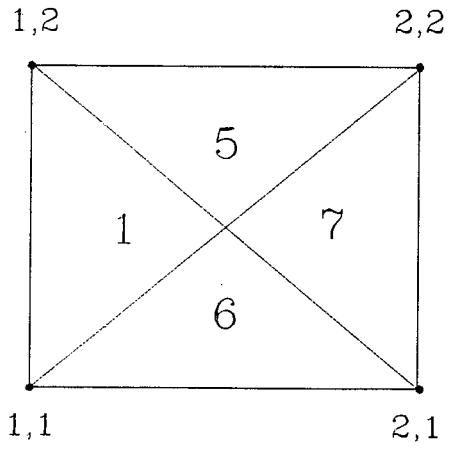
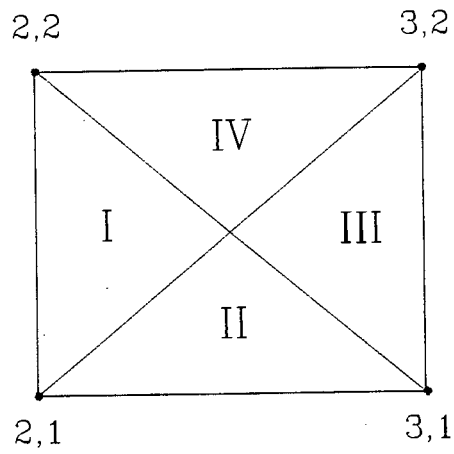


Figure B.3: Keys for Element 1

Element 2 (an even element) has been extracted from the surface and is shown in Figure B.4.



F = 2
L = 1

Figure B.4: Element 2 (even) extracted from the surface in Figure B.1

The Reference Frame is even so use equations (15) - (18) from section 3.1.1 in the main text.

Triangular Plane I for Even Element:

$$\begin{aligned} \text{Key} &= [(TL-1) \times (F-1) \times 4] + (TL-1) - (L-1) \\ &= [(3-1) \times (3-1) \times 4] + (3-1) - (1-1) \\ &= 10 \end{aligned} \qquad (15)$$

Triangular Plane II for Even Element:

$$\begin{aligned} \text{Key} &= [(TL-1) \times (F-1) \times 4] + TL + 2(L-1) \\ &= [(3-1) \times (3-1) \times 4] + 3 + 2(1-1) \\ &= 11 \end{aligned} \qquad (16)$$

Triangular Plane III for Even Element:

$$\begin{aligned} \text{Key} &= [(TL-1) \times (F-1) \times 4] - (L-1) + 3(TL-1) \\ &= [(3-1) \times (3-1) \times 4] - (1-1) + 3(3-1) \\ &= 16 \end{aligned} \qquad (17)$$

Triangular Plane IV for Even Element:

$$\begin{aligned} \text{Key} &= [(TL-1) \times (F-1) \times 4] + L + 2(TL-1) + 1 \\ &= [(3-1) \times (3-1) \times 4] + 1 + 2(3-1) + 1 \\ &= 12 \end{aligned} \tag{18}$$

The keys for element 2 are numbered below:

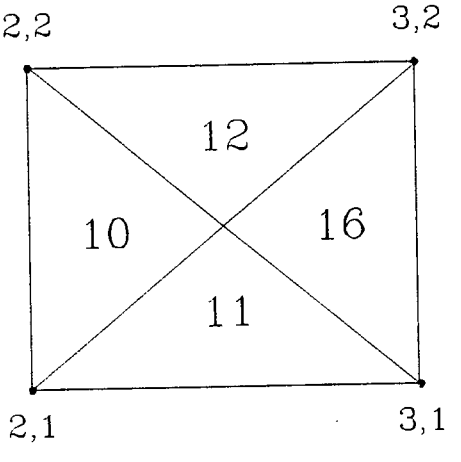


Figure B.5: Keys for Element 2

Using this procedure the keys for all the triangular planes within all the quadrilateral elements on the surface are calculated. The keys and their associated toolpositions are then sorted into ascending order, thereby obtaining the sequential toolpositions for the toolpath in the final toolfile. The calculated keys for the 2 x 2 element surface are shown in Figure B.6, with each key indicating a successive toolposition.

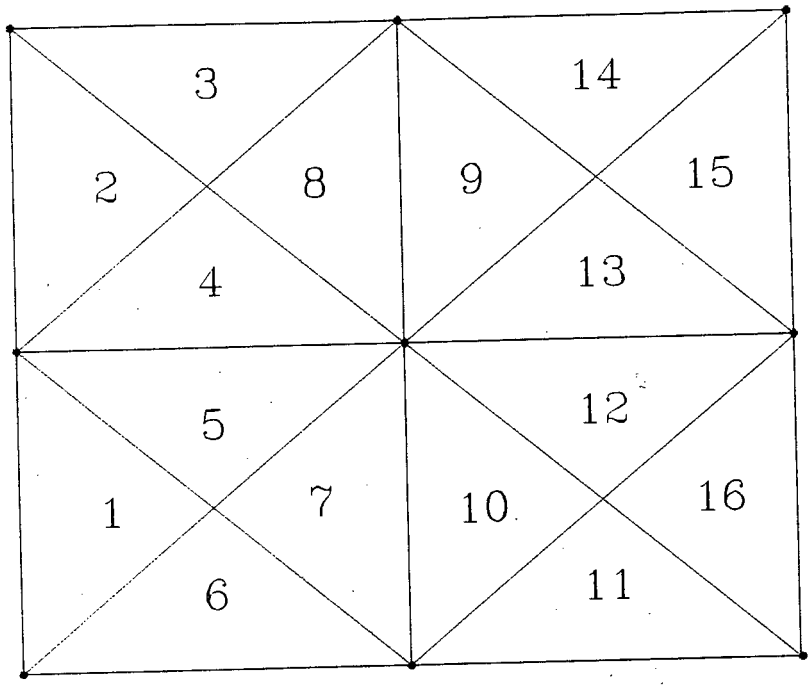


Figure B.6: 2 x 2 Element surface with each key numbering successive toolpositions

APPENDIX C

BRIEF INTRODUCTION TO DATA COMMUNICATIONS

Data communication, in the most general sense, is the exchange of information by two devices via some transmission link. In this system the devices are the IBM PC microcomputer (the local system) and the CNC computer (the remote system). They exchange data using their respective RS 232, or serial, interfaces. The term "serial interface" refers to the hardware in each device required for serial communications. The serial interfaces, or ports, are hard wired, or directly, connected as shown in Figure C.1

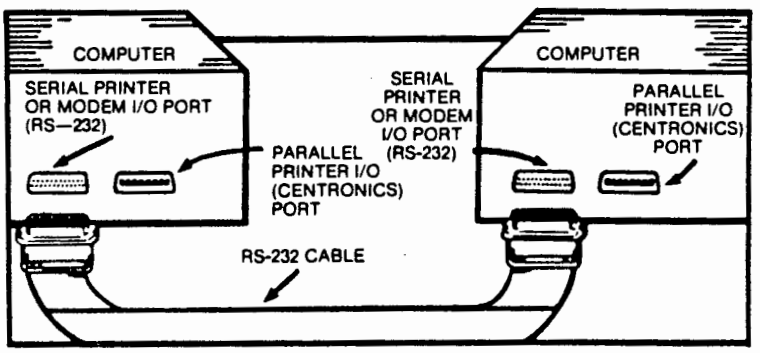


Figure C.1: Hard wire connection between serial ports

The term serial refers to the sequential manner in which the data is transmitted. The data consists of a specific number of binary digits or "bits" which may be either 1 or 0. There are codes which specify the pattern of 1's and 0's used to represent the data. The most commonly used code, the ASCII (American Society for Information Interchange) code, determines the pattern of seven 1's and 0's which represent a character set. The letter U, for example, is represented by the bit pattern 1 0 1 0 1 0 1. There are 128 (2^7) available characters in the character set, which includes the entire alphabet in upper and lower case and the numerals 0 to 9.

Each data "packet" consists of data bits, for example the bits used to represent the letter "U", which are sandwiched between a start and a stop bit. The use of a start and a stop bit to separate the data bits is known as asynchronous transmission. There is also an additional bit which is used as an error detection device and is called the parity bit. There are two types of parity checking: odd and even parity. In the case of even parity the transmitting computer counts the number of "1" bits within the packet. If there is an odd number of "1" bits the transmitting computer adds a "1" bit to the packet, making an even number of "1" bits. If there is an even number of "1" bits, the transmitting computer adds a "0" bit, thus keeping the number of "1" bits even. The receiving computer counts the number of incoming "1", bits and if the number is even then the computer has received the correct data. If, however, the number is odd then there has been a loss of data somewhere along the line and the receiving computer requests a retransmission of that packet of data. Odd parity operates in a similar manner, keeping the number of "1" bits odd.

The rate at which these packets of data are sent is known as the baud rate, and is measured in bits per second.

The last concept which needs to be explained is that of duplex. There are two possible duplex modes, half duplex and full duplex. Half duplex means that characters can be transmitted in both directions, but not simultaneously, over a single pair of wires. Full duplex, which is used in this system, allows characters to be transmitted in both directions, simultaneously, on a four-wire connection. The duplex mode determines how characters are echoed to the display of the local computer (the IBM PC in this case). If full duplex is selected then the other (remote) computer echoes any characters sent to it back to the display of the local computer. In half duplex the local system must perform this echo function.

Symptoms of incorrect duplex selection are either no characters (full duplex on a half duplex system) or double characters (half duplex on a full duplex system) displayed on the screen of the local computer.

Should the reader require more detailed information he is referred to [35].

APPENDIX D

HARDWARE DEVELOPMENT FOR THE SERIAL LINK

In order to use the IBM PC to emulate the teletype, the method used by the teletype to communicate with the CNC computer had to be determined.

D.1: Teletype Signalling Code

The description of the signalling code used by the teletype was summarised from the teletype manual [36].

The teletype transmits and receives characters by a series of 20 mA current pulses, see Figure D.1.

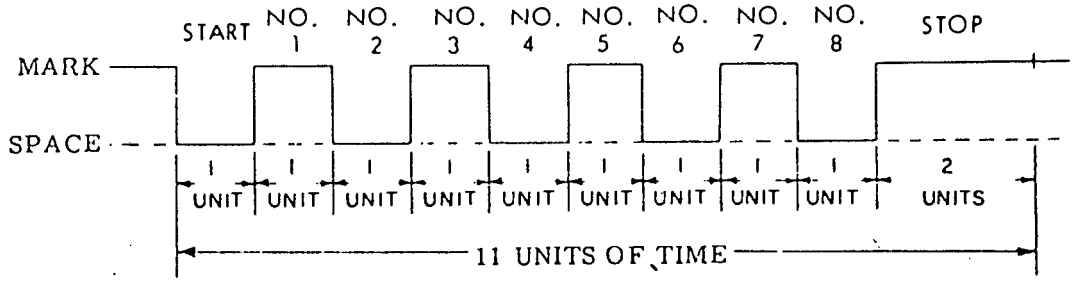


Figure D.1: Current pulse pattern of the letter "U" (with even parity) from the Teletype manual [36]

A marking pulse (or "1" bit) refers to a time interval during which there is current flowing in the transmission line, and a spacing pulse (or "0" bit) refers to the time interval when there is no current flowing. Each character is represented by eleven current pulses (the data packet). The first pulse is the start pulse and is always a spacing pulse. The next eight pulses are the data pulses, and the tenth and eleventh, which are always marking, are the stop pulses. The ASCII code is used by the teletype and, as described in Appendix C, uses only seven bits (pulses) to represent its character set.

The eighth data bit in the data packet is used as an "even parity bit" for error detection. However, if parity checking is not available on the remote device, this data bit is transmitted as a marking pulse or "1" bit.

D.2: The Teletype Connection to the CNC computer

The link between the CNC computer and the teletype uses a 20 mA current loop. There are two pairs of connections, a positive and a negative connection for transmitting and for receiving. The conventional connection between two devices in a current loop is shown in Figure D.2 below:

Transmit +	is connected to	Receive	+
Transmit -	is connected to	Receive	-
Receive +	is connected to	Transmit	+
Receive -	is connected to	Transmit	-

Figure D.2: Conventional connection between two current loop devices

Each device has its respective four connections wired to a five pin Nana male plug. Berelowitz [2] determined the configuration of the plugs on the teletype and the CNC computer communication port, as shown in Figure D.3.

	CNC port	Teletype
Description	Nana pin Number	Nana pin Number
Transmit +	4	5
Transmit -	3	2
Receive +	2	1
Receive -	1	4
Not connected	5	3

Figure D.3: 5 pin Nana plug configuration as used between the Teletype and the CNC machine

The teletype is set to full duplex mode for communication with the CNC computer.

The same connections as were used between the teletype and the CNC device could now be used to configure the serial port of the IBM PC.

D.3: Configuration and Connection of the IBM Asynchronous Communication Adaptor

The IBM Asynchronous Communication Adaptor Card is used for the serial communication of data from the IBM PC to another serial device. The card is configured for communication with the CNC computer by plugging in two shunt modules, as shown in Figure D.4 below:

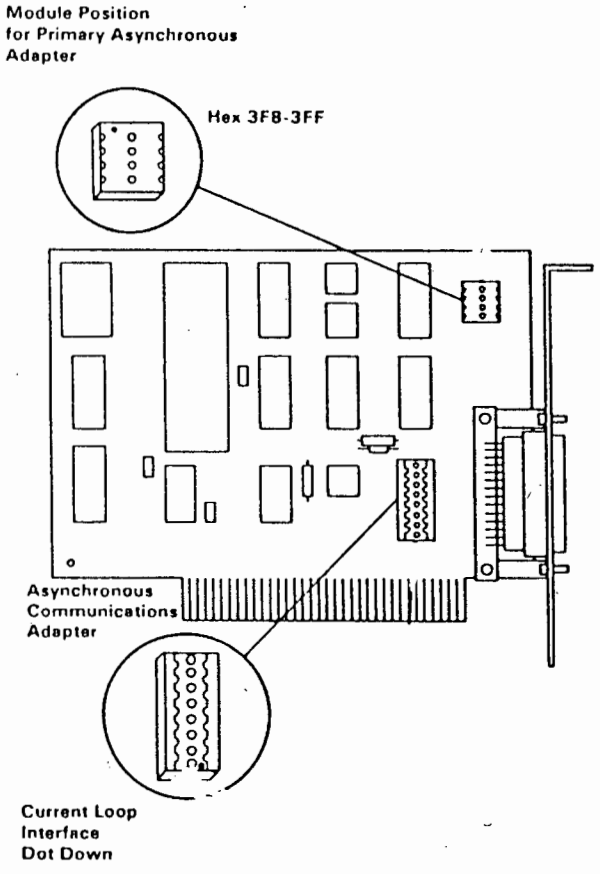


Figure D.4: Selecting the correct interface format and adaptor address on the IBM Asynchronous Communications Adaptor Card, from [37]

The topmost shunt was positioned with the locator dot up, thereby specifying the card as the primary communications adaptor, or COM1 for those more familiar with MS-DOS. The bottom shunt was positioned with its locator dot down, specifying operation as a current loop device. The adaptor is fitted with a 25 pin D-shell male connector shown in Figure D.5.

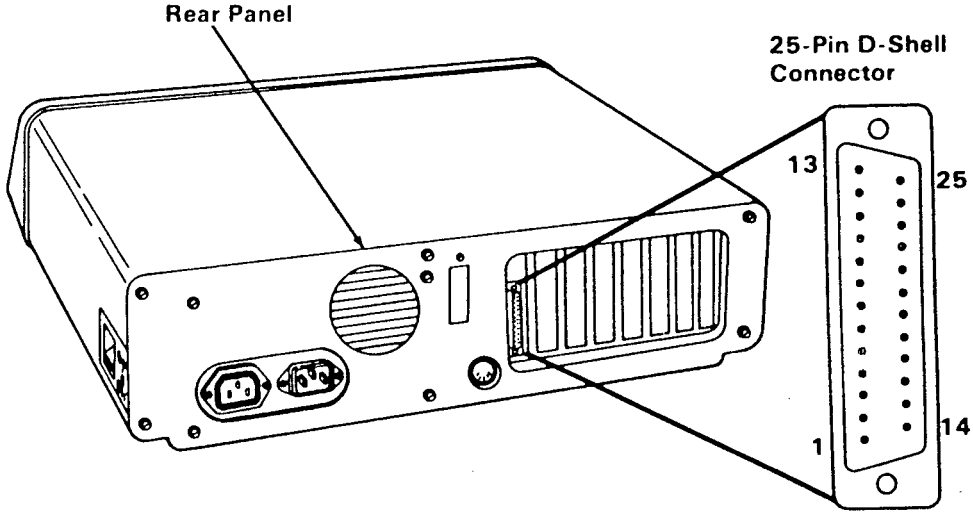


Figure D.5: 25 pin D-TYPE connector as fitted to the IBM Asynchronous Communications Card, from [37]

The configuration of the the plug, taken from the IBM Technical Reference Manual [37], is reproduced in Figure D.6.

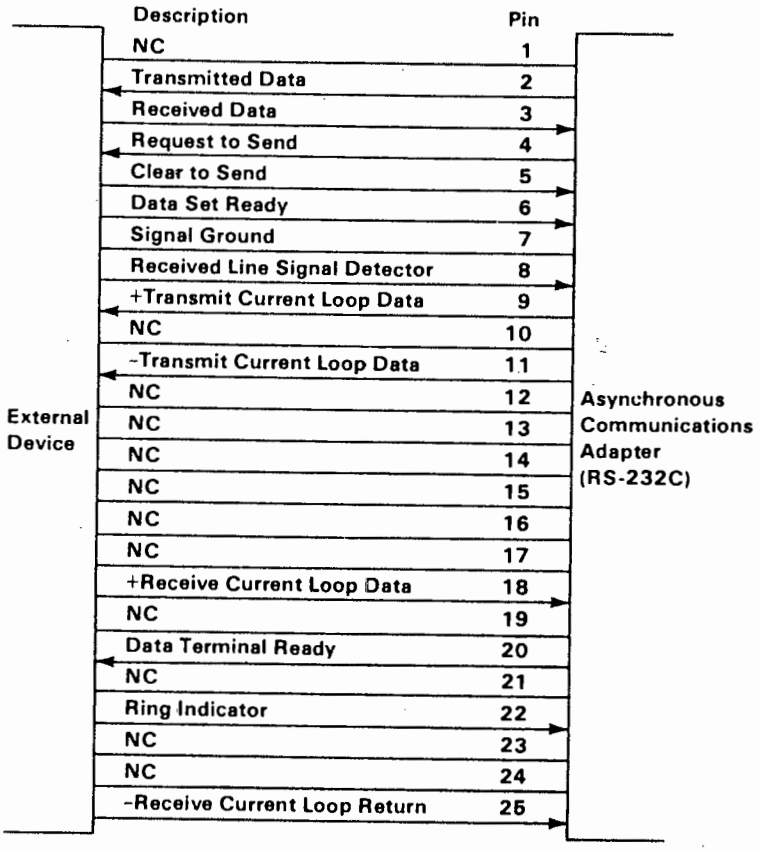


Figure D.6: Pin Configuration of the connector on the IBM Asynchronous Communications Adaptor, from [37]

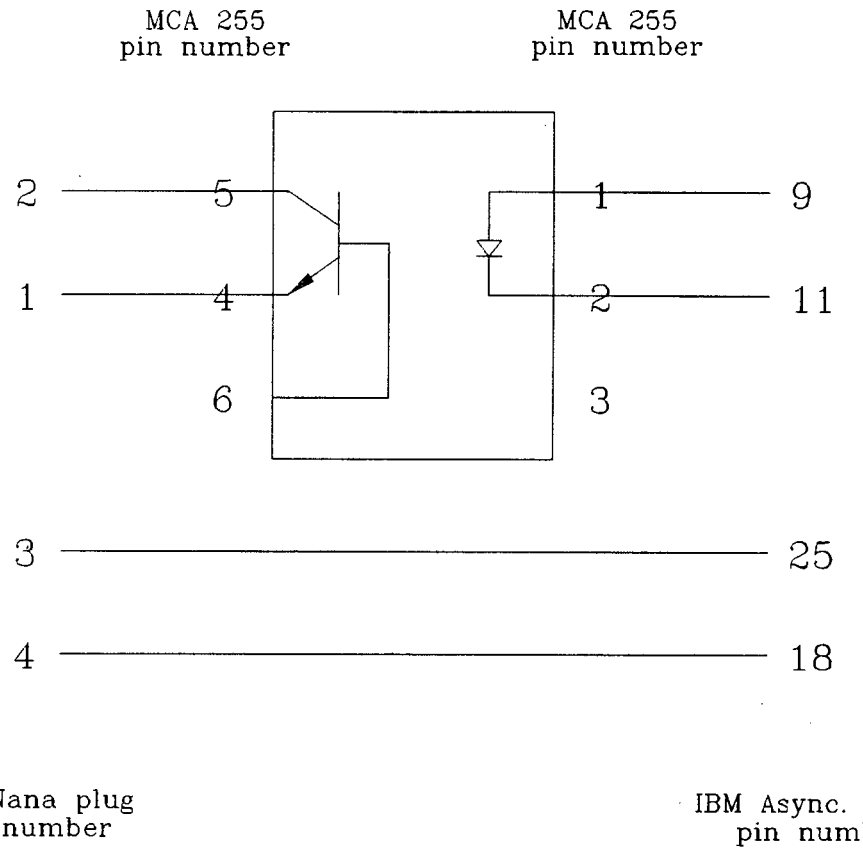
The pertinent pins of the connector with regard to current loop communication are shown in Figure D.7.

Description	Pin number on IBM Communication Card
Transmit +	9
Transmit -	11
Receive +	18
Receive -	25

Figure D.7: Relevant pins on IBM card

Originally the connection that was used was according to Figure D.2, but this proved unsuccessful.

The problem was that both devices were attempting to generate a 20 mA current on all four connections, thereby resulting in a clash of currents. The solution was to separate the connections via an MCA 255 opto-isolator, which avoided the problem of conflicting currents and allowed communication between the IBM PC and the CNC computer. The connection that was used is shown in Figure D.8.



Motorola MCA 255
Opto-isolator

Figure D.8: Circuit diagram of the plugs and cable used to connect IBM PC to the CNC computer

APPENDIX E

DATA FORMAT FOR USE WITH THE MILLING SYSTEM

The correct format of the data for use with the milling system is very important if the system is to be used with data from some external source. The milling system requires data in what is known in TRUE BASIC as a RECORD file. Each record in the file holds one data item. Because they are stored in a format peculiar to TRUE BASIC, they cannot be viewed on the computer screen. The record file is an extremely powerful format and allows the computer to read or write individual items to a file. This type of file is usually referred to as a Random Access File. However, the use of record files is not as simple as the standard text file (or ASCII file), and in order to make it as easy as possible to use the milling system with external data a utility program has been written which will translate data from a simple ASCII file into the correct record file format for use with the milling system.

E.1: The ASCII data file

An ASCII data file was chosen because it is the simplest and most widely used code for data exchange. ASCII data can be created using LOTUS 123 or even a simple wordprocessor. The data, however, must conform to some specific format which is outlined below:

E.1.1: Format of the ASCII data files

a) The data must have exactly three coordinates across; X,Y and Z. The data list may be as long as required.

b) The coordinates must be separated from each other by at least one space.

c) The data file must not have any blank lines before the beginning of the X,Y and Z values. This is particularly important when using LOTUS 123 to create the ASCII data file.

d) Each coordinate can have as many digits as are required, but three decimal places should usually suffice.

E.1.2: The order of the ASCII data

The order of the data in the list is important. Figure E.1 shows a 2 x 2 element surface with each datapoint numbered. The sequence of the datapoints in the input file (each of which consists of an X,Y and Z coordinate) must be as shown in the diagram. In addition the user must know how many frames and levels are used to define the surface.

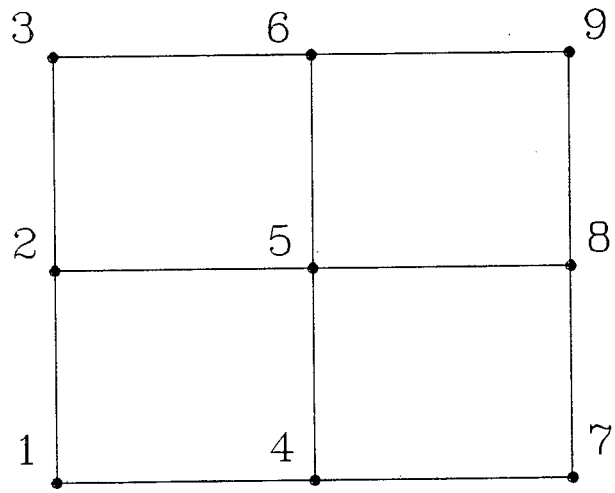


Figure E.1: Sequence of surface datapoints in the input file

Once the user has the ordered data in the ASCII file the program FILECONV.EXE, or if the user has TRUE BASIC, FILECONV.TRU, can be used to convert the ASCII file into a TRUE BASIC record file. (See Appendix N for a listing and description of this program.) The record file can then be loaded into the milling system in the normal way.

E.1.3: Hints for creating ASCII files

There are many methods which could be used to create the ASCII data files such as a BASIC program, a wordprocessor, the Framework package or Lotus 123. Using a spreadsheet facilitates the manipulation of the data, and most spreadsheets have the ability to generate ASCII files. If the user has access to LOTUS 123, then the procedure is as follows:

a) Create the spreadsheet with the data points in the correct sequence as described above.

b) Use the LOTUS command "PRINT" to print the X Y and Z range to a disk file. This creates an ASCII file in the correct format.

c) Make sure that there are no leading blank lines in the data file by using a simple text editor such as the DOS EDLIN or a wordprocessor.

d) Use the FILECONV program to convert this ASCII data file into a TRUE BASIC record file.

e) Run the milling system and use the newly created record file for the surface data.

E.2: The Record File

Should the user have access to the TRUE BASIC package and wish to write directly in the record format, the file structure must be as follows:

FILE ORGANIZATION : RECORD
RECORD SIZE : 20 bytes

There are five records for each surface data point. Each record consists of a numeric variable (as opposed to the ASCII string variable) :

1. Frame
2. Level
3. X Coordinate
4. Y Coordinate
5. Z Coordinate

The sequence of the data must be the same as shown in Figure E.1.

In order to give the user a base from which to construct his own record data files, an example program called SURFMAKE.TRU is given in Appendix M. This program used a mathematical formula to create the record file containing the data of a hemisphere on a flat plate. The user may wish to modify this program in order to create his own record data files.

APPENDIX F

DERIVATION AND CALCULATION OF THE SPHERE EQUATIONS USED IN THE MULTIPLE LINEAR REGRESSION TECHNIQUE

F.1: Derivation of the sphere equations

The standard equation for a sphere with centre offset from the origin is well known to be:

$$(x-x_0)^2 + (y-y_0)^2 + (z-z_0)^2 = R^2 \qquad (1)$$

where x_0, y_0 and z_0 are the coordinates of the centre of the sphere measured from the origin and R is the radius.

Equation (1) can be rewritten as:

$$x^2-2x_0x+x_0^2 + y^2-2y_0y+y_0^2 + z^2-2z_0z+z_0^2 = R^2 \qquad (2)$$

Equation (2) can be rearranged as:

$$x^2+y^2+z^2 = 2x_0x+2y_0y+2z_0z - (x_0^2+y_0^2+z_0^2-R^2) \qquad (3)$$

Now let

$$Y = x^2+y^2+z^2 \qquad (4)$$

$$A = 2x_0 \qquad (5)$$

$$B = 2y_0 \qquad (6)$$

$$C = 2z_0 \qquad (7)$$

$$D = - (x_0^2+y_0^2+z_0^2-R^2) \qquad (8)$$

Substituting equations (4) - (8) into (3) gives:

$$Y = Ax + By + Cz + D \qquad (9)$$

Equation (9) fits the form of a multiple linear regression in which there is one dependent variable (Y) and two or more independent variables (x,y and z).

Solving the regression yields the values of the coefficients A,B,C and D which are then used in equations (4) - (8) to give the desired values x_0, y_0, z_0 and R.

Each datapoint together with the calculated values of x_0, y_0 and z_0 is then used in equation (1) to calculate a theoretical radius Rt. The error in each datapoint is then:

$$\text{error in each datapoint} = R - R_t \qquad (10)$$

F.2: Calculation of the equation for the known surface

The measured surface datapoints of the known surface were input into LOTUS 123 and the multiple linear regression facility used to determine the coefficients. The output is shown in Table F.1.

Regression Output:			
Constant			-2478.66
Standard Error of the Y estimate			6.125
Coefficient of determination			0.999985
Number of Observations			225
Degrees of Freedom			221
Coefficients	81.6996	78.1164	-7.1917
Standard Errors	0.0287	0.0279	0.0571

Table F.1: Output of the multiple linear regression of the known surface from LOTUS 123

Using equations (4) - (8) the coefficients A - D are obtained:

$$A = 81.6996 \qquad B = 78.1164 \qquad C = -7.1917$$
$$2x_0 = 81.6996 \qquad 2y_0 = 78.1164 \qquad 2z_0 = -7.1917$$
$$x_0 = 40.8498 \text{ mm} \qquad y_0 = 39.0582 \text{ mm} \qquad z_0 = -3.5958 \text{ mm}$$

$$D = -2478.66$$

$$-(x_0^2+y_0^2+z_0^2-R^2) = -2478.66$$

$$-(40.8498^2 + 39.0582^2 + 3.5958^2 - R^2) = -2478.66$$

$$R^2 = 26.990 \text{ mm}$$

Thus the equation for the known surface was:

$$(x-40.8498)^2 + (y-39.0582)^2 + (z+3.5958)^2 = 26.990^2 \quad (11)$$

The high value of the coefficient of determination indicated that there was a good fit between the measured data and the derived sphere equation.

Because the errors in each surface datapoint were normally distributed (see discussion in Chapter 4 and Figure 4.1) the standard deviation of these errors about the mean radius, R, was valid and was therefore determined, using a built in LOTUS function. The result was:

Standard deviation = 0.1117 mm

F.3: Calculation of the equation for the machined surface

The same procedure was used with measured datapoints of the machined surface. The output is shown in Table F.2.

Regression Output:			
Constant			-1228.54
Standard Error of the Y estimate			33.794
Coefficient of determination			0.999229
Number of Observations			80
Degrees of Freedom			76
Coefficients	76.6526	36.2415	-11.6400
Standard Errors	0.2657	0.3080	0.5404

Table F.2: Output of the multiple linear regression of the machined surface from LOTUS 123

Using equations (4) - (8) the coefficients A - D are obtained:

$$A = 76.6526 \qquad B = 36.2415 \qquad C = -11.6400$$
$$2x_0 = 76.6526 \qquad 2y_0 = 36.2415 \qquad 2z_0 = -11.6400$$
$$x_0 = 38.3263 \text{ mm} \qquad y_0 = 18.1208 \text{ mm} \qquad z_0 = -5.8194 \text{ mm}$$

$$D = -1228.51$$
$$- (x_0^2+y_0^2+z_0^2-R^2) = -1228.51$$
$$- (38.3263^2 + 18.1208^2 + 5.8194^2 - R^2) = -1228.54$$
$$R^2 = 24.548 \text{ mm}$$

The equation for the machined surface was thus:

$$(x-38.3263)^2 + (y-18.1208)^2 + (z+5.8194)^2 = 24.548^2 \quad (12)$$

As in the calculation of the previous sphere equation coefficient of determination of 99.92% indicated a good fit between the measured data and the derived sphere equation.

The errors in the surface datapoints were not normally distributed (see Figure 4.2) therefore it was not valid to calculate the standard deviation for this data set.

APPENDIX G

DESCRIPTION OF THE EQUIPMENT USED

G.1: Computer

Computer type	:	IBM PC compatible
Storage Capacity	:	10 megabyte hard drive and 360 kilobyte floppy disk
Memory Capacity	:	640 kilobytes
Graphics card	:	Hercules compatible graphics card
Communications Port	:	IBM Asynchronous Communications Adaptor configured for 20 mA current loop.

G.2: Cable

The cable connecting the RS232 serial port of the IBM PC computer to the CNC computer was specially designed and incorporated a MOTOROLA MCA255 opto isolator. The circuit diagram is given in Appendix D, Figure D.8.

G.3: Computer Numerically Controlled Machine

CNC machine	:	Bridgeport Series 1 CNC
Power	:	2 H.P. AC induction motor
Spindle speeds	:	60 - 4 000 R.P.M.
Controller	:	Textron
Axes of control	:	3 axis simultaneous continuous path contouring
	:	2 axis Circular interpolation in the XY,YZ and XZ plane
Storage Capacity	:	25 metres of RS 358 standard tape (approximately 8 000 bytes)
Data input device	:	Serial line interface @ 20 ma standard or paper tape reader

APPENDIX H

USER GUIDE TO THE CNC MILLING SYSTEM

The User Guide to the milling system is divided into five parts:

- installing the system from the floppy disk
- running the milling program for a sample surface
- running the postprocessor for the conversion of the toolfile into the CNC programs
- running the communications package to transfer the CNC modules into the memory of the CNC computer
- a brief guide to the use and setting up of the milling machine to mill the surface.

1. INSTALLING THE PROGRAM FROM FLOPPY DISK

1.1: General Considerations about the Milling System

All the programs, both the compiled versions and the source code of the CNC system, are contained on two floppy disks labelled "MILLING SYSTEM 1" and "MILLING SYSTEM 2".

It is preferable to install the system on a hard disk machine because of the limitation of storage of large surfaces on floppy disks. Using a hard disk also has the benefit of loading and saving the long toolfiles and surface data files far more quickly than a floppy disk. In addition, 640 Kilobytes of memory are required to load the data files and to display the surfaces. If the memory is insufficient the system will warn the user and make suggestions to overcome the problem, if possible.

In order to follow the installation procedure the user should be familiar with DOS and the concept of directories and subdirectories.

The files with the .EXE extension have been compiled and can be run directly from the Disk Operating System, or DOS, but in order to allow the future user to modify the system the source, or original, TRUE BASIC programs have been included in the "SOURCE" subdirectory on the MILLING SYSTEM 1 floppy disk.

A list of all the files on the two floppy disks is given below, together with a brief description of their purpose.

MILLING SYSTEM 1 FLOPPY DISK:

Root Directory :

CNCMILL.EXE - a compiled version of the complete milling system which can be run from DOS
POSTPROC.EXE - a compiled version of the postprocessor which can be run from DOS

Source Subdirectory :

BEGIN.LIB - source code of the BEGIN library
CNCMILL.TRU - source code of the main program
DISPLAY.LIB - source code of the DISPLAY library
FILES.LIB - source code of the FILES library
MATH.LIB - source code of the MATH library
POSTPROC.TRU - source code of the POSTPROCESSOR
3DLIB.TRC - compiled TRUE BASIC 3D Graphics Package; this cannot be listed.

MILLING SYSTEM 2 FLOPPY DISK

Root Directory :

MIRROR.EXE - the MIRROR communications package
SPHERE.XYZ - sample record surface data file for practice
SPHERE.TOL - sample toolfile for practice

START.XTS - script file for use with MIRROR for transferring the CNC modules
STD.XTK - the MIRROR command file containing the setup parameters for the IBM Communications port

Examples Subdirectory :

SURFMAKE.TRU - program to create a record surface data file

Utility Subdirectory :

FILECONV.TRU - program to convert ASCII data files into record files for use with the Milling System (see Appendix E for more details)

By following the instructions in the next section the user will be able to install the system onto a hard disk machine.

1.2: Installing the System onto the Hard Disk

1.2.1 Turn on the hard drive machine and select the root directory of the hard disk.

CD C:\ <enter>

1.2.2 Make a directory on the hard disk called SURFMILL using the following keystrokes:

MD SURFMILL <enter>
CD SURFMILL <enter>

1.2.3 Copy the files from the "ROOT DIRECTORY" of the MILLING SYSTEM floppy disks onto the hard drive using:

COPY A:*.* C: <enter>

2. THE MILLING PROGRAM

The operation of the programs has been made as simple as possible by having pertinent messages displayed on the screen. All the message screens are self explanatory but the more important ones have been included in the User Guide to assist the novice user. The operation of the program is illustrated by using the sample surface and toolfiles, SPHERE.XYZ and SPHERE.TOL, included on the accompanying floppy disks

2.1: Running the Milling Program

2.1.1 In order to use the milling program a record file of surface data coordinates in the correct format (see Appendix E) must be present on the hard disk in the SURFMILL directory. There is a sample surface datapoint file, SPHERE.XYZ, on the MILLING SYSTEM 2 disk which can be used and which should have been copied onto the hard disk.

2.1.2 Run the milling program

CNCMILL <enter>

2.1.3 The program will display the following message on the screen.

```
You can run this program on:

   IBM Colour Graphics Adaptor ..... PRESS C
   IBM Enhanced Graphics Adaptor ... PRESS E
   Hercules Graphics Card ......... PRESS H
```

The user should respond with an "C" ,"E" or "H", depending on the graphics card in the computer being used.

2.1.4 After the graphics card has been selected the following screen is displayed:

```
                    CNC MILLING PROGRAM

     MILLING PROGRAM FOR THE MILLING OF COMPLEX THREE
     DIMENSIONAL SURFACE ON A THREE AXIS CNC MACHINE

     WRITTEN AS PART OF AN MSC IN ENGINEERING DEGREE

                  AUTHOR : D.R. BACK

           <PRESS SPACE BAR TO CONTINUE>
```

and after pressing the SPACE BAR:

```
                    CNC MILLING PROGRAM

This program was written for the Mechanical Engineering Department, U.C.T.
to enable complex three dimensional surfaces to be machined. The surface
to be machined is represented by a series of surface data points which are
fed into the milling program which then calculates the toolpath to mill
the surface. The program has a three dimensional graphical display
capability for verification of the surface and the toolpath.

Before running the program:
 1. Make sure that there is a surface data file on the disk.
 2. Make sure that there is sufficient space on the disk for the
    toolfiles.

The calculation of the toolpath for typical 20 x 20 element surface takes
approximately 30 minutes depending on the speed of the computer. If the
toolfile already exists however you can use the three dimensional display
to view the surface and toolpath.

                    PRESS SPACE BAR TO CONTINUE
```

2.1.5 The program then displays the following screen and asks the user:

CNC MILLING SYSTEM
If the toolfile already exists you can view the surface and the toolpath in three dimensions without having to recalculate the toolpositions.

Has the toolfile been previously created (Y/N)

Reply for sample surface

Has the toolfile been previously created (Y/N) : Y

The reply to this input prevents the computer from recalculating the toolpositions if the toolfile already exists.

2.1.6 The program then asks the user:

CNC MILLING SYSTEM
You need to input: 1. The number of Frames used in the surface data 2. The number of Levels used in the surface data 3. The name of the Toolfile to be viewed 4. The name of the disk file containing the surface data

Replies for sample surface

1. The number of Frames in the surface data 20
2. The number of Levels in the surface data 20
3. The name of the toolfile to be viewed SPHERE.TOL
4. The name of the disk file containing the surface data SPHERE.XYZ

If the toolfile did not exist and had to be calculated for the first time an additional input:

5. The diameter of the spherical cutting tool

would be required from the user. In the case of the sample surface a 5 mm diameter cutter was used to calculate the toolpath.

2.1.7 At this point the program loads the surface data; this may take some time so please be patient. If the toolfile exists the 3D viewer is called directly after the surface data has been loaded, otherwise the program will first calculate the toolpath and create a toolfile containing the the tool coordinates and only then call the 3D viewer.

2.1.8 When the 3D viewer is called the user has the option to create an AutoCad DXF file of the surface and toolpath. The first screen displayed by the 3D display is:

```
CNC MILLING SYSTEM
THREE DIMENSIONAL DISPLAY AND AUTOCAD DXF FILE GENERATOR

You can create an AutoCad v9 (or later) DXF file for plotting
and viewing.  Two files are created; one containing the surface
and the other the toolpath. The DXF file containing the
toolpath has the same name as the surface DXF file, but with
a T as the last letter. 2 files have been created so that
the toolpath and surface can be viewed independently.

If you wish to view them together, call the DXF files one after
another in AutoCad

Do you want AutoCad DXF files created. (Y/N)
```

Reply for sample surface

Do you want AutoCad DXF files created. (Y/N) N

2.1.9 The program follows with the 3D view selector, shown overleaf, from which the user selects the camera position to view the surface and toolpath.

```
                    CNC MILLING SYSTEM

                THREE DIMENSIONAL VIEW SELECTOR

Three views of the surface are available ;
   1 - 2-D view from directly above the surface , ie XY plane
   2 - 2-D view from the side of the surface , ie YZ plane
   3 - 3-D view from the X Y Z extremity
   4 - Specify your own Camera Position
   5 - End viewing session

        SELECT VIEW BY PRESSING A NUMBER FROM 1-5
```

Reply for sample surface

Which view do you want? 3

If option 3 is chosen the screen display should be similar to Figure H.1

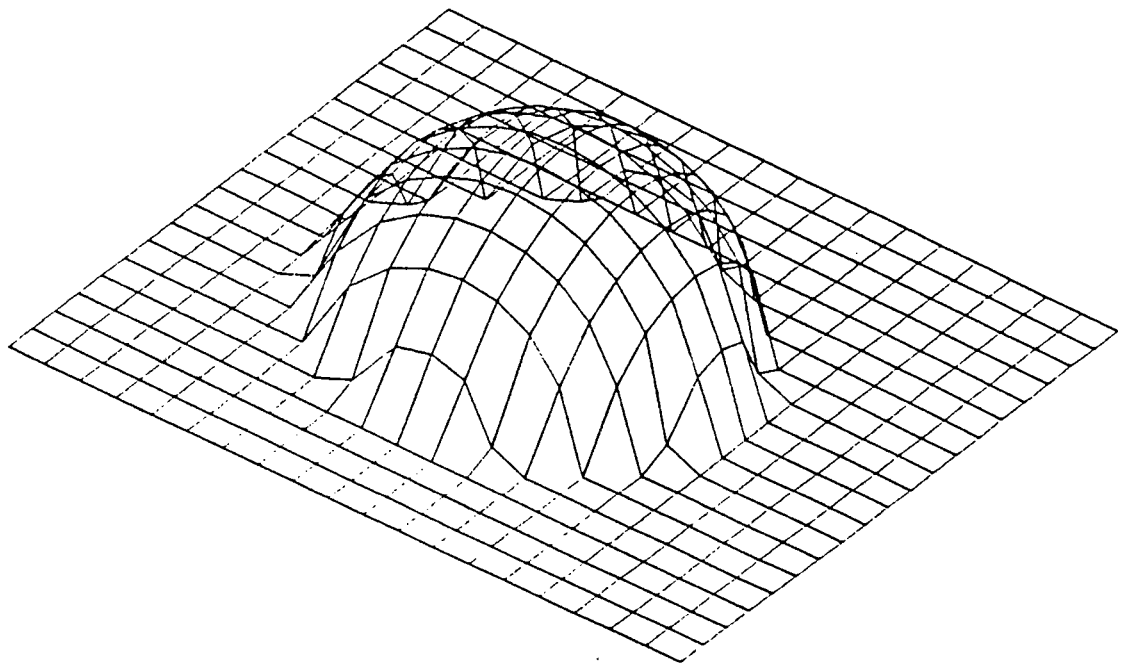


Figure H.1: Hemi-sphere on a flat plane

Press the SPACE BAR to return to the view selector and either choose another view or select option 5 to end the viewing session.

2.1.10 After the user has finished viewing the surface the program terminates and returns to DOS. If the toolpath was calculated the final toolfile will have been created and given the name specified by the user.

3. THE CONVERSION OF THE TOOLFILE INTO THE CNC MODULES

The final toolfile is used with the postprocessor to create the CNC modules. The postprocessor is called POSTPROC.EXE and can be run directly from DOS.

3.1: Running the Postprocessor

3.1.1 The postprocessor should have been copied onto the hard disk in Section 1.2 and is run by typing:

POSTPROC <enter>

3.1.2 When the postprocessor begins it displays the following screen and asks the user to input the information required.

```
CNC MILLING SYSTEM
POSTPROCESSOR FOR CONVERSION OF TOOLFILE INTO CNC PROGRAMS

You need to input the following information:

        1. Name of the toolfile to convert
        2. Number of Frames in the surface
        3. Number of Levels in the surface
        4. Name of the CNC program files

The Bridgeport CNC files are numbered consecutively and they all
have the file extension .PRG to identify them on the disk
```

Replies for the sample surface

Name of toolfile to convert?	SPHERE.TOL
Number of Frames in the surface?	20
Number of Levels in Toolfile?	20
Name of the Bridgeport CNC module to create?	ABCD

3.1.3 The postprocessor then creates a series of sequentially numbered CNC modules, "ABCD1.PRG", "ABCD2.PRG", "ABCD3.PRG", etc until the entire toolfile has been converted. When the postprocessor has completed the conversion it displays the following screen to inform the user that the files have been converted and then the computer returns to DOS.

CNC MILLING SYSTEM
POSTPROCESSOR FOR CONVERSION OF TOOLFILE INTO CNC PROGRAMS
All the CNC modules have been created

4. TRANSFERRING THE CNC MODULES INTO THE CNC MACHINE

4.1: Connecting the IBM PC to the CNC computer

4.1.1 The IBM PC computer with the correctly configured and installed RS232 card as described in Appendix D is connected to the CNC computer using the cable marked "IBM - CNC CABLE".

4.1.2 The CNC machine is turned on and the Red "LIMIT" button is pushed to initialise the CNC machine. The machine is then set up in the standard way. Section 5 gives some hints for the inexperienced user.

4.2: Running the Communications Package

4.2.1 The communications package is run by typing:

MIRROR <enter>

This loads the MIRROR communications package and uses the STD.XTK command file to automatically configure the IBM PC and to run the START.XTS script file.

4.2.2 The script file displays the following message:

```
Transfer of files from the PC to the CNC computer
```

and at the bottom of the screen asks:

```
Are you ready to begin? (Y/N)
```

Reply for the sample surface

Are you ready to begin? (Y/N) Y

Once the user has made sure that all connections are as required the "Y" key should be pressed to establish communications.

4.2.3 The script file then displays the following list of instructions on how to set the CNC control panel in order to make the connection.

```
You first need to establish communication with
the CNC computer so follow carefully the
instructions below.

    1. Set the CNC machine to SETUP mode with the
       MODE switch on the control panel.
    2. Turn the EDIT/RDI switch to EDIT ... BUT DO NOT
       PRESS IT YET!
    3. Get ready to press the EDIT switch AFTER
       hitting ENTER on the PC.
```

The user should follow the instructions on this screen and once the connection has been made the script file will display:

```
The PC is now connected to the CNC computer,
also any programs in the CNC will be cleared
```

Establishing the connection is the most crucial part of the file transfer procedure and the user must ensure that the above message has been displayed PRIOR to continuing.

4.2.4 Thereafter the script file asks:

```
Enter the name of the Program file to send to the CNC machine
```

Reply for the sample surface

Enter the name of the Program file to send to the CNC machine ABCD1.PRG

The user types in the name of the CNC module and the script file automatically transfers it to the CNC computer.

4.2.5 Once the module has been transferred the script file displays the following:

```
Finished sending module (filename), time to
start the machine.

1. Turn the MODE switch to AUTO
2. Turn the FUNCTION switch to RESTART and
   press it
3. Turn the FUNCTION switch to START CONTINUE
   and press to begin the program.


When the machining operation is finished turn
the MODE switch back to SETUP and the
EDIT/RDI switch to EDIT (BUT DO NOT PRESS IT!).

Now:  1. Press "C" on the PC and then the EDIT switch
         to send the next module
OR
      2. Press "F" on the PC to disconnect and
         terminate the communications program.
```

4.2.6 The user should follow the procedure above, and once all the CNC modules have been used to fully machine the surface the "F" key should be pressed, whereupon the IBM PC disconnects itself from the CNC computer and displays:

```
You have finished the machining procedure and the PC
has disconnected itself from the CNC computer. To reconnect
you must rerun this program from the start.
```

The script file returns the computer to DOS on completion of the transfer process.

5. A BRIEF GUIDE TO SETTING UP THE MILLING MACHINE

Although the use of the CNC machine with this system is no different from conventional milling, there are some guidelines below which could prove helpful to the novice user.

5.1: Setting up the Milling Machine

5.1.1 The origin from which the data points are measured should always be X=0, Y=0, Z=0. This corresponds to the lower left corner of the surface to be milled. The user should set the zero point appropriately and also ensure that the toolchange position X=-20 , Y=-20 is within the machine limits. Figure 3.23 in the main text shows the zero and the toolchange position.

5.1.2 The Tool Length Offset (TLO) should be set so that there is sufficient movement possible in the Z axis to mill the highest point of the surface. Also the last 15mm of travel in the Z axis are performed at a very slow speed and this should be taken into account when setting the TLO. A formula (with all values in millimeters) to help with the setting of the TLO is :

TLO = (Greatest Z height - Lowest Z height) + 15

As mentioned in 5.1 the lowest Z height should be 0.

5.1.3 Once the TLO has been set, the height of the milling table is adjusted so that the cutter tip just touches the surface of the material to be milled.

The height of the table is noted down. The table is then raised by a small amount until the correct depth of cut for the material being milled is obtained. The CNC module is then run, and the milling cutter will machine the surface. After each run of the CNC module the table is raised again until, after a few passes, the table has been raised by an amount equal to the Z height of the surface.

Then the table is lowered back to the starting height and the next CNC module is loaded and run. In this way all the CNC modules are used to machine the surface.

APPENDIX I

PROGRAM LISTING OF THE CNC MILLING SYSTEM

This Appendix contains the source code listing of the CNC milling system, including the subroutine libraries. The only library not listed is the commercially available TRUE BASIC 3D Graphics package as this would be an infringement of copyright. However, details of all the subroutine calls made to this library by the milling system are given.

The postprocessor and the communications package are listed separately and can be found in Appendices J, K and L.

The source code of the milling system on the two floppy disks does not include the detailed commentary appearing in this Appendix because it would make the program unnecessarily long and cumbersome. All the program statements are, however, identical to those is this Appendix and all the important subroutines, arrays, variables and program steps have been comprehensively described, which should make it easy for any future user to adapt the system to his requirements.

The exclamation mark is used to identify a line, or part of a line, as a comment rather than a TRUE BASIC program statement.

```
PROGRAM : CNCMILL.TRU
PURPOSE : This is the main control program for the entire CNC milling system.
          It does the following:
            1. Reads the surface data file
            2. Calculates the toolpositions
            3. Displays in 3D the surface & toolpath
            4. Creates a toolfile containing the toolpositions
```

The subroutine libraries used by this program are:

1. Begin.lib - initialisation & surface loading
2. Files.lib - file creation & initialisation
3. Math.lib - surface fitting & calculation of the toolpath
4. Display.lib - 3D display and AutoCAD v9 DXF file generation
5. 3Dlib.trc - TRUE BASIC 3D Graphics package. A TRUE BASIC compiled version of this library is used in the milling system.

The arrays used by the system are:

TOOLPTS(1,3) - 1x3 column array for the X,Y and Z coordinates of the toolpositions. It is redimensioned in the DISPLAY library according to the number number of toolpositions required.

AA(4), BB(4), CC(4), DD(4) - for the 4 coefficients (A,B,C,D) of the 4 triangular planes within each quadrilateral element of the surface

KEY(4) - 4 keys used to sort the toolpositions into the correct sequence for milling

XEL(5), YEL(5), ZEL(5) - contains the 4 corner coordinates and the calculated fifth point for each quadrilateral element

XTL(4), YTL(4), ZTL(4) - contains the coordinates of the 4 calculated toolpositions for each of the triangular elements of the surface

XCC(4), YCC(4), ZCC(4) - contains the coordinates of the 4 calculated centroids for the 4 triangular planes within each quadrilateral element

XCoord(43,43), YCoord(43,43), ZCoord(43,43) - contains the coordinates of all the the surface data points, referenced by the Frame and Level. The size of these arrays, arbitrarily set to 43, determines the number of datapoints that can be used to define the surface. The maximum size of these arrays depends on the memory of the computer being used, but for a 640 K machine is 70 x 70.

A list of the important variables used in the system are given below:

alldone$ - flag used to tell system whether or not to create toolfile

Filename$ - contains the name of the surface data file

MAXXCoord - the next 6 variables contain the maximum and minimum values of the
MAXYCoord X,Y and Z coordinates. They are used in the 3D package to define
MAXZCoord the viewing volume according to the size of the surface being
MINXCoord viewed.
MINYCoord
MINZCoord

NumberFrames - the number of frames in the surface
NumberLevels - the number of levels in the surface

toolfile$ - always contains the name of the temporary toolfile "TEMP.TOL"

toolsort$ - contains the name of the final toolfile

TRAD - the diameter in mm of the hemispherical milling tool

XTICKINC - the next 3 variables contain the tick interval for the axes in the
YTICKINC 3D display. There are always 5 ticks per axis.
ZTICKINC

x1,y1,z1 - the X,Y,Z camera position for the 3D display

Each subroutine in the MILLING SYSTEM uses certain variables and arrays which are passed to it via the CALL statement. Each CALL statement therefore has a list of parameters which are used by the subroutine being called. This is useful because the variables affecting each subroutine can be easily identified.

A list of the subroutines called in this section and their purpose is given below:

SCREEN - displays the first 2 title screens, requires no parameters
QUICKINIT - initialises the system only for viewing the surface and toolpath
GETSURFDAT - loads the surface data coordinates from the disk into memory
MAXMINDAT - calculates the maximum and minimum values of the x,y and z used for scaling the 3D display.
DISPLAY - performs the 3D display and the generation of the AUTOCAD files
INIT - a slightly more lengthy initialisation requiring more information from the user.
OPENANDMAKETOOLFILE - opens 2 toolfiles on the disk
GETPOINTS - retrieves the 4 corner points of each quadrilateral element from the array for use by the mathematical procedure
FIFTHPOINT - calculates the 5th point within each quadrilateral element
SURFACE - calculates the plane equations for each of the 4 triangular planes in each of the quadrilateral elements
TOOLPOSITION - using the previously calculated plane equations, this subroutine calculates the x,y, and z coordinates of the toolpositions required to mill the surface
WRITETOOLPOSITION - writes the coordinates of each toolposition to the disk
CLOSETOOLFILE - closes the 2 toolfiles
TOOLFILESORT - sorts the temporary toolfile, "TEMP.TOL" into the final toolfile, by using the keys, into the correct sequence
ENDPROG - calls the end subroutine and displays the appropriate message

```
!
!  The section dimensions the arrays required for
!  the system, however, some arrays which are
!  specific to a particular subroutine are
!  dimensioned in that subroutine.
!
```

```
DIM TOOLPTS(1,3)
DIM AA(4)
DIM BB(4)
DIM CC(4)
DIM DD(4)
DIM KEY(4)
DIM XEL(5)
DIM YEL(5)
DIM ZEL(5)
DIM XTL(4)
DIM YTL(4)
DIM ZTL(4)
DIM XCC(4)
DIM YCC(4)
DIM ZCC(4)
DIM XCoord(43,43)
DIM YCoord(43,43)
DIM ZCoord(43,43)

library "begin.trc"      !the following 5 lines join all subroutine libraries to the main program
library "files.trc"
library "math.trc"
library "display.trc"
library "3dlib.trc"
```

```
!
!  The section below asks the user what type of
!  graphics card is in the machine being used.
!  It is used by TRUE BASIC to set the optimum
!  mode for the graphical display.
!
```

```
CLEAR
PRINT TAB(10,13);" You can run this program on:"
PRINT TAB(12,13);"   IBM Colour Graphics Adaptor ..... PRESS C "
PRINT TAB(13,13);"   IBM Enhanced Graphics Adaptor ... PRESS E "
PRINT TAB(14,13);"   Hercules Graphics Card ......... PRESS H "

GET KEY COMPCARD      ! read the key number from the keyboard

IF COMPCARD=ORD("C") THEN SET MODE "HIRES"
IF COMPCARD=ORD("c") THEN SET MODE "HIRES"
IF COMPCARD=ORD("E") THEN SET MODE "EGAHIRES"
IF COMPCARD=ORD("e") THEN SET MODE "EGAHIRES"
IF COMPCARD=ORD("H") THEN SET MODE "HIRES"
IF COMPCARD=ORD("h") THEN SET MODE "HIRES"
```

```
!
!  The title screens are called and after which
!  the first input screen is displayed.
!
```

```
CALL SCREEN      ! Calls the title screen

CLEAR            ! Clears the display for the first input screen
```

```
PRINT "      ┌──────────────────────────────────────────────────────────┐"
PRINT "      │                      CNC MILLING SYSTEM                  │"
PRINT "      ├──────────────────────────────────────────────────────────┤"
PRINT "      │  If the toolfile already exists you can view the surface and the  │"
PRINT "      │  toolpath in three dimensions without having to recalculate the   │"
PRINT "      │  toolpositions.                                           │"
PRINT "      └──────────────────────────────────────────────────────────┘"
PRINT
PRINT
PRINT

!╔═══════════════════════════════════════════════╗
!║ The section below determines whether the      ║
!║ toolfile exists and the user only wants to use ║
!║ the 3D viewer or whether a toolfile must be   ║
!║ created for the surface.                      ║
!╚═══════════════════════════════════════════════╝

INPUT prompt " Has the Toolfile been previously created (Y/N) ":alldone$

 IF alldone$ = "Y" THEN     ! allows capital or small "y" to
  LET alldone$="y"          ! be used as a response
 END IF

!╔═══════════════════════════════════════════════╗
!║ If the toolfile exists only 3D viewing is     ║
!║ required and therefore only the subroutines   ║
!║ needed for the 3D display are called.         ║
!╚═══════════════════════════════════════════════╝

IF alldone$="y" THEN ,  !only the viewing of an existing toolfile and surface is required

 CALL QUICKINIT (NumberFrames,NumberLevels,toolsort$,Filename$)

 CALL GETSURFDAT (XCoord(,),YCoord(,),ZCoord(,),NumberFrames,NumberLevels,Filename$)

 CALL MAXMINDAT (XCoord(,),YCoord(,),ZCoord(,),MINXCoord,MAXXCoord,MINYCoord,MAXYCoord,MINZCoord,
                 MAXZCoord,NumberFrames,NumberLevels,XTICKINC,YTICKINC,ZTICKINC)

 CALL DISPLAY (XCoord(,),YCoord(,),ZCoord(,),XEL(),YEL(),ZEL(),XCC(),YCC(),ZCC(),XTL(),YTL(),ZTL(),
               MINXCoord,MAXXCoord,MINYCoord,MAXYCoord,MINZCoord,MAXZCoord,x1,y1,z1,XTICKINC,
               YTICKINC,ZTICKINC,NumberFrames,NumberLevels,toolsort$,TOOLPTS(,))

!╔═══════════════════════════════════════════════╗
!║ After the user has finished viewing the surface ║
!║ the program ends.                             ║
!╚═══════════════════════════════════════════════╝

 CALL ENDPROG   !ends the program

END IF

!╔═══════════════════════════════════════════════╗
!║ If the toolfile does not exist a full         ║
!║ calculation of the toolpath is required       ║
!║ before the 3D viewing of the surface          ║
!╚═══════════════════════════════════════════════╝

CALL INIT (NumberFrames,NumberLevels,TRAD,toolfile$,toolsort$,Filename$)

CALL GETSURFDAT (XCoord(,),YCoord(,),ZCoord(,),NumberFrames,NumberLevels,Filename$)

CALL MAXMINDAT (XCoord(,),YCoord(,),ZCoord(,),MINXCoord,MAXXCoord,MINYCoord,MAXYCoord,
                MINZCoord,MAXZCoord,NumberFrames,NumberLevels,XTICKINC,YTICKINC,ZTICKINC)

CALL OPENANDMAKETOOLFILE (#5,#6,toolfile$,toolsort$)
```

```
CLEAR
PRINT "
PRINT "      ┌──────────────────────────────────────────────────────────────┐"
PRINT "      │                        CNC MILLING SYSTEM                    │"
PRINT "      ├──────────────────────────────────────────────────────────────┤"
PRINT "      │                                                              │"
PRINT "      │          Calculating the Toolpath for Surface ";Filename$;
PRINT TAB(5,74);"|"
PRINT "      │                                                              │"
PRINT "      │                                                              │"
PRINT "      │                                                              │"
PRINT "      │                                                              │"
PRINT "      │                                                              │"
PRINT "      │                                                              │"
PRINT "      └──────────────────────────────────────────────────────────────┘"
```

The main calculating loop which retrieves each surface data point and calculates the toolpositions for each quadrilateral element.

```
FOR Frame = 1 TO NumberFrames-1
  FOR Level = 1 TO NumberLevels-1

    Print TAB (8,18); "Calculating Frame ";Frame; "and Level ";Level

    CALL GETPOINTS(Frame,Level,XEL(),YEL(),ZEL(),XCoord(,),YCoord(,),ZCoord(,))

    CALL FIFTHPOINT(XEL(),YEL(),ZEL())

    CALL SURFACE (Frame,Level,XEL(),YEL(),ZEL(),AA(),BB(),CC(),DD(),KEY(),XCC(),YCC(),ZCC(),
                  XTL(),YTL(),ZTL(),NumberLevels,NumberFrames)

    CALL TOOLPOSITION(KEY(),AA(),BB(),CC(),DD(),XCC(),YCC(),ZCC(),XTL(),YTL(),ZTL(),TRAD)

    CALL WRITETOOLPOSITION(#5,#6,KEY(),XTL(),YTL(),ZTL())

  NEXT LEVEL
NEXT FRAME
```

The temporary toolfile has been created, the program now sorts the temporary toolfile, using the keys, into the final toolfile.

```
CALL CLOSETOOLFILE(#5,#6)

CALL TOOLFILESORT(toolfile$,toolsort$,NumberFrames,NumberLevels)
```

Once the toolfile has been sorted the three dimensional display allows the user to verify that the toolpath and surface are correct.

```
CALL DISPLAY(XCoord(,),YCoord(,),ZCoord(,),XEL(),YEL(),ZEL(),XCC(),YCC(),ZCC(),XTL(),YTL(),ZTL(),
             MINXCoord,MAXXCoord,MINYCoord,MAXYCoord,MINZCoord,MAXZCoord,x1,y1,z1,XTICKINC,YTICKINC,
             ZTICKINC,NumberFrames,NumberLevels,toolsort$,Toolpts(,))

CALL ENDPROG
```

After the user has finished viewing the surface the program ends.

```
---------------------------------- ENDPROG SUBROUTINE -------------------------------

SUB ENDPROG
CLEAR

IF alldone$="Y" THEN
 ELSEIF alldone$="y" THEN

PRINT "       ┌──────────────────────────────────────────────────────────────┐"
PRINT "       │                        CNC MILLING SYSTEM                    │"
PRINT "       ├──────────────────────────────────────────────────────────────┤"
PRINT "       │                    Program terminated normally.              │"
PRINT "       └──────────────────────────────────────────────────────────────┘"

PAUSE 5
STOP
END IF
CLEAR
PRINT "       ┌──────────────────────────────────────────────────────────────┐"
PRINT "       │                        CNC MILLING SYSTEM                    │"
PRINT "       ├──────────────────────────────────────────────────────────────┤"
PRINT "       │                                                              │"
PRINT "       │  The program has ended normally, and the toolfile ";toolsort$;
PRINT TAB(5,75);"|"
PRINT "       │  has been successfully created.                              │"
PRINT "       │                                                              │"
PRINT "       └──────────────────────────────────────────────────────────────┘"

PAUSE 10

STOP

END SUB

-------------------------------------------*-------------------------------------------

END     ! END OF THE PROGRAM CNCMILL
```

```
LIBRARY : BEGIN.LIB
PURPOSE : This subroutine library contains the initialisation and other utility
          subroutines.
          The subroutines contained in this library are:
            1. INIT       - initialises variables used if the toolpath is to be
                            calculated
            2. QUICKINIT  - a shorter version of INIT used when only 3D viewing is
                            required
            3. GETSURFDAT - loads the surface data from disk into the memory of the
                            IBM PC.
            4. MAXMINDAT  - determines the maximum and minimum values of the X,Y and
                            Z coordinates which are used to define the 3D viewing
                            volume. Also the tick interval for the axes in the 3D
                            display are calculated.
            5. SCREEN     - the introductory screens are displayed.

EXTERNAL   !tells TRUE BASIC that this is a library outside of the main program

---------------------------------- INIT SUBROUTINE --------------------------------

SUB INIT (NumberFrames,NumberLevels,TRAD,toolfile$,toolsort$,Filename$)   !If toolfile to be
                                                                          !calculated this
                                                                          !subroutine is used
CLEAR
PRINT "      ┌──────────────────────────────────────────────────────────────┐"
PRINT "      │                      CNC MILLING SYSTEM                      │"
PRINT "      ├──────────────────────────────────────────────────────────────┤"
PRINT "      │  You need to input:                                          │"
PRINT "      │     1. The number of Frames used in the surface data         │"
PRINT "      │     2. The number of Levels used in the surface data         │"
PRINT "      │     3. The Diameter of the Spherically Ended cutter to be used  │"
PRINT "      │        to machine the surface.                               │"
PRINT "      │     4. The name of the Toolfile to be created by the system  │"
PRINT "      │     5. The name of the disk file containing the surface data.  │"
PRINT "      └──────────────────────────────────────────────────────────────┘"
PRINT
PRINT
PRINT
INPUT PROMPT "              1. Number of Frames of Data. ":NumberFrames
PRINT
INPUT PROMPT "              2. Number of Levels of Data. ":NumberLevels
PRINT
INPUT PROMPT "              3. Diameter of Spherical Ended Cutter. ":TRAD
PRINT
INPUT PROMPT "              4. Name of the Toolfile to be created. ":toolsort$
PRINT
INPUT PROMPT "              5. Name of the surface data disk file.  ":Filename$

LET toolfile$="TEMP.TOL"   ! always the name of the Temporary toolfile
SOUND 2000,0.1
CLEAR
PRINT TAB (12,8);".    Please note: The Milling Program uses a temporary toolfile"
PRINT TAB (13,8);"                  which is called TEMP.TOL. This file may be "
PRINT TAB (14,8);"                  deleted once the program has finished."
PAUSE 3

END SUB
```

```
-------------------------------- QUICKINIT SUBROUTINE -----------------------------

SUB QUICKINIT(NumberFrames,NumberLevels,toolsort$,Filename$)   !If toolfile already exists
                                                               !this subroutine is used.
CLEAR
PRINT "      ┌──────────────────────────────────────────────────────────────┐"
PRINT "      │                        CNC MILLING SYSTEM                    │"
PRINT "      ├──────────────────────────────────────────────────────────────┤"
PRINT "      │  You need to input:                                          │"
PRINT "      │     1. The number of Frames used in the surface data         │"
PRINT "      │     2. The number of Levels used in the surface data         │"
PRINT "      │     3. The name of the Toolfile to be viewed                 │"
PRINT "      │     4. The name of the disk file containing the surface data │"
PRINT "      └──────────────────────────────────────────────────────────────┘"
PRINT
PRINT
PRINT

INPUT PROMPT "              1. Number of Frames of Data ":NumberFrames
PRINT
INPUT PROMPT "              2. Number of Levels of Data ":NumberLevels
PRINT
INPUT PROMPT "              3. Name of the Toolfile to be viewed ":toolsort$
PRINT
INPUT PROMPT "              4. Name of the surface data disk file  ":Filename$
PRINT
PRINT

END SUB

-------------------------------- GETSURFDAT SUBROUTINE ----------------------------

SUB GETSURFDAT(XCoord(,),YCoord(,),ZCoord(,),NumberFrames,NumberLevels,Filename$)

DO
LET ERR=0          !clears the error flag before beginning
WHEN ERROR IN      !If the user inputs an invalid filename, or the frames & levels are wrong the
                   !program reports an error and allows the user to correct it before continuing.

OPEN #1:NAME Filename$,Create Newold,Organization Record
CLEAR
PRINT TAB(10,13);" Loading the Surface Data .... please be patient "
SOUND 1000,0.1

 FOR Frame=1 to NumberFrames
   FOR Level=1 to NumberLevels
     READ #1: Frame
     READ #1: Level
     READ #1: Xcoord[Frame,Level]
     READ #1: Ycoord[Frame,Level]
     READ #1: Zcoord[Frame,Level]
   NEXT Level
 NEXT Frame

 CLOSE #1  !closes the file after reading it into memory

USE   !If there is an error the following procedure is used
CLEAR
SOUND 1000,0.2
PRINT "      ┌──────────────────────────────────────────────────────────────┐"
PRINT "      │         ERROR .... ERROR .... ERROR .... ERROR .... ERROR    │"
PRINT "      ├──────────────────────────────────────────────────────────────┤"
PRINT "      │                                                              │"
PRINT "      │   THE PROBLEM IS LIKELY TO BE ONE OF THE FOLLOWING           │"
PRINT "      │                                                              │"
PRINT "      │       1. THE DATA FILE DOES NOT EXIST IN THE DIRECTORY ON THE│"
PRINT "      │          DISK                                                │"
PRINT "      │       2. THE NUMBER OF FRAMES AND LEVELS IS INCORRECT WITH   │"
PRINT "      │          RESPECT TO THE FILE                                 │"
PRINT "      │                                                              │"
PRINT "      │   THE SOLUTION IS :                                          │"
PRINT "      │                                                              │"
PRINT "      │       1. REDO THE PROCEDURE, MAKING SURE THAT THE            │"
PRINT "      │          SURFACE DATA FILENAME IS CORRECT.                   │"
PRINT "      │       2. MAKE SURE THAT THE NUMBER OF FRAMES AND LEVELS      │"
PRINT "      │          SPECIFIED IS CORRECT WITH RESPECT TO THE DATA FILE   │"
PRINT "      │          YOU ARE TRYING TO USE.                              │"
PRINT "      └──────────────────────────────────────────────────────────────┘"
PRINT
```

```
INPUT PROMPT "              Name of the surface data disk file  ":Filename$  !this allows the user
INPUT PROMPT "              Enter the Number of Frames  ":NumberFrames       !to re-enter the data
INPUT PROMPT "              Enter the Number of Levels  ":NumberLevels

LET ERR = 1   !sets the error flag to loop again
CLOSE #1
END WHEN

LOOP Until Err = 0    !end the loop if the error flag is clear

END SUB

--------------------------------- MAXMINDAT SUBROUTINE -----------------------------

SUB MAXMINDAT (XCoord(,),YCoord(,),ZCoord(,),MINXCoord,MAXXCoord,MINYCoord,MAXYCoord,MINZCoord,
               MAXZCoord,NumberFrames,NumberLevels,XTICKINC,YTICKINC,ZTICKINC)
LET MAXXCoord = XCoord(1,1)
LET MAXYCoord = YCoord(1,1)
LET MAXZCoord = ZCoord(1,1)
LET MINXCoord = XCoord(1,1)
LET MINYCoord = YCoord(1,1)
LET MINZCoord = ZCoord(1,1)

 FOR I = 1 TO NumberFrames           !scan through the data and store the maximum and minimum
   FOR J = 1 TO NumberLevels         !values of the X,Y and Z coordinates

   IF Xcoord(I,J) > MAXXCoord THEN LET MAXXCoord = XCoord(I,J)
   IF Ycoord(I,J) > MAXYCoord THEN LET MAXYCoord = YCoord(I,J)
   IF Zcoord(I,J) > MAXZCoord THEN LET MAXZCoord = ZCoord(I,J)
   IF XCoord(I,J) < MINXCoord THEN LET MINXCoord = XCoord(I,J)
   IF YCoord(I,J) < MINYCoord THEN LET MINYCoord = YCoord(I,J)
   IF ZCoord(I,J) < MINZCoord THEN LET MINZCoord = ZCoord(I,J)

 NEXT J
NEXT I

  LET XTICKINC = (MAXXCoord+5-MINXCoord)/5  !always 5 Ticks for each axis
  LET YTICKINC = (MAXYCoord+5-MINYCoord)/5
  LET ZTICKINC = (MAXZCoord+5-MINZCoord)/5

END SUB

---------------------------------- SCREEN SUBROUTINE ------------------------------

SUB SCREEN

CLEAR
PRINT
PRINT
PRINT
PRINT "          ╔═══════════════════════════════════════════════════════════╗"
PRINT "          ╟                    CNC MILLING PROGRAM                    ╢"
PRINT "          ╟                                                           ╢"
PRINT "          ╟                                                           ╢"
PRINT "          ╟     MILLING PROGRAM FOR THE MILLING OF COMPLEX THREE      ╢"
PRINT "          ╟     DIMENSIONAL SURFACE ON A THREE AXIS CNC MACHINE       ╢"
PRINT "          ╟                                                           ╢"
PRINT "          ╟                                                           ╢"
PRINT "          ╟     WRITTEN AS PART OF AN MSC IN ENGINEERING DEGREE       ╢"
PRINT "          ╟                                                           ╢"
PRINT "          ╟                 AUTHOR : D.R. BACK                        ╢"
PRINT "          ╟                                                           ╢"
PRINT "          ╟                                                           ╢"
PRINT "          ╟           <PRESS SPACE BAR TO CONTINUE>                   ╢"
PRINT "          ╟                                                           ╢"
PRINT "          ╚═══════════════════════════════════════════════════════════╝"

  DO
   GET KEY TEMPP
  LOOP UNTIL TEMPP=32   !display the message until the spacebar is pressed.

CLEAR
```

```
PRINT "╔═══════════════════════════════════════════════════════════════════════╗"
PRINT "║                          CNC MILLING PROGRAM                          ║"
PRINT "╠═══════════════════════════════════════════════════════════════════════╣"
PRINT "║  This program was written for the Mechanical Engineering Department, U.C.T. ║"
PRINT "║  to enable complex three dimensional surfaces to be machined. The surface  ║"
PRINT "║  to be machined is represented by a series of surface data points which are ║"
PRINT "║  fed into the milling program which then calculates the toolpath to mill   ║"
PRINT "║  the surface. The program has a three dimensional graphical display        ║"
PRINT "║  capability for verification of the surface and the toolpath.              ║"
PRINT "║                                                                           ║"
PRINT "║  Before running the program:                                              ║"
PRINT "║   1. Make sure that there is a surface data file on the disk.             ║"
PRINT "║   2. Make sure that there is sufficient space on the disk for the         ║"
PRINT "║      toolfiles.                                                           ║"
PRINT "║                                                                           ║"
PRINT "║  The calculation of the toolpath for typical 20 x 20 element surface takes ║"
PRINT "║  approximately 30 minutes depending on the speed of the computer. If the   ║"
PRINT "║  toolfile already exists however you can use the three dimensional display ║"
PRINT "║  to view the surface and toolpath.                                        ║"
PRINT "║                                                                           ║"
PRINT "║                        PRESS SPACE BAR TO CONTINUE                        ║"
PRINT "║                                                                           ║"
PRINT "╚═══════════════════════════════════════════════════════════════════════╝"

DO
 GET KEY TEMPP
LOOP UNTIL TEMPP=32   !display screen until spacebar is pressed

END SUB
```

```
LIBRARY : FILES.LIB
PURPOSE : This subroutine library contains the file utilities, such as the creation
          of the toolfile on the disk and the sorting of the toolfile according to
          keys.
          The subroutines contained in this library are:
            1. OPENANDMAKETOOFILE - opens the temporary and the final toolfiles
                                    on disk
            2. WRITETOOLPOSITION  - for each triangular plane in each quadrilateral
                                    element the X,Y and Z coordinate of the
                                    toolposition is written to both the temporary as
                                    well as the final toolfile. This ensures that
                                    they are identical which facilitates the sorting
                                    procedure.
            3. TOOLFILESORT       - sorts the temporary toolfile "TEMP.TOL" into
                                    the final toolfile according to the keys
            4. CLOSETOOLFILE      - closes both the toolfiles on the disk.

EXTERNAL

--------------------------- OPENANDMAKETOOLFILE SUBROUTINE -------------------------

SUB OPENANDMAKETOOLFILE(#5,#6,toolfile$,toolsort$)

   OPEN #5:Name toolfile$,Create NewOld,Organization Record
   Erase #5                                    !Erases the contents of the file and sets the
   Set #5:Recsize 20                           !pointer to the beginning of the file

   OPEN #6:Name toolsort$,Create NewOld,Organization Record
   Erase #6
   Set #6:Recsize 20

END SUB

---------------------------- WRITETOOLPOSITION SUBROUTINE ---------------------------

SUB WRITETOOLPOSITION (#5,#6,KEY(),XTL(),YTL(),ZTL())

 FOR Triangle = 1 to 4                       !For the 4 triangular planes within the
                                             !quadrilateral, write the toolposition to both
   WRITE #5:Key(Triangle)                    !the temporary toolfile and the final toolfile
   WRITE #5:Xtl(Triangle)
   WRITE #5:Ytl(Triangle)
   WRITE #5:Ztl(Triangle)

   WRITE #6:Key(Triangle)
   WRITE #6:Xtl(Triangle)
   WRITE #6:Ytl(Triangle)
   WRITE #6:Ztl(Triangle)

 NEXT Triangle

END SUB

------------------------------ TOOLFILESORT SUBROUTINE ------------------------------

SUB TOOLFILESORT (toolfile$,toolsort$,NumberFrames,NumberLevels)

   OPEN #1:Name toolfile$,Create NewOld,Organization Record
   OPEN #2:Name toolsort$,Create NewOld,Organization Record
CLEAR

Reset #1:Begin                               !Reset both the temporary and the
Reset #2:Begin                               !final toolfiles to the beginning
```

```
Print "Sorting .. ";toolfile$;" into ";toolsort$
 LET Numtoolpos = (NumberFrames-1)*(NumberLevels-1)*4   !Calculate how many toolpositions there are
                                                        !& use it do determine how many repetitions
                                                        !of the sorting loop there must be

  FOR Numrecs = 1 to Numtoolpos

   READ #1:KKey                    !read from the temporary toolfile
   READ #1:XXtl
   READ #1:YYtl
   READ #1:ZZtl

   LET FilePos = KKey*3-2          !calculate the position in the final toolfile of the current
                                   !toolposition based on the key. The fileposition = key * 3
                                   !(for the X,Y, and Z) - 2 (to begin at 1)
   SET #2:Record Filepos           !set the filepointer to the position calculated and...
   WRITE #2:XXtl,YYtl,ZZtl         !overwrite the existing data with the sorted data.

 NEXT NumRecs                      !repeat until all toolpositions have been written.

PRINT "Closing Files ..."
CLOSE #1
CLOSE #2

PRINT "Created Sorted Toolfile ";toolsort$      !inform the user that the final
                                                !toolfile has been created

END SUB

------------------------------ CLOSETOOLFILE SUBROUTINE -----------------------------

SUB CLOSETOOLFILE (#5,#6)

 CLOSE #5
 CLOSE #6

END SUB

-------------------------------------------*-------------------------------------------
```

```
LIBRARY : MATH.LIB
PURPOSE : This subroutine library contains the mathematical routines to calculate
          plane equations and the centroids for each of the 4 triangular planes
          within each quadrilateral element of the surface. The results of these
          calculations are used to calculate the 4 toolpositions for each
          quadrilateral element.
          The subroutines contained in this library are:
            1. SURFACE      - calculates the plane coefficients A,B,C and D for each
                              of the triangular planes in each quadrilateral element.
                              It makes use of two other subroutines, CENTROID and
                              SETUPDET.

            2. CENTROID     - calculates the centroid for each triangular plane
            3. SETUPDET     - sets up the determinant used to calculate the plane
                              equation coefficients
            4. FIFTHPOINT   - determines the 5th point in the quadrilateral element
            5. TOOLPOSITION - determines the X,Y and Z coordinates of the
                              toolposition using the normal vector through the
                              centroid of each triangular plane.
            6. GETPOINTS    - retrieves the 4 X,Y and Z corner coordinates of each
                              quadrilateral element from the appropriate arrays
                              stored in the memory of the IBM PC computer.
```

```
EXTERNAL

!---------------------------------- SURFACE SUBROUTINE -------------------------------

SUB SURFACE (FRAME,LEVEL,XEL(),YEL(),ZEL(),AA(),BB(),CC(),DD(),KEY(),XCC(),YCC(),ZCC(),
             XTL(),YTL(),ZTL(),NumberLevels,NumberFrames)

LET F = FRAME
LET L = LEVEL
IF REMAINDER (F,2) > 0 THEN LET FRAMEVEN = 0 ELSE LET FRAMEVEN = 1   !determines whether the element
                                                                     !is odd or even

!Coordinate 1
LET PNT1 = 1                                    !number the points for retrieval of the
LET PNT2 = 5                                    !appropriate value from the arrays
LET PNT3 = 4

IF FRAMEVEN=1 THEN LET KEY(1)=((NumberLevels-1)*(F-1)*4+(NumberLevels-1))-(L-1) ELSE LET
KEY(1)=((NumberLevels-(1))*(F-1)*4)+L                                  !calculates the key based on
                                                                       !the frames and level

CALL SETUPDET(PNT1,PNT2,PNT3,XEL(),YEL(),ZEL(),A,B,C,D)
LET AA(1) = A
LET BB(1) = B
LET CC(1) = C
LET DD(1) = D

CALL CENTROID(PNT1,PNT2,PNT3,XEL(),YEL(),ZEL(),XC,YC,ZC)
LET XCC(1) = XC
LET YCC(1) = YC
LET ZCC(1) = ZC

!Coordinate 2
LET PNT1 = 1
LET PNT2 = 2
LET PNT3 = 5

IF FRAMEVEN=1 THEN LET KEY(2)=(F-1)*(NumberLevels-1)*4 + NumberLevels + 2*(L-1) ELSE
LET KEY(2)=(F-1)*(NumberLevels-1)*4 + (NumberLevels-1) +2*(NumberLevels-1)-2*(L-1)

CALL SETUPDET(PNT1,PNT2,PNT3,XEL(),YEL(),ZEL(),A,B,C,D)
LET AA(2) = A
LET BB(2) = B
LET CC(2) = C
LET DD(2) = D
```

```
CALL CENTROID(PNT1,PNT2,PNT3,XEL(),YEL(),ZEL(),XC,YC,ZC)
LET XCC(2) = XC
LET YCC(2) = YC
LET ZCC(2) = ZC

!Coordinate 3
LET PNT1 = 2
LET PNT2 = 3
LET PNT3 = 5

IF FRAMEVEN= 1 THEN LET KEY(3)=((NumberLevels-1)*(F-1)*4)+(NumberLevels-1)-(L-1)+
((NumberLevels-1)*3) ELSE LET KEY(3)=((NumberLevels-1)*(F-1)*4)+L+((NumberLevels-1)*3)

CALL SETUPDET(PNT1,PNT2,PNT3,XEL(),YEL(),ZEL(),A,B,C,D)
LET AA(3) = A
LET BB(3) = B
LET CC(3) = C
LET DD(3) = D

CALL CENTROID(PNT1,PNT2,PNT3,XEL(),YEL(),ZEL(),XC,YC,ZC)
LET XCC(3) = XC
LET YCC(3) = YC
LET ZCC(3) = ZC

!Coordinate 4
LET PNT1 = 3
LET PNT2 = 4
LET PNT3 = 5

IF FRAMEVEN=1 THEN LET KEY(4)=((F-1)*(NumberLevels-1)*4)+NumberLevels+(2*(L-1)+1) ELSE LET
KEY(4)=((F-1)*(NumberLevels-1)*4)+(NumberLevels-1)+(2*(NumberLevels-1))-(2*(L-1))-1

CALL SETUPDET(PNT1,PNT2,PNT3,XEL(),YEL(),ZEL(),A,B,C,D)
LET AA(4) = A
LET BB(4) = B
LET CC(4) = C
LET DD(4) = D

CALL CENTROID(PNT1,PNT2,PNT3,XEL(),YEL(),ZEL(),XC,YC,ZC)
LET XCC(4) = XC
LET YCC(4) = YC
LET ZCC(4) = ZC

END SUB

---------------------------------- CENTROID SUBROUTINE -----------------------------

SUB CENTROID (PNT1,PNT2,PNT3,XEL(),YEL(),ZEL(),XC,YC,ZC)

LET XC = ((XEL(PNT1)+XEL(PNT2)+XEL(PNT3))/3)
LET YC = ((YEL(PNT1)+YEL(PNT2)+YEL(PNT3))/3)
LET ZC = ((ZEL(PNT1)+ZEL(PNT2)+ZEL(PNT3))/3)

END SUB

---------------------------------- SETUPDET SUBROUTINE -----------------------------

SUB SETUPDET(PNT1,PNT2,PNT3,XEL(),YEL(),ZEL(),A,B,C,D)

DIM SURFDET(3,3)                !dimensions a 3x3 matrix to hold the coefficients of the plane
LET SURFDET(1,1) = YEL(PNT1)    !equation and stores them in the appropriate place in the matrix
LET SURFDET(1,2) = ZEL(PNT1)
LET SURFDET(1,3) = 1
LET SURFDET(2,1) = YEL(PNT2)
LET SURFDET(2,2) = ZEL(PNT2)
LET SURFDET(2,3) = 1
LET SURFDET(3,1) = YEL(PNT3)
LET SURFDET(3,2) = ZEL(PNT3)
LET SURFDET(3,3) = 1
LET A = DET (SURFDET)           !TRUE BASIC allows the calculation of the determinant directly.
                                !this statement finds the value of the A coefficient
```

```
LET SURFDET(1,1) = XEL(PNT1)
LET SURFDET(1,2) = ZEL(PNT1)
LET SURFDET(1,3) = 1
LET SURFDET(2,1) = XEL(PNT2)
LET SURFDET(2,2) = ZEL(PNT2)
LET SURFDET(2,3) = 1
LET SURFDET(3,1) = XEL(PNT3)
LET SURFDET(3,2) = ZEL(PNT3)
LET SURFDET(3,3) = 1
LET B = -DET (SURFDET)          !the B coefficient

LET SURFDET(1,1) = XEL(PNT1)
LET SURFDET(1,2) = YEL(PNT1)
LET SURFDET(1,3) = 1
LET SURFDET(2,1) = XEL(PNT2)
LET SURFDET(2,2) = YEL(PNT2)
LET SURFDET(2,3) = 1
LET SURFDET(3,1) = XEL(PNT3)
LET SURFDET(3,2) = YEL(PNT3)
LET SURFDET(3,3) = 1
LET C = DET (SURFDET)           !the C coefficient

LET SURFDET(1,1) = XEL(PNT1)
LET SURFDET(1,2) = YEL(PNT1)
LET SURFDET(1,3) = ZEL(PNT1)
LET SURFDET(2,1) = XEL(PNT2)
LET SURFDET(2,2) = YEL(PNT2)
LET SURFDET(2,3) = ZEL(PNT2)
LET SURFDET(3,1) = XEL(PNT3)
LET SURFDET(3,2) = YEL(PNT3)
LET SURFDET(3,3) = ZEL(PNT3)
LET D = -DET (SURFDET)          !the D coefficient

END SUB

------------------------------ FIFTHPOINT SUBROUTINE -----------------------------

SUB FIFTHPOINT(XEL(),YEL(),ZEL())

DIM EQN(2,2)     !2 x 2 matrix of coefficients of the equation used to determine the shortest
                 !point between the 2 quadrilateral diagonals in 3 space

DIM CONST(2,1)   !2 x 1 matrix of the constant terms of the equation used to determine the
                 !shortest point between the 2 lines in 3 space

DIM TEMP(1,1)    !Temporary matrix used for the inversion of the coefficient matrix

DIM SOLN(2,1)    !Solution matrix of mu and lambda. This gives the scalar multipliers used
                 !to determine the closest points on each line

DIM LINE_1_PNT(1,3)    !Two 1 x 3 matrices which hold the X,Y & Z coordinates
DIM LINE_2_PNT(1,3)    !of the closest points on each of the diagonals

LET EQN(1,1)=(XEL(4)-XEL(2))^2
LET EQN(1,1)=EQN(1,1)+(YEL(4)-YEL(2))^2
LET EQN(1,1)=EQN(1,1)+(ZEL(4)-ZEL(2))^2

LET EQN(1,2)=((XEL(3)-XEL(1))*(XEL(4)-XEL(2)))
LET EQN(1,2)=EQN(1,2)+((YEL(3)-YEL(1))*(YEL(4)-YEL(2)))
LET EQN(1,2)=EQN(1,2)+((ZEL(3)-ZEL(1))*(ZEL(4)-ZEL(2)))
LET EQN(1,2)=-EQN(1,2)

LET EQN(2,1)=((XEL(4)-XEL(2))*(XEL(1)-XEL(3)))
LET EQN(2,1)=EQN(2,1)+((YEL(4)-YEL(2))*(YEL(1)-YEL(3)))
LET EQN(2,1)=EQN(2,1)+((ZEL(4)-ZEL(2))*(ZEL(1)-ZEL(3)))

LET EQN(2,2)=(XEL(1)-XEL(3))^2
LET EQN(2,2)=EQN(2,2)+((YEL(1)-YEL(3))^2)
LET EQN(2,2)=EQN(2,2)+((ZEL(1)-ZEL(3))^2)

LET CONST(1,1)=((XEL(2)-XEL(1))*(XEL(4)-XEL(2)))
LET CONST(1,1)=CONST(1,1)+((YEL(2)-YEL(1))*(YEL(4)-YEL(2)))
LET CONST(1,1)=CONST(1,1)+((ZEL(2)-ZEL(1))*(ZEL(4)-ZEL(2)))
LET CONST(1,1)=-CONST(1,1)
```

```
LET CONST(2,1)=((XEL(2)-XEL(1))*(XEL(1)-XEL(3)))
LET CONST(2,1)=CONST(2,1)+((YEL(2)-YEL(1))*(YEL(1)-YEL(3)))
LET CONST(2,1)=CONST(2,1)+((ZEL(2)-ZEL(1))*(ZEL(1)-ZEL(3)))
LET CONST(2,1)=-CONST(2,1)

MAT TEMP = INV(EQN)                                          !solves for mu and lambda
IF DET(EQN)<0.5 THEN PRINT "WARNING , DETERMINANT NEAR ZERO"  !if the determinant is near to zero
MAT SOLN = TEMP*CONST                                        !the surface is undefined.

LET LINE_1_PNT(1,1) = XEL(1)+(SOLN(2,1)*(XEL(3)-XEL(1)))
LET LINE_1_PNT(1,2) = YEL(1)+(SOLN(2,1)*(YEL(3)-YEL(1)))
LET LINE_1_PNT(1,3) = ZEL(1)+(SOLN(2,1)*(ZEL(3)-ZEL(1)))

LET LINE_2_PNT(1,1) = XEL(2)+(SOLN(1,1)*(XEL(4)-XEL(2)))
LET LINE_2_PNT(1,2) = YEL(2)+(SOLN(1,1)*(YEL(4)-YEL(2)))
LET LINE_2_PNT(1,3) = ZEL(2)+(SOLN(1,1)*(ZEL(4)-ZEL(2)))

LET XEL(5) = (LINE_1_PNT(1,1) + LINE_2_PNT(1,1))/2    !the 5th point is half-way between the two
LET YEL(5) = (LINE_1_PNT(1,2) + LINE_2_PNT(1,2))/2    !closest points on each diagonal
LET ZEL(5) = (LINE_1_PNT(1,3) + LINE_2_PNT(1,3))/2

END SUB

------------------------------ TOOLPOSITION SUBROUTINE ------------------------------

SUB TOOLPOSITION(KEY(),AA(),BB(),CC(),DD(),XCC(),YCC(),ZCC(),XTL(),YTL(),ZTL(),TRAD)

LET TOOLRAD = TRAD/2       !the radius of the cutting tool
FOR PLANE = 1 TO 4
LET KEY(PLANE) = KEY(PLANE)
LET   A = AA(PLANE)
LET   B = BB(PLANE)
LET   C = CC(PLANE)
LET   D = DD(PLANE)
LET  XC = XCC(PLANE)
LET  YC = YCC(PLANE)
LET  ZC = ZCC(PLANE)

LET alpha = A/SQR(A^2+B^2+C^2)
LET beta  = B/SQR(A^2+B^2+C^2)
LET gamma = C/SQR(A^2+B^2+C^2)

LET XTL(PLANE) = XC + TOOLRAD*alpha    !the x,y and z coordinates of the toolposition
LET YTL(PLANE) = YC + TOOLRAD*beta
LET ZTL(PLANE) = ZC + TOOLRAD*gamma
NEXT PLANE

END SUB

--------------------------------- GETPOINTS SUBROUTINE ------------------------------

SUB GETPOINTS(Frame,Level,XEL(),YEL(),ZEL(),XCoord(,),YCoord(,),ZCoord(,))

LET XEL(1)=XCoord(Frame,Level)
LET YEL(1)=YCoord(Frame,Level)
LET ZEL(1)=ZCoord(Frame,Level)

LET XEL(2)=XCoord(Frame+1,Level)
LET YEL(2)=YCoord(Frame+1,Level)
LET ZEL(2)=ZCoord(Frame+1,Level)

LET XEL(3)=XCoord(Frame+1,Level+1)
LET YEL(3)=YCoord(Frame+1,Level+1)
LET ZEL(3)=ZCoord(Frame+1,Level+1)

LET XEL(4)=XCoord(Frame,Level+1)
LET YEL(4)=YCoord(Frame,Level+1)
LET ZEL(4)=ZCoord(Frame,Level+1)

END SUB

-----------------------------------------*-----------------------------------------
```

```
CALL PARAWINDOW (MINXCoord,MAXXCoord+5,MINYCoord,MAXYCoord+5,
                 MINZCoord,MAXZCoord+5,work$)                    !sets up the 3D viewing volume
CALL SETREFPT3 ((MAXXCoord-MINXCoord)/2,(MAXYCoord-MINYCoord)/2,
                0,work$)                                         !sets the camera reference point

DO UNTIL view = 5   !repeat the viewing routine until the user wishes to end the program
  CLEAR
  PRINT "      ┌──────────────────────────────────────────────────────────┐"
  PRINT "      │                    CNC MILLING SYSTEM                    │"
  PRINT "      ├──────────────────────────────────────────────────────────┤"
  PRINT "      │                THREE DIMENSIONAL VIEW SELECTOR           │"
  PRINT "      ├──────────────────────────────────────────────────────────┤"
  PRINT "      │                                                          │"
  PRINT "      │    Three views of the surface are available ;            │"
  PRINT "      │       1 - 2-D view from directly above the surface , ie XY plane  │"
  PRINT "      │       2 - 2-D view from the side of the surface , ie YZ plane     │"
  PRINT "      │       3 - 3-D view from the X Y Z extremity              │"
  PRINT "      │       4 - Specify your own Camera Position               │"
  PRINT "      │       5 - End viewing session                            │"
  PRINT "      │                                                          │"
  PRINT "      │                                                          │"
  PRINT "      │            SELECT VIEW BY PRESSING A NUMBER FROM 1-5     │"
  PRINT "      │                                                          │"
  PRINT "      └──────────────────────────────────────────────────────────┘"
  PRINT
  PRINT

  INPUT prompt "Which view do you want? ":view
  CLEAR
   SELECT CASE view

    Case is = 1                                  !positions the camera directly above the surface
     Let x1 = (MAXXCoord-MINXCoord)/2            !aiming straight down onto the surface
     Let y1 = (MAXYCoord-MINYCoord)/2
     Let z1 =  MAXZCoord+2

   Case is = 2                                   !positions the camera at the side of the surface
    Let x1 = 0                                   !looking directly at the reference point
    Let y1 = (MAXYCoord-MINYCoord)/2
    Let z1 = 0

   Case is = 3                                   !positions the camera at the extremity of
   Let x1 = MAXXCoord                            !the surface looking down onto the surface to
   Let y1 = MAXYCoord                            !give a true 3D representation
   Let z1 = MAXZCoord*2

   Case is = 4
    Input prompt "Give the Camera X position ":x1        !allows the user to enter his
    Input prompt "Give the Camera Y position ":y1        !own camera coordinates
    Input prompt "Give the Camera Z position ":z1

  Case is = 5
   CLEAR
   PRINT " Viewing session ended - goodbye "
   EXIT SUB                                      !this ends the viewing session and returns to
                                                 !CNCMILL.TRU to terminate the program

END SELECT

CALL SETCAMERA3 (x1,y1,z1,work$)                 !this call uses the values of x1,y1 and
                                                 !z1 to actually position the camera
CALL TICKS3 (XTICKINC,YTICKINC,ZTICKINC,work$)   !draws the ticks on the x,y and z axes

   FOR Frame = 1 to NumberFrames-1                !loop to draw each quadrilateral element
     FOR Level = 1 to NumberLevels-1
      PLOT                                          !turns the drawing "beam" off
      CALL PLOTON3 (XCoord[Frame,Level],YCoord[Frame,Level],
                    ZCoord[Frame,Level],work$)      !move to X,Y,Z point of element
                                                    !without drawing line

      IF PLOTFLAG$="Y" THEN                         !if the AUTOCAD file is created then
                                                    !the coordinates of the edge are also
                                                    !written to the DXF file

      PRINT #9:"0"
      PRINT #9:"3DLINE"                             !tells AUTOCAD to expect a 3D line
      PRINT #9:"8"
      PRINT #9:"0"
```

```
PRINT #9:"10"
LET TEMP$=STR$(XCoord[Frame,Level])                    !write the first X coordinate
PRINT #9:TEMP$

PRINT #9:"20"
LET TEMP$=STR$(YCoord[Frame,Level])                    !write the first Y coordinate
PRINT #9:TEMP$

PRINT #9:"30"
LET TEMP$=STR$(ZCoord[Frame,Level])                    !write the first Z coordinate
PRINT #9:TEMP$

END IF

CALL PLOTON3 (XCoord[Frame+1,Level],YCoord[Frame+1,Level],
              ZCoord[Frame+1,Level],work$)             !draw line to second X,Y,Z point
IF PLOTFLAG$="Y" THEN                                  !of element
PRINT #9:"11"
LET TEMP$=STR$(XCoord[Frame+1,Level])                  !write the second X coordinate
PRINT #9:TEMP$

PRINT #9:"21"
LET TEMP$=STR$(YCoord[Frame+1,Level])                  !write the second Y coordinate
PRINT #9:TEMP$

PRINT #9:"31"
LET TEMP$=STR$(ZCoord[Frame+1,Level])                  !write the second Z coordinate
PRINT #9:TEMP$

PRINT #9:"0"
PRINT #9:"3DLINE"                                      !let the first coordinates of the
PRINT #9:"8"                                           !next line be the second coordinates
PRINT #9:"0"                                           !of the first line, overlapping of
                                                       !points is required by AUTOCAD as
                                                       !each line is drawn separately
PRINT #9:"10"
LET TEMP$=STR$(XCoord[Frame+1,Level])
PRINT #9:TEMP$

PRINT #9:"20"
LET TEMP$=STR$(YCoord[Frame+1,Level])
PRINT #9:TEMP$

PRINT #9:"30"
LET TEMP$=STR$(ZCoord[Frame+1,Level])
PRINT #9:TEMP$
END IF

CALL PLOTON3 (XCoord[Frame+1,Level+1],YCoord[Frame+1,Level+1],
              ZCoord[Frame+1,Level+1],work$)           !draw line to third X,Y,Z point
IF PLOTFLAG$="Y" THEN
PRINT #9:"11"
LET TEMP$=STR$(XCoord[Frame+1,Level+1])
PRINT #9:TEMP$

PRINT #9:"21"
LET TEMP$=STR$(YCoord[Frame+1,Level+1])
PRINT #9:TEMP$

PRINT #9:"31"
LET TEMP$=STR$(ZCoord[Frame+1,Level+1])
PRINT #9:TEMP$

PRINT #9:"0"
PRINT #9:"3DLINE"
PRINT #9:"8"
PRINT #9:"0"

PRINT #9:"10"
LET TEMP$=STR$(XCoord[Frame+1,Level+1])
PRINT #9:TEMP$

PRINT #9:"20"
LET TEMP$=STR$(YCoord[Frame+1,Level+1])
PRINT #9:TEMP$
```

```
      PRINT #9:"30"
      LET TEMP$=STR$(ZCoord[Frame+1,Level+1])
      PRINT #9:TEMP$

      END IF

      CALL PLOTON3 (XCoord[Frame,Level+1],YCoord[Frame,Level+1],
                    ZCoord[Frame,Level+1],work$)            !draw line to fourth X,Y,Z point
      IF PLOTFLAG$="Y" THEN
      PRINT #9:"11"
      LET TEMP$=STR$(XCoord[Frame,Level+1])
      PRINT #9:TEMP$

      PRINT #9:"21"
      LET TEMP$=STR$(YCoord[Frame,Level+1])
      PRINT #9:TEMP$

      PRINT #9:"31"
      LET TEMP$=STR$(ZCoord[Frame,Level+1])
      PRINT #9:TEMP$

      PRINT #9:"0"
      PRINT #9:"3DLINE"
      PRINT #9:"8"
      PRINT #9:"0"

      PRINT #9:"10"
      LET TEMP$=STR$(XCoord[Frame,Level+1])
      PRINT #9:TEMP$

      PRINT #9:"20"
      LET TEMP$=STR$(YCoord[Frame,Level+1])
      PRINT #9:TEMP$

      PRINT #9:"30"
      LET TEMP$=STR$(ZCoord[Frame,Level+1])
      PRINT #9:TEMP$

      END IF

      CALL PLOTON3 (XCoord[Frame,Level],YCoord[Frame,Level],
                    ZCoord[Frame,Level],work$)              !draw last line of element
      IF PLOTFLAG$="Y" THEN
      PRINT #9:"11"
      LET TEMP$=STR$(XCoord[Frame,Level])
      PRINT #9:TEMP$

      PRINT #9:"21"
      LET TEMP$=STR$(YCoord[Frame,Level])
      PRINT #9:TEMP$

      PRINT #9:"31"
      LET TEMP$=STR$(ZCoord[Frame,Level])
      PRINT #9:TEMP$
      END IF

      PLOT                              !turn beam off and go to next element
      NEXT Level                        !repeat for all the surface elements
    NEXT Frame

    IF PLOTFLAG$="Y" THEN               !after loop write the ending of the DXF
     PRINT #9:"0"                       !file as specified by AUTOCAD
     PRINT #9:"ENDSEC"
     PRINT #9:"0"
     PRINT #9:"EOF"
     CLOSE #9

    END IF

DO
 LET Err = 0                            !clears the error flag allowing the user to re-enter the
   WHEN ERROR IN                        !filename should there be a mistake

   OPEN #1:Name toolsort$,Create NewOld,Organization Record
   RESET #1:BEGIN
   LET Numtoolposs = ((NumberFrames-1)*(NumberLevels-1)*4)

   MAT READ #1:ToolPts(Numtoolposs,3)       !TRUE BASIC statement which reads the final toolfile
   CLOSE #1                                 !AND dynamically redimensions the ToolPTS array to
                                            !the correct length
```

```
USE                                          !use the following procedure should there be an error

 CLOSE #1
 SOUND 1000,0.2                              !beep to warn user of error

 IF Extype = 5000 THEN                       !this error type occurs if the surface is too
                                             !large for the memory capacity of the IBM PC

   PRINT " Out of Memory , you can't Display this surface and toolpath "
   PRINT " You cannot display a surface with greater than 70x70 elements"
   PRINT " Try reducing the size of the arrays declared in the CNCMILL program"

   ELSE

   PRINT "ERROR .. toolfile doesn't exist on the disk."   !alternatively the toolfile may not be
   PRINT " "                                              !be correctly specified

   INPUT PROMPT "Give the correct name of the toolfile please ":toolsort$

 END IF

LET Err = 1                                               !allow user to re-enter data
END WHEN

LOOP Until Err = 0                                        !exit loop if no error found

 IF PLOTFLAG$="Y" THEN                                    !if AUTOCAD DXF file required then also
                                                          !write the corresponding toolfile
   FOR acadtoolpos = 1 TO NumtoolPoss-1                   !write all the toolpositions

    PRINT #10:"0"
    PRINT #10:"3DLINE"
    PRINT #10:"8"
    PRINT #10:"0"

    PRINT #10:"10"                                !write X coordinate of start of line segment
    LET Temp$=Str$(ToolPts(acadtoolpos,1))
    PRINT #10:Temp$

    PRINT #10:"20"                                !write Y coordinate of start of line segment
    LET Temp$=Str$(ToolPts(acadtoolpos,2))
    PRINT #10:Temp$

    PRINT #10:"30"                                !write Z coordinate of start of line segment
    LET Temp$=Str$(ToolPts(acadtoolpos,3))
    PRINT #10:Temp$

    LET Endline=acadtoolpos+1

    PRINT #10:"11"                                !write X coordinate of end of line segment
    LET Temp$=Str$(ToolPts(Endline,1))
    PRINT #10:Temp$

    PRINT #10:"21"                                !write Y coordinate of end of line segment
    LET Temp$=Str$(ToolPts(Endline,2))
    PRINT #10:Temp$

    PRINT #10:"31"                                !write Z coordinate of end of line segment
    LET Temp$=Str$(ToolPts(Endline,3))
    PRINT #10:Temp$

   Next acadtoolpos                          !repeat until all the toolpositions have been written

    PRINT #10:"0"                            !the next 4 items specify the end of the DXF file
    PRINT #10:"ENDSEC"
    PRINT #10:"0"
    PRINT #10:"EOF"
    CLOSE #10

   END IF

SET COLOR "Red"                              !sets the toolpath to red for colour systems
PLOT
CALL MATLINES3 (ToolPts(,),work$)            !TRUE BASIC has ONE matrix statement to plot the 3D
PLOT                                         !line segments comprising the toolpath
SET COLOR "Green"
PRINT TAB(23,20);"Press <SPACE BAR> to Select Next View"
```

```
    DO                          !allow the user to look at the view on the screen until spacebar pressed
      GET KEY TEMPKEY
    LOOP until TEMPKEY=32

   LOOP

END SUB

-------------------------------------------*-------------------------------------------
```

APPENDIX J

PROGRAM LISTING OF THE POSTPROCESSOR

This Appendix contains the listing of the postprocessor used to convert the toolfile into a series of CNC modules.

PROGRAM : POSTPROC.TRU
PURPOSE : The postprocessor function is to convert the toolfile into a series of 7800 byte CNC modules which can be loaded into the memory of the Bridgeport CNC computer.
The postprocessor does the following:

1. Reads the X,Y and Z coordinates of each toolposition from the toolfile
2. Constructs syntactically correct commands for the CNC machine using these coordinates.
3. Writes these commands to a disk file from where they can be transferred into the memory of the CNC computer

There are two types of subroutines in this program, control and utility subroutines The control subroutines provide the overall program logic while the utility subroutines do the work of assembling each CNC command into a string ready for writing to the CNC module.

Calls are made to the following control subroutines:

- INITIAL - displays the introductory screen and initialises the required variables. It also assembles the name of each CNC module and opens each module file on the disk
- FLINIT - sets the FEEDRATE and writes the MOVERAPIDLY section of each CNC module
- SURFMILL - this is the main part of the postprocessor which writes all the toolpositions to the CNC module files
- TERMINATE - after all the CNC modules have been created the program ends by using this subroutine

Calls are made to the following utility subroutines:

- START - writes the starting string to each CNC module which puts the CNC machine into absolute and metric modes. It also tell the CNC where the toolchange position is located.
- MOVERAPIDLY - writes the command causing the machine to move quickly to a position above the first milling point
- FEED - writes the command which sets the correct feedrate and uses the machines linear interpolation mode to move the cutter to the next toolposition.
- COORDINATES - this utility subroutine is used by both the MOVERAPIDLY routine and the FEED routine. Its function is to write the coordinates of each toolposition in the correct format.
- CHANGETAPE - this routine is used to end one CNC module when its length exceeds 7800 bytes and begin the next CNC module
- FINISH - writes the last line of each CNC module which makes the machine move to the origin on the surface

This is the main control section of the program providing the overall logic flow. As most of the "work" is performed in the utility subroutines this section is brief.

```
CALL INITIAL
CALL FLINIT
CALL SURFMILL
CALL TERMINATE

---------------------------------- INITIAL SUBROUTINE ----------------------------------

SUB INITIAL

CLEAR
PRINT "      ┌──────────────────────────────────────────────────────────────┐"
PRINT "      │                      CNC MILLING SYSTEM                      │"
PRINT "      ├──────────────────────────────────────────────────────────────┤"
PRINT "      │     POSTPROCESSOR FOR CONVERSION OF TOOLFILE INTO CNC PROGRAMS │"
PRINT "      ├──────────────────────────────────────────────────────────────┤"
PRINT "      │  You need to input the following information:                │"
PRINT "      │                                                              │"
PRINT "      │            1. Name of the toolfile to convert                │"
PRINT "      │            2. Number of Frames in the surface                │"
PRINT "      │            3. Number of Levels in the surface                │"
PRINT "      │            4. Name of the CNC program files                  │"
PRINT "      │                                                              │"
PRINT "      │  The Bridgeport CNC files are number consecutively and they all │"
PRINT "      │  have the file extension .PRG to identify them on the disk   │"
PRINT "      │                                                              │"
PRINT "      └──────────────────────────────────────────────────────────────┘"
PRINT
INPUT PROMPT "  Name of toolfile to convert ":toolsort$
PRINT
INPUT PROMPT "  Number of Frames in the surface ":Numtlflframes
PRINT
INPUT PROMPT "  Number of Levels in Toolfile ":Numtlfllevels
PRINT
INPUT PROMPT "  Name of the Bridgeport CNC program file to create (max 5 letters) ":progname$


IF LEN(progname$) > 5 THEN                !make sure that the CNC module name given by the user is
  SOUND 1000,0.2                          !less than 5 characters long
  PRINT " THE LENGTH OF THE FILENAME IS TOO LONG PLEASE SHORTEN "
  INPUT PROMPT "  Name of the Bridgeport CNC program file to create (max 5 letters) ":progname$
END IF

LET cncflno = 1
LET cncfile$ = progname$ & STR$(cncflno) & ".PRG"    !assembles the filename with the .PRG extension

OPEN #1:NAME cncfile$,CREATE NEWOLD,ORGANIZATION TEXT     !creates the first CNC module
OPEN #2:NAME toolsort$,CREATE NEWOLD,ORGANIZATION RECORD  !opens the final toolfile
CLEAR
PRINT "      ┌──────────────────────────────────────────────────────────────┐"
PRINT "      │                      CNC MILLING SYSTEM                      │"
PRINT "      ├──────────────────────────────────────────────────────────────┤"
PRINT "      │     POSTPROCESSOR FOR CONVERSION OF TOOLFILE INTO CNC PROGRAMS │"
PRINT "      ├──────────────────────────────────────────────────────────────┤"
PRINT "      │                                                              │"
PRINT "      │  Creating the Bridgeport CNC module ";cncfile$;" ... please wait"
PRINT TAB (7,74);"│"
PRINT "      │                                                              │"
PRINT "      └──────────────────────────────────────────────────────────────┘"

LET Numtoolposs = ((Numtlflframes-1)*(Numtlfllevels-1)*4)-1  !determine the number of tool positions

END SUB

---------------------------------- FLINIT SUBROUTINE ----------------------------------

SUB FLINIT

ERASE #1
LET RET$=CHR$(32)
```

```
LET GFLAG$="G1"

CALL START(#1,GFLAG$,BYTECOUNT,RET$)
LET FEEDRATE = 0                    !this is put here to get the program to write F correctly
LET NEWFEEDRATE=800                 !the feedrate must be changed here if a different one is required

Read #2:XTL
Read #2:YTL
Read #2:ZTL
LET XTL = Truncate(XTL,2)   !get the toolcoordinate into the correct format
LET YTL = Truncate(YTL,2)
LET ZTL = Truncate(ZTL,2)

LET ZTL=ZTL+10
CALL MOVERAPIDLY(#1,GFLAG$,XTL,YTL,ZTL,BYTECOUNT,RET$)
LET ZTL=ZTL-10
CALL FEED(#1,GFLAG$,XTL,YTL,ZTL,BYTECOUNT,FEEDRATE,NEWFEEDRATE,RET$)

END SUB

---------------------------------- SURFMILL SUBROUTINE -----------------------------

SUB SURFMILL

FOR ToolPos = 1 to Numtoolposs

 Read #2:XTL
 Read #2:YTL
 Read #2:ZTL

 LET XTL = Truncate(XTL,2)
 LET YTL = Truncate(YTL,2)
 LET ZTL = Truncate(ZTL,2)

 CALL FEED(#1,GFLAG$,XTL,YTL,ZTL,BYTECOUNT,FEEDRATE,NEWFEEDRATE,RET$)

  IF bytecount > 7800 THEN                                  !check the length of the CNC module,
   CALL CHANGETAPE(#1,GFLAG$,XTL,YTL,ZTL,BYTECOUNT,RET$)    !if>7800 bytes then start a new one
  END IF

NEXT ToolPos

END SUB

---------------------------------- TERMINATE SUBROUTINE-----------------------------

SUB TERMINATE

CALL FINISH(#1,GFLAG$,BYTECOUNT,RET$)
CLOSE #1
CLOSE #2

CLEAR
PRINT "      ┌──────────────────────────────────────────────────────────────┐"
PRINT "      │                      CNC MILLING SYSTEM                      │"
PRINT "      ├──────────────────────────────────────────────────────────────┤"
PRINT "      │     POSTPROCESSOR FOR CONVERSION OF TOOLFILE INTO CNC PROGRAMS │"
PRINT "      ├──────────────────────────────────────────────────────────────┤"
PRINT "      │                                                              │"
PRINT "      │              All the CNC modules have been created           │"
PRINT "      │                                                              │"
PRINT "      └──────────────────────────────────────────────────────────────┘"
PAUSE 3

END SUB

------------------------------- COORDINATES SUBROUTINE ----------------------------

SUB COORDINATES(#1,GFLAG$,XTL,YTL,ZTL,BYTECOUNT,RET$)

  PRINT #1:"X";
  PRINT #1:USING$("<###.##",XTL);   !get the toolcoordinates in the correct format
  PRINT #1:"Y";
  PRINT #1:USING$("<###.##",YTL);
  PRINT #1:"Z";
  PRINT #1:USING$("<###.##",ZTL);
  LET BYTECOUNT=BYTECOUNT+26        !increase the counter

END SUB !of coordinates
```

```
SUB FEED(#1,GFLAG$,XTL,YTL,ZTL,BYTECOUNT,FEEDRATE,NEWFEEDRATE,RET$)

  IF GFLAG$="G1" THEN     !do not rewrite the character "G1" if they have already been written
    ELSE
        LET GFLAG$="G1"
        PRINT #1: GFLAG$;
        LET BYTECOUNT=BYTECOUNT+2
  END IF

 CALL COORDINATES(#1,GFLAG$,XTL,YTL,ZTL,BYTECOUNT,RET$)

  IF NEWFEEDRATE <> FEEDRATE THEN    !if the feedrate should change then write "F" and the new
     LET FEEDRATE=NEWFEEDRATE        !the new feedrate
     PRINT #1: "F";FEEDRATE;
     LET BYTECOUNT=BYTECOUNT+4
  END IF

  PRINT #1

END SUB

-------------------------------- MOVERAPIDLY SUBROUTINE-----------------------------

SUB MOVERAPIDLY(#1,GFLAG$,XTL,YTL,ZTL,BYTECOUNT,RET$)

  IF GFLAG$="G0" THEN
   PRINT " " !do nothing
    ELSE
        LET GFLAG$="G0"
        PRINT #1: GFLAG$;
        LET BYTECOUNT=BYTECOUNT+2
  END IF

  CALL COORDINATES(#1,GFLAG$,XTL,YTL,ZTL,BYTECOUNT,RET$)
  PRINT #1:RET$

END SUB   !of moverapidly

---------------------------------- START SUBROUTINE--------------------------------

SUB START(#1,GFLAG$,BYTECOUNT,RET$)

  PRINT #1: "G0G90G71X-20.00Y-20.00T1M6"    !write the starting string to the CNC module to select
  LET BYTECOUNT=BYTECOUNT+33                !absolute, metric and rapid movement
  LET GFLAG$="G0"

END SUB   !of start

--------------------------------- FINISH SUBROUTINE -------------------------------

SUB FINISH(#1,GFLAG$,BYTECOUNT,RET$)

  PRINT #1: "G0G90G71X0.0Y0.0M2"     !write the finishing string to send the machine to the
  LET GFLAG$="G0"                    !origin.
  LET BYTECOUNT=BYTECOUNT+24
  PRINT #1: "$"

END SUB    !of finish

-------------------------------- CHANGETAPE SUBROUTINE ------------------------------
```

```
SUB CHANGETAPE(#1,GFLAG$,XTL,YTL,ZTL,BYTECOUNT,RET$)
  CALL FINISH(#1,GFLAG$,BYTECOUNT,RET$)
  CLOSE #1

  SET #2:POINTER SAME  !These statements move the file pointer so that
  SET #2:POINTER SAME  !the coordinates of the last point in file "X" are the
  SET #2:POINTER SAME  !coordinates of the first point in file "X+1". This prevent the loss of a
                       !toolposition between CNC modules

  LET cncflno = cncflno + 1
  LET cncfile$ = progname$ & STR$(cncflno) & ".PRG"
  OPEN #1:NAME cncfile$,CREATE NEWOLD,ORGANIZATION TEXT
  CALL FLINIT
  PRINT TAB (7,9);"Creating the Bridgeport CNC module ";cncfile$;" ... please wait"

END SUB

-------------------------------------------*-----------------------------------------

END
```

APPENDIX K

MIRROR COMMAND FILE

The MIRROR command file is used to configure the communications package with the correct settings for file transfer to the CNC computer.

```
PROGRAM : STD.XTK
PURPOSE : This is the MIRROR command file.
```

```
The important variables in this file are:
   MODE CALL     - sets the IBM PC to transmit rather than wait for data
   DUPLEX FULL   - allows 2 way communication
   PARITY EVEN   - sets even parity in the data packet
   SPEED 0110    - sets the baud rate
   STOP 2        - sets 2 stop bits in the data packet
```

```
The only command in this file is:
   DO START      - after MIRROR has configured itself using the settings contained
                   in this file the START.XTS script file is loaded and run
                   automatically
```

```
NAme      ""
NUmber    ""
ACcept    Everything
ANswback  Off
ATten     Esc
BReak     End
SWitch    Home
CWait     None
LWait     None
DEbug     Off
DPrefix   ""
DSuffix   |
EMulate   None
FIlter    -------++++-++------------------
INfilter  On
LFauto    Off
MOde      Call
POrt      1
PWord     ""
TImer     On
TUrnarnd  Enter
BKsize    1
CApture   off
COmmand   ETX (^C)
DAta      8
DNames    200
DUplex    Full
OUtfiltr  On
PArity    Even
PMode     1
PRinter   Off
SPeed     0110
STop      2
TAbex     Off
BLankex   Off
UConly    Off
```

```
FK 1 ""
FK 2 ""
FK 3 ""
FK 4 ""
FK 5 ""
FK 6 ""
FK 7 ""
FK 8 ""
FK 9 ""
FK 10 ""
FK S1 ""
FK S2 ""
FK S3 ""
FK S4 ""
FK S5 ""
FK S6 ""
FK S7 ""
FK S8 ""
FK S9 ""
FK S10 ""
FK C1 ""
FK C2 ""
FK C3 ""
FK C4 ""
FK C5 ""
FK C6 ""
FK C7 ""
FK C8 ""
FK C9 ""
FK C10 ""
FK A1 ""
FK A2 ""
FK A3 ""
FK A4 ""
FK A5 ""
FK A6 ""
FK A7 ""
FK A8 ""
FK A9 ""
FK A10 ""
DO START
```

APPENDIX L

SCRIPT FILE USED WITH THE MIRROR PACKAGE

The script file to be used with the MIRROR program is given below. The numbers appearing on the extreme left do not form part of the program and are used solely for reference purposes.

The semi-colon appearing in front of the description of each subroutine is required by the MIRROR program to identify the text as a comment rather than as a program command.

```
PROGRAM : START.XTS
PURPOSE : The script file used with the MIRROR program for transmitting
          the CNC modules from the IBM PC to the memory of the CNC computer
          It does the following:
            1. Establishes a connection between the IBM PC and the CNC computer
            2. Reads each CNC module and transmits it to the CNC computer.
            3. Displays a message to the user telling him precisely which
               controls to press on the CNC machine.
            4. Disconnects the IBM PC from the CNC computer after all the CNC
               modules have been transmitted.
```

```
As with previous programs this program has two sections, the main logic section
and the subroutine section. To call a subroutine the Script file uses the "JUMP"
command rather than the "CALL" command used in TRUE BASIC.
The calls to subroutines in this program are given below:
    SETTUP    - used for initialising and starting the file transfer process
    SETCONN   - sets up the initial connection between the IBM PC and the CNC
                computer. It also displays a detailed screen telling the user
                exactly which CNC controls to use and when to use them during the
                transfer of the CNC modules.
    PROGSEND  - this transmits each CNC module from the disk to the memory of the
                CNC machine.
     FIN      - finishes the files transfer and disconnects the IBM PC from the
                CNC computer.
```

```
---------------------------------- CONTROL PROGRAM ---------------------------------

1   LABEL ONE                                          ;main control logic section
2   JUMP SETTUP
3   LABEL  TWO
4   JUMP SETCONN
5   LABEL  THREE
6   JUMP  PROGSEND
7   LABEL  FOUR
8   JUMP  FIN
```

```
---------------------------------- SETTUP SUBROUTINE -----------------------------

9   LABEL  SETTUP
10  SCREEN  T                                   ;changes to the text screen for displaying messages
11  CLEAR
12  MESSAGE                                     ;equivalent of a BASIC "Print" statement

        Transfer of files from the PC to the CNC computer.
.
13  ASK Are you ready to begin? (Y/N)
14  IF Yy JUMP TWO                              ;allows capital or small letter Y
15  JUMP FIN

---------------------------------SETCONN SUBROUTINE -----------------------------

16  LABEL SETCONN
17  CLEAR
18  MESSAGE
            You first need to establish communication with
            the CNC computer so follow carefully the
            instructions below.

        1. Set the CNC machine to SETUP mode with the
           MODE switch on the control panel.
        2. Turn the EDIT/RDI switch to EDIT ... BUT DO NOT
           PRESS IT YET!
        3. Get ready to press the EDIT switch AFTER
           hitting ENTER on the PC.
.                                                   ;marks the end of the message

19  ALARM  3 NOW                                    ;sounds a beep to alert the user to press
20  MESSAGE                                         ;the controls in the correct sequence

            NOW PRESS ENTER ON THE PC TO CONNECT AND THEN
            PRESS THE EDIT SWITCH ON THE CNC
.

21  ASK Press Enter to connect
22  GO LOCAL                                          ;establishes the connection
23  WAIT STRING  BOSS 4.1                             ;waits for the CNC to acknowledge the
                                                      ;connection
24  MESSAGE
           The PC is now connected to the CNC computer,
           also any program in the CNC will be cleared.
.
25  ALARM  3 NOW
26  WAIT  QUIET  15                         ;waits for 1.5 seconds of no activity
                                            ;on the transmission lines to ensure that all the
                                            ;information has been received by the IBM PC

27  REPLY  K|                               ;sends a "K" and a return character
                                            ;which is the CNC command to erase any programs
                                            ;which may be in the memory of the CNC computer
28  WAIT  QUIET  15
29  JUMP  THREE

--------------------------------- PROGSEND SUBROUTINE ----------------------------

30  LABEL  PROGSEND
31  CLEAR
32  ALARM  3 NOW
33  MESSAGE
            Enter the Name of the Program file to send to
            the CNC machine.
.
34  ASK @F10  What is the Name of the Program file to send?

35  SEND @F10                                   ;the MIRROR command to transmit a disk file
36  WAIT DELAY 30                               ;waits 3 seconds to ensure no data loss
37  CLEAR
38  REPLY  ^Z                                   ;the CNC command to take the CNC machine out of
                                                ;the EDIT mode and put it into the machining
                                                ;mode to execute the CNC module
39  WAIT  QUIET  15
40  ALARM  2 NOW
```

```
41  MESSAGE
            Finished sending module @F10,  time to
            start the machine.

            1. Turn the MODE switch to AUTO

            2. Turn the FUNCTION switch to RESTART and
               press it

            3. Turn the FUNCTION switch to START CONTINUE
               and press to begin the program.


            When the machining operation is finished turn
            the MODE switch back to SETUP and the
            EDIT/RDI switch to EDIT (BUT DO NOT PRESS IT!).

      Now:  1. Press "C" on the PC and then the EDIT switch
               to send the next module
      OR
            2. Press "F" on the PC to disconnect and
               terminate the communications program.
.

42  ASK  Press C or F to continue:
43  IF F JUMP FOUR                              ;either continue and send the next
                                                ;CNC module or finish the program
44  WAIT STRING  BOSS 4.1
45  WAIT QUIET  15
46  REPLY |
47  WAIT QUIET  15
48  REPLY K|
49  WAIT QUIET  15
50  JUMP  PROGSEND                              ;loop back to the PROGSEND section to transmit
                                                ;the next CNC module

----------------------------------- FIN SUBROUTINE --------------------------------

51 LABEL  FIN
52 CLEAR
53 MESSAGE
            You have finished the machining procedure now
            and the PC has disconnected itself from the CNC
            computer. To reconnect you must rerun this
            program by typing "DO START" at the command
            line.
.
54 ALARM  1 NOW
55 BYE                                          ;disconnect from the CNC computer
56 QUIT                                         ;end the program

----------------------------------- END OF PROGRAM --------------------------------
```

APPENDIX M

SAMPLE PROGRAM FOR GENERATING DATA

The program SURFMAKE.TRU which is used to create the surface data for a hemispherical surface of radius 25mm centrally located on a 100mm x 100mm flat plate is listed below.

The program is provided for those users wishing to directly create record files for use with the milling system. Both the source code and a compiled .EXE version of this program are on the floppy disks accompanying the thesis.

The surface created by this program is shown in Figure M.1 below.

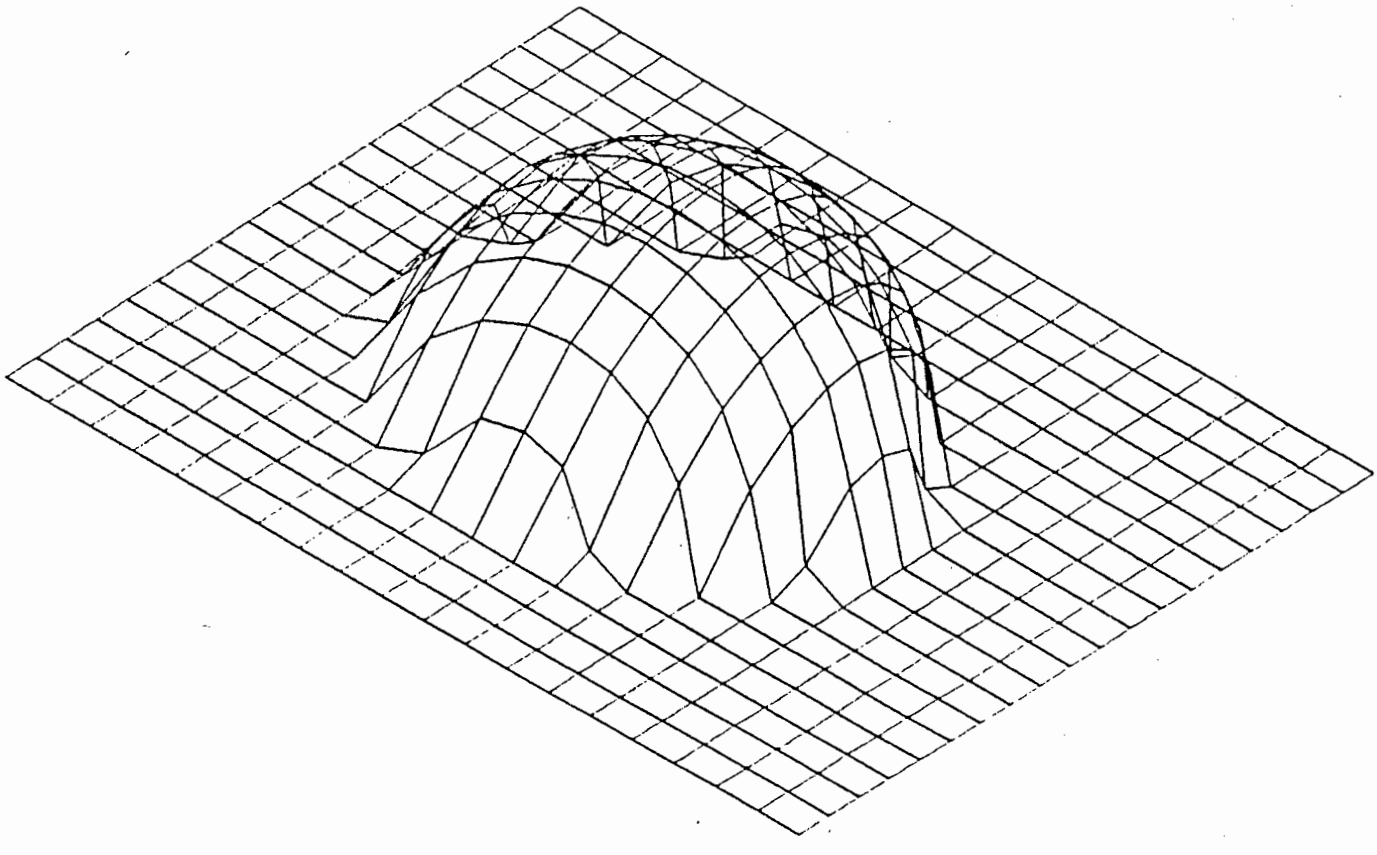


Figure M.1: Surface created by the SURFMAKE.TRU program

```
PROGRAM : SURFMAKE.TRU
PURPOSE : This program generates data for the CNC milling system according to
          a mathematical equation. It is provided as an example of the correct
          data format to use for the milling system, and the program can serve as a
          basis from which the user can construct his own data generating
          programs.
          The program structure is simple and therefore no separate subroutines
          have been used.
```

```
This section dimensions the arrays required
for the program and requests the filename for
the surface data file that is to be created.
```

```
DIM Xcoord[100,100]
DIM Ycoord[100,100]
DIM Zcoord[100,100]
CLEAR
INPUT PROMPT "Name of the Surface Data File to be created ":xyzcoord$
```

```
Creation of the RECORD file, which is the TRUE
BASIC equivalent of a random access file, with
the RECORD SIZE = 20 bytes. The user MUST
adhere to this format.
```

```
OPEN #1:NAME xyzcoord$,Create Newold,Organization Record
ERASE #1
SET #1:Recsize 20                                   !this is the only parameter specifying the
CLEAR                                               !format of the file and MUST be used
```

```
Loop to calculate the x,y and z coordinates of
the surface according to the equation:
 z = √(625 - ((x-50)^2-(y-50)^2))
    for 0 > x > 100
        0 > y > 100
    and z = 0  for  √((x-50)^2-(y-50)^2) < 0
```

```
 PRINT TAB(14,34) "CREATING THE SURFACE DATA FILE .... PLEASE WAIT"

  FOR Frame=1 to 20                                          !create a 20x20 element
   FOR Level=1 to 20                                         !surface, using 5mm x 5mm elements

    LET XCOORD[Frame,Level]=Frame*5
    LET YCOORD[Frame,Level]=Level*5
    LET EXP=625-((XCOORD[FRAME,LEVEL]-50)^2)-((YCOORD[FRAME,LEVEL]-50)^2)

    IF EXP < 0 THEN                                          !if the data point is not on the
                                                             !hemisphere then it must have 0 height
    LET ZCOORD[Frame,Level]=0
    ELSE
    LET ZCOORD[Frame,Level]=SQR(EXP)                         !calculate the Z height using the
                                                             !equation of a hemi-sphere
    END IF
```

```
    Write #1: Frame                                  !this is the data actually written to
    Write #1: Level                                  !the data file
    Write #1: Xcoord[Frame,Level]
    Write #1: Ycoord[Frame,Level]
    Write #1: Zcoord[Frame,Level]

  NEXT Level
 NEXT Frame
```

A beep sounds to alert the user that the program has finished. The disk file is closed and the program terminates.

```
SOUND 1000,0.2
CLEAR
PRINT "FINISHED CREATING THE SURFACE "
CLOSE #1

END
```

APPENDIX N

ASCII TO RECORD FILE CONVERSION PROGRAM

In order to enable externally generated data to be used with the CNC milling system, a file conversion program was written which converts ASCII data, the most commonly used format for data exchange, into the RECORD file format used by the milling system. For more detail on the data format see Appendix E.

```
PROGRAM : FILECONV.TRU
PURPOSE : This is a utility program which allows the user to convert ASCII surface
          data files into the RECORD file format required by the CNC milling system.
           The constraints on the ASCII data file are:
            a) It must not have any blank lines preceding the data.
            b) The data must consist of long list with 3 columns, corresponding to
               the X,Y and Z coordinates of the surface. There must be at least one
               space between each column.
            c) The data must be in a specific sequence as detailed in the USER GUIDE
          It does the following:
            1. Reads the ASCII data file.
            2. Writes the same data in a RECORD file format with RECORD SIZE 20 bytes

 As this program is simple no subroutines have been used.
```

```
CLEAR
PRINT "This program converts ASCII surface data files into a form"
PRINT "that can be used by the Milling System. "
PRINT

INPUT PROMPT "Give the name of the ASCII data file  ":asciifile$
INPUT PROMPT "Give the name of the File to create for the milling system  ":millfile$

OPEN #1:NAME asciifile$,ORGANIZATION TEXT                    !open the ASCII data file
OPEN #2:NAME millfile$,ORGANIZATION RECORD,CREATE NEWOLD      !create a RECORD file

SET #1:POINTER BEGIN
ERASE #2
SET #2:RECSIZE 20
PRINT
PRINT
INPUT PROMPT "Number of Frames:  ":NUMFRAMES
INPUT PROMPT "Number of Levels:  ":NUMLEVELS

CLEAR
PRINT tab(8,14); "Converting the File ";asciifile$;" into ";millfile$;" .... please wait"
```

```
FOR FRAME = 1 TO NUMFRAMES
 FOR LEVEL = 1 TO NUMLEVELS

INPUT #1:Triple$

LET Triple$=Trim$(Triple$)                          !gets rid of any leading or trailing spaces
LET Lengthtrip=len(triple$)                         !determines the length of the data string
LET posspacea=pos(triple$," ")                      !finds the position in the string of the first space
LET x$=triple$[1:posspacea]                         !the X coordinate immediately precedes the space

LET Lengthxstring=Len(x$)                           !find the length of the X coordinate
LET double$=triple$[Lengthxstring:Lengthtrip]       !make a new string, leaving out the X coordinate
LET double$=trim$(double$)                          !get rid of any leading or trailing spaces
LET posspaceb=pos(double$," ")                      !find the position of the next space
LET y$=double$[1:posspaceb]                         !the Y coordinate immediately precedes this space

LET Lengthdouble=Len(double$)
LET lengthystring=Len(y$)                           !find the length of the Y string
LET Single$=double$[lengthystring:lengthdouble]     !remove it from the shortened string
LET z$=trim$(single$)                               !what is left is the Z coordinate


 WRITE #2: Frame                                    !write the data, including the frame and level
 WRITE #2: Level                                    !to the RECORD file
 WRITE #2: Val(x$)
 WRITE #2: Val(y$)
 WRITE #2: Val(z$)

 NEXT LEVEL
 NEXT FRAME

CLOSE #1                                            !close the files
CLOSE #2
CLEAR
SOUND 1000,0.5
PRINT tab(15,14) ;" Finished creating file .... "

END
```