LINKING FINITE-ELEMENT ANALYSIS AND COMPUTER-AIDED DRAFTING PROCESSES

by

GODFREY CHUKUDI ONWUBOLU

A
thesis submitted for
the degree of

DOCTOR OF PHILOSOPHY

in the

Department of Mechanical & Production Engineering
at

THE UNIVERSITY OF ASTON IN BIRMINGHAM , U.K.

February 1985

Supervisor:
T.H. Richards

THE UNIVERSITY OF ASTON IN BIRMINGHAM

LINKING FINITE-ELEMENT ANALYSIS AND COMPUTER-AIDED DRAFTING PROCESSES

by
GODFREY CHUKUDI ONWUBOLU

A thesis submitted to
The University of Aston in Birmingham
for the degree of
Doctor of Philosophy 1985

SUMMARY

The aim of this research was to investigate the integration of computer-aided drafting and finite-element analysis in a linked computer-aided design procedure and to develop the necessary software.

The Be'zier surface patch for surface representation was used to bridge the gap between the rather separate fields of drafting and finite-element analysis because the surfaces are defined by analytical functions which allow systematic and controlled variation of the shape and provide continuous derivatives up to any required degree.

The objectives of this research were achieved by establishing :

(i) A package which interpretes the engineering drawings of plate and shell structures and prepares the Be'zier net necessary for surface representation.

(ii) A general purpose stand-alone meshed-surface modelling package for surface representation of plates and shells using the Be'zier surface patch technique.

(iii) A translator which adapts the geometric description of plate and shell structures as given by the meshed-surface modeller to the form needed by the finite-element analysis package. The translator was extended to suit fan impellers by taking advantage of their sectorial symmetry.

The linking processes were carried out for simple test structures , simplified and actual fan impellers to verify the flexibility and usefulness of the linking technique adopted.

Finite-element results for thin plate and shell structures showed excellent agreement with those obtained by other investigators while results for the simplified and actual fan impellers also showed good agreement with those obtained in an earlier investigation where finite-element analysis input data were manually prepared.

Some extensions of this work have also been discussed.

Key Words

COMPUTER-AIDED DESIGN, DRAFTING, GEOMETRIC MODELLING, F.E.M, LINKING

ACKNOWLEDGEMENTS

I hereby express my sincere gratitudes to my supervisor , Mr. T. H. Richards , Head of Mechanical Engineering Division , without whom this research would not have been completed. He was understanding , patient and kind ; he was more than a supervisor.

The author would also like to thank Mr. P. Cooley for introducing him to the basic theory of geometric modelling during the Masters programme.

I also thank all my brethren who have supported me in prayers throughout the duration of this work.

Finally , I owe my wife , Ngozi and our daughters , Chioma and Chineye much thanks for their bearing the yoke with me throughout the period of the research and especially during the very difficult times.

DEDICATION

I humbly dedicate this work to the LORD for ;

HE is gracious ;

HE is loving and

HIS mercies endureth for ever.

LIST OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOTATIONS

The following notations are used throughout the thesis :

β	Impeller blade angle
r,θ,z	Cylindrical coordinates
a,b,c,d,e,f	Parameters in various equations
a_x,b_x,c_x,d_x	Cofficients in patch equations
a_y,b_y,c_y,d_y	
a_z,b_z,c_z,d_z	
X,Y,Z	Rectangular Cartesian coordinates
l	Length
u,v	Parameters to define Be'zier patches
$P(u)$	Position vector of points on a curve
$P(u,v)$	Position vector of points on a surface

$Q(u)$	Representation of a curve resulting from vector $P(u)$
$Q(u,v)$	Representation of a surface resulting from vector $P(u,v)$
$\emptyset_u, \emptyset_v$	Linear operators
$\emptyset_{u,v}$	Bi-linear operator
$\oplus$	Boolean operator
V	Total potential energy
U_e	Element strain energy
P_e	Potential energy of loading on element
$[K]$	Stiffness matrix
$\{q\}$	Vector of generalised coordinates
$\{Q\}$	Vector of generalised forces
ξ, η	Curvilinear coordinates

f:	Arbitrary function
$B_m[f;u]$	Bernstein polynomial of univariate function
$B_{m,n}[f;u,v]$	Bernstein polynomial of bivariate function
ϕ_i, ϕ_i	Binomial probability density function
m	Degree of Bernstein polynomial
${}^mC_i$	Probability of exactly i successes in m trials
P_i	Position vector of Be'zier polygon vertex
P_{ij}	Position vector of Be'zier net vertex
V_i	Control vertex of Be'zier polygon
V_{ij}	Control vertex of Be'zier net

N_u	Number of meshes in u-direction (Be'zier surface)
N_v	Number of meshes in v-direction (Be'zier surface)
R	Radius
$N_1, N_2, N_3, \ldots$etc	Numbering for each corner of a surface padtch
n_x, n_y	Number of grid lines to define Be'zier surface
$\cup$	Union of sets
$\cap$	Inersection of sets
A	Area
$\bar{X}, \bar{Y}, \bar{Z}$	Centroidal coordinates
I_x, I_y, I_z	First area moment of inertia
I_{xx}, I_{yy}, I_{zz}	Second area moment of inertia

I_{xy}, I_{xz}, I_{yz}	Product area moments

CHAPTER 1

INTRODUCTION

Computer-Aided Design (CAD) is a relatively recent technique in which man and computer are blended into an efficient design team. A fairly recent topic of research in CAD is the integration of its features such as computer aided drafting , geometric modelling , analysis (finite-element analysis and geometric and inertial properties) and kinematics. These features have developed independently since the early 1950's and the evolution of CAD is integrating these diverse technologies.

The conventional way of conveying shape information from the designer to the manufacturing shop was in the form of engineering drawings on paper prepared on a drawing board. With the evolution of computers having interactive graphics facilities , automated drafting systems became very effective in speeding up the production of engineering drawings. Traditionally ,it was not until the first prototype became available for test , that the engineer was able to get full 3-dimensional appreciation of a design product. Errors detected when the prototype was ready were only reluctantly remedied , rather large factors of safety are used and certain compromises accepted.

Later on , 3-dimensional modelling systems emerged. In CAD , a distinction is made between the drafting system and the modeller. A drafting system permits the generation and manipulation of graphics entities such as lines , arcs and points ; little attempt is made at interpreting their meaning. With its aid , the user is able to generate objects quickly in close dialogue with the system. Graphical output gives the illusion of reality , but no built-in rules govern the consistency of the resulting objects. A modeller on the other hand generates graphical output very similar to the drafting system , but it contains additional information sufficient to assert consistency of the design within the given domain.

Geometric modelling is considered the "core" of integrated CAD systems because other CAD features use it as starting point. The geometric model can be submitted for finite-element method (FEM) stress analysis , or it may serve as an input for automated drafting to produce engineering drawings of the part , or may be used as a basis for generating Numerical Control (NC) instructions for making parts on automated machine tools.

Where a 3-dimensional modelling capability is an integral part of CAD system , the question of whether to use 3-dimensional facilities to produce 2-dimensional drawings or to produce 3-dimensional output from simple 2-dimensional drawings is often raised. The answer to this question is dictated by the efficiency with which a system produces 3-dimensional output from simple 2-dimensional drawings and the complexity of the geometry involved.

In the design of engineering products made from thin sheetmetal , 3-D surface modellers are most appropriate. Thus , centrifugal fan impellers usually fabricated from thin sheetmetal into a backsheet , conesheet and the blades between them , can be best represented by a surface modeller. One of the most successful and recent CAD modelling techniques is the Be'zier surface patch , conceived by Be'zier [7]. It was chosen for this work because it suits the definition of any surface whether simple or sculptured (free-form). Other benefits of this technique are enumerated in Chapter 2.

The combination of surface modeller (Fig. 1.1) and drafting (Fig. 1.2) on one side , and finite-element analysis (Fig. 1.3) on another side , in a CAD system is of mutual benefit for the following reasons (Fig. 1.4) : the representation of an object within the geometric modeller can be utilized directly for the definition of a finite element mesh at the data-generation stage of the analysis. This removes the need for a complete re-definition of the object or in other cases , taking the 3-dimensional model of a component previously created in the system and automatically generating meshes on the surface.

Meshed-surface modelling if seen in a new light , will make possible a closing of the gap between the rather separate fields of drafting and finite-element analysis for shell type structures. Primarily , surface modelling is used for surface representation. In an extended form it may now serve as an interface between drafting and finite-element analysis by adapting the description of an object as given by the one representation to the form needed by the other.

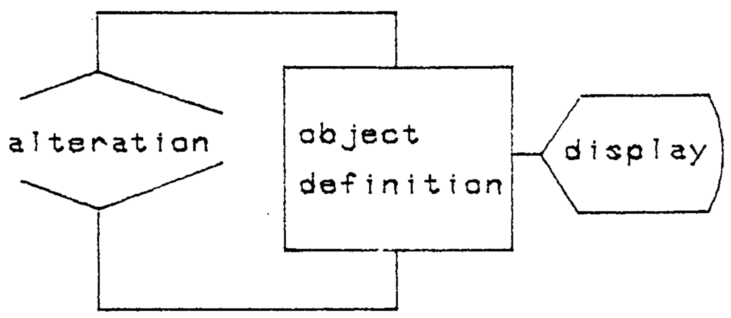


Fig. 1.1 Geometric modelling

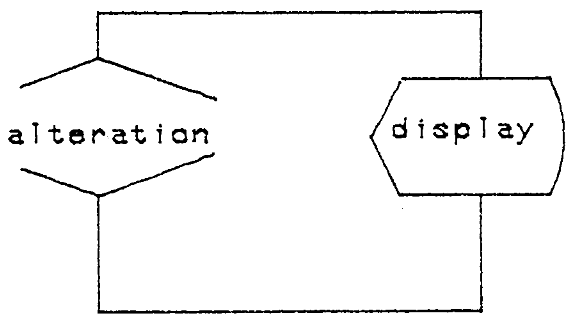


Fig. 1.2 Drafting.

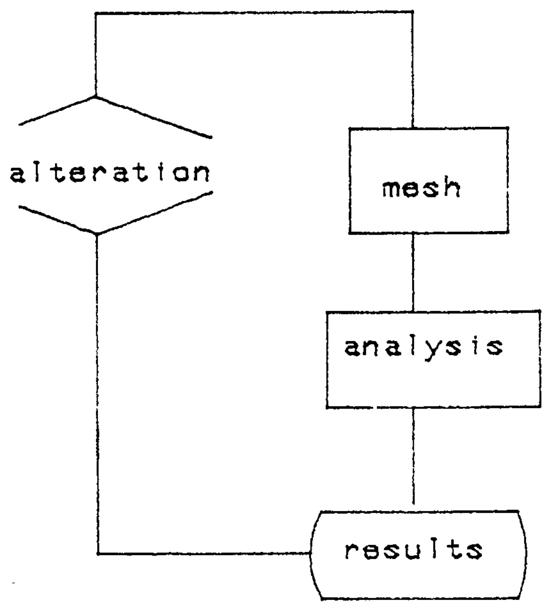


Fig. 1.3 Finite-element analysis

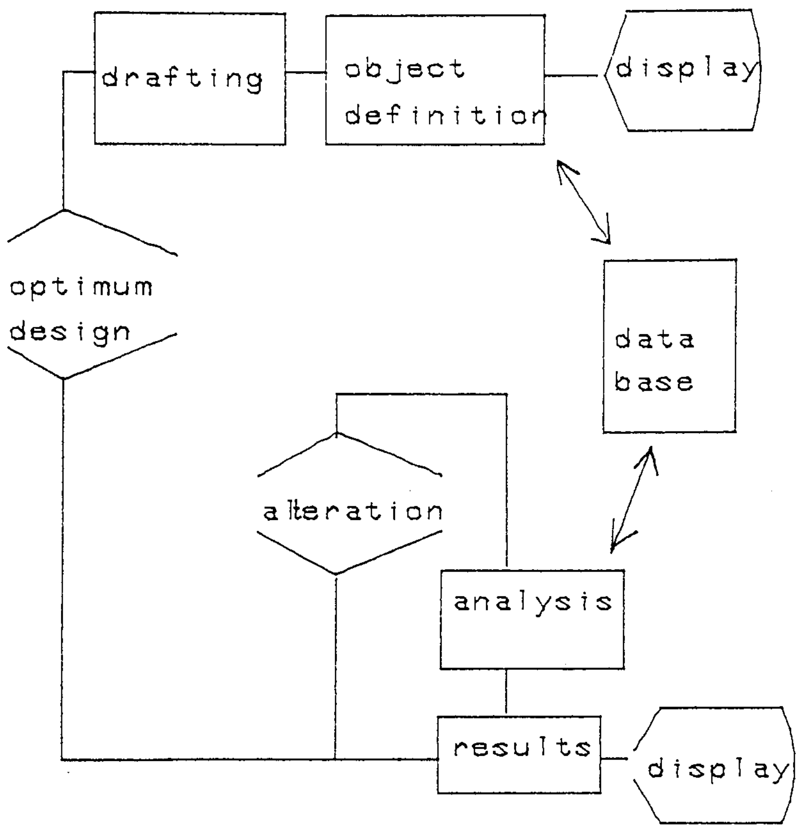


Fig. 1.4 Geometric modelling,drafting & Finite-element analysis

The objective of the research described in this thesis was the development of a surface modeller based on the Be'zier surface representation to link drafting and finite element analysis of plate/shell structures and in particular , fan impellers. In order to attain this goal , the following steps were found to be necessary :

(i) Developing a surface modeller which provides convenient geometric construction capabilities for models of simple and complex (free-form) surface-based entities.

(ii) Implementing a program to interprete orthographic views of a 2-dimensional drafting package , MEDIA [15] for input to the surface modeller.

(iii) Translation of geometric data created by the modeller for input to the finite element analysis programs , SMILOF and IMPSMF [13] for general thin plate and shell structures.

(iv) Extending the translation program to suit fan impellers by taking advantage of their sectorial symmetry.

(v) Utilization of the geometric data of the surface model for calculation of geometric and inertial properties of the model.

The choice of internal representation for the modeller is an essential factor in meeting the requirements of CAD-FEM integration. The Be'zier method adopted for this work is very adequate for the geometric representation of plate and shell structures.

The following chapters describe in detail , the role of a surface modeller as a link between drafting and a finite-element analyser of thin plates and shells. In Chapter 2 , a literature survey of the geometric and structural design of rotating fan impellers is given. The reasons for adopting the method of Be'zier technique for internal representation for the modeller are enumerated.

In Chapter 3 , the three main aspects of geometric modelling — wireframe , surface and solid modellings are treated.

Chapter 4 treats the basic steps in finite-element method. The geometric representation of thin plate and shell using modern CAD modelling techniques are discussed. The semiloof element is treated briefly as a useful element for such structures.

Chapter 5 expands the theory of Be'zier surfaces which leads to the algorithm for surface representation. A computer implementation for the Be'zier surface technique is also presented.

In Chapter 6 , review of literature relevant to the interpretation of engineering drawing for input to the surface modeller is given. The basis of interpretation are presented. A few Be'zier "nets" which are useful for surface representation are presented to show how interpreted orthographic views can be converted to the appropriate surface. Cyclic symmetry which occurs in fan impellers is taken into consideration in constructing the Be'zier nets. The program for interpretation is also presented.

Chapter 7 presents the translation of the geometric data of the modeller to finite-element analysis input data for the application programs SMILOF and IMPSMF [13] which were developed for analysis of thin plate and shell structures as well as fan impellers. The geometric data of the modeller is also used for calculating geometric and inertial properties of the model.

In Chapter 8 , a number of examples of processes starting from drafting , through interpretation , geometric modelling and translation to input to finite element analysis are shown.

In Chapter 9 , the summary of the work reported in this thesis , conclusion and suggestions for further work are presented.

CHAPTER 2

REVIEW OF THE LITERATURE RELATED TO THE GEOMETRIC AND STRUCTURAL DESIGN OF ROTATING FAN IMPELLERS

2.1 GENERAL INTRODUCTION

The design of fan impellers usually assumes two distinct stages, viz: preliminary and detailed design. During the preliminary design, the designer uses one-dimensional flow analysis based on previous experience to specify the inlet and outlet blade angles and the "skeletal" dimensions(such as inlet and outlet diameters and axial length of the impeller). This is then followed by a detailed design in which the complete impeller geometry is specified and subsequently refined by means of successive stress analyses.

During the detailed design stage the designer repeatedly adjusts the shape of the blades until he finds a suitable geometry that combines acceptable aerodynamic performance with low stress levels. The process of continual refinement of the shape can be expensive, tedious, and time-consuming, since at each stage the geometrical data for the necessary stress analysis must be prepared. This task is greatly simplified when a simple flexible system of geometry definition for the components of a fan impeller is available. The essential requirements for such a system may be summarised as follows:

(i) Be able to provide initial definition of the impeller shape.

(ii) Be able to provide geometric data for stress analyses in intervening phases.

(iii) Be able to allow consequent geometry modification and re-definition.

In this chapter several methods for the definition of the shape of fan impeller are reviewed.

2.2 THIN WALLED CENTRIFUGAL IMPELLERS

It is probable that the majority of all fans are of the centrifugal or radial flow type. Such a fan consists of an impeller running in a casing having a spirally shaped contour (See Fig.2.1). The air enters the impeller in an axial direction and is discharged at the periphery, the impeller rotation being towards the casing outlet. The amount of work done on the air, evident in the pressure development of the fan, depends primarily on the angle of the blades with respect to the direction of rotation at the periphery of the impeller. The impeller is the most highly stressed part of the fan. The stresses in the impeller are caused by rotational, aerodynamic and temperature effects , but the most important are the stresses caused by rotation.

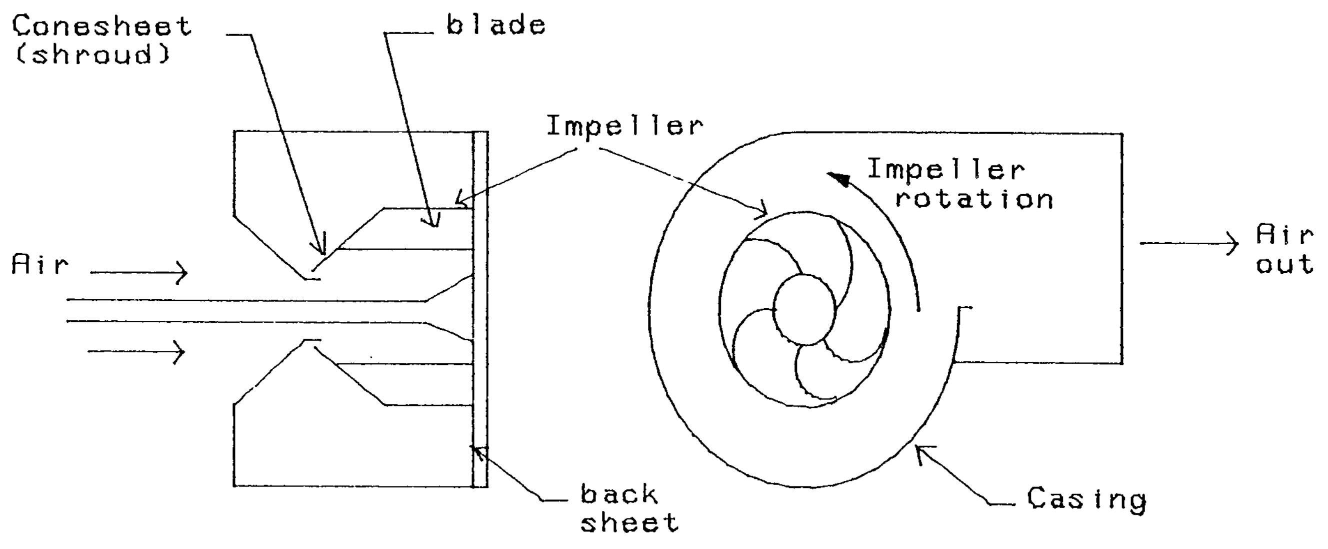
2.2.1 TYPES OF BLADES

Three main forms of blade are common and are shown in Fig. 2.1. These are:

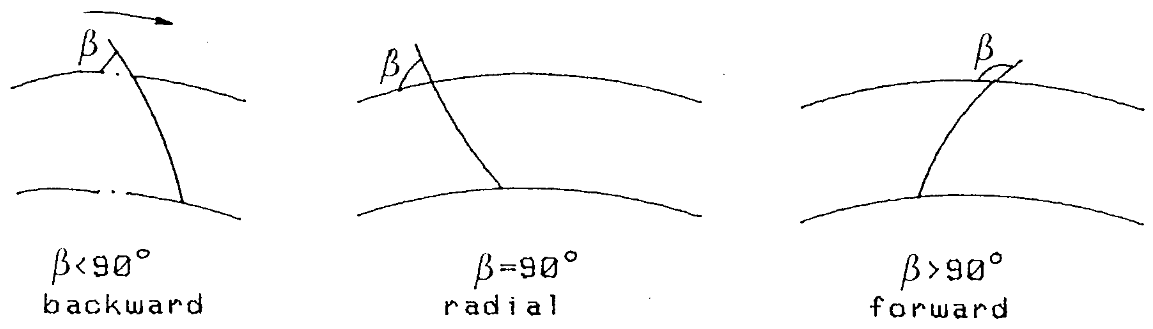
(a) the backward bladed impeller, in which the blade tips incline away from the direction of rotation, and the blade angle β is less than 90°.

(b) the radial bladed impeller, where the blade tips are radial, that is, $\beta = 90^{\circ}$.

(c) the forward bladed impeller, where the blade tips incline toward the direction of rotation, and the blade angle β is said to be greater than 90°.



Centrifugal fan configuration.



Blade forms.

Fig. 2.1 Centrifugal fan impeller.

Backward fan impellers are commonly used for ventilation of commercial buildings where their improved efficiency is of advantage , radial fan impellers being used as draught fans on large boilers and forward fan impeller are commonly used for ventilation of commercial buildings in view of their compact size for a given duty.

2.3 REVIEW OF THE RELATED LITERATURE

In this section , the literature related to the geometric and structural design of rotating fan impeller are reviewed.

2.3.1 GEOMETRIC DEFINITION OF ROTATING FAN IMPELLERS

Several methods for the definition of the shape of centrifugal fan impellers are reviewed in this section.

An early choice of many designers for the definition of impeller geometry was conic sections (for example, circular arc, ellipse, parabola, hyperbola etc), for both the hub and shroud contours and the impeller blades. A general description of the blade surface geometry using conic sections is, for example,

$$r\theta = ar^2 + 2brz + cz^2 + 2dr + 2ez + f$$

where the parameters a, b, c, d, e and f determine the inlet and outlet angles and the blade curvature. Moore [1] quoted examples of elliptical bladed impellers which fall into this category.

An alternative procedure was suggested by Whitfield et al [2]. This made use of Lamé's ovals of the general form

$$\{(\emptyset + a)/b\}^e + \{(z + c)/d\}^f = 1$$

where $\emptyset$ is either r or θ to define the blade surface shape. For any known end conditions, inlet location and slope, and exit location and slope the constants a, b, c and d can be determined and a series of possible profiles rapidly obtained by varying e and f. Because the analytic solution was difficult to handle and had a number of complex combinations where solutions did not exist, the procedure was modified by defining a Lamé oval with zero inlet slope and infinite outlet slope.

Both of these methods allow considerable freedom of shape through adjustment of the parameters a, b, c, etc., but are obviously not suitable for a general method of fan impeller design.

Jansen and Kirschner [3] described an early computer-based general work. They specified the blade shape by straight line elements from hub to shroud. The disadvantage here was that no equations were developed for the blade surface. Smith and Merryweather [4] described a similar computer-aided design method in which the impeller blade is represented by 'patches', a particular variety of three-dimensional surface studied by Coons [5]. They chose a 'cubic-linear' patch because its analytic form is such that it is imagined to be generated either by an infinite number of cubic curves lying in the direction of changing u, or by an infinite number of straight lines lying in the direction of changing v (Fig. 2.2). The co-ordinates along curves such as AB and MN (Fig. 2.2) are given by equation of the form

$$x = a_x u^3 + b_x u^2 + c_x u + d_x$$

$$y = a_y u^3 + b_y u^2 + c_y u + d_y$$

$$z = a_z u^3 + b_z u^2 + c_z u + d_z$$

with $v = 0$ along AB, $v = 1$ along MN and with $0 < v < 1$ for intermediate curves. The boundaries AM and BN are represented by linear equations:

$$x = e_x v + f_x$$

$$y = e_y v + f_y$$

$$z = e_z v + f_z$$

with $u = 0$ along AM, $u = 1$ along BN, and $0 < u < 1$ for intermediate straight lines lying in the surface.

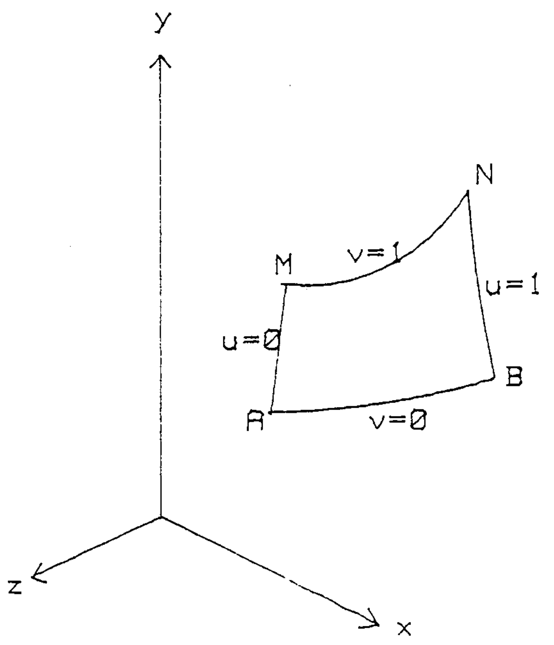


Fig. 2.2 The basic Coons' patch.

The disadvantage however is that the slope at patch corners must be known by the user.

Most recently, Casey [6] has adopted the interpolation formulae due to Be'zier [7] for the geometric definition of impeller blades.The methods of both Coons and Be'zier fall into a new class of geometries that has been recently developed for numerical controlled machining and manufacturing. Casey did not deal with the surface definition of the conesheet since he considered unshrouded impellers. In his approach, the control polygon points which are required as input data for Be'zier patches are manually supplied by the designer.

The research project described in the present thesis is aimed at implementing Be'zier patches to the geometric description of plate and shell structures and in particular , that of a fan impeller (consisting of the backsheet, blades and consheet) as well as deriving the control polygon points automatically from a two-view engineering drawing.

2.3.2 STRUCTURAL DESIGN OF ROTATING FAN IMPELLERS

One of the essential requirements of a design system is the capability to determine stresses and deflections in a structure being designed. In the past, the theoretical methods used to estimate the stress levels in rotating impellers proved to be unsatisfactory due to a general lack of knowledge regarding their effectiveness with regard to calculation accuracy. Consequently an efficient method is required which gives an accurate and overall stress pattern that exists in thin wall rotating fan impellers under the action of rotational loads. Experimental methods such as strain gauge measurements are also available for stress analysis of complicated structures such as rotating impellers. However, experimental methods are time consuming and expensive when compared with modern methods of analysis based on digital computers which have become very common and cheap.

There are few references [8] - [1Ø] dealing with the stress analysis of fan impellers, and for many years the conventional method used to obtain the stress distribution in fan impeller has been that suggested by Haerle [11]. This method involved the seperate treatment of the backsheet and consheet as rotating profiled discs with the blade loadings added at the appropriate radii. The changing thickness of section is catered for in discrete steps, and the percentage blade load to be added to the backsheet and conesheet at each step is derived from an empirical relationship depending on the blade width.

The disadvantages of this method is that it assumes that the impeller stresses are axisymmetric and take no account of the complex geometry of the impeller or the discontinous nature of the blade loading, which can in some cases cause large bending stresses in the backsheet and conesheet. Therefore, when applied to centrifugal fan impellers, this method of analysis involves approximations that result in large factors of safety having to be included.

Bells' work [12] on the rotating fan impellers was the first which attempted to apply the finite-element method for obtaining the stress level in these structures. The analysis was based on the plane triangular shell element described by Zienkiewicz [29], in which the bending and membrane behaviour are separately represented. The centrifugal force was calculated at the centre of the element and one third of the force was applied to each node of the element. There were significant discrepancies between the results for the bending moments between the experimental and finite element analysis in area of backsheet-blade and conesheet-blade intersections.

Jweeg [13] applied an approach similar to that of Bell and used the sophiscated Semiloof shell elements developed by Irons [14] : these are most suitable for analysing thin shell structures with sharps corners having junctions between blades and sheets (backsheet and conesheet). He assumed that the fan structure is cyclically symmetric and so only a sector of the fan structure needed to be considered. The use of the Semiloof element for discretization gave good agreement between the finite element predictions and experimental data for the stresses in the regions of backsheet-blade and conesheet-blade interaction.

However, Jweeg did not develop any automatic finite element mesh generation routine for discretization of the fan impeller surfaces. Although his work provides the Stress Engineer with an accurate method of assessing the stress levels of fan impellers , it has the disadvantages that manual construction of the element meshes and preparation of the vast amount of input data is required. The research project described in this present thesis is aimed at developing a method which overcomes these disadvantages by integrating the structural design with geometrical design.

2.4 THE BENEFITS OF ADOPTING BE'ZIER TECHNIQUE FOR GEOMETRIC DEFINITION OF FAN IMPELLERS.

In section 2.1, we have laid down the essential requirements for a flexible system of geometry definition which can be integrated in a modern computer-aided design system (where geometric definition system,drafting,analysis and sometimes kinematics are integrated). The method of Be'zier for surface definition meets such requirements. We want to be able to define the fan impeller geometrically, and straightaway utilize the goemetric defintion for stress analysis of the structure without recourse to any stand-alone finite element mesh generator. The particularly elegant technique developed by Be'zier has been adopted in the present work for the following reasons:

(i) The definition of the geometry of any surface, whether simple or sculptured is possible.

(ii) The backsheet, blades, and conesheet surfaces can be defined by equations of the same type.

(iii) The shapes produced are general enough to be used in the design of new fan impellers and in approximation of the geometry of existing ones.

(iv) The designer is the focal point because he can control the so-called polygon points, thereby providing a good interface between a user who does not necessarily have any mathematical skills and the mathematics representation he is controlling. This makes this technique suitable for incorporation into a computer-aided design (CAD) procedure for fans.

(v) The surfaces are defined by algebraic functions which allow systematic and controlled variation of the shape and provide continous derivatives up to any required degree.

(vi) The great advantage of using a parametric description for all of the surfaces is that the coordinates of the points on the surface can then be simply obtained by specifying the values of the parametric coordinates (u,v). By this means the geometric data of the backsheet, blades and conesheet surfaces can be generated for any number of points and in any suitable distribution for subsequent stress anaysis, especially by the finite element method.

(vii) The finite element meshes obtained are very suitable for the Semiloof element where only nodal positions are necessary for its geometric descriptions.

2.5 CLOSING REMARKS.

The discussions in Section 2.4 revealed that the work in Ref.[13] used the finite element method , together with a new class of elements which are most suitable for thin structures with intersection areas (blade-backsheet and blade-conesheet) for reasonable stress modelling. However, there was no automatic mesh generation routine for discretizating the impeller structure. Manual geometric construction of the complex impeller structure and preparation is tedious and error prone. In Section 2.3, modern techniques for surface generation which usefully lend themselves for mesh representation were reviewed.

The work reported here deals with the translation of the geometric definition of plate and shell structures and in particular ,that of the fan impeller (using Be'zier technique) from the `patch' representation into the required finite element idealisation through an interface program for use in the finite element analysis program in Ref.[13]. Other application programs such as the programs to calculate section properties of the fan (e.g surface area, centroid, area moments of inertia, principal moment of inetia) should be able to derive their input data from the gometric definition system. This aspect of the method should prove useful in the computer-aided design procedure for fan impellers.

In the next chapter, different types of geometric modellers,which are systems for geometric defintions as already mentioned in the previous section , are discussed. In chapter 4, the finite element method is briefly explained. Thereafter, the mathematical foundations of Bernstein-Be'zier surfaces are given. The fan impeller is then modelled and its geometric data translated into finite element meshes for analysis. The finite elements obtained are suitable for Semiloof elements used in Ref.[13] since only geometric representation in terms of nodal positions are needed.

CHAPTER 3

GEOMETRIC MODELLING

3.1 INTRODUCTION

Geometry is central to the design and production of mechanical parts. In the early times, artisans relied on physical models. As mass maufacturing increased, engineering drawings on paper were adopted as the primary medium of geometric specification because of their usefulness in communicating geometry between human beings.

With the advent of computers, there was a drift from drawing on paper to drawing using computers [16]. The representation, manipulation, and storage of part size and shape in a computer memory is known as geometric modelling. Geometric modelling is considered as one of the most important features of a CAD system because many design functions use it as a starting point. For example, the geometric model may be used as a framwork to create a more detailed finite element model of the structure. Or it may serve as an input for automated drafting to produce engineering drawings of the part. In another application, the geometric model may be used as a basis for generating numerical control (N C) instructions for making parts on automated machine tools.

Geometric models typically are created with pictures drawn on the screen of a CAD graphics terminal. Consequently, the user need not have knowledge of computers or programming to perform geometric modelling. As a result, the geometric pictures on CAD screens are now virtually synonymous with the geometric models they represent in computer memory.

Geometric models could be 2D for representing plane parts, 2 1/2 types for representing parts of constant section with no side-wall details, and full 3D models for representing the most general shapes. The first two are adequate for many parts but recent research in geometric modelling concentrate on developing more sophiscated 3D modelling capabilities. There are essentially three categories of 3D modelling which are treated here: wire frames, surface and solid modelling.

3.2 WIRE FRAME MODELLING.

Most 3D modelling are presently done with so-called wire frames made of inteconnected lines (or wires) to represent the edges of the physical objects being modelled. An operator at a CAD terminal creates a wire-frame model by merely specifying points and lines in space; the terminal screen being used in much the same manner as a drawing board to create various views of the model.

The system constructs various line segments based on points specified by the user and commands chosen from a function menu. The user may also produce curved lines using similar techniqus. Circles may be gnerated automatically from a centre point and a radius or three points on the circumference. In this same manner, the user also may produce conics, which are complex curves such as ellipses, hyperbolas, and parabolas. Most systems can generate splines, which are smooth curves fitted through a series of arbitrary user-specified points. Moreover application programs could scan through such internal representation to produce point-to-point NC code for drilling and punching.

In the 1970's , the 2D lines and arcs were generalised to segments of 3D space curves ; these can be projected automatically onto other views to produce orthographic, perspective and isometric views. Many other features help the user to create the model, such as duplication of specific details on the model, temporarily erasing selected lines and arcs from the screen without deleting them in the computer memory, recalling erased items to the screen at any time, enlarging a part of the model (zooming) to add minute details and later reducing it to the proper size.

While such 3D wire frame systems are clearly useful, they exhibit some serious deficiences. An example (Fig. 3.1) provided by Markowsky and Wesley [17], who described an interesting program for generating all possible solids that a valid wireframe may represent, is used to highlight some of the deficiencies of wireframe representation.

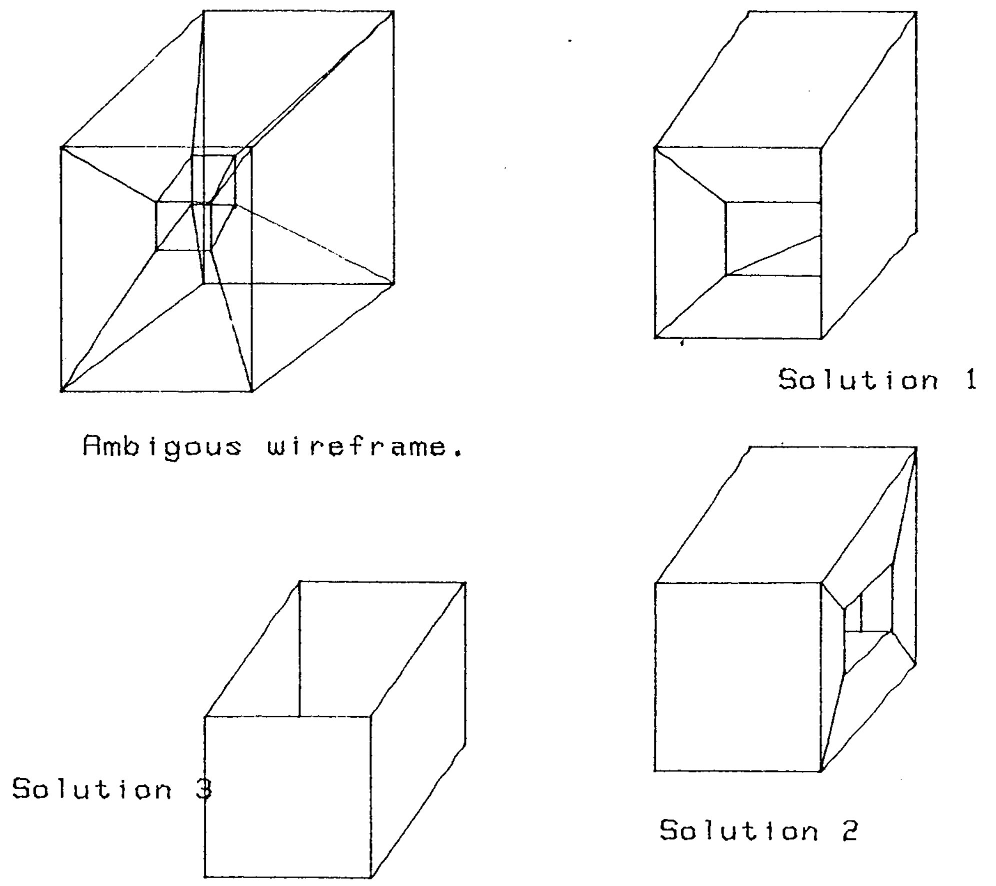


Fig. 3.1 Example of an ambigous wireframe having three valid solutions.

Some of the shortcomings of wire frame representation are as follows :

(i) The user, rather than the system, is expected to detect such anomalies as a missing edge.

(ii) The user needs to supply much information to describe simple objects.

(iii) Blended, rounded, or otherwise skewed surfaces are not readily represented. For example, considering Fig.3.2 where a simple cylinder is represented, the profile line shown is usually not included in the wire frame.

These limitations make it difficult to use a wireframe system for many forms of automatic processing such as automatic sectioning and volume, weight , centre of gravity and other mass properties calculations.

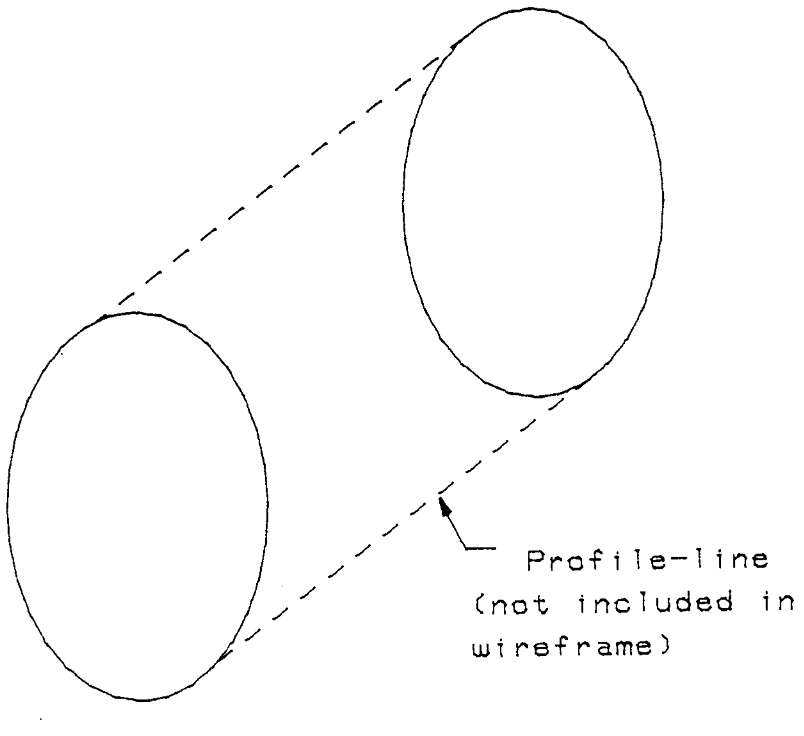


Fig. 3.2 Solid wireframe and profile lines.

3.3 SURFACE MODELLING

The next step towards increased complexity is the treatment of surfaces in space. With surface models, the outside faces of an object are represented by various surface types pieced together in the computer. Surface models are considered the best suited in applications where complex 3D geometries must be designed, especially where the application is primarily concerned with the shell of objects such as sheet-metal and thin molded-plastic parts. Moreover, the approach is often sufficient in finite-element modelling and NC programming where parts have complex contours. Therefore, surface modelling systems find much application in the design and manufacture of automobiles, aircraft, ships, machine parts, glassware, and clothing industries.

3.3.1 GENERAL PRINCIPLES OF SURFACE CONSTRUCTION

Sometimes the shape of a surface can be described analytically, e.g., in the form of cylinders, spheres, revolving conics, etc. However, in most cases, such an analytical description is not given, as in the case of car bodies, aircraft wings or ship hull, and the designer must resort to a constructive approach, creating surfaces from simpler data.

Curves can be constructed by interpolating or approximating a given set of scalar values specifying points or derivatives. Forrest [18] presented an enlightening survey of various ways of specifying curves by various combination of data. A substantial amount of literature on this topic is available [19]. However, we are concerned with a constructive approach to surface representation. Basically, there are three ways in which surfaces can be constructed from such data, and these are cartesian or tensor product, lofting and generalised Coons' or transfinite.

A curve is represented (in the general case of a curve in 3-space) in a parametric form

$$P(u)=[\ X(u)\ y(u)\ Z(u)\ ]$$

(parametric representation in ordinary coordinates).Generally, the curve-generating algorithms may be represented by an operator $\emptyset_u$ applied to the vector-valued function P(u) representing the data [18]. Hence,

$$Q(u) = \emptyset_u(P(u))$$

In the same way , a point on a surface in 3-space may be described in parametric form by the function

$$P(u) = [\ X(u,v),\ Y(u,v),\ Z(u,v)\ ],$$

and the surface-generating procedure may be symbolically denoted by

$$Q(u,v) = \emptyset_{u,v}(P(u,v)),$$

where P(u,v) represents the data from which Q(u,v) will be constructed.

These notations are used in discussing the various ways of surface construction.

3.3.2 CARTESIAN OR TENSOR PRODUCT DEFINITION

The most widely used approach to surface representation is to use the operators ϕ_u and ϕ_v to operate on the variables u and v simultaneously. Mathematically, we take the tensor product of the operators, leading to the surface approximation

$$Q(u,v) = \phi_u . \phi_v (P(u,v)).$$

The effect of this operation is that ϕ_u operates on the data $P(u_j,v)$ while ϕ_v operates simultaneously on the data $P(u,v_j)$, leading to the extraction of fixed vectors i.e vectors with constant u and contant v, and multiplying these vectors by both u and v blending functions. The fixed vectors can be associated with the nodes of a rectangular grid in the parameter (or u-v) plane as shown in Fig.(3.3). We are thus constructing a surface entirely in terms of zero-variate data since the fixed vectors will be generally point vectors or derivative vectors at fixed points.

3.3.3 LOFTING DEFINITION

Lofting is often used in applications where the surface to be constructed primarily stretches in one direction (e.g in aircraft or ship building industries). The surface is defined in terms of a single family of curves. The surface definition is simply by one of the two operations

$$Q(u,v) = \phi_u(P(u,v))$$

$$\text{or } Q(u,v) = \phi_v(P(u,v))$$

In this surface definition, once a surface has been defined by one family of functions, say functions of constant u, then a check is made on the other family of univariate functions, of constant v. These are then adjusted iteratively until both families of curves are satisfactory. Fig.3.4 and Fig.3.5 illustrate the lofting technique.

3.3.4 GENERALIZED COONS' OR TRANSFINITE DEFINITION

Instead of approximating a bivariate operator as a product of two sets of univariate operators (Section 3.3.2), or simply as one of the univariate operators (Section 3.3.3), one may employ the principle of superposition on two compatible families of univariate curves. Thus the method enables us to loft a surface in two directions simultaneously, thereby combining the capabilities of both methods of lofting ϕ_u and ϕ_v. We are thus using data wich make up all the grid lines (Fig. 3.6) rather than just one set as with lofting. A simple addition of the two operators , ϕ_u and ϕ_v , however, would render , at the points of intersection of the two families of intepolates, twice the value of the approximand.To correct this, the following operation must be performed [20]:

$$Q(u,v) = \phi_u \oplus \phi_v (P(u,v)) = [\phi_u + \phi_v - \phi_u \cdot \phi_v](P(u,v)).$$

where $\phi_u \oplus \phi_v$ is called the boolean sum approximation.

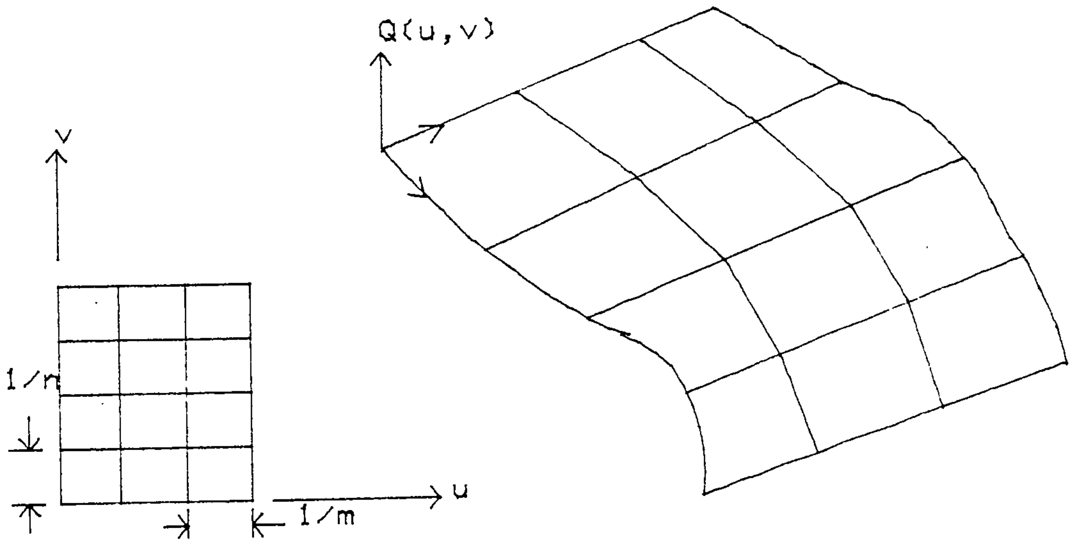


Fig. 3.3 Cartesian product surface definition.

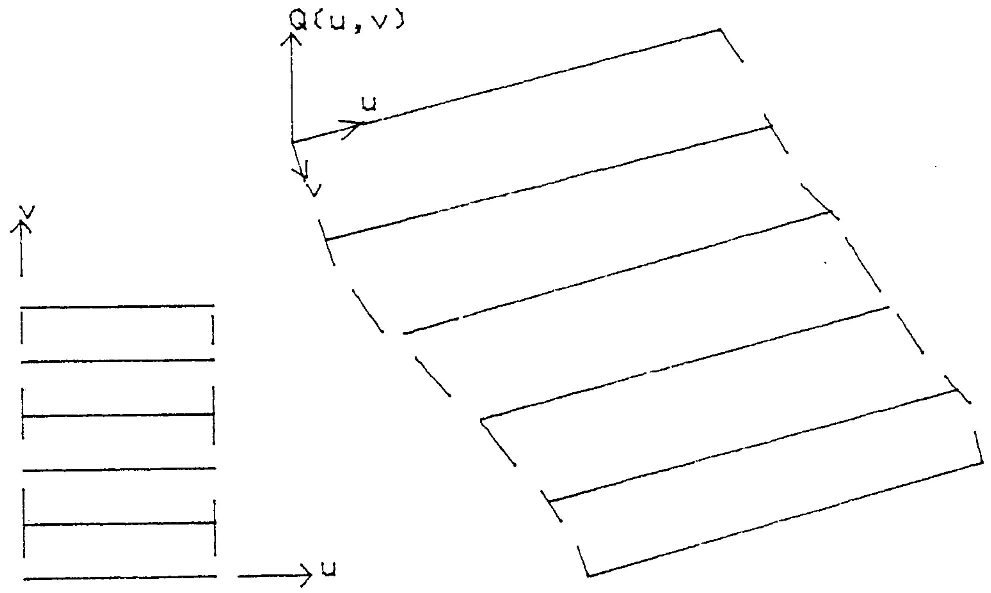


Fig. 3.4 Lofting (in u-direction)

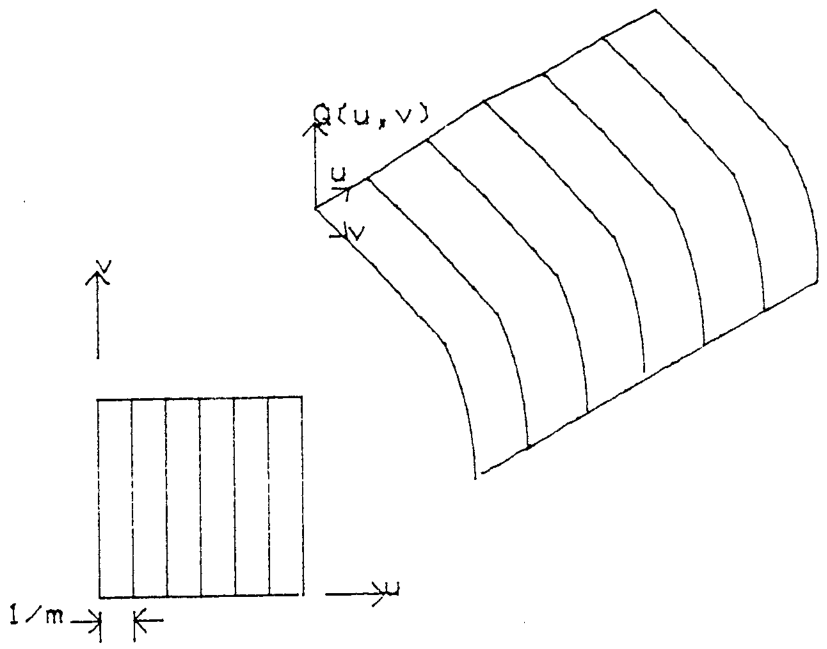


Fig. 3.5 Lofting (in v-direction)

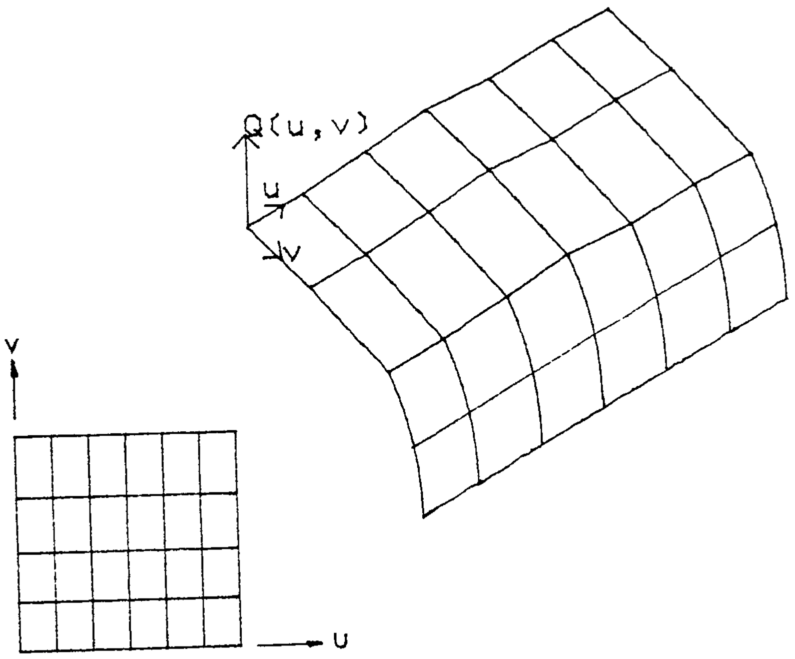


Fig. 3.6 Generalised Coons or transfinite definition

3.3.5 HIERARCHY OF SURFACE DEFINITION.

At the top of the hierarchy is the definition of the surface directly as a surface, e.g by specifying that the surface shall be conical shell,cylindrical or sculptured (free-form). Forrest [21] has shown that the cartesian and lofting forms of surface definitions are particular types of the extended Coons' surface definition. Therefore, in the hierarchy comes Coons' method followed by lofting and finally the cartesian product definition.

The choice of the type of surface definition depends on the application. For example, in the aircraft industry where the aerodynamic behaviour of the fuselage and wings are important, lofting is chosen. When stress analysis is the primary application, the cartesian product definition will suffice.

3.4 SOLID MODELLING

Three-dimensional solid geometry plays an important role in many engineering fields, and is vital to the industries which produce discrete goods. Solid modelling is a computer representation of physical solids and such reprsentation is a source of data for procedures which compute useful properties of the object. Solid modelling evolved in the early 1960's and could be traced to Robert's work [22]. Since then progress in solid modelling has been very rapid. The history of the field is well treated by Requicha and Voelcker [23] and we shall not attempt to repeat such treatment here. Solid modelling began with little theoretical support and various unrealiable theories were devised, however, an adequate theory, based on rigorous mathematics of topology was published by Requicha [24] in the late 1970's. Since then commercially accepted solid modellers have been tested by such a mathematical framework. Details of the mathematical framework for characterizing certain important representations are contained in reference [24].

Representation schemes for solid objects should satisfy the following criteria [24]:

Validity: it is required that there exist a real 3-D object corresponding to any given representation. We want to avoid a single line dangling in space or nonesense 3-D objects.

Completeness: All the operations we provide in a system must be applicable to all possible representations of solids within the schema used. For example, if hidden-line removal could be applied to convex solids but not to non-convex ones, we would consider the schema as incomplete.

Uniqueness: there should exist only one 3-D object corresponding to any given representation. It therefore means that a representation scheme that is ambiguous is not unique.

Conciseness: the schema for representation of solids should not contain redundant information in the database.

Ease of creation and modification: in order to minimize computational effort during interaction with the 3-D model, the internal representation should be as close as possible to the mental schema that the operator prefers when building or mofifying a solid or an assembly of solids.

Efficiency: the internal representation determines the efficiency of the algorithms operating on the internal representation of solids. Since different representations may be better suited for different algorithms, it may be preferable to maintain some redundancy in the data model, while violating the principle of conciseness for the sake of greater efficiency.

Requicha [24] has classified six families of unambiguous schemes for representing solids. However, in reality only three of them are used by most of the commercial solid modellers: these are constructive solid geometry (CGS), boundary representation (B-rep) and sweep representation. These three schemes are discussed here and the reader is advised that the sweep representation is the least understood, in a theoretical sense.

3.4.1 CONSTRUCTIVE SOLID GEOMETRY (CSG) REPRESENTATION

In this representation an object is described in terms of elementary shapes, or primitives. The constructive solid geometry representation is based on a two-level scheme. On the second level, bounded primitive volumes are combined by Boolean set operations. The leaf nodes are either primitive leaves which represent subsets of three-dimensional Euclidean spaces (solid primitive shapes sized and positioned in space), or transformation leaves which contain the defining arguments of rigid motions. The branch nodes are operators, which may be either regularized union,intersection,or difference [25] or may be rigid motions (Fig.3.7). An explantion of why regularization is necessary is given in reference [26]. Each subtree that is not a transformation leaf represents a set resulting from applying the indicated combinational or motional operators to the sets represented by the primitive leaves. On the lower level, bounded volume primitives are defined on the bases of half-spaces (one half-space for a sphere, three for a circular cylinder, six for a square block). In simple cases, as when only rectangular blocks are used, the half-spaces may be defined by parameters (like size, position and orientation) associated with the volume primitive, rather than being represented explicitly in the schema.

The solid scheme is the most compact of all known representations, at least for the class of commonly machined parts. For a complex part, the number of leaf nodes is, in practice, approximately equal to the number of distinct surfaces on the part. Other possible advantages are the ability to emulate the manufacturing process in the tree and the ability to handle genus of parts efficiently. The main advantage of CGS is that it guarantees the validity of uniqueness of the model: a boundary representation can always be derived in a unique way.

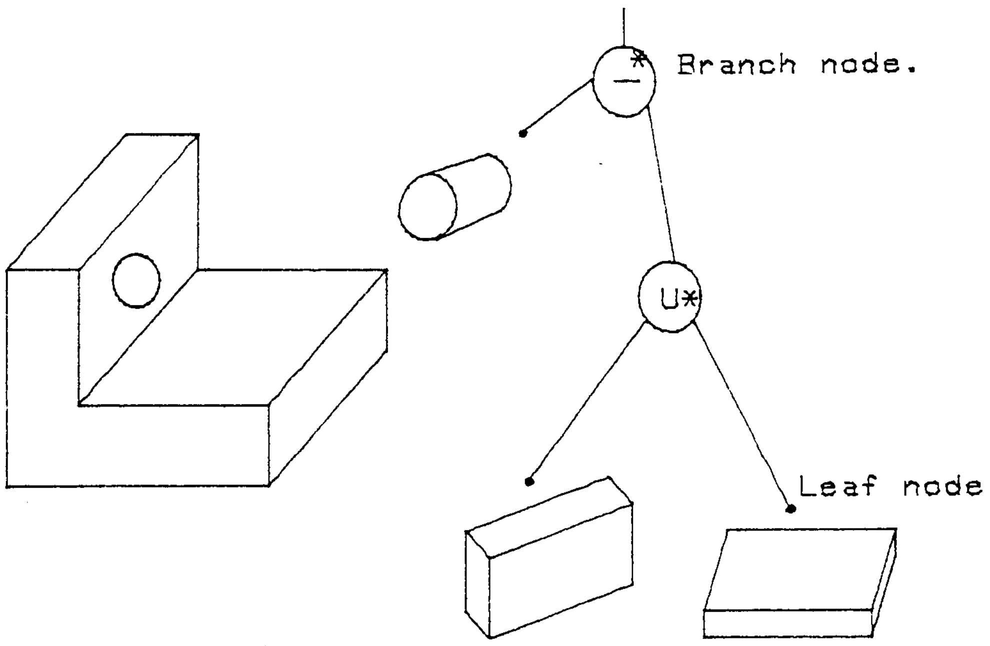


Fig.3.7 Constructive solid geometry (CSG).

3.4.2 BOUNDARY REPRESENTATION (B-REP) SCHEME.

In this representation, a solid is defined by its boundaries. Each surface is planar or sculptured and bounded by edges of an adjacent boundary. Boundaries of a solid usually are represented as unions of faces, with each face represented in terms of its boundary (union of edges) , together with data which define the surface in which the face lies. We distinguish between planar face and sculptured surface boundaries:

3.4.2.1 PLANAR FACE BOUNDARY REPRESENTATION

Fig.3.8 provides an example of a planar face boundary representation which is a graph containing object, face, edge and vertex nodes. Here, the information associated with the components of surface (face, edge and vertex) can be considered in two parts [27]. One is the geometry, including the physical dimensions and locations in space of each component. The other is the topology, describing the connections between the components. The geometry defines points,lines and planes. Topology regards a point as a vertex that bounds a line to define an edge. Similarly, a ring of edges provides bounds on a surface to define a face. Both geometry and topology are necessary for a complete shape description when the face is planar.

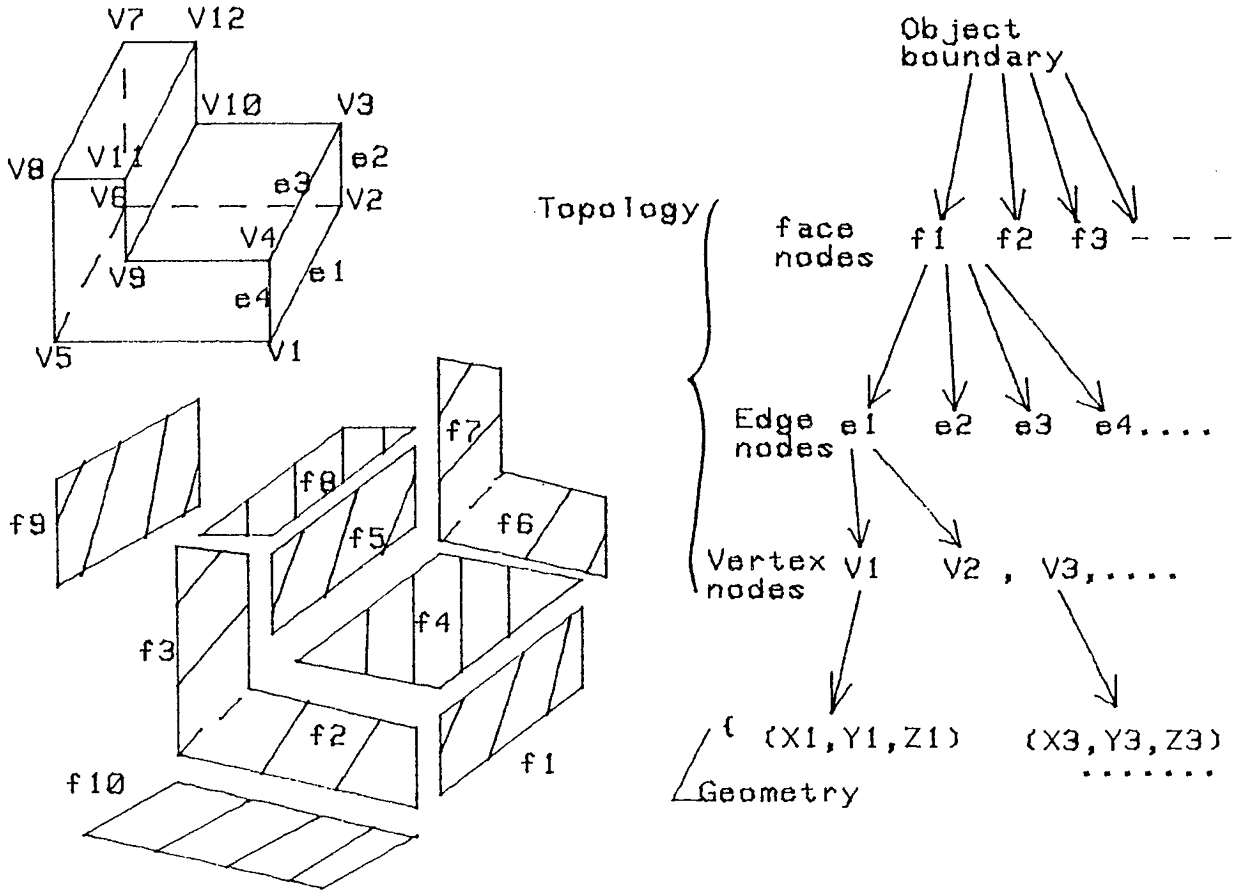


Fig. 3.8 A boundary representation

(graph of object,face,edge,and vertex nodes)

3.4.2.2 CURVED FACE BOUNDARY REPRESENTATION

Most practical engineering objects have planar as well as curved surfaces. Therefore there is the need to integrate simple and doubly-curved (sculptured) surface design capability into solid modelling functions. Even if we restrict our attention to the treatment of rather conventional machine parts, we must sometimes process various kinds of complex surfaces which arise as a result of rounding and filleting operations.

Kimura [28] has incorporated a complex surface design capability in a 3-D solid modelling system. A brief summary of the technique for handling such boundary representation is as follows:

(i) Use a local shape modification operations (such as "swing" (Fig.3.9a) and "round" (Fig.3.9b)) to construct a solid with curved surface

(ii) Generate pertinent surface patches by any known method of free-form surface design (e.g Be'zier surface technique).

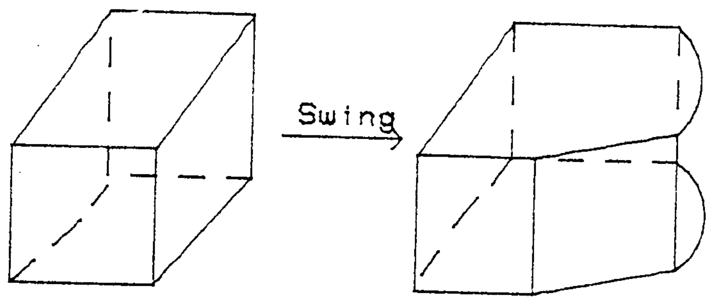


Fig. 3.9a Swing modification

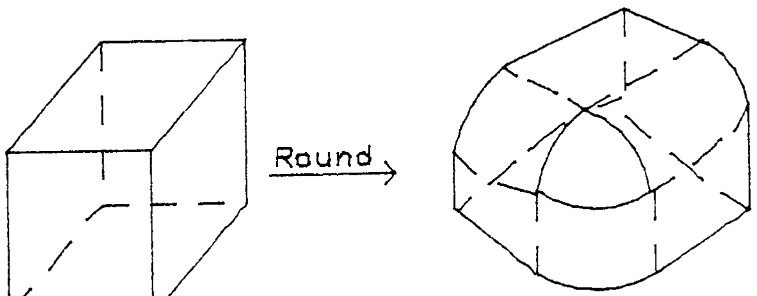


Fig.3.9b Round modification

Fig.3.9 Representation of local shape modification operations

3.4.3 SWEEP REPRESENTATIONS

The basic notion embodied in sweep schemes is that a "solid set" of points can be unambiguously represented as the cartesian product of an "area set" and a "trajectory set". The three main types of sweep schemes are translational, rotational and general sweeping.

3.4.3.1 TRANSLATIONAL SWEEPING

In this case, consider a 2-D set area A (see Fig.3.10) lying in a plane and a line segment B perpendicular to the plane, known as the trajectory set. As the area set moves along the trajectory, a solid is swept out. Representing such a 3-D solid reduces to that of representing the 2-D area set since the representation of the trajectory in this case is trivial.

3.4.3.2 ROTATIONAL SWEEPING.

A rotational sweeping scheme may be described in an analogous manner to translational sweeping except that the trajectory in this case is rotating. Such a scheme is shown in Fig.3.11 and is useful in representing axi-symmetric solids.

3.4.3.3 GENERAL SWEEPING.

Here, a curve may be swept along another curve as in Fig. 3.12. A dangling edge may occur which makes the resulting solid not to be homogenous. This type of sweep representation is not popular.

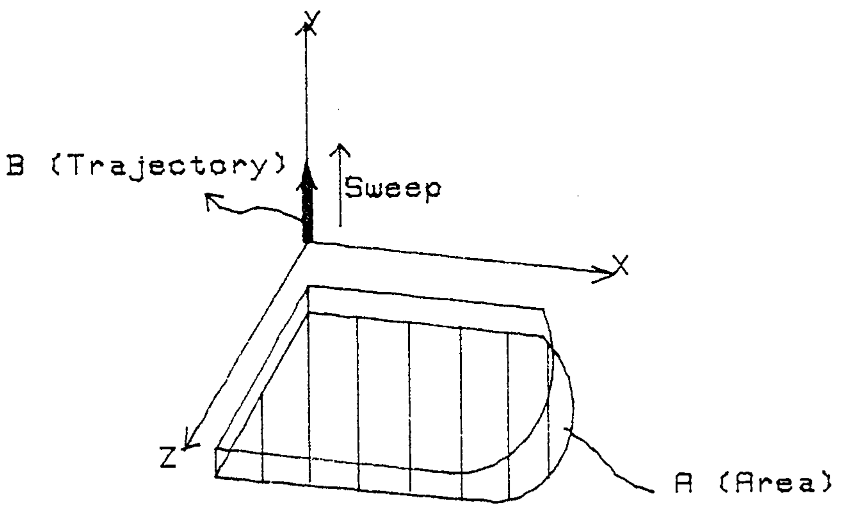


Fig. 3.10 Translational sweeping

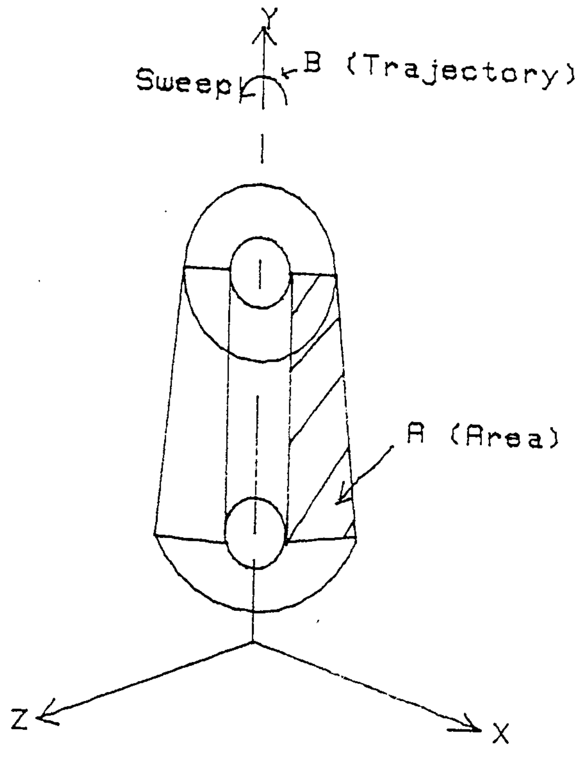


Fig. 3.11 Rotational Sweep

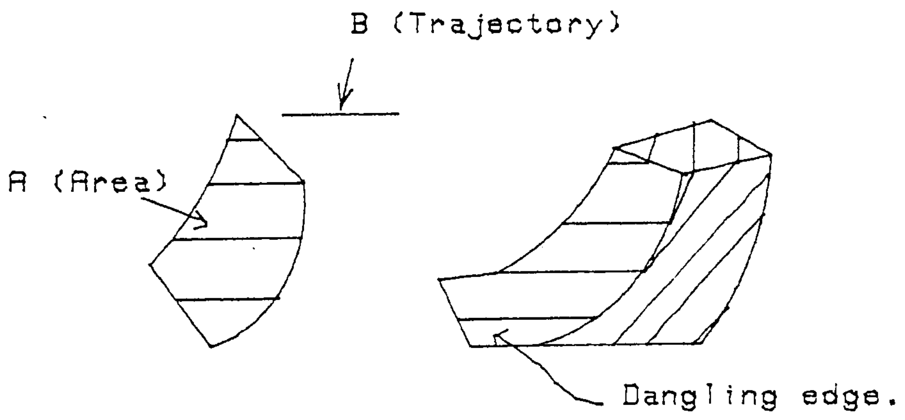


Fig. 2.12 General sweeping (may produce dangling edge).

3.5 CLOSING REMARKS

Since the present work is concerned with thin plate and shell structures , the surface modelling method was chosen for surface representation. 3D wireframe and solid modelling techniques are not suitable for the modelling of thin plates and shells.

The cartesian product and transfinite definitions in surface modelling are very useful tools when the geometry of the surface model is to be translated into fnite element analysis data since they produce meshed-surfaces. However , the cartesian product was chosen for this work because it is simple and yet very suitable for meshed surface representation.

In Chapter 5 , the Be'zier technique for surface modelling is presented. The mathematical foundations are summarized and a computer implementation based on the cartesian product definition is presented.

CHAPTER 4

FINITE ELEMENT MODELLING FOR STRESS ANALYSIS IN DESIGN: APPLICATION TO PLATE/SHELL STRUCTURES AND FAN IMPELLERS

4.1 INTRODUCTION

The finite element method is one of the most powerful methods for analysing a structure on a computer. In the finite element method, a continuum, with infnite degrees of freedom is approximated to be an assemblage of sub-regions each with a specified but now finite number of degrees of freedom. The behaviour of each sub-region or element is described by a set of assumed functions representing the stresses or displacement. The assumed functions are usually in the form of polynomials and an acceptable representation of the overall real situation is obtained by using a sufficient number of elements. In contrast to the Ritz method, all the integrations required to define the appropriate functional must be evaluated in a piecewise manner from element to element and the total contribution obtained by summation.

The advent of the digital computer in the early 1950's enabled the stress engineer, using the techniques of matrix algebra, to deal with problems which were considered complex. This represented the beginning of the finite element method as a significant tool for the stress engineer. It was not until the early 1960's that the stiffness finite element was formulated in terms of the principle of stationary total potential energy. Since then, the finite element method has been the subject of a very extensive literature with references such as [29] and [30] being typical of the many books available.

Until recently, the finite element method was so expensive that it was restricted to industries such as aerospace or nuclear engineering that required precise analysis almost regardless of cost. But now, mainframe computers and special programs required to do the analysis are being offered at low cost through timesharing and leasing. Minicomputers and terminals are more powerful and less expensive than ever. And modelling techniques have been developed to provide accurate results with lower labour and computer-processing costs. With these new features, finite-element analysis is spreading rapidly throughout general industry. Most new automobiles are analysed with the finite-element method. And the technique is being applied to construction machinery, agricultural equipment, pumps, air compressors, machine tools, home appliances, electric motors, fans, turbines, and numerous other products. Moreover, the technique is even being applied to many solid components that were hitherto considered too complex for rigorous stress analysis. These include parts such as engine blocks, engine heads and manifolds.

4.2 OUTLINE OF THE FINITE ELEMENT PROCESS

The displacement (stiffness) formulation is by far the most widely used form of the finite-element technique. The sub-regions or elements behave according to a prescribed function which describes an assumed displacement and / or stress field. The process can be summarised as follows:

4.2.1 DEFINITION OF THE FINITE ELEMENT MESH.

The continuum is discretized into appropriate sub-regions or elements. Elements are connected at points called nodes that form a network known as a mesh. The total pattern of elements representing the entire structure is called the model. Different types of elements are available such as rods, shells or cubes, depending on the geometry of the structure. In the case of a three-dimensional surface such as shell, it may be divided up into triangles and quadrilaterals. Distribution of elements may be uniform but they have to be clustered around regions of high stress level. A typical model is shown in Fig. 4.1.

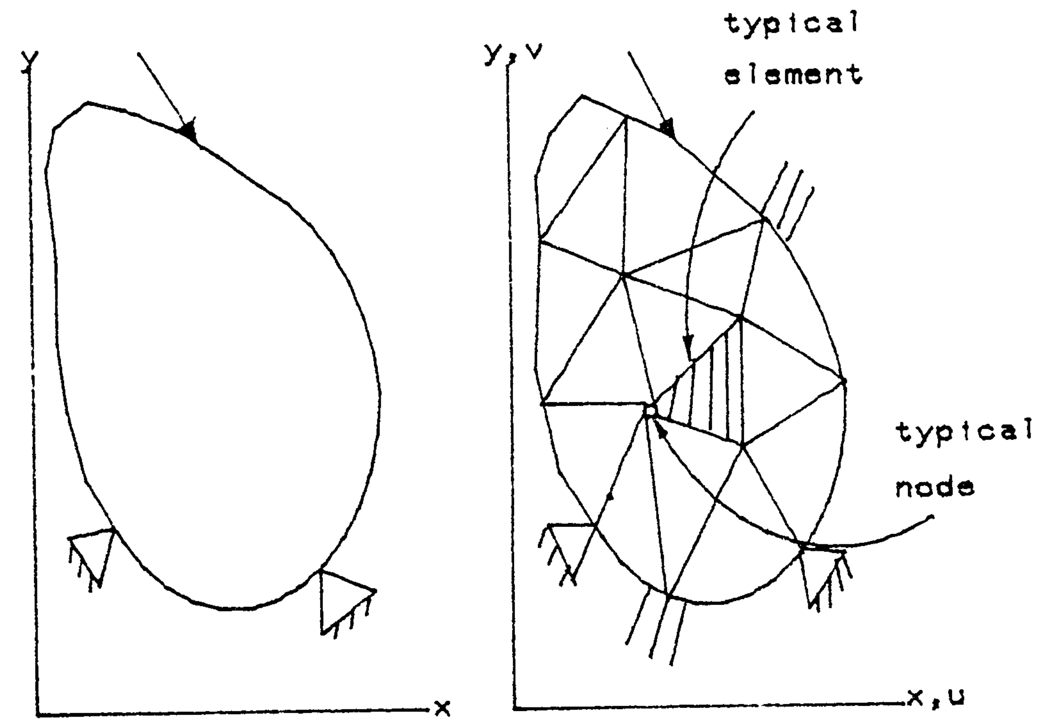


Fig. 4.1 Idealisation of two-dimensional structure using triangular elements

4.2.2 DISPLACEMENT FUNCTION.

The choice of the displacement function is crucially important as it governs the assumed element behaviour. Polynomial series are the most widely used approximating functions in the finite element method and the degree of the polynomial governs the ability of the element to approximate the true displacement field. For a desired accuracy in any particular problem, fewer higher order elements are required as compared to simpler ones. Usually, the same form of function is used in all the elements of discretisation, but mixed element types are acceptable. When different types of elements are used, then these functioms need to be different in the different types of element.

To ensure convergence to the correct result the choice of a suitable polynomial must meet the following requirements:

(i) The approximating function and certain of its derivatives must be continuous within the element and there must be compatibility between adjacent elements. This means that the chosen approximating function does not imply openings or overlaps between elements.

(ii) Displacement pattern must be able to accomodate rigid body movements.

(iii) The function must include the state of constant strain.

4.2.3 FORMULATION OF THE STIFFNESS EQUATION.

Based on the assumed displacement of the previous section, the strain distribution, and consequently the total potential energy of the discrete approximation to the continuum may be determined from

$$V = U + P = \sum_e (U_e + P_e) \qquad \ldots\ldots \quad 4.1$$

where U_e is element strain energy and P_e is the potential energy of the loading on the element.

The equilibrium condition $\delta V = 0$ leads to the stiffness equations

$$[K]\{q\} = \{Q\} \; . \qquad \ldots\ldots \quad 4.2$$

Here, [K] is a stiffness matrix, whilst {q} and {Q} are vectors of generalised coordinates and generalised forces respectively.

4.2.4 SOLUTION OF THE STIFFNESS EQUATIONS

A number of routines are available for solving the stiffness equation (4.2). These routines are commonly based on the Gaussian elimination or Cholesky decomposition processes. The stiffness matrix is symmetric and sparsely populated, and by suitable choice of nodal numbering scheme, the non-zero elements of the stiffness matrix are clustered in a narrow band along its leading diagonal.

Therefore, efficient routines exploit these properties in order to reduce the storage requirements demanded of the computer.

4.2.5 DETERMINING ELEMENT STRESSES AND STRAINS.

The element strains are easily calculated from the displacement shape function using the normal strain-displacement relations once the nodal displacements have been computed. Hooke's law is then applied to obtain the stresses. Equilibrium conditions are only satisfied in some average manner; consequently the stress field is discontinuous from element to element with the usual models employed.

4.3 REPRESENTATION AND DESCRITIZATION OF ARBITRARY SURFACES FOR FINITE ELEMENT ANALYSIS OF THIN PLATE AND SHELL STRUCTURES.

4.3.1 INTRODUCTION.

The finite-element method has proven to be a powerful tool that can be applied not only to thin plate and conventional thin shell structures with simple mathematical descriptions but also to so-called thin arbitrary shells. In practice, however, while the method has been widely used for structures in the former category, the relatively few reported applications to arbitrary shapes have been hampered by the difficulty in generating digital descriptions of the structures suitable for the analysis phase. Finite-element analysis requires the definition of a network of nodal points, each of which must be characterised by spatial coordinates.

Examples of arbitrary shells of practical interest include portions of various transportation structures such as aerospace, ships and rail vehicles, with the most obvious instance being automobile bodies.

There is a tendency for analysis of arbitrary or complex shell structures to use finite-element formulations based on simple geometry concepts, such as flat or facet elements for which only the three spatial coordinates of each node are required to define the geometry [14]. In addition, some formulations of thin doubly-curved or thin plate-shell structures having sharp corners and multiple junction regions are based upon geometric descriptions in terms of nodal positions only, such as the Semiloof element [13] , [14].

Related to the problem of the geometric representation of the thin plate-shell structures is the question of discretization for stress analysis. Not only do finite-element approaches require the identification of certain geometric parameters at nodes and other locations, but selection of the arrangement of nodes and elements is also a concern that becomes crucial when the geometry becomes complex or arbitrary. A capability for rapid mesh generation which embodies the calculation of all required geometric information is desirable if the finite-element method is to be implemented effectively for the analysis of arbitrary plate-shell structures.

Such a facilty permits significant shortening of the preprocessing and, when coupled with graphic devices allows mesh design, i.e. the consideration of a number of alternative descritisations prior to analysis.

The generation of a mesh on a surface in three dimensions may be considered a generalisation of the discretisation process used for two-dimensional planar domains. Three popular methods have been employed: laplacian, isparametric and transfinite mappings. In the laplacian mapping, one may generate locations of a number of specific boundary and interior points and construct the mesh by joining the points [31]. Kamel and Shanta [32] have described the application of this approach to three-dimensional surfaces. The second technique is isoparametric mesh generation in which curvilinear coordinates are constructed on regions with boundaries consisting of ralatively low order polynomials [33].The third approach, known as transfinite interpolation is an extension of the isoparametric curvilinear coordinate technique, in which generalized multivariate interpolation is used to map a domain such as a unit square onto the domain to be described. Gordon and Hall [34] demonstrated how this technique can be applied to surfaces in three-dimensional space.

The objective of the research project described in the present thesis has been to develop a unified approach to the geometric problems associated with the finite-element analysis of arbitrary thin plate-shell structures of which the fan impeller is typical. In particular, the representation of arbitrary surfaces has been effected in a manner which leads to natural approaches for the automatic discritization into a mesh of finite elements. The basis for the unified approach is the approximate representation of the surface geometry as discussed in section 3.3.

4.3.2 SURFACE REPRESENTATION AND SURFACE MESH FOR STRESS ANALYSIS.

Recall that the various surface representations have been discused in section 3.3, and that our objective is to effect a chosen representation of arbitrary surfaces in a manner which leads to natural approaches for the automatic discretization into a mesh of finite elements.In section 2.5, reasons for choosing the Be'zier technique (which is a cartesian product surface) for the geometric definition of fan impellers were given.

Therefore, given a Be'zier surface representation as shown in Fig.4.2 , the first step in the further preprocessing is the defintion of a finite element mesh appropriate for analysis.

The approach adopted in this research project is to use the parametric lines also employed in the display mesh for the finite-element description as shown in Fig. 4.2. In other words, the finite element mesh is just a simple mapping from a rectilinear mesh in parametric space, i.e., the u-v plane. The element boundaries always conicide with constant curvilinear coordinate lines. The numbering of the nodal points, which affects the bandwidth of the global stiffness matrix and consequently the computational efficiency, can be made nearly optimum.

Usually, such a mesh is made up of quadrilaterals. We must include facilities to obtain other types of elements such as triangular meshes, as well as being able to upgrade the simple quadrilateral and triangular meshes respectively. Details of such techniques for transfer of the simple quadrilateral to triangular, as well as upgrading them are discussed in section 7.

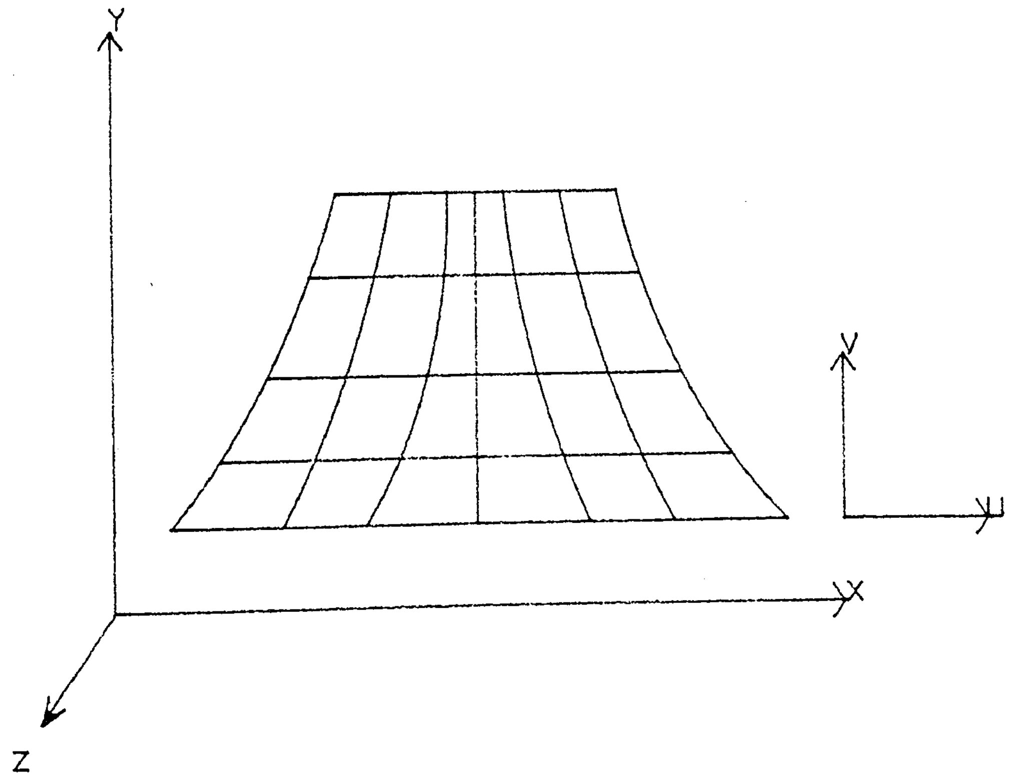


Fig. 4.2 Be´zier surface representation: (a possible scheme for generating finite element mesh on a surface)

4.3.3 THE SEMILOOF ELEMENT.

Since the Semiloof element was adopted for the development of the finite element stress analysis program to which the geometric modeller developed in this work is linked , it became necessary to summarize the geometric description of the element.

A large number of plate and shell structures are thin with sharp corners and multiple junction regions. The Semiloof thin shell element was developed by Irons [14] to meet the practical needs of analysing such complicated structures. The element is non-conforming and therefore owes its usefulness to the fact that it has passed the patch test [29]. Basically, the element adopts the well-known isoparametric concepts for geometrical and generalised displacement definitions and the version coded to date is an 8-noded parabolic model. The element has proved to be very efficient for engineering thin shell applications.

4.3.3.1 GEOMETRY AND NODAL CONFIGURATION.

The quadrilateral and triangular Semiloof shell elements are shown in Fig. 4.3 where the local coordinate system , isoparametric curvilinear coordinate system and global reference system are also illustrated. It is seen that three types of nodes are considered :

(i) corner and midside nodes at which three global displacement components are taken as nodal parameters.

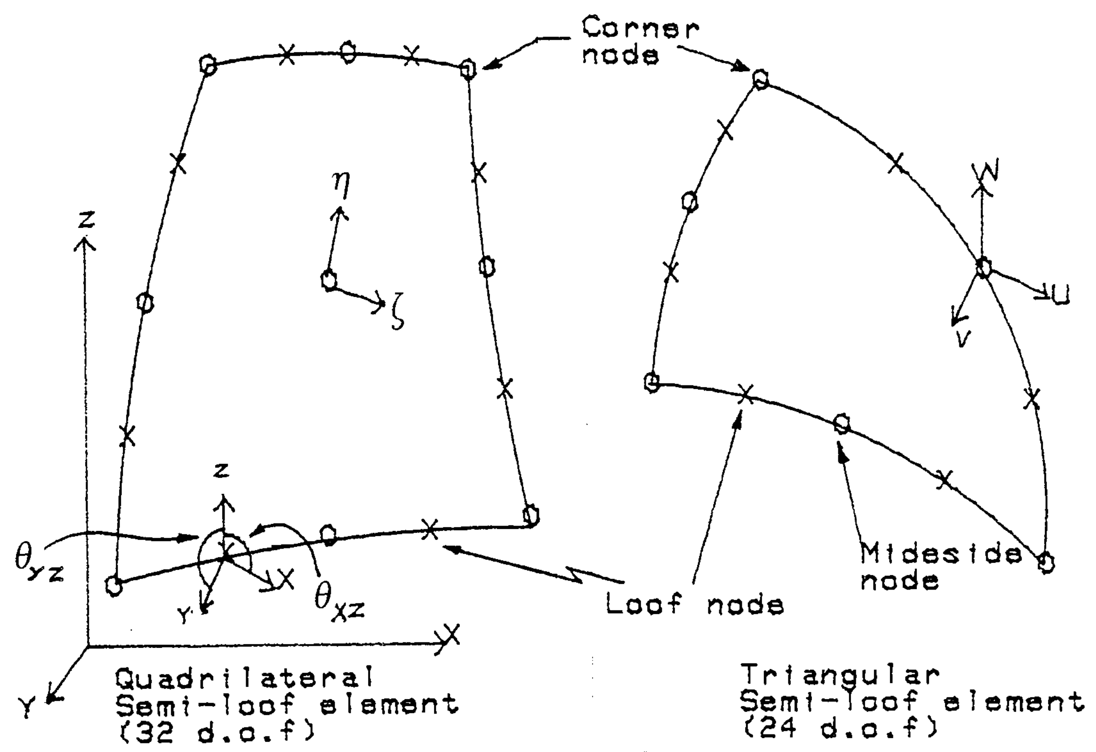

Corner node
η
ζ
z
z
θ_{yz}
θ_{xz}
X
Y
X
Y
W
U
V
Loof node
Midesside node
Quadrilateral Semi-loof element (32 d.o.f)
Triangular Semi-loof element (24 d.o.f)

Fig. 4,3 Semi-loof configuration.

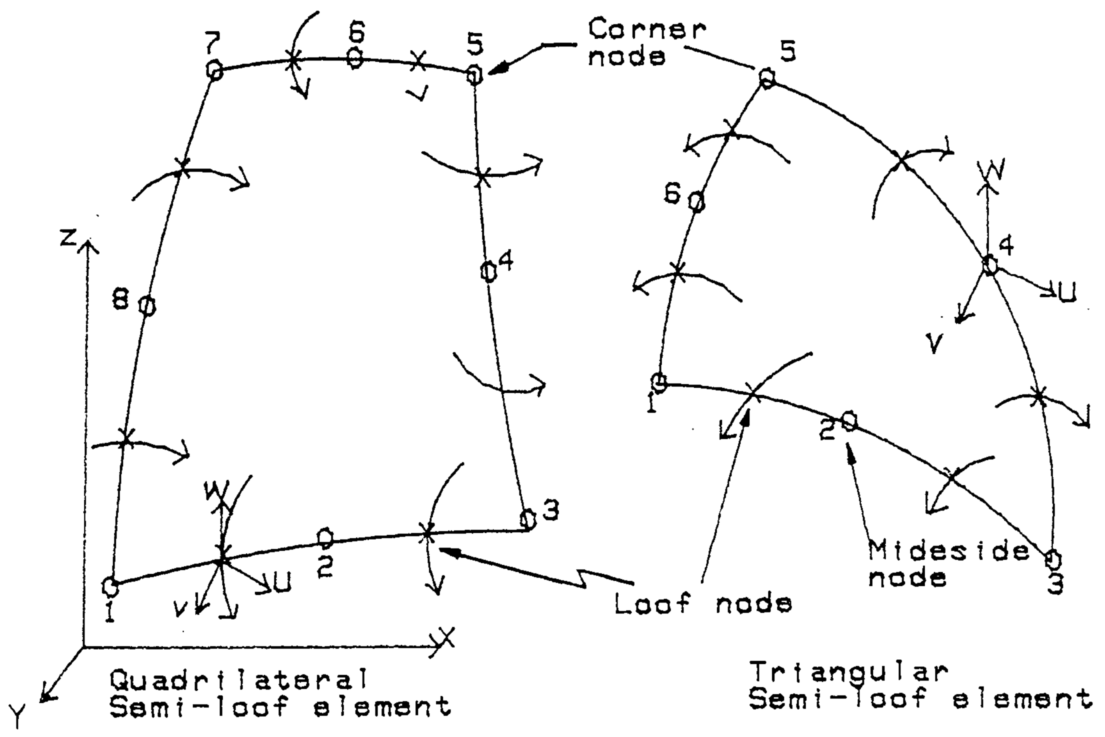


Fig. 4.4 Semi-loof element topology

(ii) loof nodes which are located at the Gauss points along the element sides. The nodal variables at the loof nodes are two rotations normal to the side at two points along each side, positioned at a distance of $1/2\sqrt{3}$ (side length) from the centre.

(iii) central node at which the nodal parameters are chosen to be the three local displacement components together with the two rotations.

These will give a total of 32 d.o.f. for a quadrilateral and 24 for a triangular element. The numbering sequence is ordered by starting at any corner node and progressing round the element as shown in Figs. 4.4 (a) and 4.4 (b).

4.3.3.2 SHAPE FUNCTION POLYNOMIALS.

Semiloof element is a non-conforming which passes the patch test and the element adopts the well-known isoparametric 8-noded parabolic model. Some measure of C^1-continuity is maintained by the introduction of the normal rotation variables at the loof nodes on the element periphery.

The following shapes functions in terms of curvilinear coordinate system (ξ,η) are used for quadrilateral elements ($\xi_o = \xi_i \xi$, $\eta_o = \eta_i \eta$). (refer to Fig. 4.5)

(a) For corner nodes

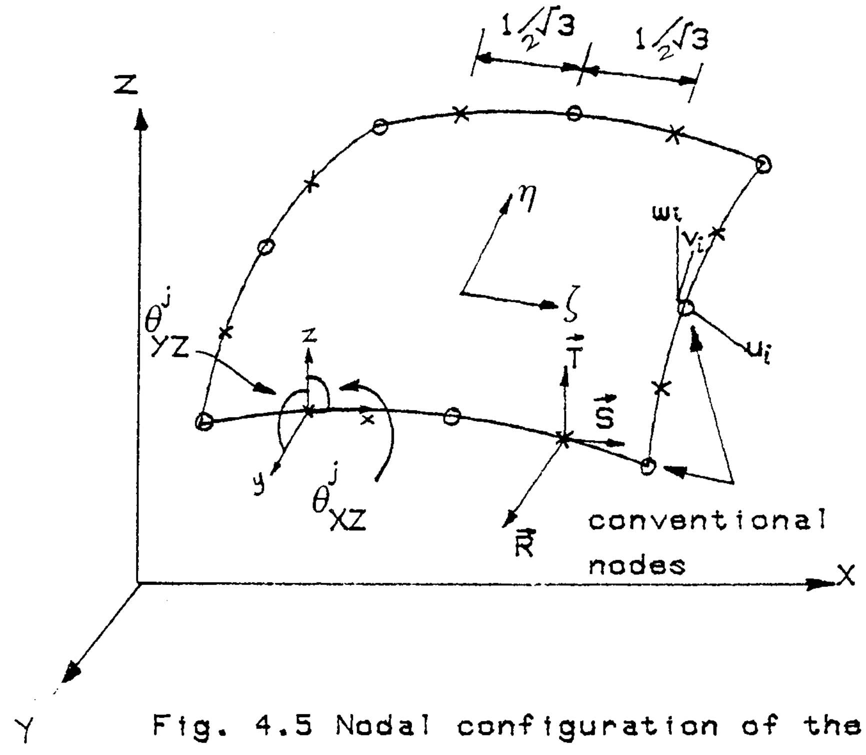


Fig. 4.5 Nodal configuration of the semiloof shell element (quadrilateral type)

$$N_i = 1/4\left(1 + \xi_o\right)\left(1 + \eta_o\right)\left(\xi_o + \eta_o - 1\right) \qquad \cdots 4.3$$

(b) For midside nodes

$$N_i = 1/2\left(1 - \xi^2\right)\left(1 + \eta_o\right), \quad \xi_i = 0 \qquad \ldots 4.4$$

$$N_i = 1/2\left(1 - \eta^2\right)\left(1 + \xi_o\right), \quad \eta_i = 0 \qquad \cdots 4.5$$

(c) For loof nodes (given in Ref [35])

$$N_i = 3/32\left(3\xi^2 - \eta^2\right) + \frac{1}{8}\left[3\xi_o\left(1 - \eta^2\right) + 3\eta_o\left\{3\xi^2 + \eta_o - 1 + \frac{3}{32}\xi\left(\xi^2 - \eta^2\right)\right\}\right]$$

$$\xi_i = \pm 1 \qquad 4.6$$

$$N_i = 3/32\left(3\eta^2 - \xi^2\right) + \frac{1}{8}\left[3\eta_o\left(1 - \xi^2\right) + 3\xi_o\left\{3\eta^2 + \xi_o - 1 + \frac{3}{32}\eta\left(\eta^2 - \xi^2\right)\right\}\right]$$

$$\eta_i = \pm 1 \qquad \ldots 4.7$$

(d) Central node : the bubble function

$$N_c = \left(1 - \xi^2\right)\left(1 - \eta^2\right) \qquad \ldots 4.8$$

The consideration of this shape function has been given in Ref.(36).

4.4 CLOSING REMARKS

In this Chapter, the outline of the finite element process was presented. An important features of the process is the definition of the finite element mesh. This feature forms a basis for linking drafting and the rest of the finite element analysis process via geometric modelling. This is because bulk of the input data to a finite element analysis program is made up of the geometric and topological definitions of the finite element mesh while the remaining are the material properties , boundary and loading conditions. Therefore , a geometric modeller such as the one discussed in Chapter 5 which meshes the model should be most suitable for bridging drafting/finite element analysis gap.

The meshes resulting from the surface modeller of Chapter 5 can be translated for use as semiloof elements since only the nodal coordinates are required for the definition of semiloof elements which adopt the well-known isoparametric concepts for geometric definitions.

CHAPTER 5

THREE-DIMENSIONAL SURFACE MODELLING USING BERNSTEIN-BE'ZIER METHODS AND COMPUTER IMPLEMENTATION.

5.1 INTRODUCTION.

For a long time , the automobile , aircraft and ship-building industries have developed various practical schemes, based largely upon the theory of linear approximation, for the mathematical description of arbitrary geometric shapes. This is part of a study that Forrest [37] has termed "computational geometry". According to Forrest,first we need a satisfactory and suitable general mathematical method for describing or , more appropriately, defining very general free-form curves and "sculptured" surfaces such as those found for instance on hulls, automobile and aircraft bodies. Moreover, an adequate interface between the undelying mathematical technique and the designer or user who may have little mathematical training is essential. A successful computer-aided design (CAD) system should appeal to the designer. It must be simple , intuitive and easy to use. Ideally, an interactive design system makes no mathematical demand on the user other than those to which he has been formerly accustomed through drafting and design experience.

One such CAD system - SYSTE'ME UNISURF , has been developed by Be'zier [38]. The system is different from most CAD systems in several ways. Be'zier approached the system from an instinctive geometric direction. The designer sketches a curve on the board , and draws an (open) polygon about the sketched curve (See fig. 5.1) which in a crude way mimics the gross shape property of the curve. Then , the vertices of the polygon are digitized and the corresponding "Be'zier curve" computed and automatically drawn by an automatic computer-driven drafting machine. Because of some descrepancy between the expected and drawn curves, the polygon is adjusted iteratively until an acceptable curve is obtained.

This way of designing curves is easily generalised to a method for designing surfaces simply by using the cartesian product form of the one dimensional case. The (open) Be'zier polygon becomes the "Be'zier net" , that is , a piecewise bilinear net. The relationship between a Be'zier net and a Be'zier surface is illustrated in Fig. 5.2.

The Be'zier technique is described in details in this chapter. A computer program was implemented using the Be'zier technique for surface representation. The computer program is capable of representing any free-form surface. In particular, we have applied it to the geometric representation of a fan impeller : reasons for and advantages of the choice of the Be'zier technique have been enumerated in section 2.4.

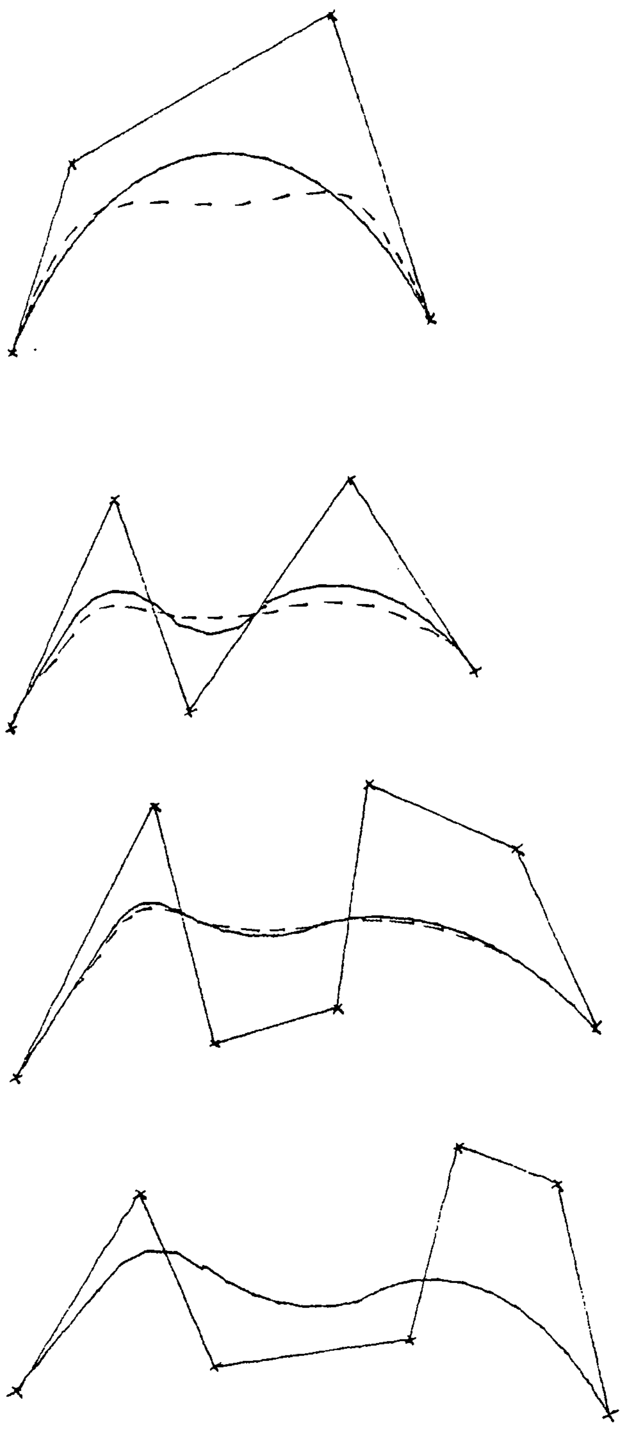


Fig. 5.1 Sequence of Bezier curves(full line) approximating a hand drawn curve (dashed line).

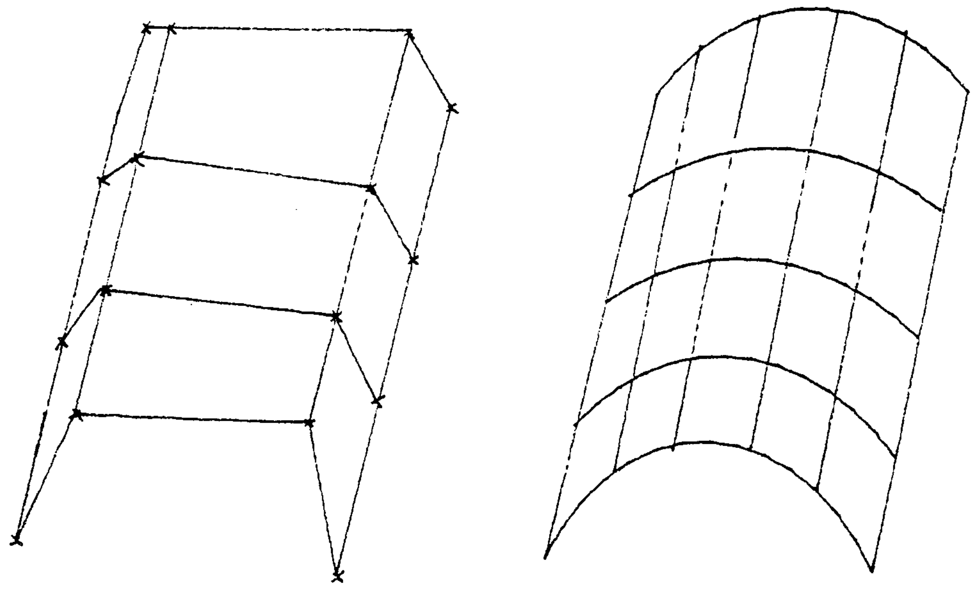


Fig. 5.2 Typical Be´zier surface and net

5.2 A REVIEW OF BERNSTEIN APPROXIMATION.

The basis of the Be'zier technique is the Bernstein polynomial approximation. Therefore , in order to understand the geometric properties of Be'zier surfaces , an understanding of the properties of the Bernstein polynomial approximation is essential. The approximation theoretic basis of Bernstein approximation is described here.

The Bernstein polynomial [39] approximation of degree m to an arbitrary function $f:[\emptyset,1] \rightarrow R$ is :

$$B_m[f;u] = \sum_{i=\emptyset}^{m} f(i/m)\ \emptyset_i(u) \qquad \cdots\cdots \quad (5.1)$$

where the weighting functions $\emptyset_i$ are , for $u \in [\emptyset,1]$, the discrete binomial probability density functions for a fixed probability u , given by

$$\emptyset_i(u) = {}^{m}C_i u^i (1-u)^{m-i} ; \qquad i=\emptyset,1,..,m \qquad \cdots\cdots \quad (5.2)$$

Fig.5.3 shows the graphs of the weights as a function of u.

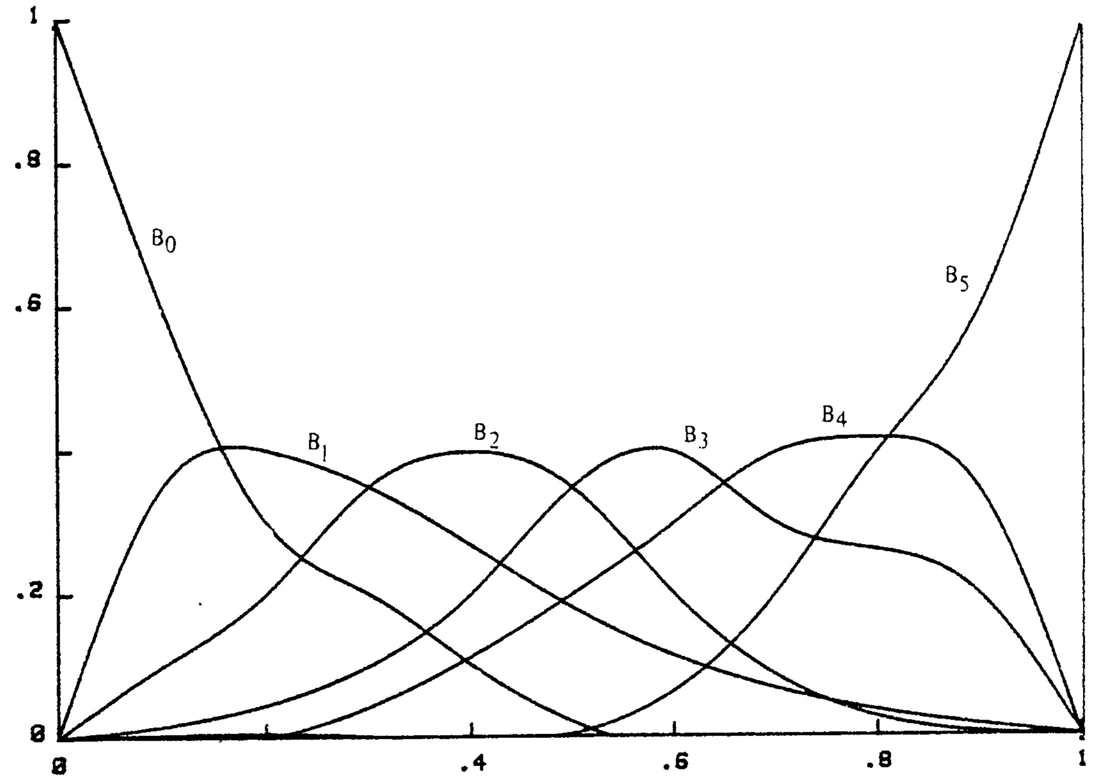


Fig. 5.3 The Bernstein basis functions for a linear space P_5 , i.e polynomials of degree $\leqslant 5$.

5.2.1 THE CONVERGENCE PROPERTIES

Consider a second degree (i.e. $m = 2$) Bernstein approximation to the function $f(u) = u^2$, where $u \in [\emptyset,1]$. Consider the following nodal points : $f(\emptyset)$, $f(1/2)$, $f(1)$ having corresponding nodal function values :

$f(\emptyset) = \emptyset$

$f(1/2) = 1/4$

$f(1) = 1$

Making use of these nodal values in equation 5.1 , we have :

$$B_m[u^2;u] = u^2 + u(1-u)/2 \qquad \ldots\ldots \quad 5.3$$

More generally , the mth degree Bernstein approximation to $f(u) = u^2$ is

$$B_m[u^2;u] = u^2 + u(1-u)/m \qquad \ldots\ldots \quad 5.4$$

which clearly shows the slow convergence (like i/m) of the Bernstein approximation $B_m[f]$ to the primitive f.

5.2.2 PROBABILISTIC INTERPRETATION.

Bernstein approximation theory has a probabilistic interpretation. From probability theory and statistics , we can give the following meaning to $B_m[f;u]$:

Let u be the probability of the occurrence of a given event in each of m Bernoulli trials. Then the probabilities of Ø,1,....,m occurrences of this are $\phi_{\emptyset}(u)$, $\phi_1(u)$, ..., $\phi_m(u)$, respectively. Moreover , if f(i/m) represents the "value" of obtaining exactly i occurences , then expression (5.1) is the expected "value" of the m+1 trials. Of course for u = Ø the expected value is simply f(Ø) ; and for u = 1 it is f(1).

In general , for any value of $u \in [\emptyset,1]$, $B_m[f;u]$ is a convex linear combination of the values of f at the m+1 nodal points $f(\emptyset)$,$f(i/m)$,...., $f(1/m)$, $f(1)$. Moreover, the approximation can be thought of as a statistically smooth convex linear combination of these nodal points.

5.2.3 THE BERNSTEIN BASIS.

The polynomials $B_m[f;u]$ are called Bernstein polynomials, and they form the Bernstein basis for the function f with degree m at most. Recalling some of the elementary properties of the binomial probability densities :

$$\phi_i(u) \geqslant \emptyset \qquad (i=\emptyset,1,..,m) \quad \sum_{i=\phi}^{m}\phi_i(u) = 1 \text{ for } u \in [\emptyset,1] \text{ —— } 5.5.$$

$$\phi_\emptyset(\emptyset) = 1 \; ; \; \phi_\emptyset^{(j)}(1) = \emptyset \quad ; \quad (j=\emptyset,1,....,m-1)$$

$$\left.\begin{array}{ll} \phi_i^{(j)}(\emptyset) = \emptyset \quad ; & (j=\emptyset,1,....,i-1) \\ \phi_i^{(i)}(\emptyset) = m! \,/\, (m-i)! & \\ \phi_i^{(j)}(1) = \emptyset & (\, j=\emptyset,1,.....,m-i-1) \\ \phi_i^{(m-i)}(1) = (-1)^{m-i}.\; m! \,/\, (m-i)! & \end{array}\right\} \quad \begin{array}{l} i = 1,2,..,m-i \\ \text{——— } 5.6 \end{array}$$

$$\phi_m(1) = 1 \; ; \; \phi_m^{(j)}(\emptyset) = \emptyset \quad (j=\emptyset,1,\ldots,m-i)$$

These relations serve completely to characterise the Bernstein basis $\{\phi_i(u)\}_i^m = \emptyset$ for the linear space P_m of the polygons of degree m.

Forrest [4∅] has shown that the maximum of such a function $\phi_i(u)$ occurs at the value $u=i/m$ $(i \neq \emptyset , m)$ and is

$$\phi_i(i/m) = {}^mC_i \, i^i (m-i)^{m-i}/m^m \qquad \ldots\ldots \qquad 5.7$$

Observe that for $i \neq \emptyset , m$:

$$\phi_i(u) < 1 \text{ for } u \in [\emptyset,1], \qquad \ldots\ldots \qquad 5.8$$

and that $\phi_\emptyset(\emptyset) = 1$, $\phi_\emptyset(1) = \emptyset$, and $\phi_m(\emptyset) = \emptyset$, $\phi_m(1) = 1$, which immediately imply that the two endpoint values $f(\emptyset)$ and $f(1)$ are , in general , the only values which are interpolated by the Bernstein polynomial. The graph of the Bernstein basis functions for m = 5 is shown in Fig. 5.3.

From the above relations involving the values of the $\phi_i(u)$ and their derivatives at the endpoints of the unit interval it is a simple matter to see that the endpoint derivatives of the Bernstein polynomial itself are given by :

$$\text{At } u = \emptyset : \frac{d^i}{du^i} B_m[f;u]\Big|_{u=\emptyset} = \frac{m!}{(m-i)!}\sum_{j=\phi}^{i}(-1)^{i-j}.{}^{i}C_j f(i/m)$$

$$\text{At } u = 1 : \frac{d^i}{du^i} B_m[f;u]\Big|_{u=1} = \frac{m!}{(m-i)!}\sum_{j=\phi}^{i}(-1)^{j}.{}^{i}C_j f((m-j)/m) \quad \ldots 5.9$$

It is seen from these last expressions that the i'th derivatives at the endpoints u = Ø , 1 are determined by the values of f(u) at that endpoint and at the i points nearest to the endpoint. For the first derevatives (i=1) :

$$B_m'[f;u]\Big|_{u=\emptyset} = m[f(1/m) - f(\emptyset)]$$

$$B_m'[f;u]\Big|_{u=1} = m[f(1) - f((m-1)/m)] , \qquad \ldots\ldots 5.1\emptyset$$

which means that the polynomial is tangent at the endpoints to the straight line joining the endpoint to the neighbouring interior point.

5.2.4 THE BERNSTEIN OPERATOR

B_m is a linear operator and has the property :

$$B_m[af + bg] = aB_m[f] + bB_m[g]$$

where a and b are real numbers.

Recall example 5.2.1 , for a second degree Bernstein approximation to the function $f(u) = u^2$:

$$B_m[u^2;u] = u^2 + \frac{u(1-u)}{m} = f + \frac{u(1-u)}{m} > f$$

We observe that $B_m[f;u] \neq f$.

5.2.5 BERNSTEIN APPROXIMATION FOR FUNCTIONS OF MORE THAN ONE VARIABLE.

The natural cartesian (tensor) product generalisation of the Berstein operator for the case of a bivariate function is the linear operator

$$B_{m,n}[f;u,v] = B_m B_n[f;u,v] = \sum_{i=\emptyset}^{m}\sum_{j=\emptyset}^{n} \emptyset_i(u)\emptyset_j(v)f(i/m,j/n) \qquad \cdots\cdots \quad 5.11$$

where $\emptyset_i$ and $\emptyset_j$ are the binomial densities of equation 5.2. $B_{m,n}$ is simply a product of the univariate operators B_m and B_n. These operators are commutative , as long as f is continuous :

$$B_m B_n[f] = B_n B_m[f] \qquad \text{......5.12}$$

The properties of the univariate Bernstein polynomial are carried over to the bivariate form. For example , $B_{m,n}[f]$ coincides with f , in general, only at the four corners of the (u,v) unit square — (Ø,Ø) , (Ø,1) , (1,Ø) , (1,1). However , along the four boundaries of the square , $B_{m,n}[f]$ reduces to the appropriate Bernstein polynomial approximation to the value of f on that edge. For example,

$$B_{m,n}[f;u,v]\Big|_{v=\emptyset} = B_m[f;u,\emptyset] = \sum_{i=\emptyset}^{m} \emptyset_i(u) f(i/m,\emptyset) \qquad \text{....5.13}$$

In the interior, the values of $B_{m,n}[f]$ are simply convex weighted combinations of the values of f at the (m+1)x(n+1) points of the cartesian product partition.

Four possibilites are considered in the Bernstein approximation for functions of more than one variable. These are the operators B_m and B_n themselves in parametric form , the cartesian product operator $B_m B_n$ in (5.11) and the Boolean sum operator given by

$$(B_m \oplus B_n)[f] = B_m[f] + B_n[f] - (B_m B_n)[f] \qquad \cdots\cdots \; 5.13$$

$$= \sum_{i=\emptyset}^{m} \emptyset_i(u) f(i/m, v) + \sum_{j=\emptyset}^{n} \emptyset_j(v) f(u, j/n) - \sum_{i=\emptyset}^{m} \sum_{j=\emptyset}^{n} \emptyset_i(u) \emptyset_j(v) f(i/m, j/n) \qquad \cdots\cdots \; 5.14$$

The cartesian product of equation 5.11 is a very particular case of equation 5.14 by replacing the arbitrary univariate functions f(i/m,v) and f(u,j/n) by their Bernstein approximations of degree n and m , respectively.

The Bernstein approximation for functions forms the basis for Be'zier curves and surfaces. The Be'zier polygon and net vertices correspond to the nodal points (See section 5.2.1) of the function being approximated.

5.3 BE'ZIER CURVES : GEOMETRIC APPLICATION OF BERNSTEIN APPROXIMATION.

Be'zier surfaces are made up of Be'zier curves in the u and v directions. Therefore , it is usual to study Be'zier curves in some detail and then extend the idea to Be'zier surfaces.

An arbitrary curve in the plane or space cannot be regarded as a single-valued function of one of the coordinates , irrespective of how the coordinate system is chosen. If we decide to represent all our curves as polynomial functions , of one variable , then the function becomes indeterminate when the curve has a vertical tangent at a point. Axis independence , and tangency conditions have led to the use of parametric or vector-valued methods for the representation of curves and surfaces. Thus instead of defining all our curves in the form $y = f(x)$ and surfaces in the form $z = f(x,y)$ we make use of what are known as vector-valued functions.

Planar curves simply take the form :

$$P(u) = [X(u) , Y(u)].$$

Space curves simply take the form :

$$P(u) = [X(u) , Y(u) , Z(u)]$$

and surfaces take the form :

$$P(u , v) = [X(u , v) , Y(u , v) , Z(u , v)]$$

where P is a point vector. Throughout the discussion that follows , the parametric form will be used.

5.3.1 VECTOR-VALUED DEFINITIONS

Let $P_i (i = \emptyset , 1 , \ldots\ldots , m)$ be (m+1) ordered points in three-dimensional Euclidean space. In what follows we will refer to the (open) polygon formed by joining successive points as the Be'zier polygon $P = P_{\emptyset}P_1 \ldots . P_m$.

The Be'zier curve associated with the Be'zier polygon P is the vector-valued Bernstein polynomial $B_m(u)$ given by

$$B_m(u) = \sum_{i=\emptyset}^{m} \emptyset_i(u)P_i \qquad \cdots\cdots \quad 5.15$$

where the $\emptyset_i(u)$ are the Bernstein basis functions of equation 5.2. Some examples of Be'zier curves are shown in Figs 5.4 and 5.5. It is observed that the curves are tangent at the endpoints to the straight line joining the endpoint to the neighbouring interior point (See section 5.2.3)

The right hand side of the expression (5.15) is essentially the formulation due to Forrest [4Ø]. This is an approximation to the discrete data $\{P_i\}$ rather than to a continuous primitive function that is defined over the interval [Ø,1].

Figs. 5.4 and 5.5 illustrate the relationship between the Be'zier polygon and the plane curves. Some interesting observations are apparent from these figures : that the curve $B_m(P_{\emptyset}, P_1, \ldots, P_m)$ is tangent at $P_{\emptyset}$ to the leg P_oP_1 and is tangent to the leg $P_{m-1}P_m$ at the second endpoint. This can be verified analytically by recalling expression 5.1Ø for the first derivative of a Berstein polynomial. From that , it is clear that the tangent vector at $P_{\emptyset}$ is

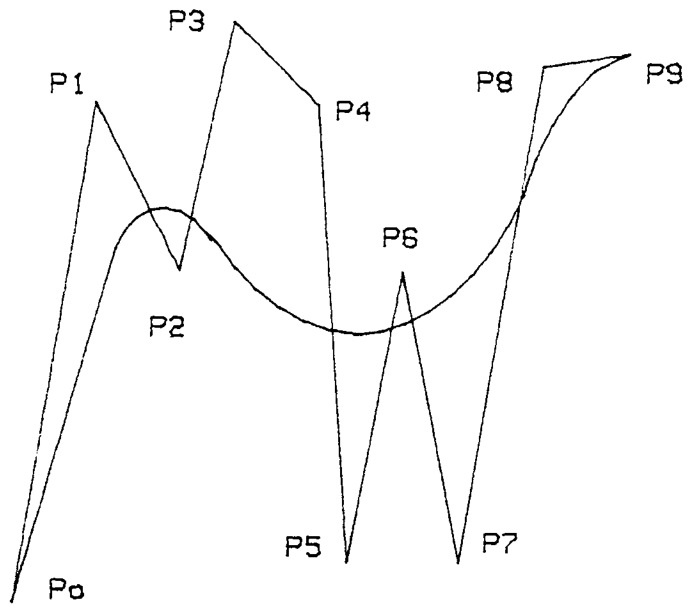


Fig. 5.4 A Bezier curve for
a 9-sided polygon

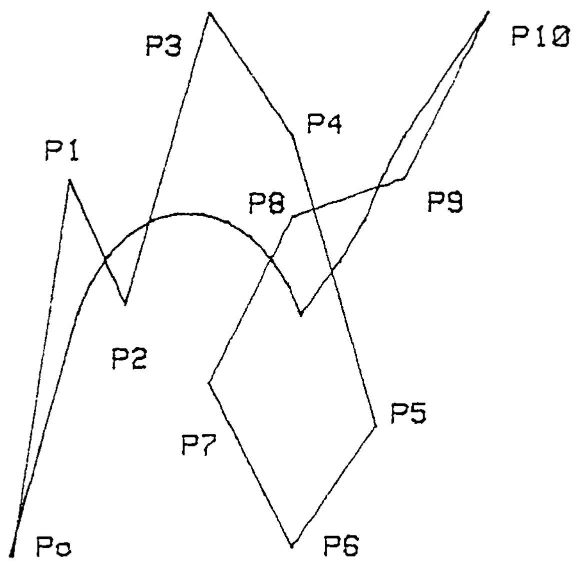


Fig5.5 A Bezier curve for a
10-sided polygon.

in the direction $P_1 - P_\emptyset$, and at P_m the tangent vector has the direction $P_m - P_{m-1}$.

5.4 BE'ZIER SURFACES.

The idea of Be'zier curves is extended to Be'zier surfaces by merely introducing vector-value functions for f(u,v) in the basic surface formulae for the cartesian product approximation (equation 5.11) and for the Boolean sum (equation 5.14). In the cartesian form of the extension , the Be'zier polygon is replaced with the Be'zier "net". $P = \{P_{ij}\ ,\ i = \emptyset\ ,\ 1\ ,\ldots.m\ ;\ j = \emptyset\ ,\ 1\ ,\ldots.n\}$

If we substitute the function

$$f(i/m, j/n) = P_{ij} \qquad \ldots\ldots 5.16$$

into equation 5.11 the result is the Be'zier surface B[P] that corresponds to the Be'zier net P.

$$B_{m,n}(u,v) = \sum_{i=\emptyset}^{m}\sum_{j=\emptyset}^{n} \phi_i(u).\psi_j(v).P_{ij} \qquad \ldots\ldots 5.17$$

where the ϕ_i and ψ_j are the usual binomial density functions of equation 5.2.

Consider now the cubic Be'zier curve. Since it is cubic , m = n = 3 and there is a control polygon consisting of the four control vertices $[V_{ij}]$, where i = Ø ,....,3 and j = Ø ,....,3. From equation 5.2 , the Bernstein polynomial is

$$Q_\emptyset(u) = (1-u)^3$$

$$Q_1(u) = 3u.(1-u)^2$$

$$Q_2(u) = 3u^2.(1-u)$$

$$Q_3(u) = u^3 \quad , \text{ for } u \quad [\emptyset,1] \qquad \text{........5.18}$$

Combining equations 5.2 and 5.15 , the Be'zier curve is

$$B_3(u) = (1-u)^3.P_\emptyset + 3u.(1-u)^2.P_1 + 3u^2.(1-u).P_2 + u^3.P_3$$

...5.19

These equations can be recast in matrix notation. From equation 5.15 , the curve can be expressed as

$$B_3(u) = [Q_\emptyset(u)\ Q_1(u)\ Q_2(u)\ Q_3(u)]\ [P_\emptyset ,\dots\dots,P_3]^t$$

From equation 5.18 , the polynomials can be written as

$$[Q_\emptyset(u)\ Q_1(u)\ Q_2(u)\ Q_3(u)] = [u^3\ u^2\ u\ 1].[B]$$

where

$$B = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & \emptyset \\ -1 & 3 & \emptyset & \emptyset \\ 1 & \emptyset & \emptyset & \emptyset \end{bmatrix}$$

From equation 5.19 , the curve can be rewritten in the following matrix form :

$$B_3(u) = [u^3\ u^2\ u\ 1].[B].[P_\emptyset, \ldots, P_3]^t \qquad \ldots\ldots \qquad 5.2\emptyset$$

One of the premises in the development of the Be'zier formulation was the relationship between the derivatives of the polynomial and the edges of the control polygon. From equation 5.19 , it is observed that

$$B_3(\emptyset) = P_\emptyset$$

$$B_3(1) = P_3$$

$$B_3^{(1)}(\emptyset) = m(P_1 - P_\emptyset)$$

$$B_3^{(1)}(1) = m(P_3 - P_2) \quad \text{,where } m = 3$$

In statement form , the curve begins at the first vertex ($V_Ø$) and ends at the last vertex (V_3) and is tangent to the control polygon at these vertices (see section 5.2.3).

A Be'zier surface is a tensor product of Be'zier curves. It is defined by a set of control vertices, in three - dimensional x,y,z space that is organised as a two dimensional graph with a rectangular topology. The set of control vertices is called a control hull (Be'zier net).(Refer to fig. 5.2.)

Given a Be'zier net [$P_{ØØ}$, $P_{Ø1}$,...,$P_{m,n}$] , equation 5.17 for the Be'zier surface can be written in matrix form as :

$$B_{m,n}(u,v) = [Q_Ø(u) \ ,Q_1(u) \ ....,Q_m(u)].[P] \ [\not{V}_Ø(v),..\not{V}_n(v)]^t \qquad ..... \quad 5.21$$

$$\text{where } P = \begin{bmatrix} P_{ØØ} & \cdots\cdots & P_{Øn} \\ & \cdots\cdots & \\ & \cdots\cdots & \\ & \cdots\cdots & \\ P_{mØ} & \cdots\cdots & P_{nm} \end{bmatrix}$$

The total number of Be'zier mesh in a Be'zier surface patch is determined by the number of subdivisions in the u and v directions. Suppose there are N_u divisions in u-direction and N_v divisions in v-direction as shown in Fig. 5.6. Then there a N_ux N_v rectangular grid is generated.

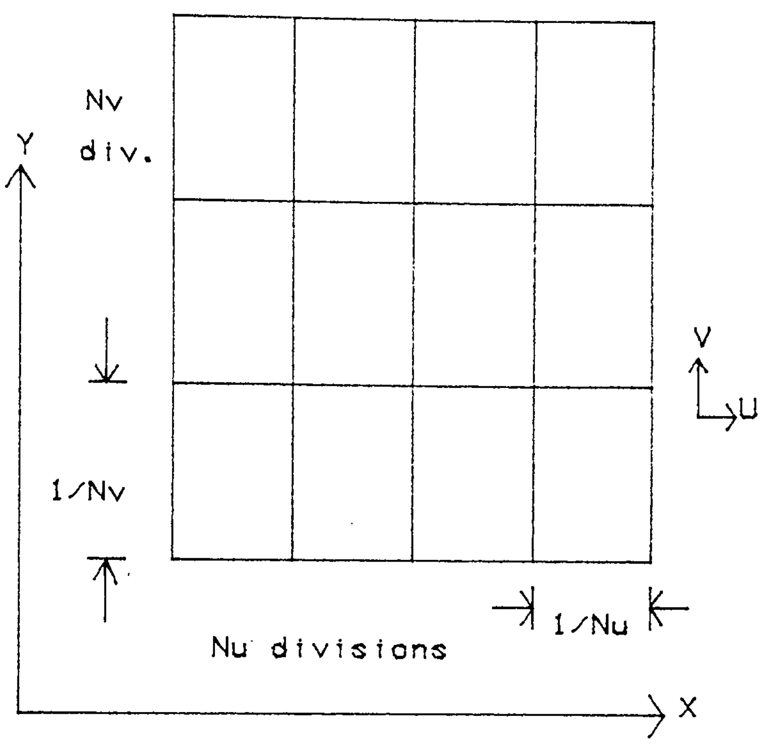


Fig. 5.6 Relationship between number of subdivisions in u-v directions and number of grids in a Bezier surface patch.

5.5 CONCENTRATING BE'ZIER MESH.

The surface modelling technique described so far is based on uniformly spaced nodes on each side of a Be'zier mesh. There are many instances , however , where , it is necessary to concentrate the Be'zier mesh (and consequently the finite element mesh after translation) since certain areas of the model could have stress concentration and a fine mesh will lead to greater accuracy in calculating stresses at such points. A Be'zier mesh concentration factor may be specified , to the surface modelling program developed in this section in both x and y directions which permits a Be'zier mesh to be concentrated in any part of the model. A model is shown in Fig. 5.7 with mesh concentrated in two parts by specifying a concentration factor S_x in the x-direction and S_y in the in the y-direction. Fig. 5.8a shows a sector with the Be'zier meshes concentrated in the inner edge (see fig. 5.8b) in order to increase the accuracy in calculating stresses in the model.

The global coordinates x_i and y_i of the nodal points in the basic mesh are given as follows :

$$X_i = \frac{S^{(i_x-1)} - 1}{S_x^{(n_x-1)} - 1}$$

$$\text{Similarly, } Y_i = \frac{S^{(i_y-1)} - 1}{S_y^{(n_y-1)} - 1} \qquad \ldots\ldots \quad 5.22$$

The optimum value of the scale factor is $1 < S_x, S_y < 3$ [65].

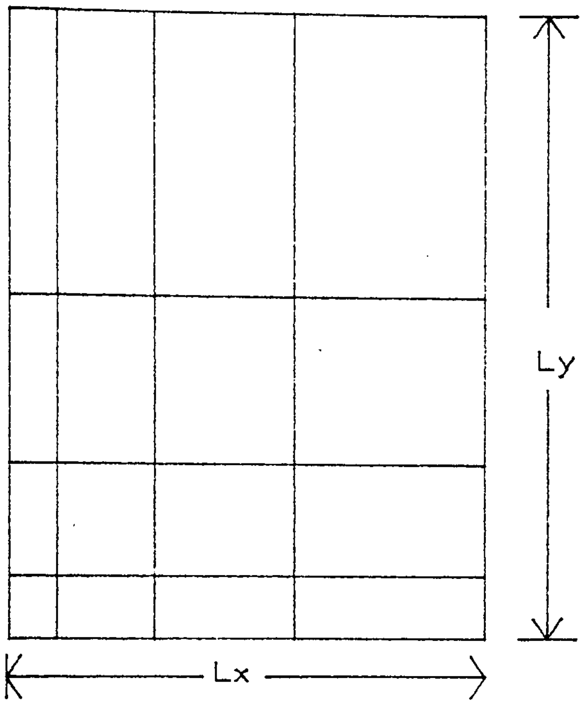


Fig5.7 Mesh concentrated around lower left-hand side of patch.

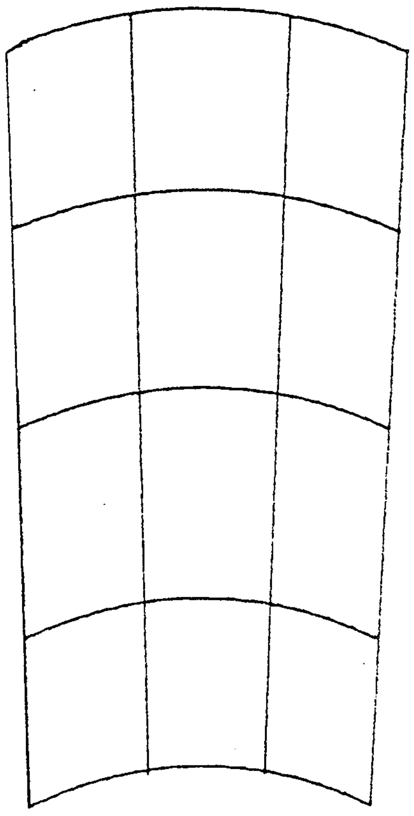


Fig. 5.8(a) Bezier surface with uniform grid size.

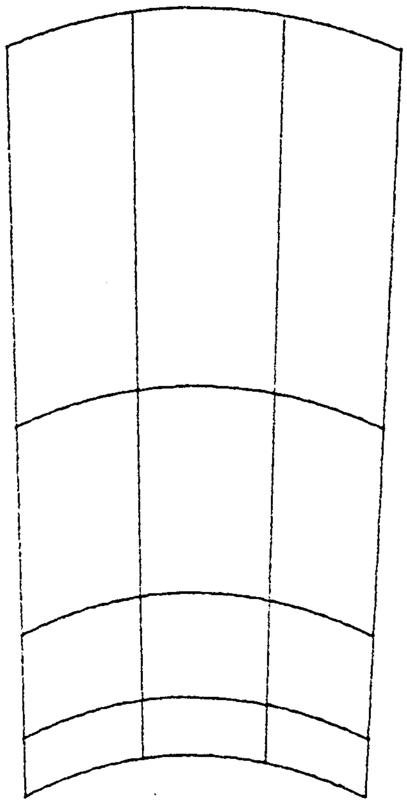


Fig. 5.8(b) The same patch with radial mesh concentration.

5.6 SOFTWARE IMPLEMENTATION OF THE 3-D SURFACE MODELLER

5.6.1 COMPUTER SYSTEM CONFIGURATION

The total package produced in this work is mounted on Hewlett-Packard 9845B(System 45) desktop computer. The system 45 has a usable screen size of 200mm by 162.5mm which can accomodate 560x455 dot resolution. The refresh rate is 60 Hz and display speed is 3180 mm/s.

The language available is an enhanced version of BASIC. The system ROM has 120K bytes and the read / write memory is 187K bytes. There are two tape cartridge units having capacity of 217K bytes each and an HP9845 flexible disc drive for disks having 500K bytes storage capacity.

Graphics display can be dumped onto either the built-in lineprinter , the external Hewlett-Packard 9827A flat-bed plotter or the Benson drum plotter.

5.6.2 STRUCTURE OF THE PROGRAM

This section describes the structure and operation of the program SUFACE developed to generate irregular surfaces in three-dimensional space using the Be'zier methods discussed already in this chapter. Recall that there are three basic ways in which a surface can be represented (section 3.3). The method employed in the work reported here is the cartesian (tensor) method of section 5.2.5 in which the surface is defined by a grid of points , only four of which (the surface corner points) lie on the surface. Examples of such cartesian product surfaces are UNISURF surfaces [38] (Be'zier operators) , Numerical Master Geometry [41] (spline operators) and FMILL [42] (hybrid of bicubic Hermite and spline operators). Of the three methods of surface representation , the cartesian (tensor) method is probably the best suitable for interactive design [4Ø] because no knowledge of the tangents at the endpoints of the Be'zier polygon is required by the user.

The objectives of the 3-D surface modelling system are :

1. The system should provide convenient geometric construction capabilities for models composed of surface-based entites.

2. The geometric data created by the modeller should be usable as input to application programs , such as finite element modelling and surface area and section properties calculations thereby making this system the "core" of a computer-aided engineering environment.

3. The interface should be compatible with a 2-D drafting system so that some "control" data needed for surface modelling could be derived from the 2-D drafting system when so desired.

5.6.2.1 SUFACE

This is the master program and it has been developed in a highly interactive manner so that the user finds it easy to use. The function keys of the HP9845 have been used so that the user only selects the key(s) appropriate to his/her need. When each function is completed , the model created is displayed and no further action takes place until another function key is depressed. The various function keys and the corresponding options available are shown in the flowchart of this program.

The remaining part of this chapter describes the subprogram SURFACE which is accessed using function keys #7 or #12 for the representation of models.

5.6.2.2 <u>Subprogram SURFACE</u>

This is the subprogram for surface representation and the following steps indicate how the processes carried out in the subprogram are organised :

1. The Be'zier net coordinates , the angle of orientation of the surface and the number of subdivisions in u and v directions are read .

2. The combination of m (the number of vertices in the u-direction)taken i at a time , where i = Ø , 1 , 2 ,.....,m ,is obtained and stored in an array. The same is done for n , where j = Ø , 1 , 2 ,...,n and these values stored in another array.

3. The boundary curves are drawn.

4. From the input data as well as the combinations obtained in step 2 , the curves of constant v are drawn in a piecewise form.

5. Steps 2 - 3 are repeated for drawing curves of constant u.

6. The nodal coordinates of grid corners are stored in an array in some logical order for further interpretation and are used in application programs such as finite element analysis , surface area and inertial properties of the model.

7. Input data could be modified and steps 2 - 6 repeated until the designer is satisfied with the model.

Following the flowchart for the program , it can be seen that the first thing it does is to read the coordinate values for the Be'zier net , starting from the left-hand position of the polygon (open) in X-Y plane and working to the right , then progressing from one polygon to the other in the X-Z plane. These , x , y , z coordinates are stored in arrays in an ordered form. These , together with the number of divisions in the u and v directions of a patch as well as the angles at which the surface is to be oriented in space , are input to the program via the keyboard or from a database if the suface modeller is interfaced with a 2-D drafting package.

From the given number of divisions in the u and v directions , the corresponding combinations are respectively obtained and the dot product of these combinations is obtained and stored in an array G for use in most of the subroutines.

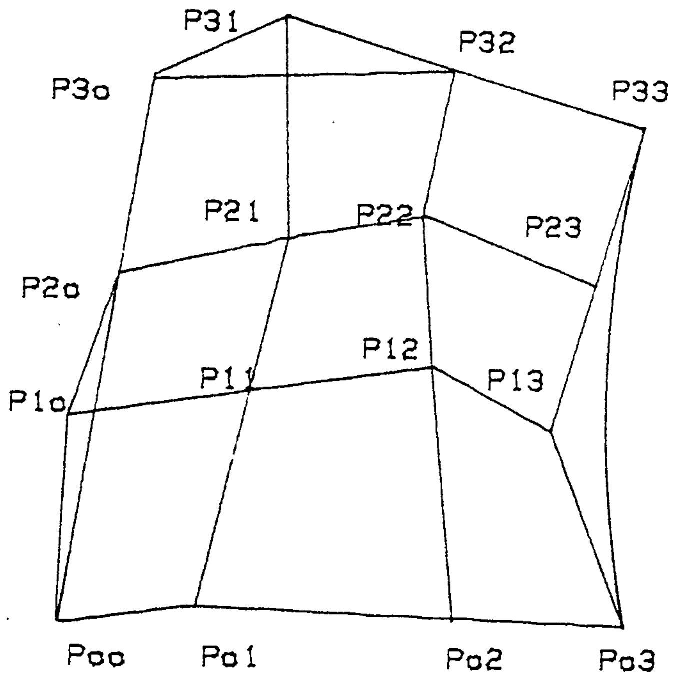
The next procedure to be accessed is XYBY , which determines the boundary curves in the u-direction. The first coordinate values in the first and last polygon (open) of the Be'zier net are required. The local coordinates can be specified using this information and the corresponding global cartesian coordinates are generated through the probability density function Q(u). A loop (having u values from zero to unity in steps of Ø.1) is established which controls this operation. Within the loop , each local coordinte value returned from the routine XYBY is termed a "new" coordinate value. The routine DRAW is then accessed to draw from the preceeding (old) coordinate to the new coordinate. In this manner the curves are drawn in a piecewise fashion.

A similar operation to that of the preceding paragraph is carried out for determining the boundary curves in the v-direction. This time , the procedure YZBY is accessed.

Having drawn the boundary curves , the internal grids have to be represented. Therefore the next procedure to be accessed is MESHXY , which determines the internal curves in the u-direction.The input data for this routine are the number of Be'zier meshes in the u-direction and the x , y , z coordinates of the polygon (open) within the Be'zier net. The curves are dealt with in turn , starting from the left-hand corner and traversing each polygon (open) , then progressing from one row of the polygon (open) to the other . The local coordinates of the piece-wise linear curves in the u-direction can be obtained using this informaton and the corresponding global cartesian coordinates generated , through the probabililty density function Q(u) and the corresponding combinations ${}^{m}C_{i}$. There are two loops (each control variable varying from zero to unity) that control this
operation , namely U which is an outer loop and V an inner loop. Within the loops the routine DRAW is accessed in a similar manner as the boundary curves representation.

A similar operation is carried out for determining the internal curves in the v-direction. However , the procedure MESHYZ is accessed. The corresponding global cartesian coordinates are generated through the probability density function Q(v) and the corresponding combinations ${}^{n}C_{j}$. Two loops (each control variable varying from zero to unity) control the operation. The outer loop is U while the inner loop is V. The DRAW routine is again accessed several times within these loops to draw the curves required.
Fig. 5.9 shows a Be'zier net and the corresponding surface.

Because we are interested in more than just a graphical representation of the surface , it is essential to record the coordinate values at the corners of the grids. To avoid duplication , this is done only after the routine MESHYZ has been accesed. These coordinate values are important for application programs that rely on the 3-D surface modelling database.



Bezier net

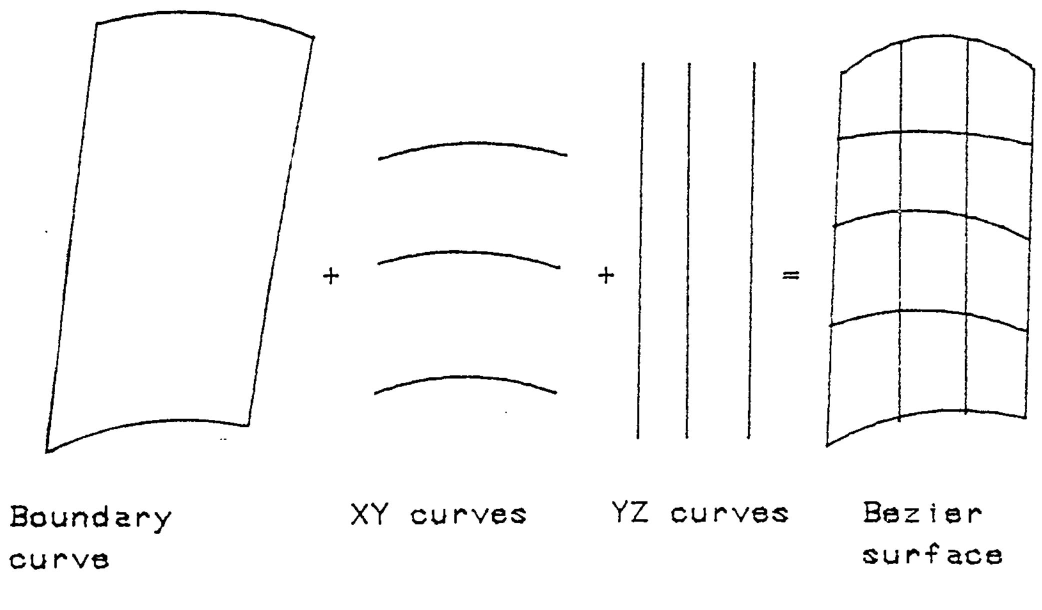


Fig. 5.9 Bezier surface (made up of boundary ,XY and YZ curves).

Fig.5.10 Flowchart for main program SUFACE

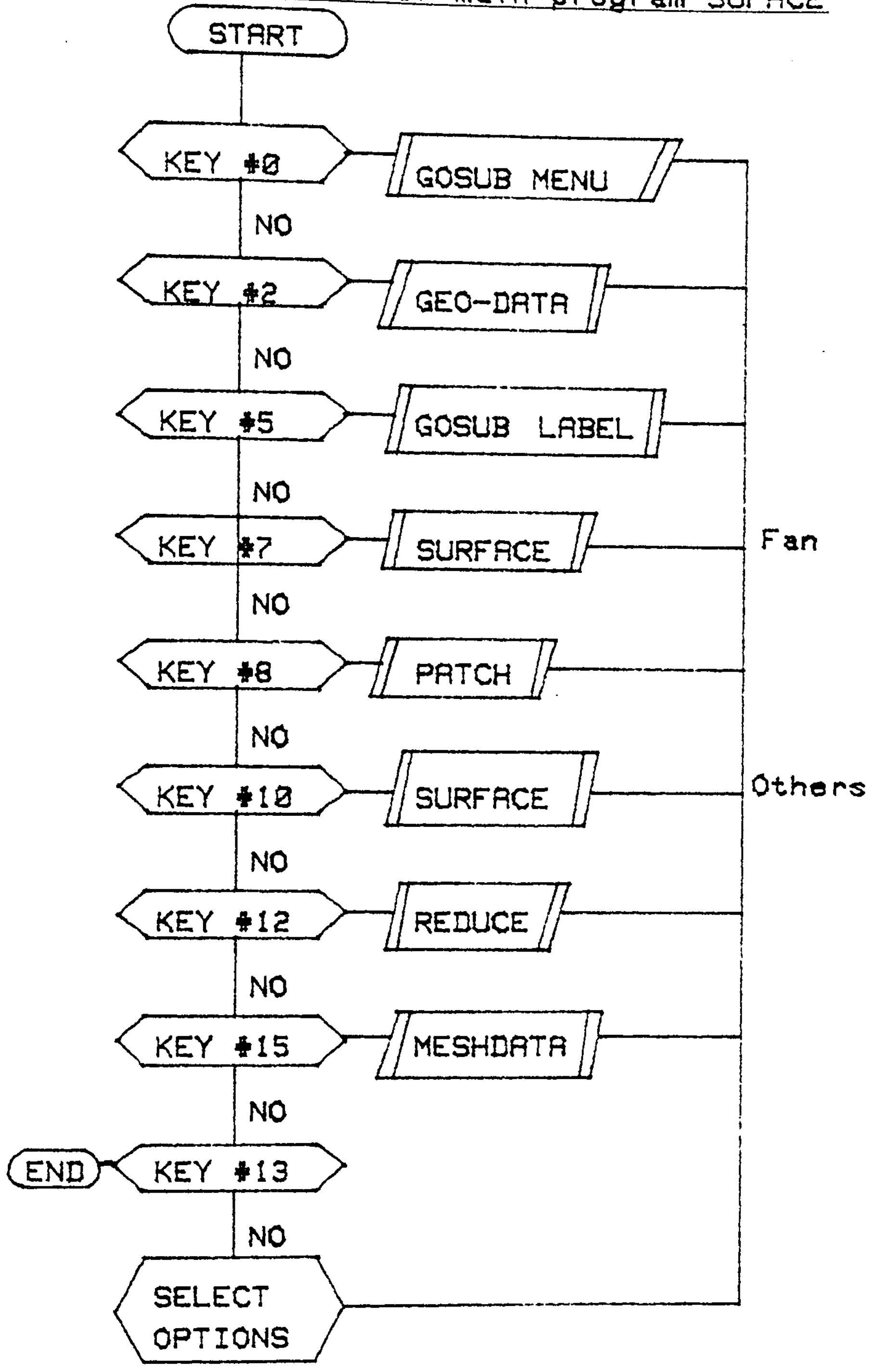
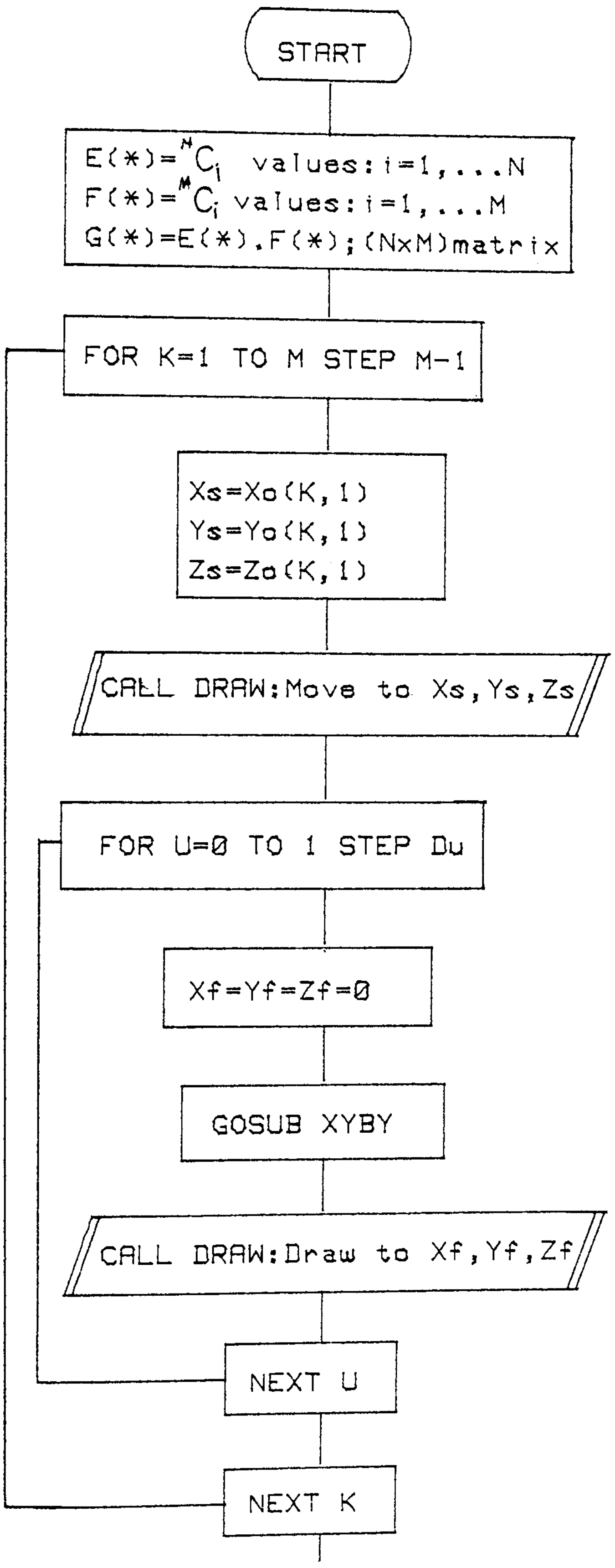
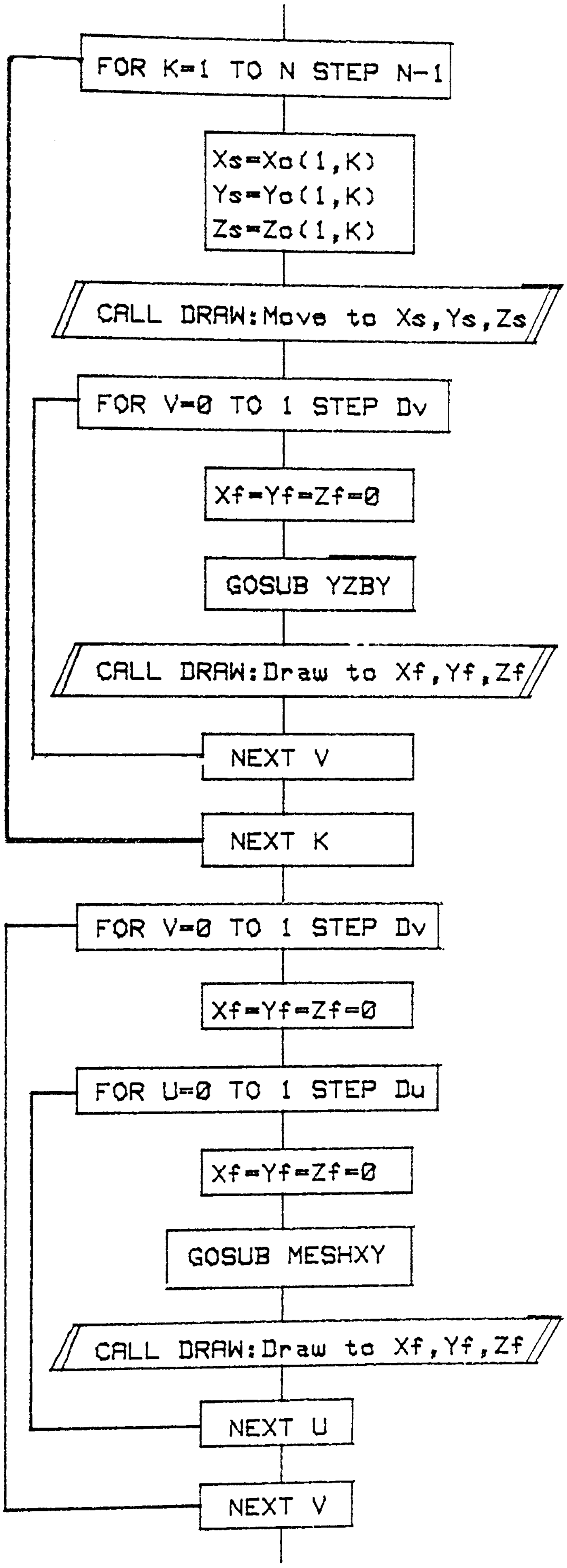
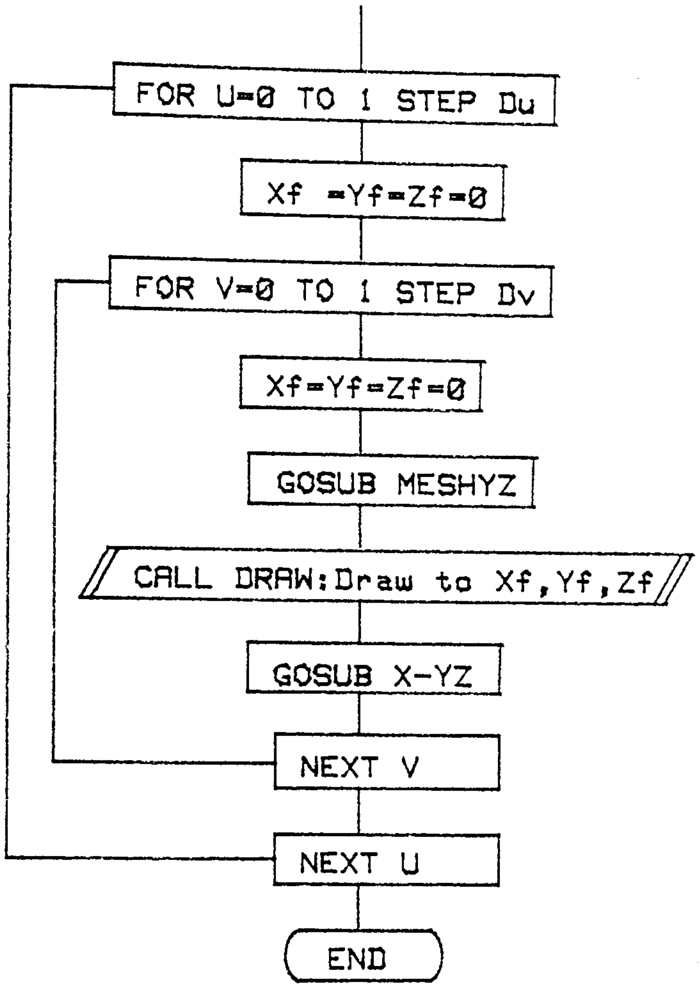


Fig. 5·11 Flowchart for subprogram SURFACE


START
E(*)=^NC_i values: i=1,...N
F(*)=^MC_i values: i=1,...M
G(*)=E(*).F(*);(NxM)matrix
FOR K=1 TO M STEP M-1
Xs=Xo(K,1)
Ys=Yo(K,1)
Zs=Zo(K,1)
CALL DRAW:Move to Xs,Ys,Zs
FOR U=0 TO 1 STEP Du
Xf=Yf=Zf=0
GOSUB XYBY
CALL DRAW:Draw to Xf,Yf,Zf
NEXT U
NEXT K

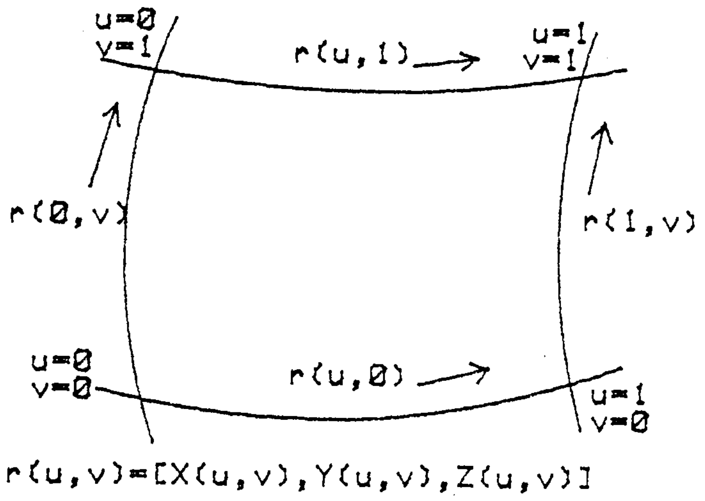

FOR K=1 TO N STEP N-1
Xs=Xo(1,K)
Ys=Yo(1,K)
Zs=Zo(1,K)
CALL DRAW:Move to Xs,Ys,Zs
FOR V=0 TO 1 STEP Dv
Xf=Yf=Zf=0
GOSUB YZBY
CALL DRAW:Draw to Xf,Yf,Zf
NEXT V
NEXT K
FOR V=0 TO 1 STEP Dv
Xf=Yf=Zf=0
FOR U=0 TO 1 STEP Du
Xf=Yf=Zf=0
GOSUB MESHXY
CALL DRAW:Draw to Xf,Yf,Zf
NEXT U
NEXT V



FOR U=Ø TO 1 STEP Du
Xf =Yf=Zf=Ø
FOR V=Ø TO 1 STEP Dv
Xf=Yf=Zf=Ø
GOSUB MESHYZ
CALL DRAW:Draw to Xf,Yf,Zf
GOSUB X-YZ
NEXT V
NEXT U
END

5.6.2.3 Subroutines XYBY and YZBY

Both of these subroutines function in the same way and compute the x , y , and z coordinates for the ends of the segments generated along the two boundary u-curves and the two boundary v-curves of each surface patch with parameters u and v running from zero to unity along the relevant boundaries as shown in the figure below.



The coordinates are computed from the vector-valued Bernstein-Be'zier polynomial written as

$$X = X(u,v) = \sum_{i=\emptyset}^{p} Q_i(u,v)x_i \quad ,$$

$$Y = Y(u,v) = \sum_{i=\emptyset}^{p} Q_i(u,v)y_i \quad , \text{ when } P = N \text{ , } u = \emptyset,..1 \text{ ;}$$

$$Z = Z(u,v) = \sum_{i=\emptyset}^{p} Q_i(u,v)z_i \quad , \text{ when } P = M \text{ , } v = \emptyset,..1 \text{ ;}$$

$Q_i(u,v)$ are the Bernstein basis functions and x_i , y_i and z_i are the coordinates of the control points on the Be'zier polygon along the boundary under consideration.

The steps for the routine are as follows :

1. The counter loop I or J is constructed and represents the number of control points in a Be'zier polygon along the boundary under consideration.
2. Within the loop , for the two u-curves , the Bernstein polynomials $Q_\emptyset(u,\emptyset)$, $Q_1(u,\emptyset),.....,Q_N(u,\emptyset)$ are evaluated with the value of u running from zero to unity. For the two v-curves , the Bernstein polynomials $Q_\emptyset(\emptyset,v)$, $Q_1(\emptyset,v),.....Q_M(\emptyset,v)$ are also evaluated with the value of v running from zero to unity. These are multiplied each time by the corresponding control vertex coordinates and summed up to give blended Be'zier curves.

Boundary u-curves and v-curves are very useful in representing the outline of a 3-D surface model when meshes in the interior of the surface patches are not required. Boundary curves can also be used in conjunction with subdivision to develop a fast occlusion algorithm for surface / surface intersections. If the boundary curves of two surface patches do not occlude one another , neither can the interior meshes of the surface patches. However , if they do interfere , subdivision can be used as a means of resolving the interference.

Fig.5.12 Flowchart for subroutine XYBY

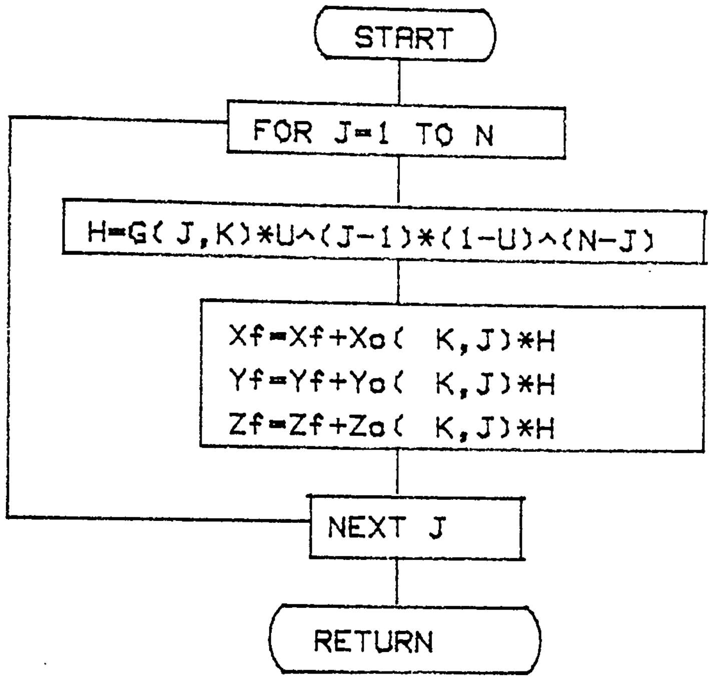
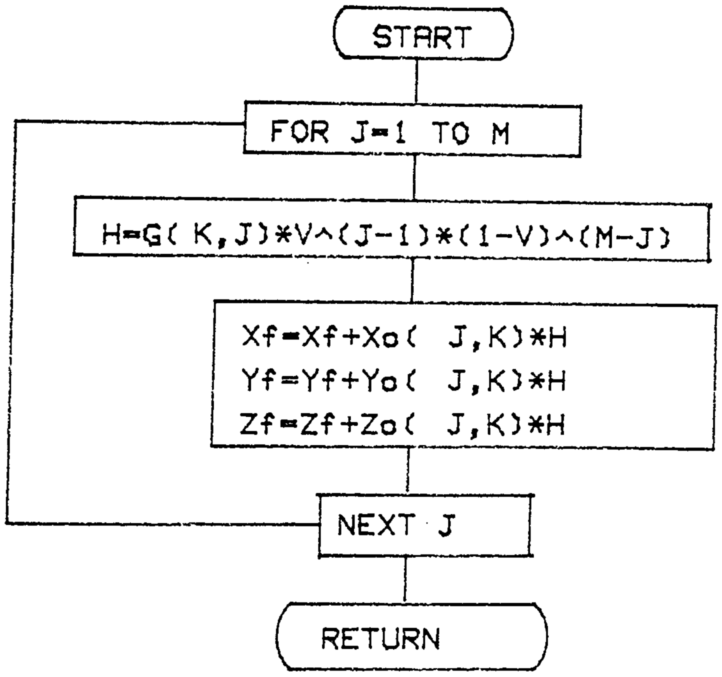
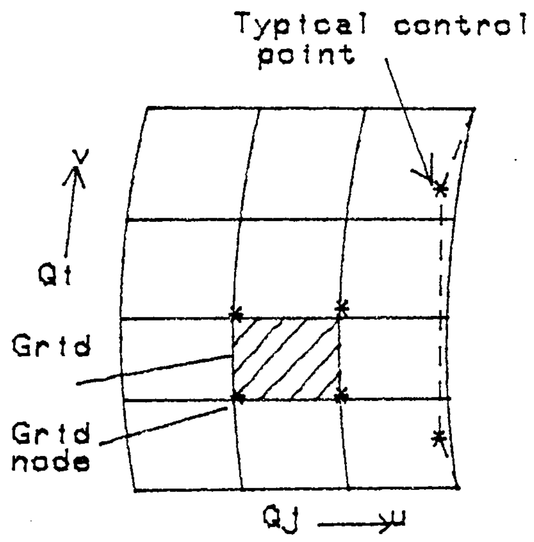


Fig. 5.13 Flowchart for subroutine YZBY



5.6.2.4 Subroutines MESHXY and MESHYZ

Again , both of these subroutines function in the same way and compute the x , y and z coordinates for the four corners of the surface grids. The technique adopted by this routine is the cartesian product form in which we extract vectors with constant u and v blending functions. The fixed vectors can be associated with the nodes of a rectangular grid in the parameter plane as shown in the diagram below. The fixed vectors are point vectors and are computed from the Bernstein-Be'zier surface approximation given by



$$X = X(u,v) = aX_{ij}$$

$$Y = Y(u,v) = aY_{ij}$$

$Z = Z(u,v) = aZ_{ij}$

where $a = \sum_{i=\emptyset}^{m}\sum_{j=\emptyset}^{n} Q_i(u,v)Q_j(u,v)$, and $Q_i(u,v)$, $Q_j(u,v)$ being the Bernstein blending functions. X_{ij} , Y_{ij} , Z_{ij} are the control vertices of the Be'zier net taken in order.

The expression $X = X(u,v) = \sum_{i=\emptyset}^{m}\sum_{j=\emptyset}^{n} Q_i(u,v)Q_j(u,v)X_{ij}$(applicable to y and z) is a weighted average of the $(m + 1)*(n + 1)$ control vertices with the Bernstein polynomials being the weighting factors and it defines an $(m \times n)^{th}$ - degree Be'zier surface [43]. In otherwords, suppose there are MxN vertices defining a Be'zier net , then the surface represented is an $(M-1)*(N-1)^{th}$degree Be'zier surface.

The steps of the routine can be identified as follows :

1. The outer loop has a counter L which represents the number of control points in a Be'zier net along one of the directions. The loop runs from zero to the number of control points in that direction minus one.

2. Along the direction under consideration , only one term of the blending function is obtained at a time and this is determined by the counter L. For example if $L = \emptyset$, then $Q_\emptyset(u,v)$ is obtained ; if $L = 1$, then $Q_1(u,v)$ is obtained , and so on.

3. An inner loop J is constructed , within which the blending functions in the other direction are determined , multiplied together with the blending function from step 2 , and in conjuction

with the associated control points , the global coordinates of each node of the surface grids are found.

Fig. 5.14. Flowchart for subroutine MESHXY.

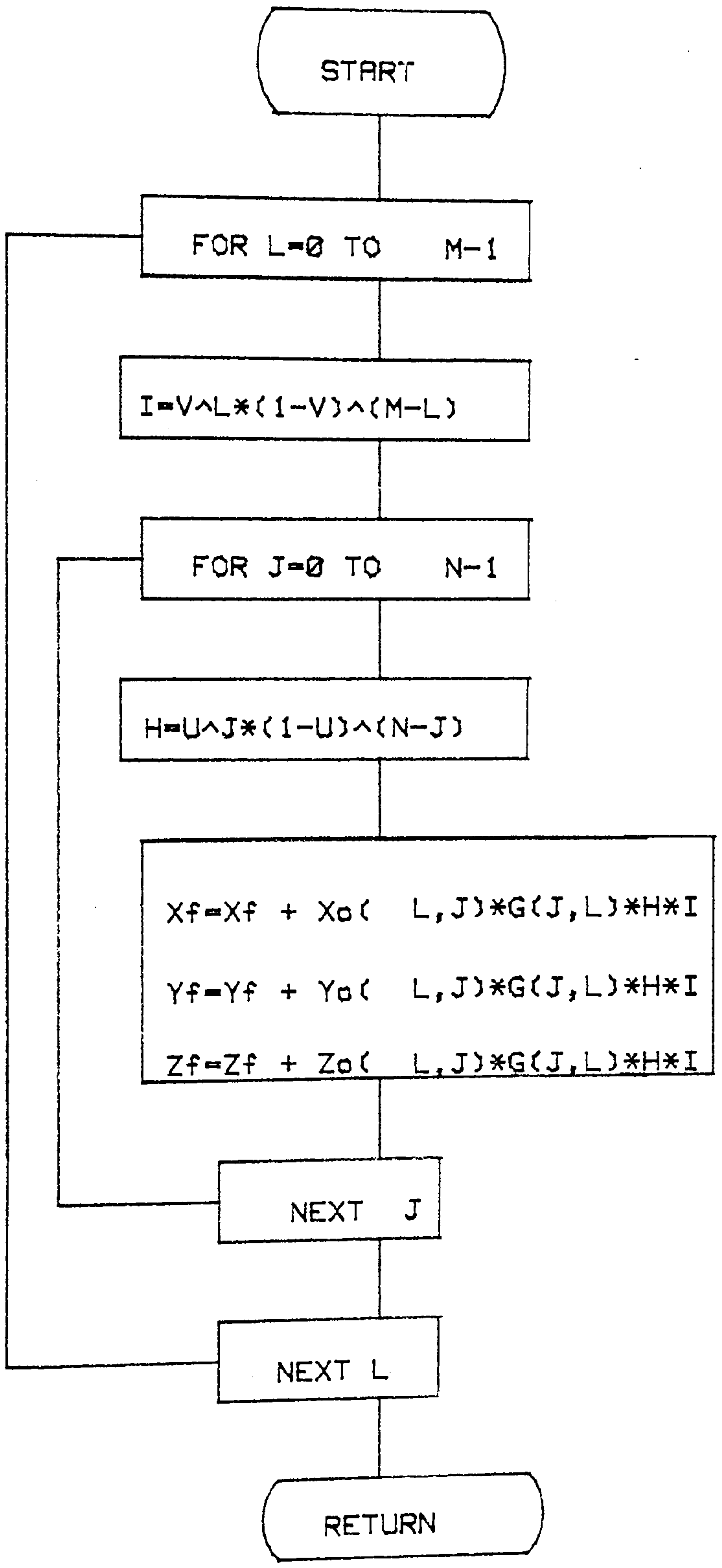
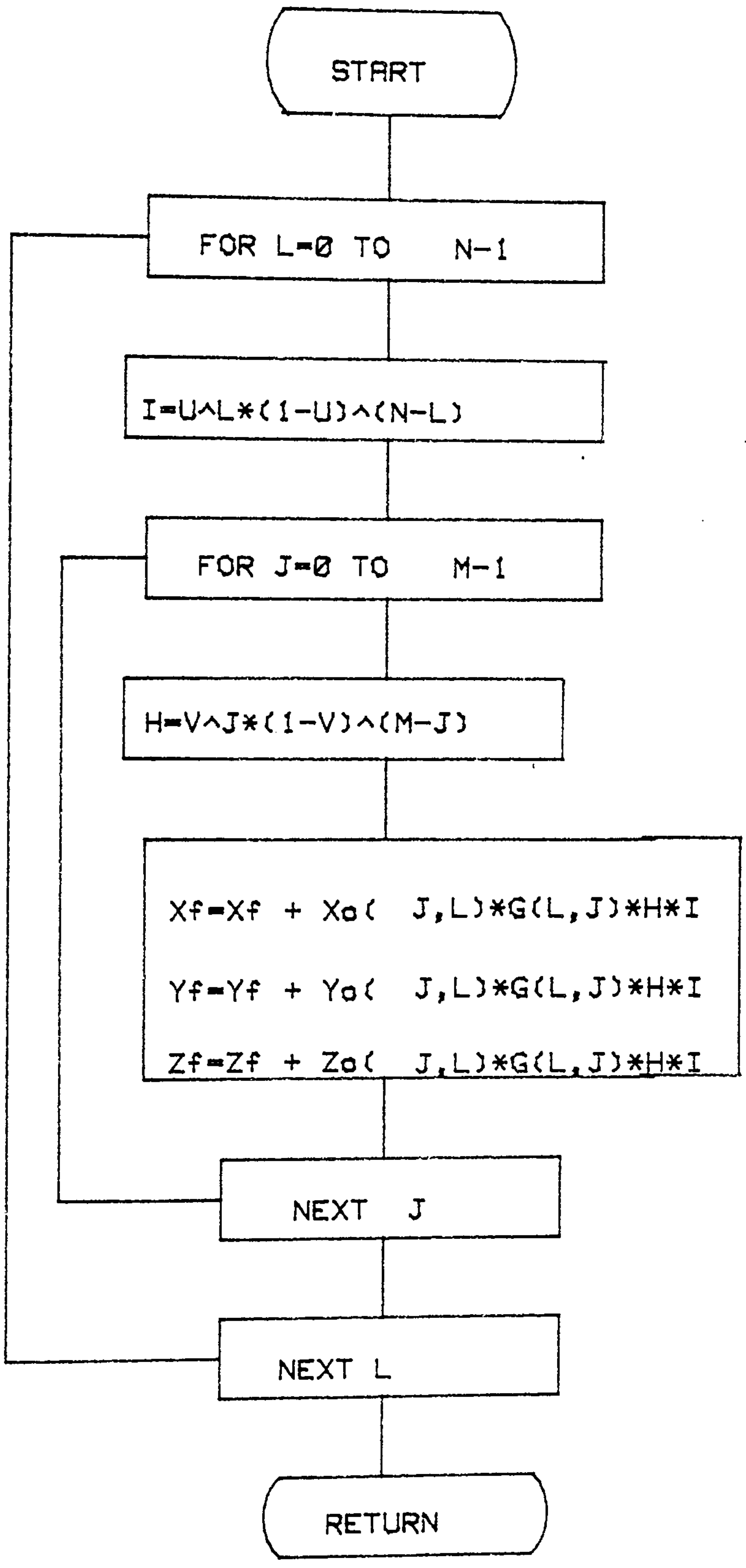


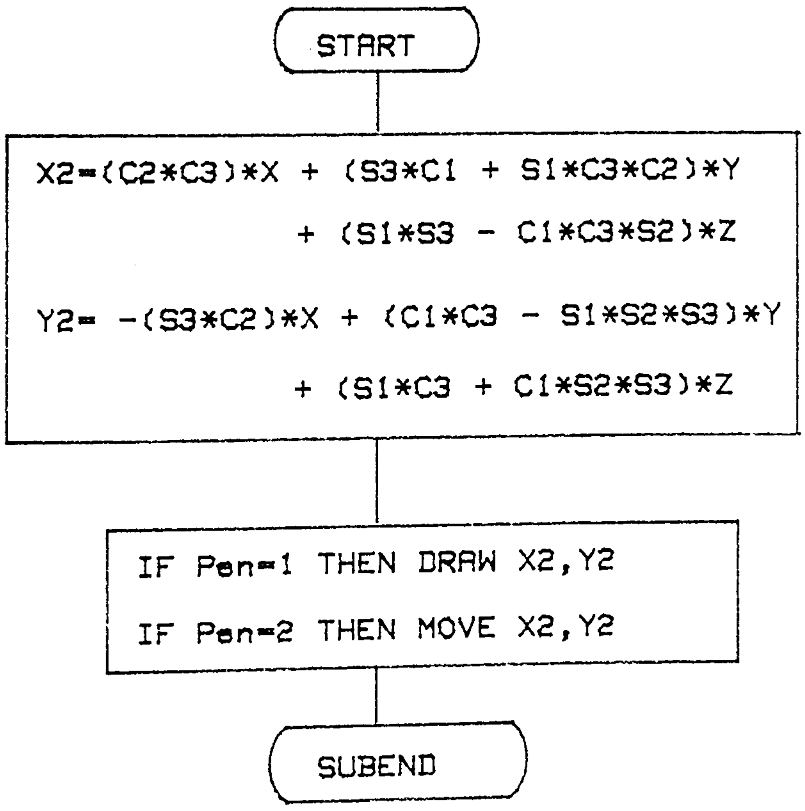
Fig. 5-15. Flowchart for subroutine MESHYZ.



5.6.2.5 Subprogram DRAW

The rotations about the x , y and z axes can be combined into a general rotation matrix [R]. Expressions for these rotations are available from standard graphics textbooks. The rotation matrix is very useful for rendering three-dimensional surface models. A compact form of the general rotation matrix was used for the subprogram DRAW. The COSINE of the angles of rotations about the x , y and z axes are given by C1 , C2 and C3 repectively while the SINE of these angles are given by S1 , S2 and S3 respectively.

Fig. 5.16. Flowchart for subprogram DRAW



5.7 CLOSING REMARKS

In this chapter , the mathematical foundation of Be'zier surface technique has been presented. The development of a stand-alone general purpose surface modelling package has been described. The master program SUFACE is highly interactive and accesses different modules by means of the HP9845 special function keys. The subprogram SURFACE is the heart of the surface modeller. The subprogram DRAW is repeatedly accessed to display the model.

The package is highly versatile and can receive control data from a 2-D engineering drawing interpretation module such as the one described in the next chapter or directly from the key board. By suppplying control data on the key board , the user can model any complicated surface such as car bodies or aeroplane fuselage and wings.

CHAPTER 6

AUTOMATIC INTERPRETATION OF ENGINEERING DRAWING FOR 3-D SURFACE REPRESENTATION

6.1 INTRODUCTION

Since earliest times engineers have used drawings to communicate and record ideas so that they would not be forgotten. For every manufactured object there are drawings that describe its physical shape completely and accurately , communicating the drafter's ideas to the machinist or foundry worker. For this reason , engineering drawing is referred to as the language of industry.

The majority of drawings used in mechanical drafting for completely describing an object are multiview drawings. The drafter represents the part which appears as three-dimensional (width , height , depth) to the eye on the flat plane of the drawing paper or board. Multiviews of the object are systematically arranged on the paper or board to convey the necessary information to the reader. Features are projected from one view to another. This type of drawing is called an orthographic projection.

In one-view drawing , the third dimension may be expressed by a note or descriptive words. Simple parts are represented by one-view drawings. Frequently the drafter decides that only two views are necessary to explain fully the shape of an object. For this reason , some drawings consist of two adjacent views only. Two views are usually sufficent to explain fully the shape of axisymmetric objects.If three views were used , two of them would be identical.

With the advent of computers , one of the possible ways of specifying the geometry of mechanical parts in CAD follows a two-step procedure : input of a graphic - oriented 2-D representation corresponding to the orthographic views ; and interpretation of this data to give the explicit 3-D description. Assuming that the user has already drawn the orthographic views , we concern ourselves here with the problem of automating the interpretation step.

Work on this problem has so far concentrated on 3-D solid objects. Even for the 3-D solid objects , emphasis have been on the class of polyhedral parts. There are little or no automatic capabilities at all for 3-D surface objects such as are encountered in fan impellers where the substructures are described by their middle surfaces and thickness

Generally , available 3-D surface modellers require the user manually to define the coordinates of the boundary curves on the surface. These values are keyed in from the computer keyboard or digitized on a digitizing table. These input data are then manipulated as many times as necessary until the desired model is obtained. In some cases , stylised methods are used manually to extract 3-D data from orthographic views.

This thesis presents a new approach for automatic construction of a 3-D surface model from engineering projections. By construction we mean being able to obtain the Be'zier nets for the Be'zier surface on the basis of the line / arc segments in the engineering drawing. The principles of interpreting a 3-D solid or surface object remain the same up to a point. When a substructure of a 3-D surface model is fully interpreted , the control polygon (when Be'zier surface technique is used) has to be constructed. But some difficulties arise at this stage : appropriate Be'zier nets have to be constructed for any structure to be fully interpreted , taking into account the nature of the surface.

The basic concepts , principles and step by step description of the algorithm are presented.

6.2 <u>REVIEW OF LITERATURE RELATED TO INTERPRETATION OF ENGINEERING DRAWINGS</u>

The problem of reconstruction of an object from multiple engineering projections has been dealt with by several researchers. There are three basic methods available. One of them is manual , in which the user points the cursor to related points in the different views. The other two are the use of algebraic topology and heuristic techniques. Methods which reconstruct an object from photographs and X-rays are deliberately left out since these are not in themselves engineering drawings which are the concern here.

Sutherland [44] has assembled a data input system which makes it quite easy and convenient to enter three-dimensional data from multiple two-dimensional views of the object being digitized. The system uses a large-area tablet to provide the simultaneous use of multiple views of the object. The user indicates points in three dimensions by pointing to them in two views simultaneously with two digitizing pens. A program deduces which views are being used from the positions of the two pens and consequently derives the three-dimensional positon of the point indicated. The limitation of this method of interpreting engineering drawings is that the user interprets the drawing for the computer program.

Idesawa [45] described a wireframe reconstruction of a solid figure from three views. The method used is based on finding sets of coplanar edges and fitting them together to form solid objects. Any edge line which can not be any boundary of faces is eliminated as a ghost line. The limitations in his reconstruction method are the constraint that only polyhedra are dealt with and ambiguities are not handled. When a solid contains a curved surface , in order to identify the surface , some proper lines are added on the curved surface , and each segment of the surface divided by the lines is regarded as a different face .

The historical trend has been to free the user of as many constraints as possible. However , the relaxation of constraints has led to the possibility of multiple solutions to a given problem and workers have tended to concentrate on heuristic approaches to find a probable solution. Preiss [46] reported such a heuristic approach for plane-faced body that allows complete freedom of input. In his approach , after the input data has been suitably organised , the loops of lines that represent the projections of each face in each view are found. The leftmost lower point for each view is found and at each subsequent point the first right turn is taken until the loop closes. This process is repeated until all 2-D loops are identified. Each line of each loop are interpreted by appropriate calculations using each view. All the edges interpreted using appropriate conditions from a 2-D loop must create a valid 3-D face. Hidden faces are then evaluated until the interpretation is complete. The faces are then assembled into a solid body.

Adelfeld [47] has combined model-guided recognition and heuristic techniques for interpreting three-view representations. In the model-guided recognition concept , some classes for the modelling objects are defined in such a way that a variety of shapes are included within each single class. They could be uniform thickness or rotationally symmetric objects. In his approach , Aldefeld considered uniform thickness objects only , with each object oriented such that its base is parallel to one of the coordinate planes. All elementary loops are found and a score is assigned to each loop according to the heuristic search strategy. The loop with the highest score is assumed to represent the base silhouette of one or more objects. All the interpretation of this candidate loop is sought with the aid of the model - guided recogniton algorithm. The shortcomings of his work include the fact that his algorithm can not distinguish between solid and hollow objects , restrictions are imposed on the spatial orientaion of the objects and a restricted classes of objects is handled.

Lafue [48] also described a program that generates solids from 2-D projections for polyhedra.The user is constrained to draw the views in a face-wise manner and to draw all faces projected onto a view whether they are visible or not. Holes in faces are only acceptable when the user adds artificial edges to the faces containing them ; moreover every closed loop of edges that does not separate the object into two regions requires that such artificial edge be added. During the search for solution these artificial edges are ignored. The identification of an object starts with the tablet-coordinates and aggregating them into 3-D vertices by associating each vertex of a view with those of other views sharing with it the same value of common coordinates. The next stage is to identify true and false faces. The general principle here is to aggregate faces having a common property (such as containing a given edge or a given vertex) in sets and impose on these sets some rules which determine the maximum and minimum number of possible true faces contained in the sets. Whenever several of the sets contain at least one common face , a "mini-theorem prover" is run to choose the right faces. When all true faces have been identified , then they are combined into a body. The limitations of Lafue's technique are that the views must be described in terms of faces and that "artificial edges" need to be included.

Unlike all the methods that have been discussed so far , Wesly and Markowsky [17] solved the reconstruction problem for a plane-faced body by using the concept of algebraic topology. In their algorithm , each vertex in the wireframe (the ordered pair of vertices and edges) is considered and a list is formed of edges for which the vertex is an endpoint. For each noncollinear pair of edges in the list , the plane equation is computed and a list formed of distinct planes at the vertex. The planar edge and vertex lists are then processed to obtain virtual faces which are candidates for faces of the object. Illegal intersections between virtual faces are checked and handled accordingly. Then virtual faces are fitted together to form virtual blocks. The final stage of the object description is to fit the virtual blocks together. The algorithm is restricted to straight edged and planar faced objects.

None of these techniques discussed automatically interprets engineering drawings describing 3-D objects composed of shells. One of the objectives of the research project desribed in the present thesis is aimed at formulating algorithms that can automatically interprete such engineering drawings and represent their surfaces in a computer.

6.3 BASIC CONCEPTS

The concepts defined in this section are useful in understanding how the algorithm discussed in section 6.7 works.

6.3.1 REFERENCE LINE

It is necessary to define a "reference line" on the elevation , preferably along the lowest edge of the elevation. It does not matter whether or not the lower edge is horizontal or inclined. The user defines the reference line by digitizing two points on the lowest edge of the elevation. Recall that to define a 3-D surface requires x , y , z coordinates. Each view has only two of these space coordinates at the nodes , therefore the third is found from another view. If the reference line is placed along the lowest edge of the view in the elevation , then the z coordinates can be obtained. Considering a "convex cone" (in this section and in fan impeller work , "cone" may not be a true straight sided cone) and referring to Fig. 6.1 , we see that the x and y coordinates are easily read from the plan and the corresponding z coordinate for each node is read from the elevation very easily if the reference line is drawn at the lower edge of this view. In other words , all nodes along this line have zero value with respect to others in the elevation.

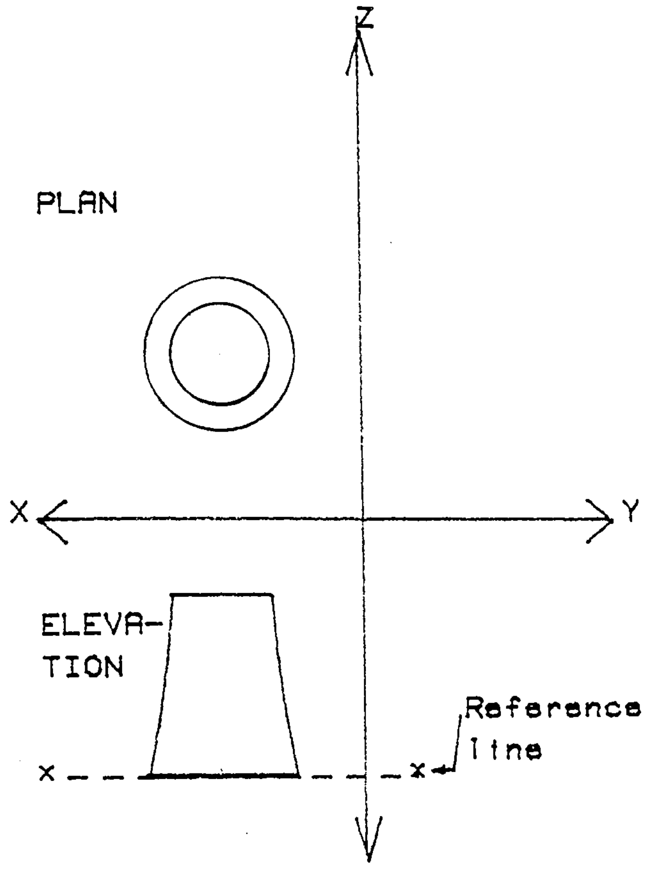


Fig. 6.1 Reference line in elevation helps to obtain z-value

6.3.2 CONTROL POLYGON DATA FOR VARIOUS USEFUL SHAPES

Recall that the Be'zier surface is drawn using control polygons. Once a sub-structure is fully interpreted , its corresponding Be'zier net data need to be calculated. Therefore in this section , we consider some shapes which are useful in engineering applications and which are useful for constructing the surfaces of general plate and shell structures. The Be'zier nets for these shapes are also constructed.

(i) <u>Flat and straight edged shapes</u>

In this and all other examples to be discussed , it should be noted that the minimum number of control polygons required to define a surface patch is four , each having four vertices.

A control polygon (Fig. 6.2a) can be distorted to give the shape of the curve desired. In order to represent a straight line , all the vertices on a control polygon are made collinear (Fig. 6.2b). It is an easy matter to verify that the condition of end points tangency is met. In fact , it is not necessary to space the vertices equally along the distorted polygon. The start and finish points only can be used fully to define the characteristic polygon. Taking advantage of this distortion characteristic , edges of a flat straight edged object can be fully defined as in Fig. 6.3

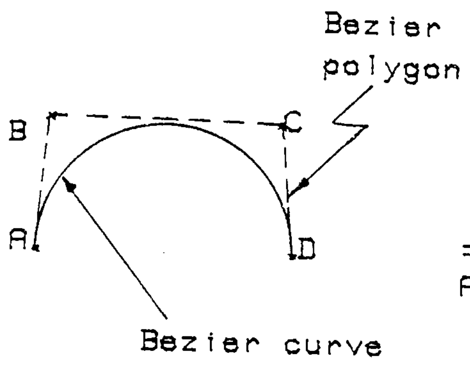


Fig. 6.2a A Bezier control polygon and curve

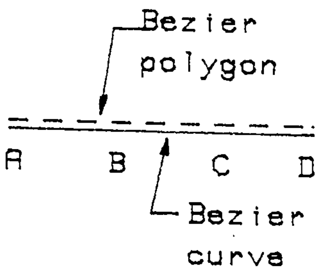
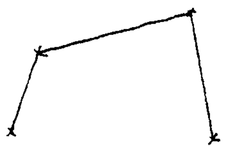
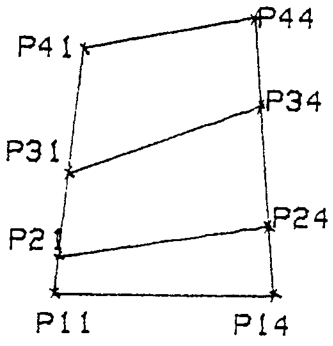


Fig. 6.2b The same polygon distorted until points become collinear

Fig. 6.2 Distortion of Bezier polygon



(a) A control polygon



(b) Bezier net

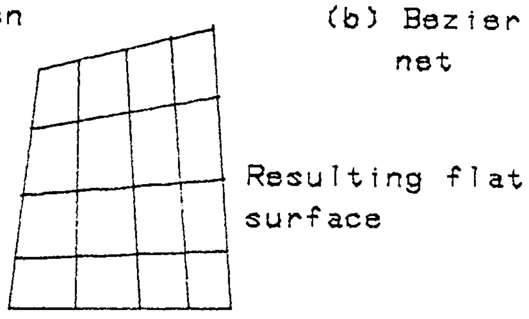


Fig. 6.3 Flat straight edge possible with Bezier technique

(ii) Disc with hole

Consider a disc of radius R measured from the centre as shown in Fig. 6.4a. It is logical to consider the surface mapped out into increasingly overlapping rectangles from near the centre outwards as shown in Fig. 6.4b. For a constant value of y , arcs are obtained which have increasing diameter from near the centre towards the outer diameter. For example the arc A1......D1 has a radius R1 which is smallest and the radii of various arcs are in inreasing magnitude until the maximum value of R4 for arc A4.......D4 is reached. A1.....D1 is the first characteristic polygon while A4.....D4 is the last. The corresponding arcs are drawn in that order also.

A closer observation of Fig. 6.4b shows how to represent a circular shape having a hole. The hole is represented by A1.....D1. The smaller the dimensions of A1.....D1 , the smaller the dimension of the hole. The designer can then control the size of the hole by controlling the dimensions of the control polygon A1.....D1.

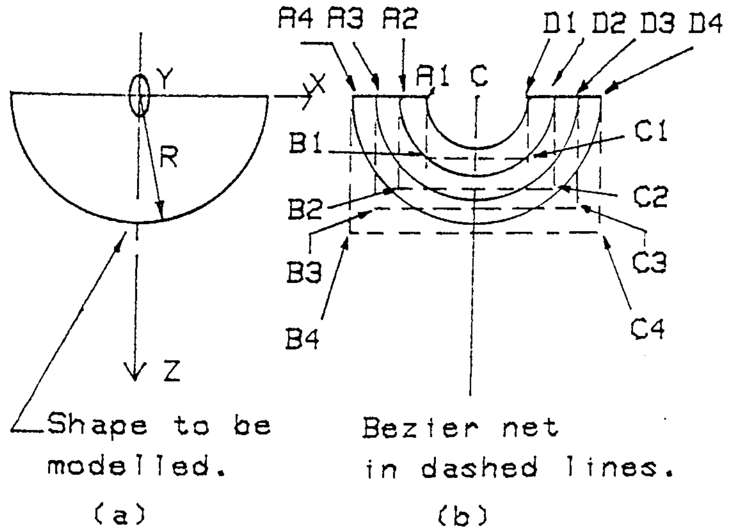
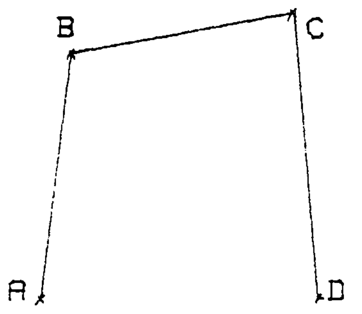


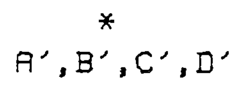
Fig. 6.4 Flat curved edged shape possible with Bezier technique

(iii) Disc without a hole

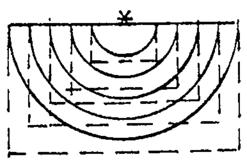
The excellence of Be'zier technique for surface design lies on the flexibility which the designer has in manipulating the polygons to obtain any desired surface pattern. Advantage is taken of the degenerate property of Be'zier surface. Consider Fig. 6.5a having the characteristic polygon defined by vertices A ,B , C and D. The polygon could be distorted until the vertices coincide at a point. This concept is very useful in the representation of a disc without a hole. We simply make A1.....D1 to coincide at the centre of the disc. Such a simple intuitive alteration results in the same disc , but this time not having a hole as shown in Fig. 6.5c



(a) Bezier polygon
A,B,C,D



(a) The same
polygon
distorted
until it
becomes a
point.



(c) Resulting shape
without hole

Fig. 6.5 Flat curved shape
without hole possible(Bezier tech)

iv) Curved surfaces

The preparation of the data for the characteristic polygon of curved surfaces is more complex than all that have been discussed so far. This is not surprising because the mathematics of curved surfaces is naturally more complex than those of simpler shapes. We need to know before hand how the surface should look and then graphically represent a cross-section of it. This then gives us an idea of how the z-coordinate has to vary.

Consider the cross-section of a "conical" shell as shown in Fig. 6.6. The cross-section enables us to visualize how z varies for given x and y values. The Be'zier net can then be constructed with the z coordinates varying according to the graph of the cross-section.

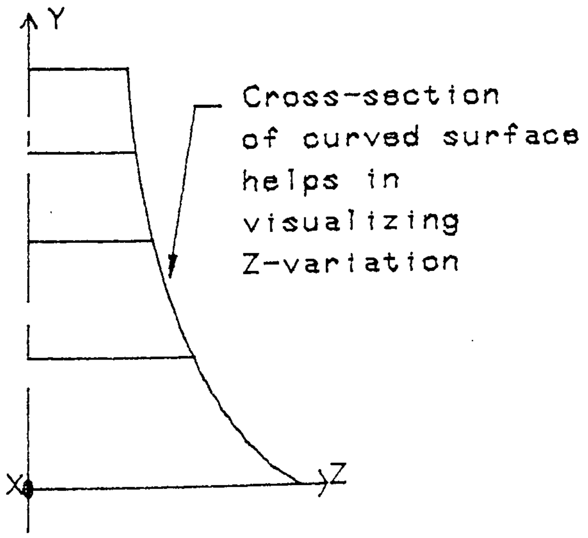

Y
Cross-section of curved surface helps in visualizing Z-variation
X
Z

Fig. 6.6 Cross-sectional curve makes any shell shape possible with Bezier tech.

6.4 CYCLIC SYMMETRY (ROTATIONALLY PERIODIC STRUCTURES)

Rotationally periodic structures consist of identically coupled sub-structures positioned symmetrically about an axis. In these structures , one can recognise a repetition of geometry. If the geometry of the structure is defined for any radial or axial position at some angle θ , it will be identical at ($\theta + n\theta_{\emptyset}$) , where $\theta_{\emptyset}$ is $2\pi/N$; n and N being integers , and N is structure dependent (the number of identical substructures that constitute the structure). Structures which posses the property of cyclic symmetry include rotating fan impellers , bladed turbine discs, centrifugal pumps and cooling towers.

Fig. 6.7 shows a sector of a disc . If the geometry is completely defined for an angular segment of $2\pi/8$, then the rest of the disc can be generated by repeated rotation of the segment through $2\pi/8$ radians. In geometric modelling terms , the sector is replicated 8 times. As we shall see in Chapter 7 , the use of a sector for a cyclically symmetric structure such as fan impellers conserves computer time and storage. Many CAD systems implement this technique and finite element analysts have found the technique very useful.

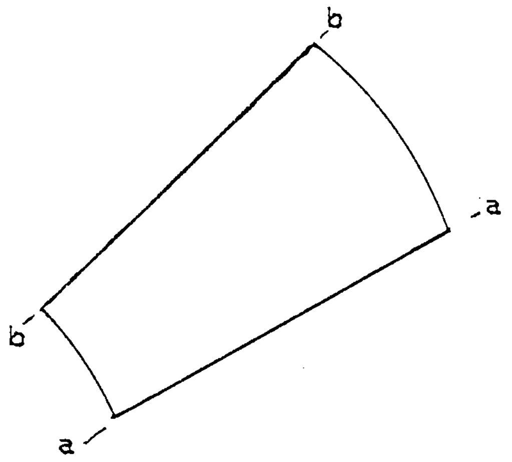


Fig. 6.7 A typical sector of disc (aa--bb boundaries)

The principle of cyclic symmetry is used in the construction of a Be'zier net when the interpretation of the sub-structures is complete. The program automatically constructs the Be'zier nets and frees the user of the task.

6.5 MATCHING

Sub-parts of a total object are inter-related via some matching conditions which follow from the object's shape and from the rules governing the preparation of engineering drawings. For instance , a match between primitives (e.g. , line or arc) P_1 and P_2 in two different views can be defined by

$$P_1 = P_2 \text{ if } ABS(P_1) = ABS(P_2) ,$$

where ABS denotes the absolute length of the primitive that the two views have in common.

6.6 SEPARATING ENGINEERING DRAWING VIEWS

Before begining the interpretation of an engineering drawing , the 2-D drafting data must be suitably organised. For the MEDIA drafting program , an interface program has been written which separates engineering drawing views and organise the data into a suitable form for the interpreting module. A summary of the relevant steps are as folows :

1. "Mask" a view.

2. Isolate the data of the view.

3. Store the data of the view.

4. When steps 1 --- 3 are completed for the different views that completely describe the object , interprete the views and extract the 3-D data.

It was necessary to include steps 1 --- 3 in the data preparation stage for the interpretation process because the 2-D drafting package MEDIA , does not descriminate between the data of one view from that of another view.

Each of these steps is treated briefly to show the underlying principles involved. Details of standard graphics operations are omitted since these are well treated in text books in references [49] and [5Ø].

6.6.1 "BOXING" ROUND A VIEW

Before the geometry of a view is isolated , a box is drawn round the view. The user only needs to digitize the coordinates of diagonal corners of the box. The subprogram BOX then automatically calculates the remaining corner coordinates of the box and draws a dashed line around the perimeter of the box. The corner coordinates of the box are stored in the arrays B_x and B_y and these are used in subsequent steps.

Assuming X_A , Y_A and X_C , Y_C are the coordinates of the diagonal corners of the box , then

$$X_B = X_C$$

$$Y_B = Y_A$$

$$X_D = X_A$$

$$Y_D = Y_C$$

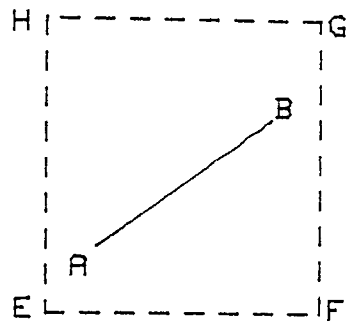
6.6.2 ISOLATING DATA OF A VIEW

Once a view is masked , the next task is to remove details in the geometry of the view which may unecessarily complicate the interpretation of the object and then store the relevant geometry in an array.

The subprogram TRANSFER was written for this task and the way it operates is that it searches through the database of the 2-D drafting program. Each record (except the first ten) of the file storing the drafting data , stores the "start" and "finish" coordinates of items drawn. The first two data are the "start" x and y coordinates while the last two are the "finish" x and y coordinates (not necessarily so for arcs and labels). The signs for the data in each file record are used as codes for identifying which items are lines(full , dashed or chained) , arcs (full , dashed or chained) , labels etc. Therefore , the subprogram TRANSFER is able to identify and extract data related to these items and ignore items such as label , centre lines , dimmension lines etc . The relevant lines and arcs are then stored sequentially in the order in which they were drawn , in array B.

The start and finish points of the relevant items are tested if they are completely enclosed by the "box" drawn round the view under consideration. Subprogram CLIP is accessed for this task and the way the test is carried out is illustrated as follows :

Consider a primitive AB inside the "box" EFGH



A set of conditions must be met if the primitive AB is enclosed by the "box" :

$X_A , X_B > X_E$

$X_A , X_B < X_G$

$Y_A , Y_B > Y_E$

$Y_A , Y_B < Y_G$

All primitives whose start and finish points lie inside the "box" are stored in the array B.

The subprogram DRAW is accessed which uses the corner coordinates of the "box" as the "window corners" and so resets the

LOCATE and SCALE values of the drafting program in such a way that the item when redrawn occupies the whole of the screen. In graphics terms , the view has been "windowed".

6.6.3 STORING DATA OF THE VIEW

A separate subprogram VIEWS-DATA was written to store the data of the view under consideration. This is because the user could change his mind when a view has already been isolated and may decide to make modifications in the drawing. However , when the user is ready to store the data of a view , this subprogram , when accessed , CREATES and ASSIGNS a temporary file to store the number of primitives and the geometric data obtained from section 6.6.2.

6.7 THE PROGRAM STRUCTURE

The program structure of the processes involved in separation of engineering drawing views of the MEDIA drafting program and the interpretation is discussed here. Following the flowchart of the program INTERFACE , it is observed that when a view is to be "boxed" , then the subprogram BOX is accessed after which control transfers to an appropriate point in the 2-D drafting program. When isolation of a view is required , two subprograms TRANSFER and DRAW are accessed. Then for storing the data of such views , the subprogram VIEW-DATA is accessed. These different operations are not sequential , rather the user selects the required operaton from a menu. This makes the process of separating the view very interactive. When the views that fully describe the object have been separated , the user can then decide to instruct the program to begin interpretation.

The interpretation of the drawing for 3-D surface representation will be complete when each sub-structure that makes up the 3-D surface structure is identified and the "key" coordinates defining the sub-structure are found. Two subprograms have been written : one subprogram C ,caters for plate structures while the other subprogram Co , caters for shell structures.The user has plate or shell options. The subprograms for general plate structures is capable of interpreting intersecting plate structures which are commonly encountered in engineering applications. On the other hand , the subprogram for shell structures has been developed to handle general folded shells. It has been further developed to be able to interprete a fan impeller which is an assemblage of plate and shell structures. The steps showing how the subprograms are organised are as follows:

1. The input data is organised by separating the drawing into views and establishing the number of primitives in each view.

2. The principles of matching primitives in the different views and where applicable , cyclic symmetry , are used for interpretation of the substructures of an object.This is then followed by identifying these substructures.

3. Construction of the Be'zier nets for each interpreted sub-structure.

Steps 1 and 3 are common to each of the subprograms and therefore are treated in details once. Step 2 is treated for general plate and shell structures.

Temporary files which were used to store the views data are then purged and the geometric modeller is accessed.

The flowcharts for the different subprograms whose operations have been fully treated in sections 6.6.1 — 6.6.3 are also included in this section. Moreover , the details and flowcharts for the subprograms C and Co which interprete the plate and shell structures respectively are also given.

Fig. 6.8. Flowchart for program INTERFACE

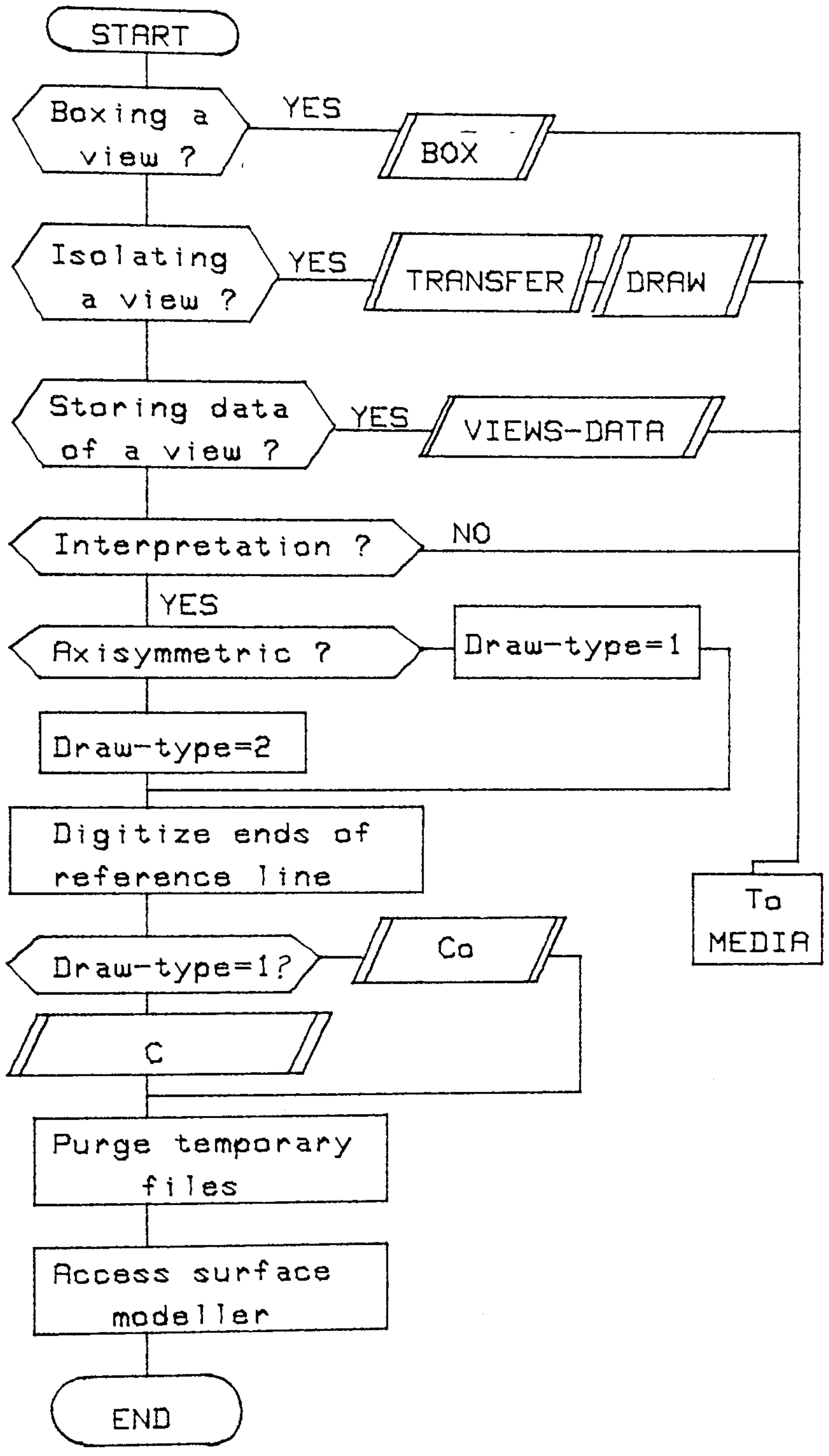


Fig. 6.9.   Flowchart for subprogram TRANSFER

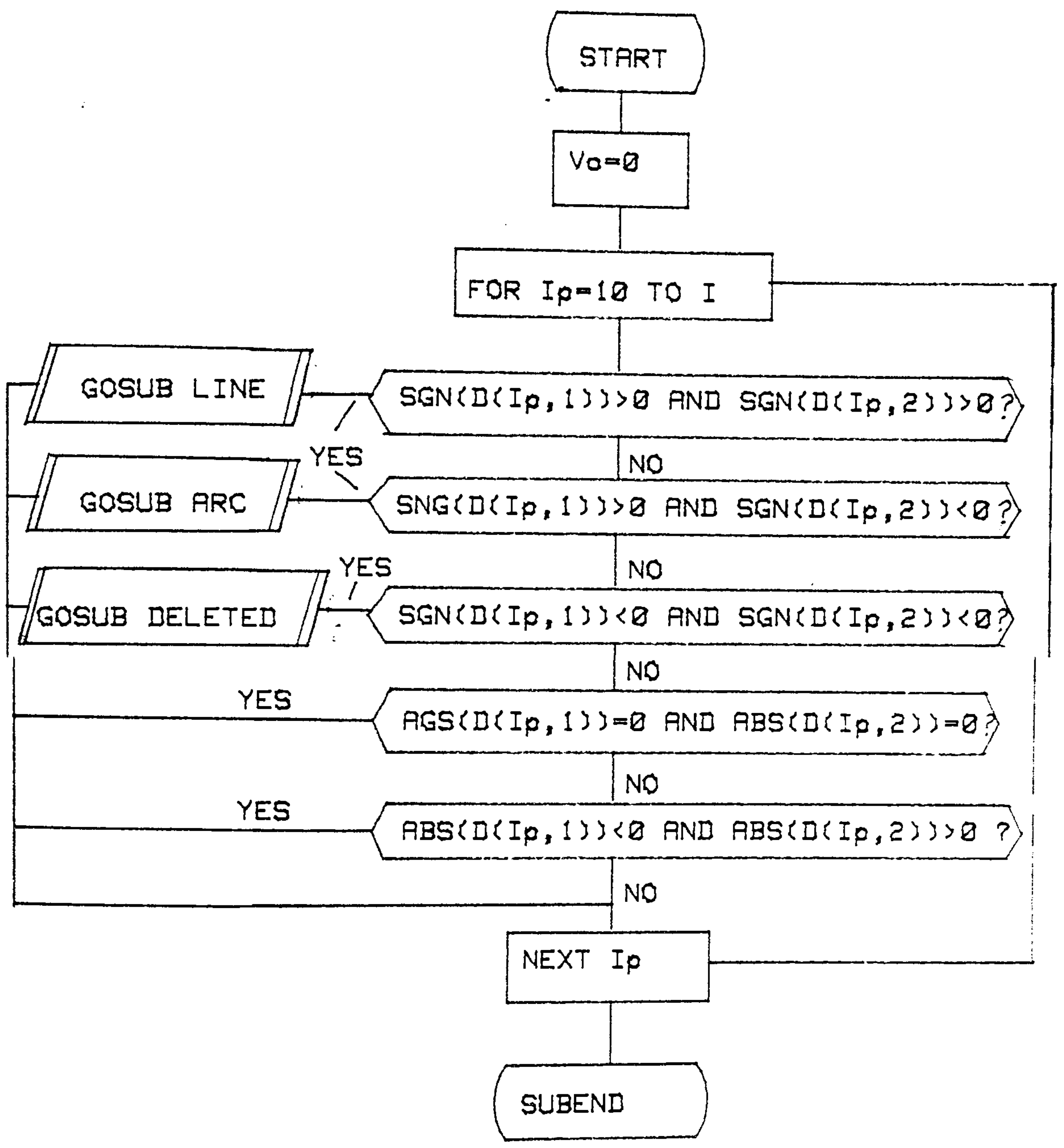

START
Vo=0
FOR Ip=10 TO I
SGN(D(Ip,1))>0 AND SGN(D(Ip,2))>0?
GOSUB LINE
YES
NO
SNG(D(Ip,1))>0 AND SGN(D(Ip,2))<0?
GOSUB ARC
YES
NO
SGN(D(Ip,1))<0 AND SGN(D(Ip,2))<0?
GOSUB DELETED
NO
YES
AGS(D(Ip,1))=0 AND ABS(D(Ip,2))=0?
NO
YES
ABS(D(Ip,1))<0 AND ABS(D(Ip,2))>0 ?
NO
NEXT Ip
SUBEND

Fig. 6.10. Flowchart for subroutine LINE

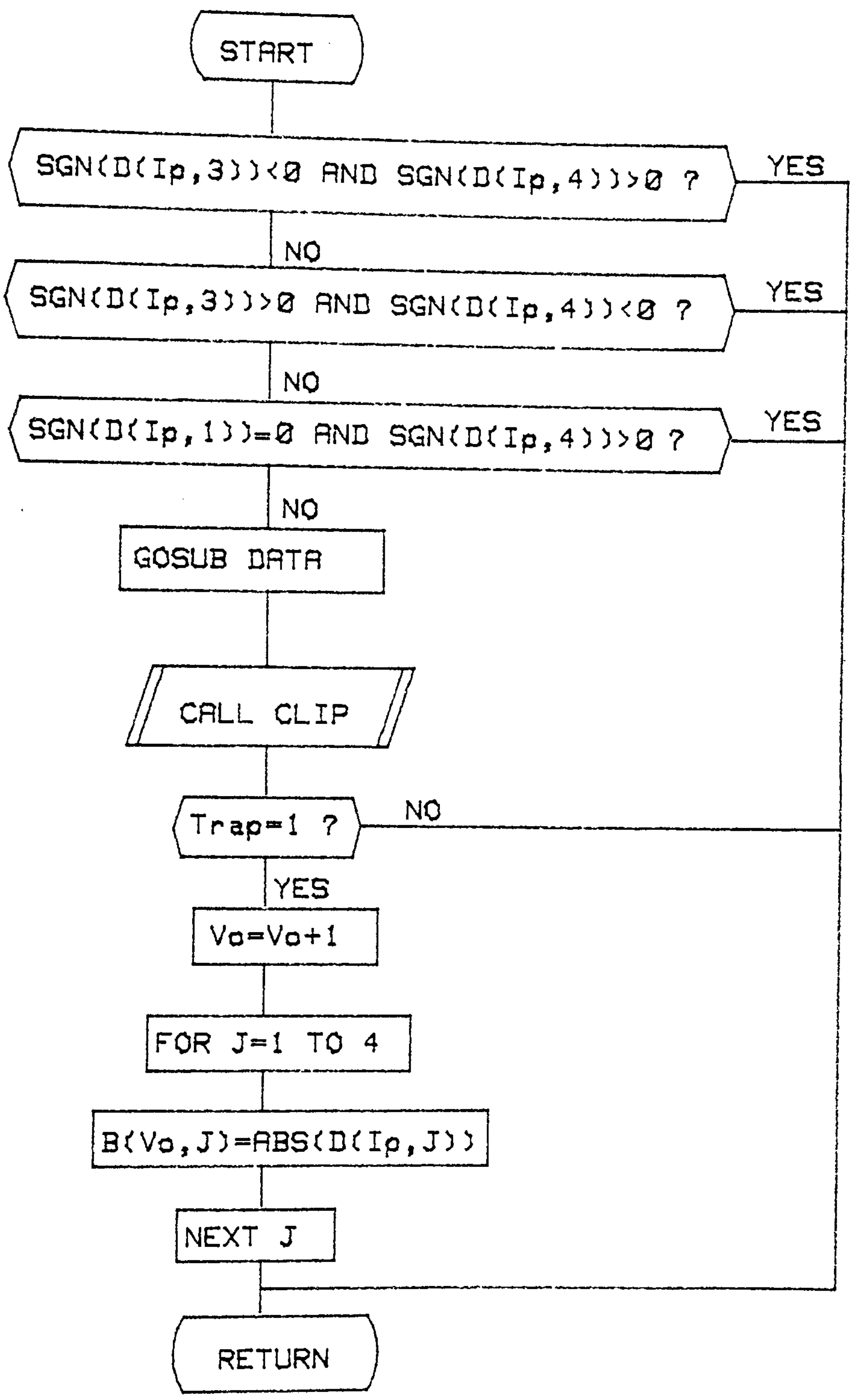

START
SGN(D(Ip,3))<0 AND SGN(D(Ip,4))>0 ?
YES
NO
SGN(D(Ip,3))>0 AND SGN(D(Ip,4))<0 ?
YES
NO
SGN(D(Ip,1))=0 AND SGN(D(Ip,4))>0 ?
YES
NO
GOSUB DATA
CALL CLIP
Trap=1 ?
NO
YES
Vo=Vo+1
FOR J=1 TO 4
B(Vo,J)=ABS(D(Ip,J))
NEXT J
RETURN

Fig. 6.11. Flowchart for subroutine ARC

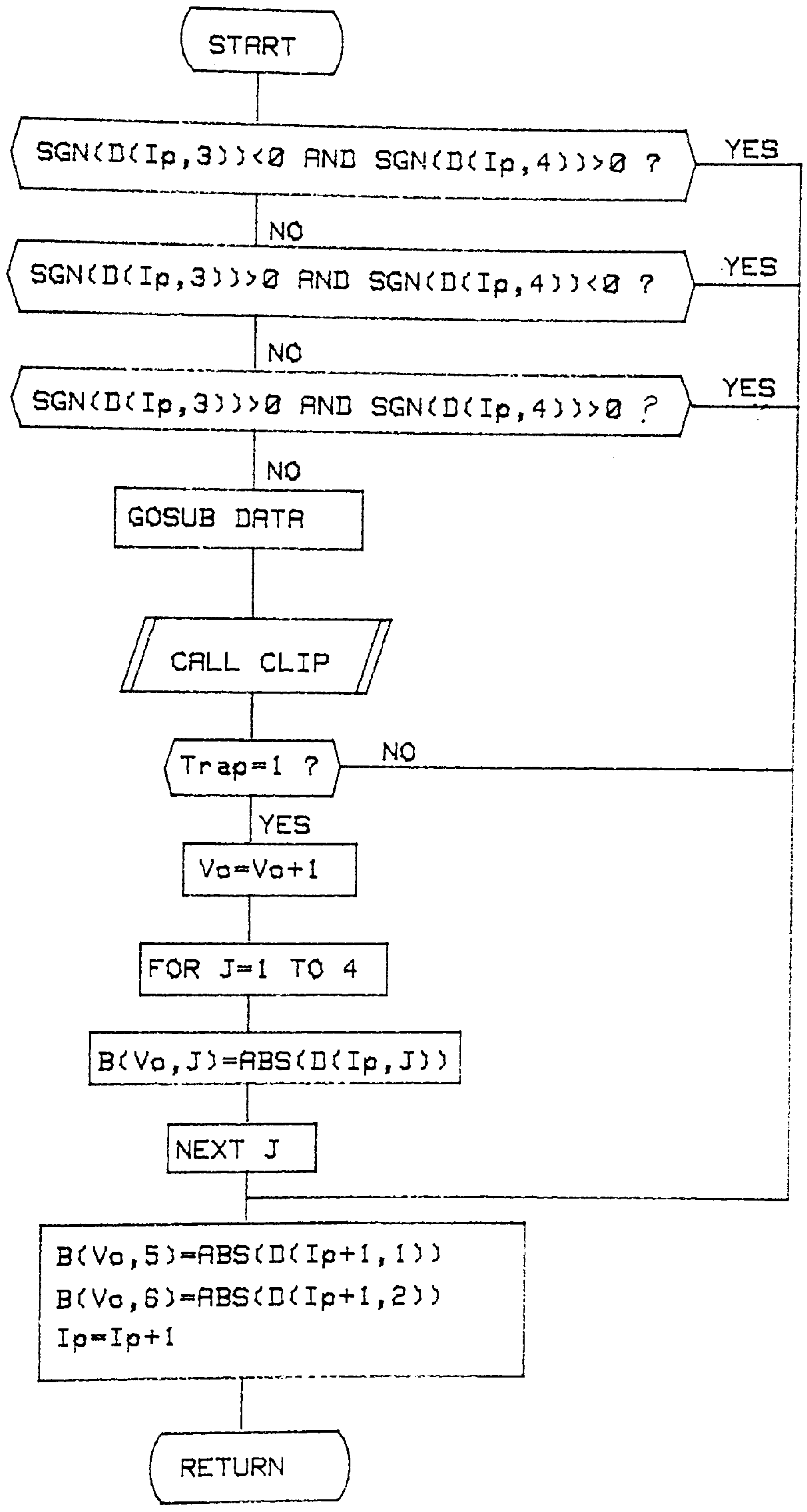
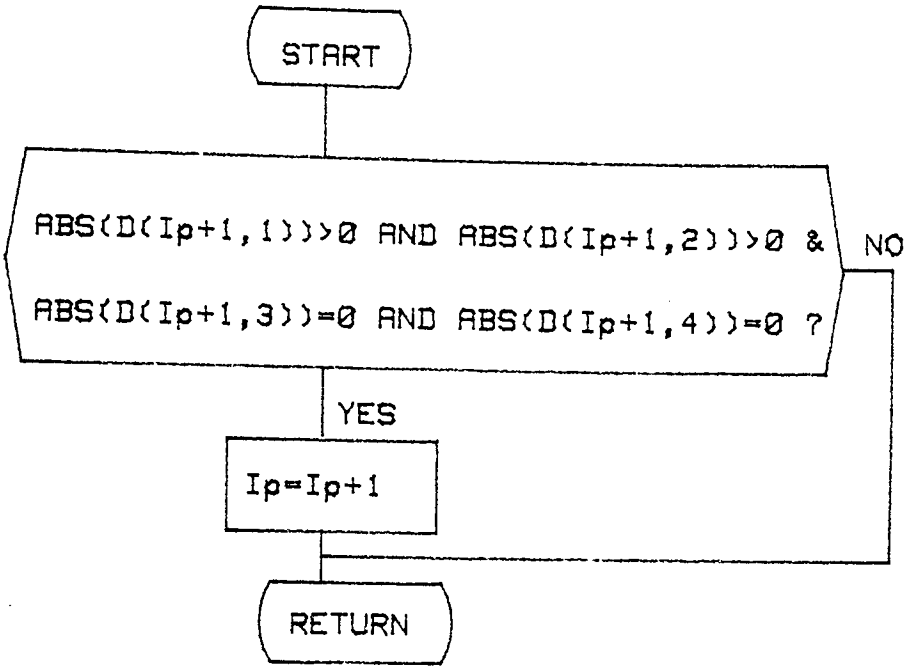


Fig. 6.12. Flowchart for subroutine DELETED



START
ABS(D(Ip+1,1))>0 AND ABS(D(Ip+1,2))>0 &
ABS(D(Ip+1,3))=0 AND ABS(D(Ip+1,4))=0 ?
NO
YES
Ip=Ip+1
RETURN

Fig. 6.13. Flowchart for subroutine DATA

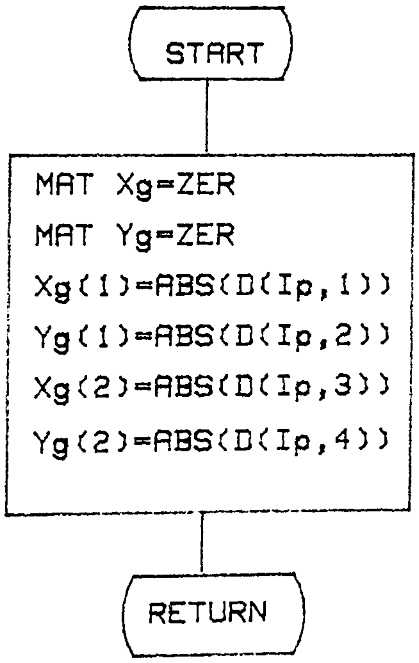


Fig. 6.14. Flowchart for subroutine BOX

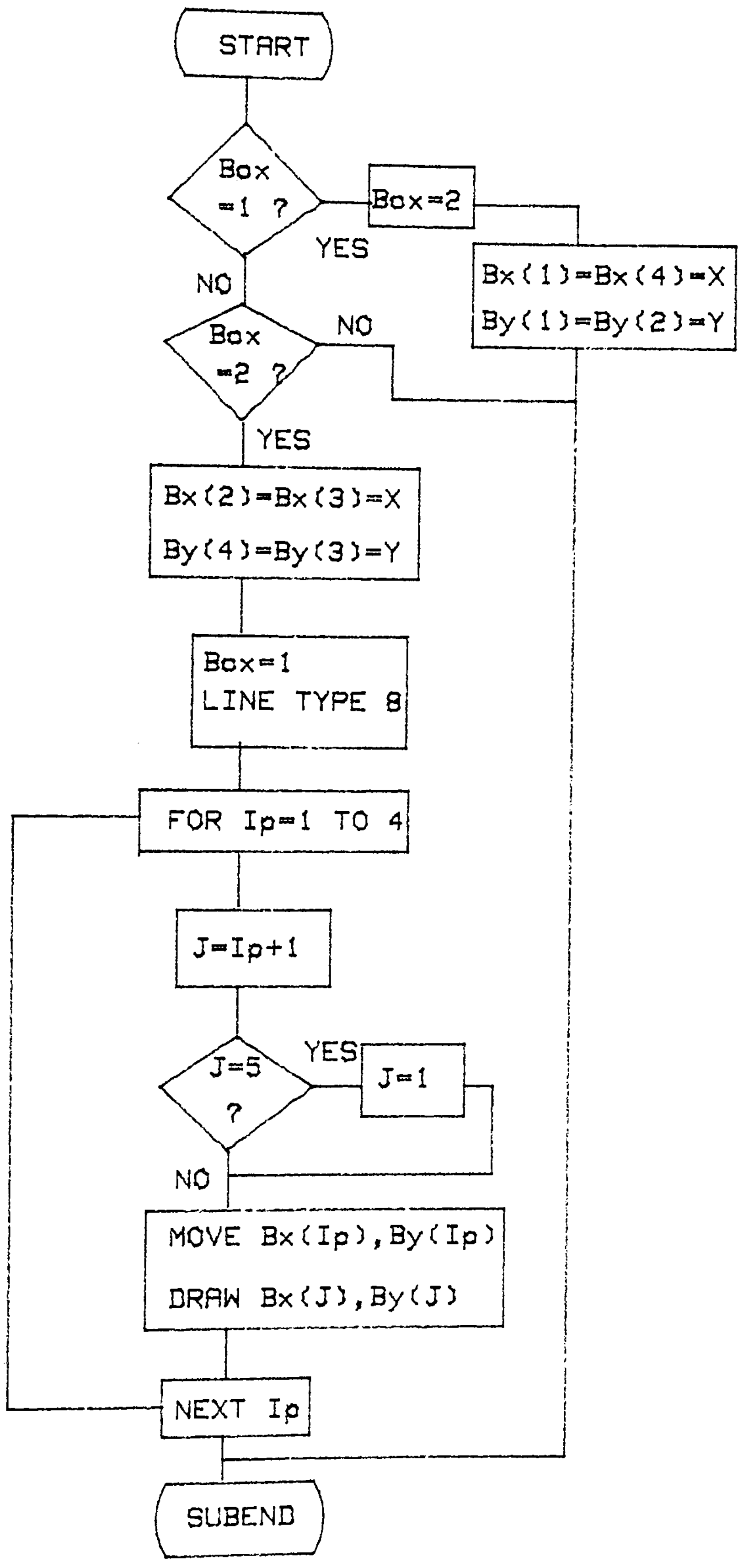


Fig. 6.15. Flowchart for subprogram CLIP

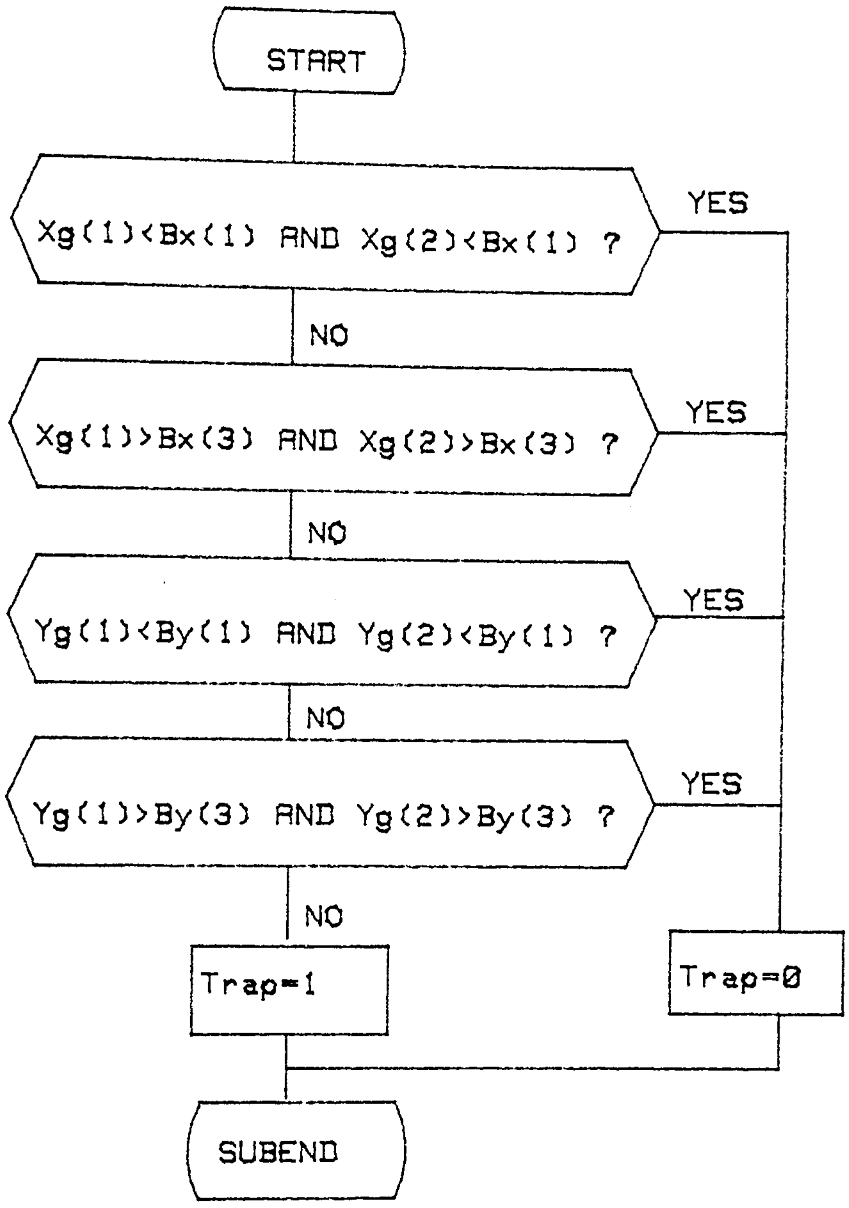

START
Xg(1)<Bx(1) AND Xg(2)<Bx(1) ?
YES
NO
Xg(1)>Bx(3) AND Xg(2)>Bx(3) ?
YES
NO
Yg(1)<By(1) AND Yg(2)<By(1) ?
YES
NO
Yg(1)>By(3) AND Yg(2)>By(3) ?
YES
NO
Trap=1
Trap=0
SUBEND

Fig. 6.16. Flowchart for subprogram VIEWS-DATA

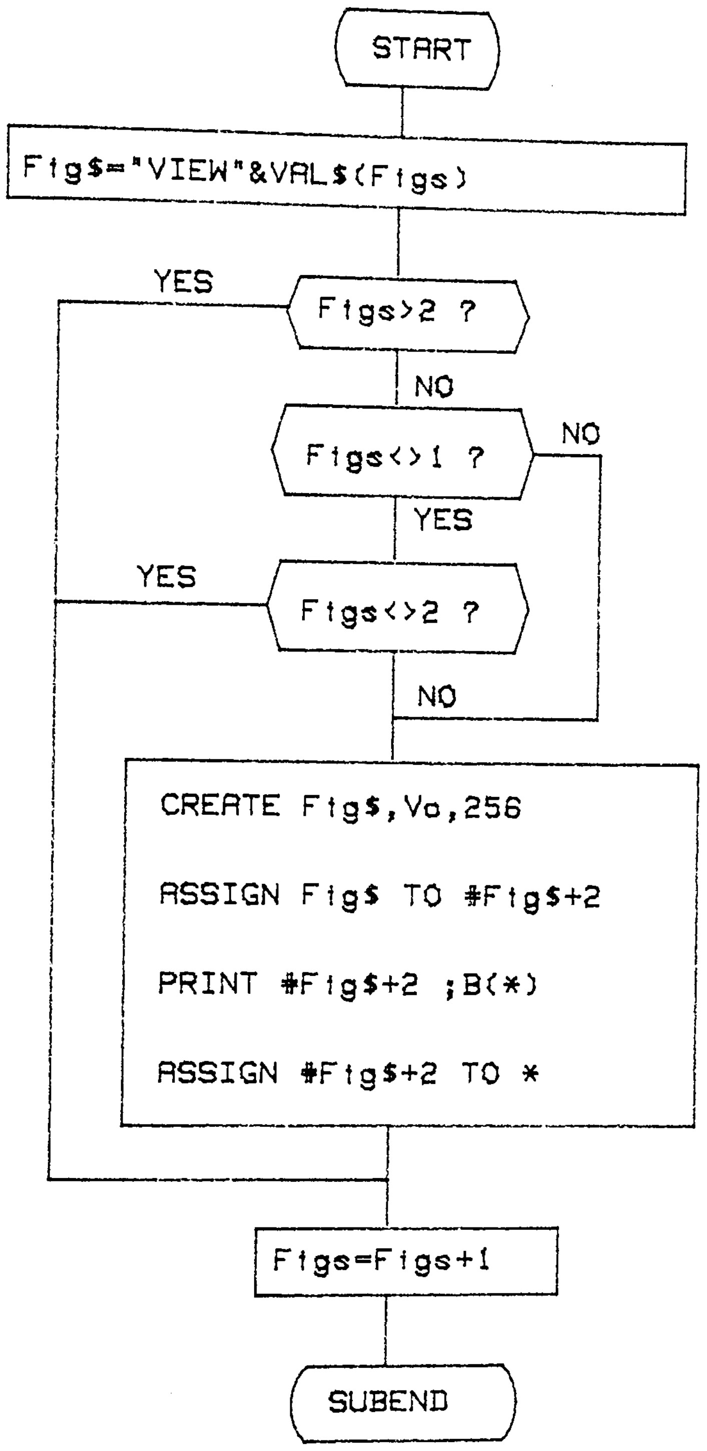
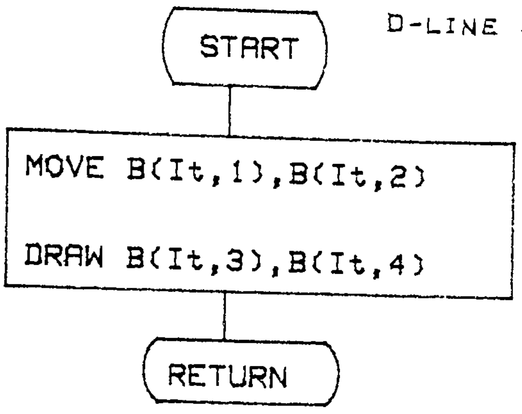


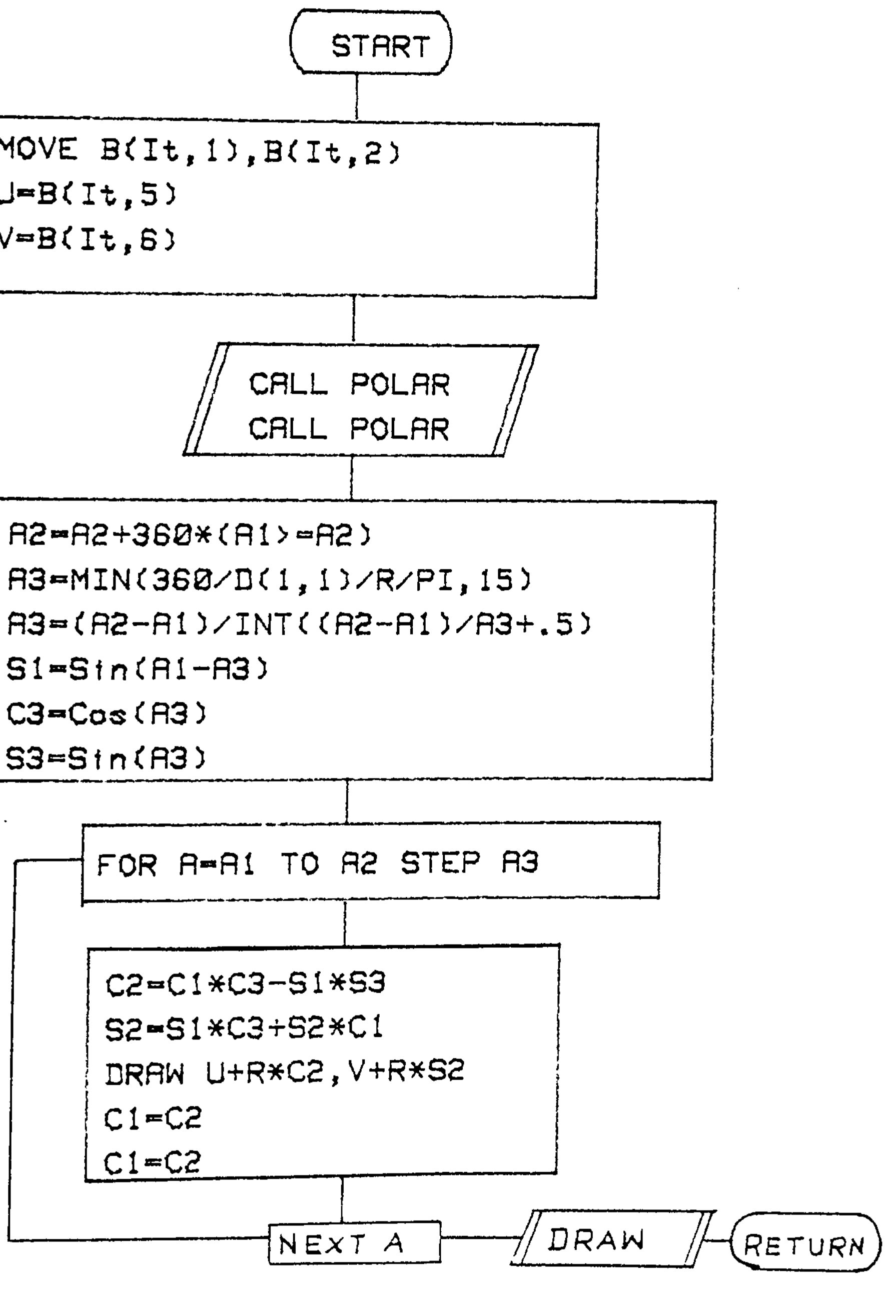
Fig. 6.17. Flowcharts for subroutines D-LINE & D-ARC

D-LINE :



```
MOVE B(It,1),B(It,2)
DRAW B(It,3),B(It,4)
```

D-ARC;



```
MOVE B(It,1),B(It,2)
U=B(It,5)
V=B(It,6)
```

```
A2=A2+360*(A1>=A2)
A3=MIN(360/D(1,1)/R/PI,15)
A3=(A2-A1)/INT((A2-A1)/A3+.5)
S1=Sin(A1-A3)
C3=Cos(A3)
S3=Sin(A3)
```

```
FOR A=A1 TO A2 STEP A3
```

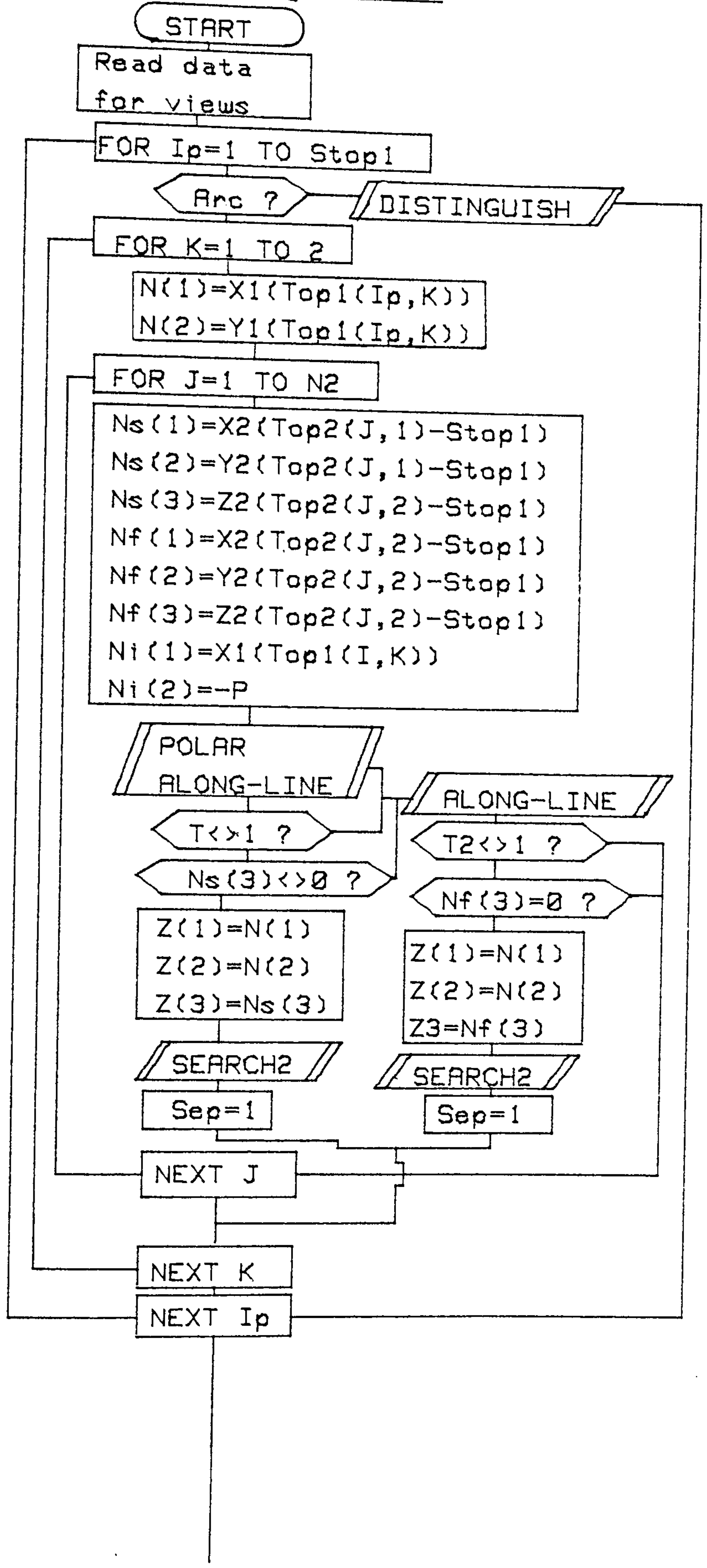
```
C2=C1*C3-S1*S3
S2=S1*C3+S2*C1
DRAW U+R*C2,V+R*S2
C1=C2
C1=C2
```

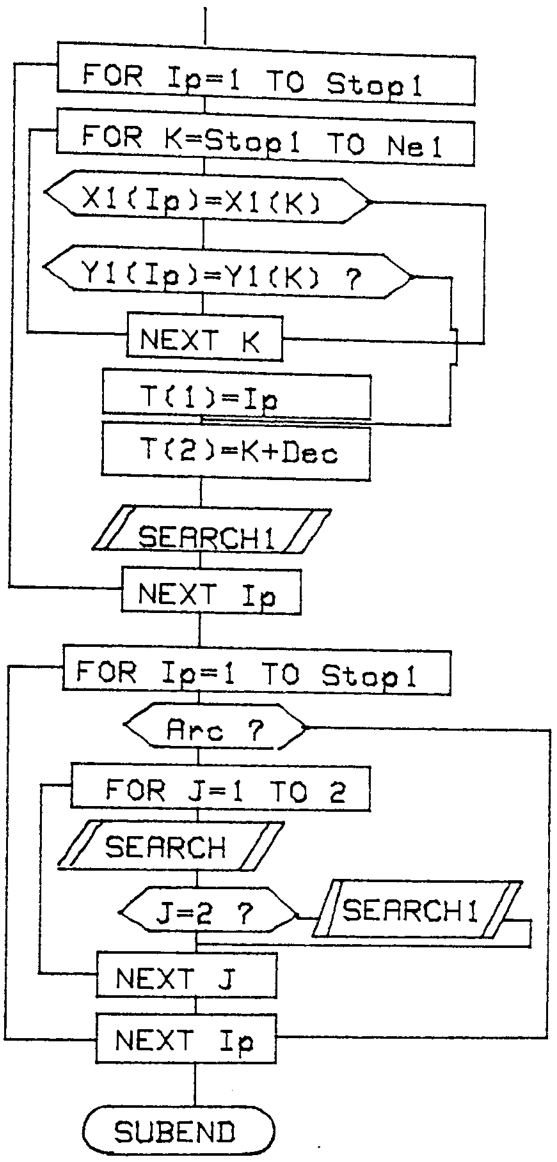
6.7.1 Subprogram C

The flowchart for this subprogram shows that its first major task is to read the geometric and topological data for each view obtained from section 6.6. The geometric data for the elevation is transformed through the routine IDENTIFYNODE2 to obtain the z-coordinates of primitives in this view by making use of the reference line (see section 6.3.1) as a datum.

Within the loop I_P , the program distinguishes between lines and arcs. For a line , the end-points coordinates are used when the main subroutine of the interpretation module , ZED is accessed. The subroutine ZED has a loop J which checks the end-point coordinates against all the points in an adjacent view if they match. The matching concept of section 6.5 is used but this time P is a point rather than the length of a primitive. In this way , the program extracts the z-coordinate values for all points in a view. The subprogram SEARCH2 is then accessed to store the x , y , z coordinates. When all primitives have been treated in this way , the program then obtains the topology of the different substructures. The loops I_P control this operation and the subroutine SEARCH1 is accessed to update the topology array , TOP1.

Fig.6.18. Flowchart for subprogram C




FOR Ip=1 TO Stop1
FOR K=Stop1 TO Ne1
X1(Ip)=X1(K)
Y1(Ip)=Y1(K) ?
NEXT K
T(1)=Ip
T(2)=K+Dec
SEARCH1
NEXT Ip
FOR Ip=1 TO Stop1
Arc ?
FOR J=1 TO 2
SEARCH
J=2 ?
SEARCH1
NEXT J
NEXT Ip
SUBEND

6.7.2 Subprogram Co

The program reads the data as in section 6.7.1 and then commences the interpretation of the views by using the concept of matching descussed in section 6.5. For all the primitives (lines and arcs) in the elevation of the engineering drawing , denoted by view V_2 , a search is made for all the circles in the plan , denoted by view V_1 for a match. During the matching proces all circles in view V_1 are represented by their corresponding diameters. Each time a match is found , the topology (start and centre points) of the circles in the plan view are recorded together with the corresponding z-values in the elevation. An outer loop is used for the elevation data while an inner loop is established for extracting the data for the plan and executing the matching operations.

Next , the "depth" search strategy is used to identify the circles which meets the match condition. This is done by comparing the values of the z-coordinates. When a sub-structure nearest to the reference line is found , it is identified by a code equals 1. Other substructures are given codes as they are identified and these codes are useful for assembling the substructures. Using these codes and a control variable the subprogram is able to identify whether the model is a fan impeller or not. For models other than fan impeller, the subroutine CONTROL is accessed to construct the relevant Be'zier nets after which the subprogram exits.

However , the interpretation of a fan impeller is discussd in detail here.For fan impeller ,two loops are established to carry out this function. Within the outer loop I_p , the start and finish points of each line in the plan are obtained. Within the loop J , a routine ALONG-ARC is accessed to find out whether or not the start point of each line (within the outer loop) lie on the outer part of the base (cone). A flag is returned to determine the state of the line element. If the flag is equal to unity then a further test is required to determine if the end point lies on the outer part of the base (cone). If the flag again equals unity , then the line element is a candidate that defines a blade. The angles at the start and finish of the line are obtained to determine whether the blade is radial or inclined. Moreover , the length of the blade is evaluated. Because of symmetry , only half of the blades are processed. This is important for the construction of the Be'zier net of the blades.

Once the interpretation is completed , the Be'zier nets for the sub-structures (backsheet , blade and conesheet) are constructed. The first thing that is done before the construction commences is to determine from the z-coordinates of the two circles bounding the cone , the value of the z-coordinates at the inner and outer parts of the cone (Y_i and Y_o) as well as the inner and outer radii of the backsheet and conesheet (R_i and R_o) as shown in Fig. 6.19.

The scales of the 2-D drafting program are re-organised to give equivalent scales with which the Be'zier nets are constructed.

Using the number of control points in u and v-directions (M and N) respectively and the appropriate radii for values of control parameters Dx_1 ,....,Dx_4 , the routine CONTROL is accessed to construct the relevant data for the Be'zier net for the backsheet. The corner points coordinates of the blade on the backsheet as well as those of the conesheet together with the angle of the blade are used for input to the routine DIM-BLADE to construct the Be'zier net for the blade. Lastly, the same procedure as that of the backsheet is repeated for the conesheet using appropriate data to construct the Be'zier net. The data are held in the database for later use by the surface modeller.

Before the program exits , it checks if inconsistent dimensions were input to this module and if so aborts and informs the user through the thermal printer. However , if the interpretation and construction phase are successful , the stand-alone surface modeller is accessed and data are read from the database.

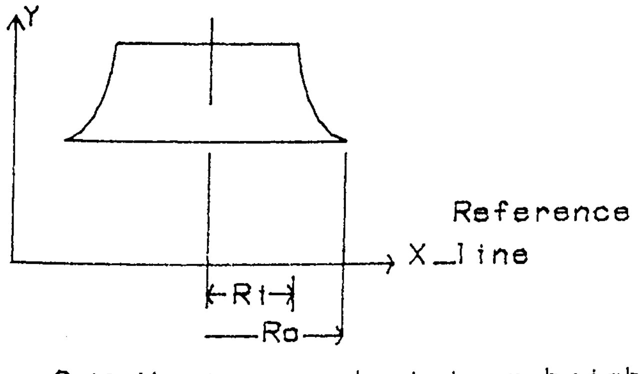
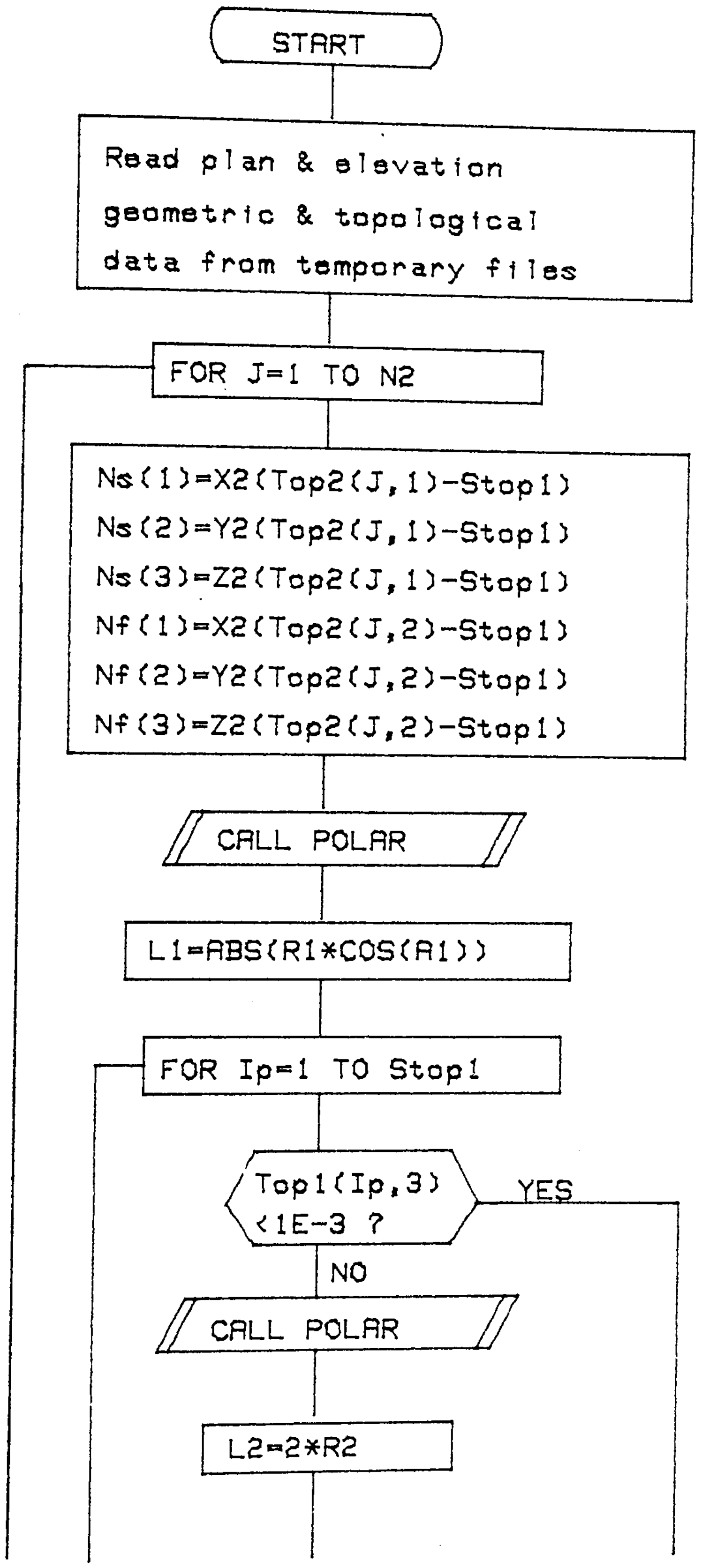
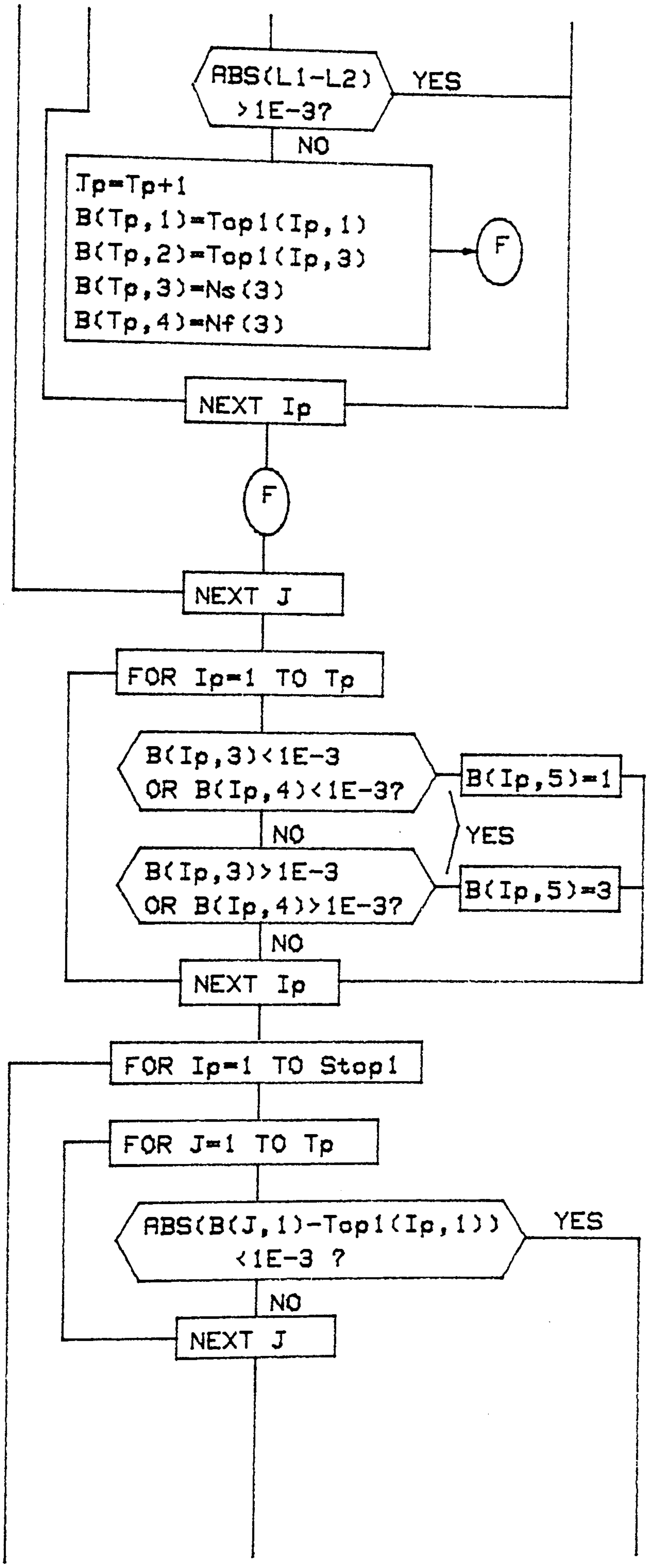


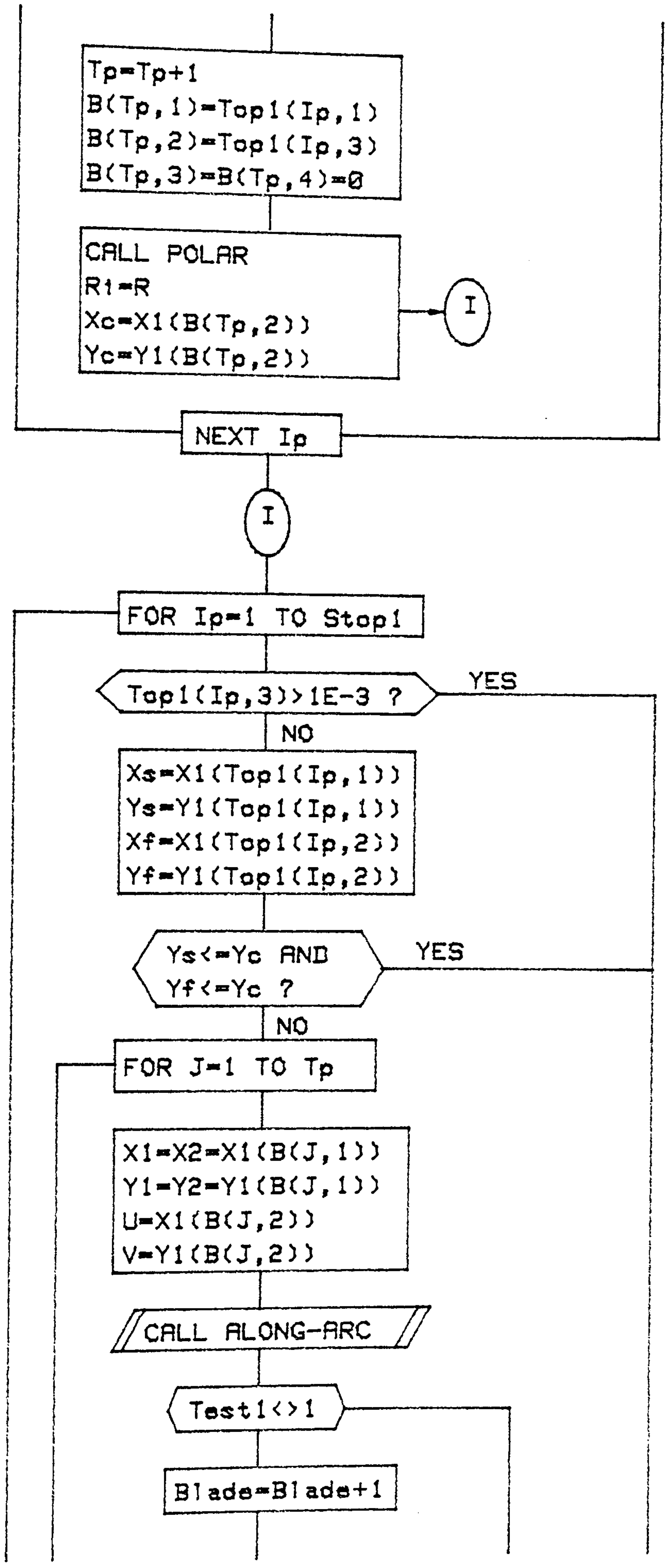
Fig. 6.19 Maximum and minimum heights at inner/outer radii of cone

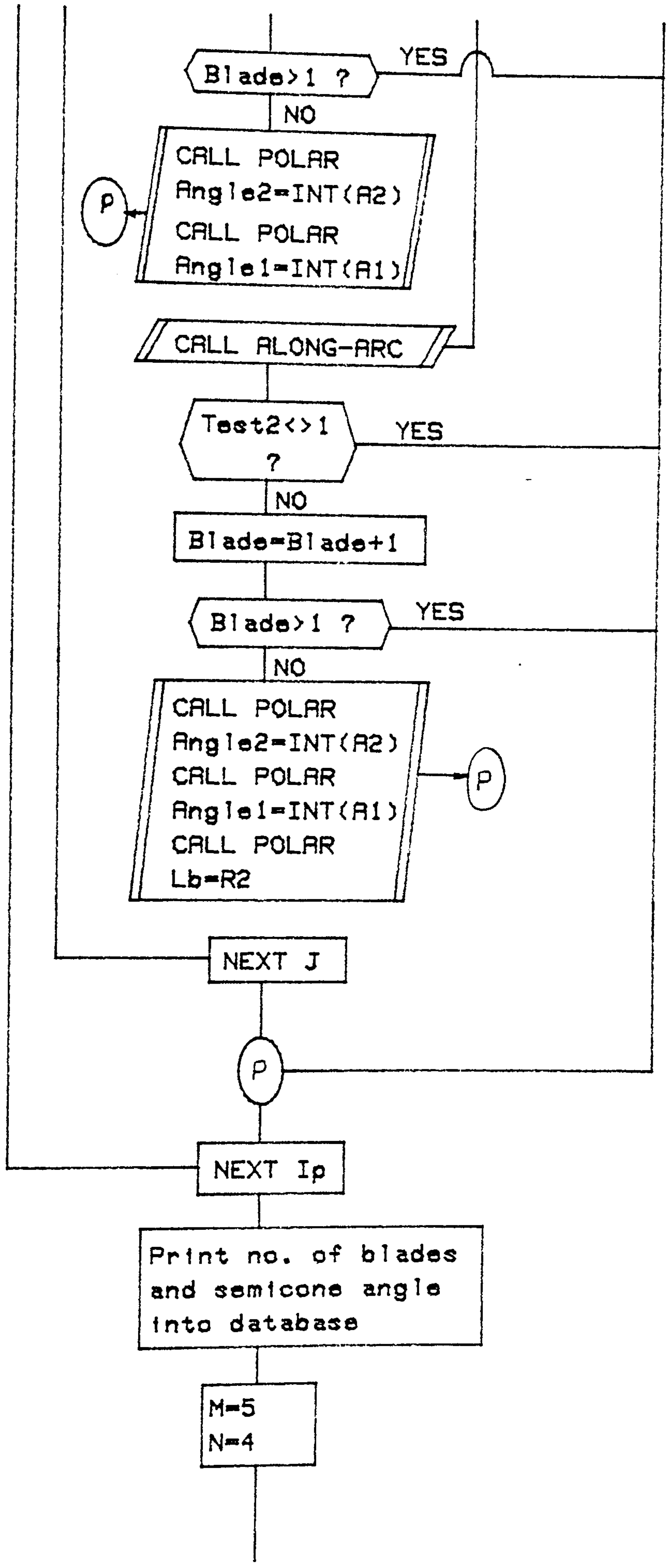
Fig. 6.20 Flowchart for subprogram Co

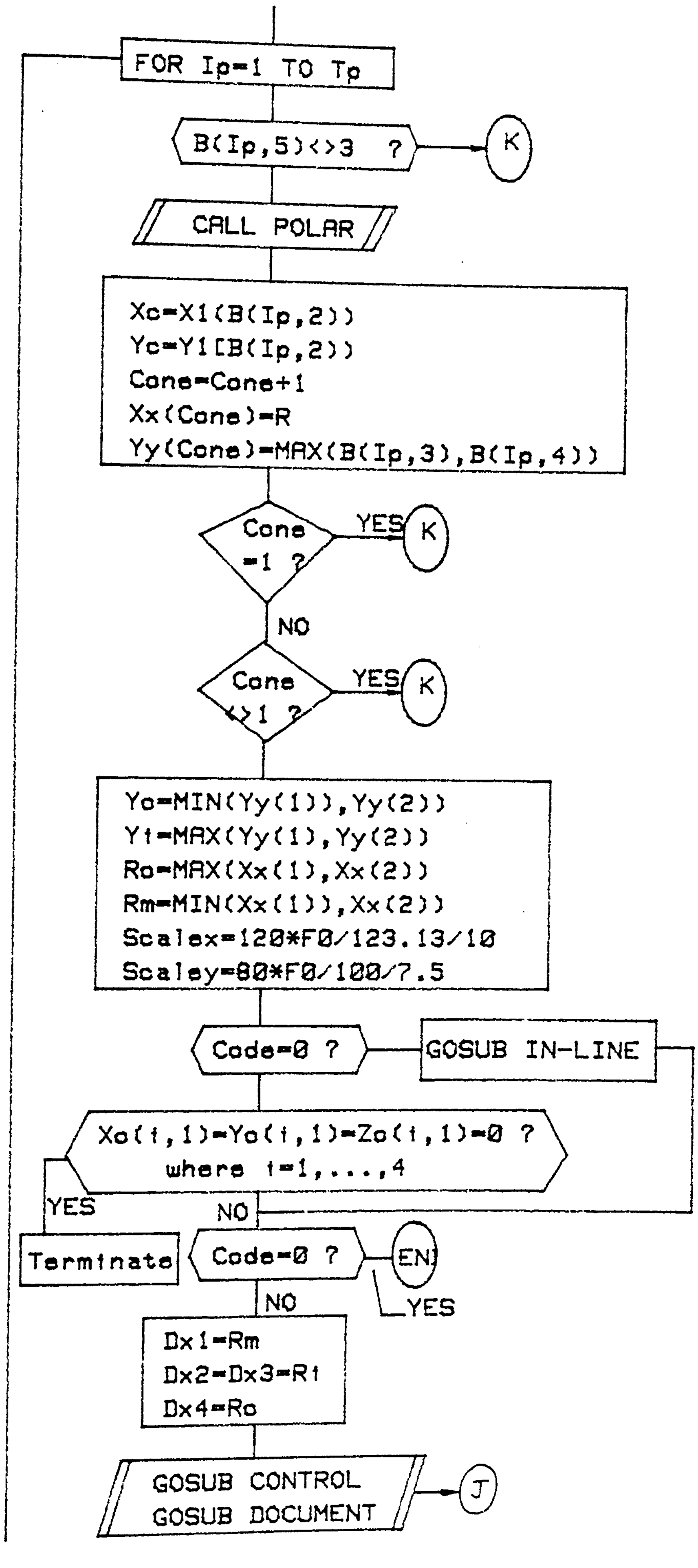

START
Read plan & elevation geometric & topological data from temporary files
FOR J=1 TO N2
Ns(1)=X2(Top2(J,1)-Stop1)
Ns(2)=Y2(Top2(J,1)-Stop1)
Ns(3)=Z2(Top2(J,1)-Stop1)
Nf(1)=X2(Top2(J,2)-Stop1)
Nf(2)=Y2(Top2(J,2)-Stop1)
Nf(3)=Z2(Top2(J,2)-Stop1)
CALL POLAR
L1=ABS(R1*COS(A1))
FOR Ip=1 TO Stop1
Top1(Ip,3) <1E-3 ?
YES
NO
CALL POLAR
L2=2*R2



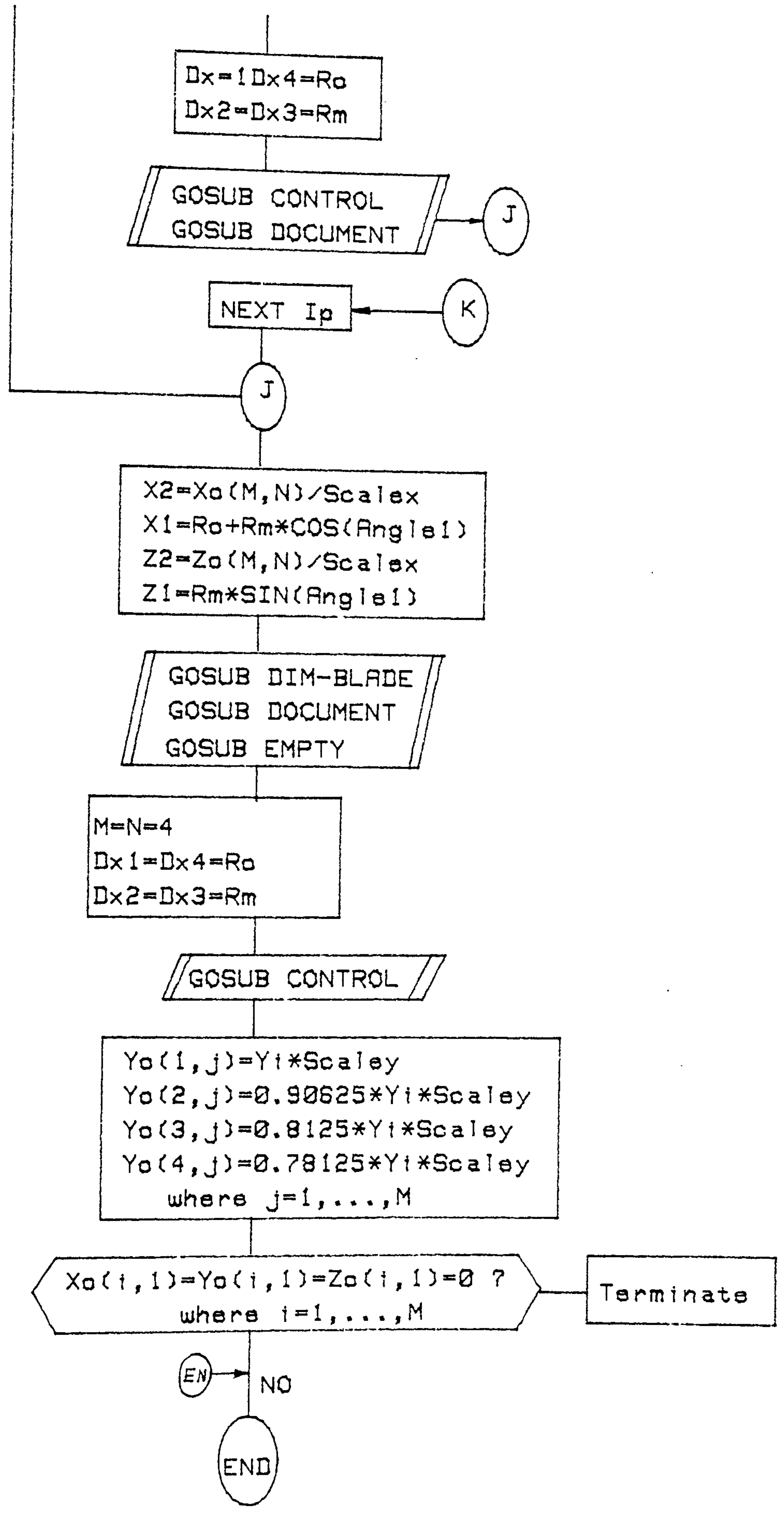
ABS(L1-L2) >1E-3?
YES
NO
Tp=Tp+1
B(Tp,1)=Top1(Ip,1)
B(Tp,2)=Top1(Ip,3)
B(Tp,3)=Ns(3)
B(Tp,4)=Nf(3)
F
NEXT Ip
F
NEXT J
FOR Ip=1 TO Tp
B(Ip,3)<1E-3 OR B(Ip,4)<1E-3?
B(Ip,5)=1
NO
YES
B(Ip,3)>1E-3 OR B(Ip,4)>1E-3?
B(Ip,5)=3
NO
NEXT Ip
FOR Ip=1 TO Stop1
FOR J=1 TO Tp
ABS(B(J,1)-Top1(Ip,1)) <1E-3 ?
YES
NO
NEXT J


Tp=Tp+1
B(Tp,1)=Top1(Ip,1)
B(Tp,2)=Top1(Ip,3)
B(Tp,3)=B(Tp,4)=0
CALL POLAR
R1=R
Xc=X1(B(Tp,2))
Yc=Y1(B(Tp,2))
I
NEXT Ip
I
FOR Ip=1 TO Stop1
Top1(Ip,3)>1E-3 ?
YES
NO
Xs=X1(Top1(Ip,1))
Ys=Y1(Top1(Ip,1))
Xf=X1(Top1(Ip,2))
Yf=Y1(Top1(Ip,2))
Ys<=Yc AND
Yf<=Yc ?
YES
NO
FOR J=1 TO Tp
X1=X2=X1(B(J,1))
Y1=Y2=Y1(B(J,1))
U=X1(B(J,2))
V=Y1(B(J,2))
CALL ALONG-ARC
Test1<>1
Blade=Blade+1


Blade>1 ?
YES
NO
CALL POLAR
Angle2=INT(A2)
CALL POLAR
Angle1=INT(A1)
P
CALL ALONG-ARC
Test2<>1 ?
YES
NO
Blade=Blade+1
Blade>1 ?
YES
NO
CALL POLAR
Angle2=INT(A2)
CALL POLAR
Angle1=INT(A1)
CALL POLAR
Lb=R2
P
NEXT J
P
NEXT Ip
Print no. of blades and semicone angle into database
M=5
N=4



FOR Ip=1 TO Tp
B(Ip,5)<>3 ?
K
CALL POLAR
Xc=X1(B(Ip,2))
Yc=Y1[B(Ip,2))
Cone=Cone+1
Xx(Cone)=R
Yy(Cone)=MAX(B(Ip,3),B(Ip,4))
Cone =1 ?
YES
K
NO
Cone <>1 ?
YES
K
Yo=MIN(Yy(1)),Yy(2))
Yi=MAX(Yy(1),Yy(2))
Ro=MAX(Xx(1),Xx(2))
Rm=MIN(Xx(1)),Xx(2))
Scalex=120*F0/123.13/10
Scaley=80*F0/100/7.5
Code=0 ?
GOSUB IN-LINE
Xo(i,1)=Yo(i,1)=Zo(i,1)=0 ? where i=1,...,4
YES
NO
Terminate
Code=0 ?
EN
YES
NO
Dx1=Rm
Dx2=Dx3=Ri
Dx4=Ro
GOSUB CONTROL
GOSUB DOCUMENT
J


Dx=1Dx4=Ro
Dx2=Dx3=Rm
GOSUB CONTROL
GOSUB DOCUMENT
J
NEXT Ip
K
J
X2=Xo(M,N)/Scalex
X1=Ro+Rm*COS(Angle1)
Z2=Zo(M,N)/Scalex
Z1=Rm*SIN(Angle1)
GOSUB DIM-BLADE
GOSUB DOCUMENT
GOSUB EMPTY
M=N=4
Dx1=Dx4=Ro
Dx2=Dx3=Rm
GOSUB CONTROL
Yo(1,j)=Yi*Scaley
Yo(2,j)=0.90625*Yi*Scaley
Yo(3,j)=0.8125*Yi*Scaley
Yo(4,j)=0.78125*Yi*Scaley
where j=1,...,M
Xo(i,1)=Yo(i,1)=Zo(i,1)=0 ?
where i=1,...,M
Terminate
EN
NO
END

6.7.2.1 Subroutine CONTROL

This subroutine is the key to the construction of the Be'zier net for the backsheet and conesheet. In order to make the routine as general as possible , the y-coordinates of the control points are defined separately. Consider a typical sector of a backsheet (ABCD) shown in Fig. 6.21 with the Be'zier net AFEB ,....,DHGC. In the u-direction there are four control points (i.e. A , F , E , B) for each row. However along the v-direction the user may specify any reasonable number of control points (usually 4 or 5 in number).

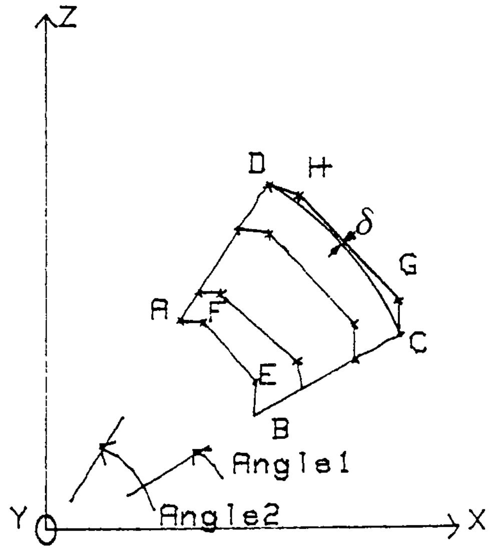
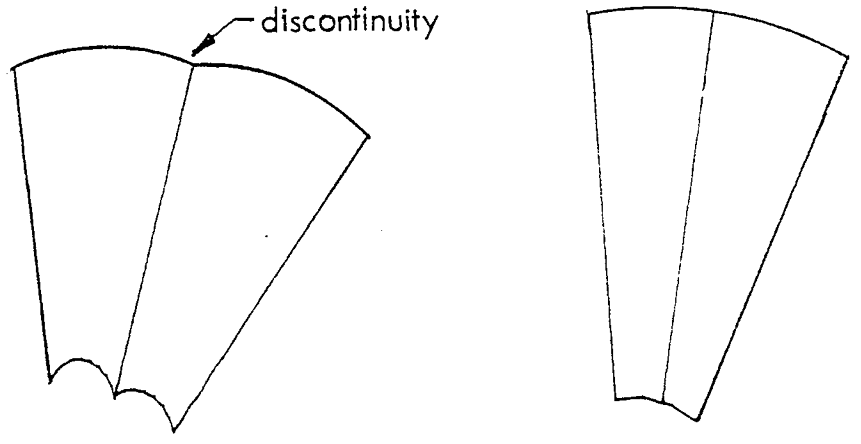


Fig. 6.21 A typical Bezier net for a sectorial structure

Notice that the value δ can be controlled to give different curvatures for the sector. The larger the value the more curved the edge becomes and this would not be acceptable because when the sector is replicated , discontinuities are observed as in Fig. 6.22a. A reasonable value of δ is therefore chosen to give the sector in Fig. 6.22b

The shape of the sector is controlled by the values of the start and finish angles Angle1 and Angle2. Angle1 takes any of the values $2\pi i/N$, $i \in [\emptyset,(N/2-1)]$, where N is the total number of sectors that constitute the complete structure. Angle2 takes any of the values $2\pi j/N$, $j \in [\emptyset,N/2]$. For example if $i = \emptyset$, $j = 1$ and $N = 12$, a $3\emptyset^{o}$- sector is obtained. However , if $j = N/2$, we obtain a semi-circular portion of the structure , i.e Angle2 $=18\emptyset^{o}$.

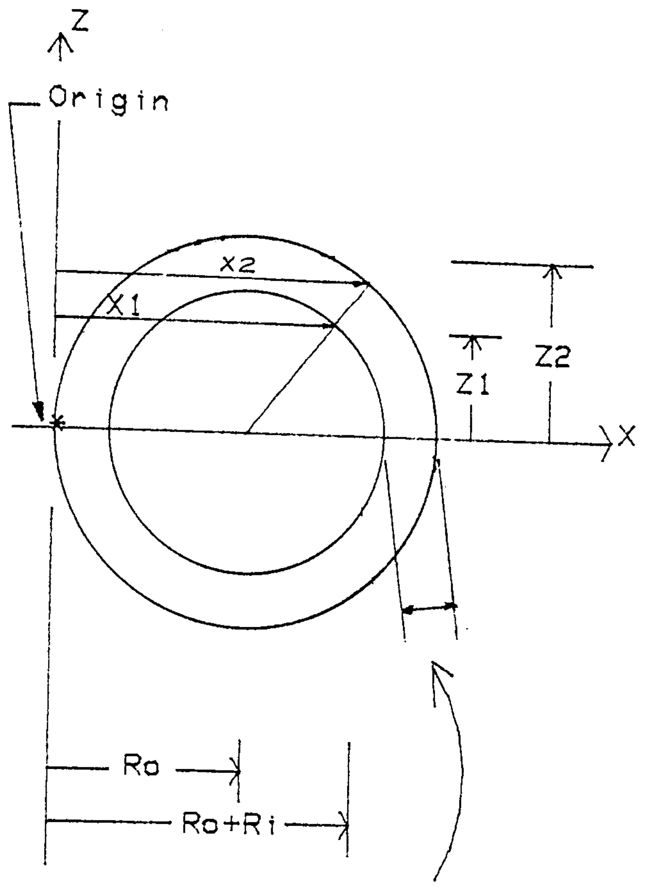


(a) Large value of δ causes discontinuity

(b) δ can be chosen to give smooth sector boundaries

Fig. 6.22 Effect of control polygon on sectorially symmetric objects

The left-hand boundary of the cone or blade along the horzontal centre line is taken to be the origin from where the coordinates of the control points are measured. The number of grids in the v-direction D_x is evaluated as shown in Fig. 6.23



Dx=ABS(Dx1-Dx2)/(M-1)

Fig. 6.23 Evaluating no. of grids in v-direction for hollow disk.

A loop J, controls the operation of this routine. Expression for ROM gives the actual coordinate for the iner part of the structure being considered , while ROM1 gives the corresponding coordinate for the control polygon point at the coordinate considered. These are used for establishing the x-coordinates. Similarly , expressions RAM and RAM1 are established for the z-coordinates. Using the angles Angle1 and Angle2 , the coordinate values for x and z are obtained automatically for control points such as A,F,E,B,....D,H,G,C as the value of J changes within the loop.

Notice from Fig. 6.24 that if the y-coordinates for all the control points are the same then a flat surface is obtained. If the y-coordinates decrease linearly from the inner to outer edge , a straight edge cone is obtained. However , if the variaton follows a curve , then a shell is obtained whose surface lies on the equation of the curve. This is the technique used for the surface definition of the conesheet. If the variation of the y-coordinates is arbitrary then the a free-form surface is obtained. For arbitrary surface , the Be'zier net data are input as numerical data on the key board.

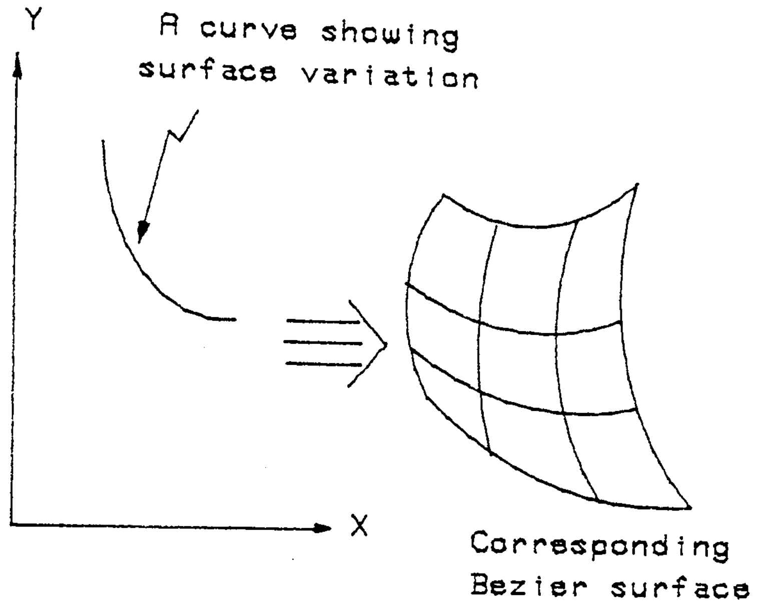
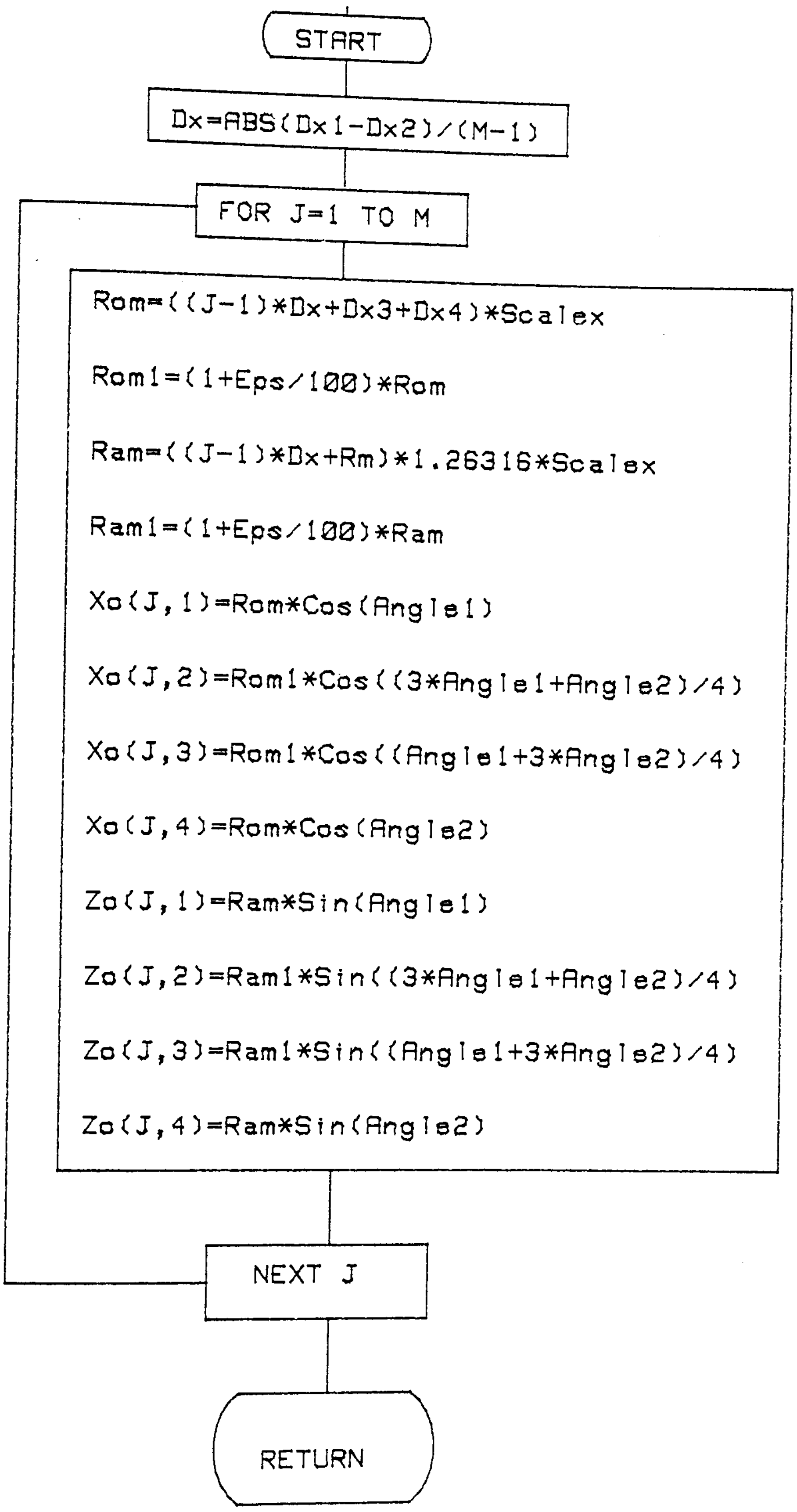

Y
A curve showing surface variation
X
Corresponding Bezier surface

Fig. 6.24 Surface variation given by a curve

Fig. 6.25 Flowchart for subroutine CONTROL



```
START
Dx=ABS(Dx1-Dx2)/(M-1)
FOR J=1 TO M
    Rom=((J-1)*Dx+Dx3+Dx4)*Scalex
    Rom1=(1+Eps/100)*Rom
    Ram=((J-1)*Dx+Rm)*1.26316*Scalex
    Ram1=(1+Eps/100)*Ram
    Xo(J,1)=Rom*Cos(Angle1)
    Xo(J,2)=Rom1*Cos((3*Angle1+Angle2)/4)
    Xo(J,3)=Rom1*Cos((Angle1+3*Angle2)/4)
    Xo(J,4)=Rom*Cos(Angle2)
    Zo(J,1)=Ram*Sin(Angle1)
    Zo(J,2)=Ram1*Sin((3*Angle1+Angle2)/4)
    Zo(J,3)=Ram1*Sin((Angle1+3*Angle2)/4)
    Zo(J,4)=Ram*Sin(Angle2)
NEXT J
RETURN
```

6.7.2.2 Subroutine DIM-BLADE

The principle underlying this subroutine is the same as that of the routine CONTROL except that this time a sector is not considered for the blade. Repetition of the details is therefore avoided. However , the key data needed for the operation are considered. The corner coordinates of the blade are supplied together with the number of control points in the u and v-directions respectively. The routine automatically calculates the coordinates of the control points of the Be'zier net. Refer to Fig. 6.26.

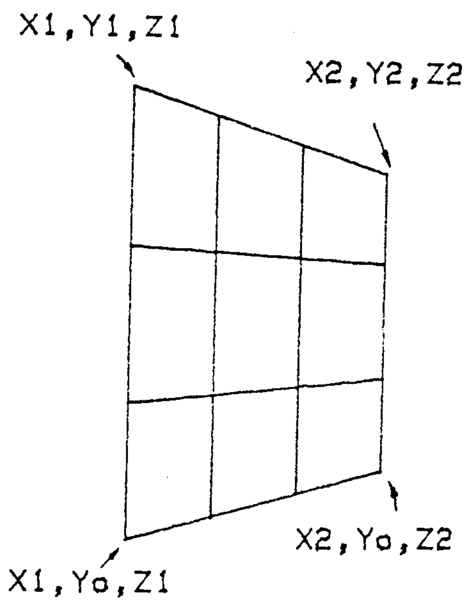
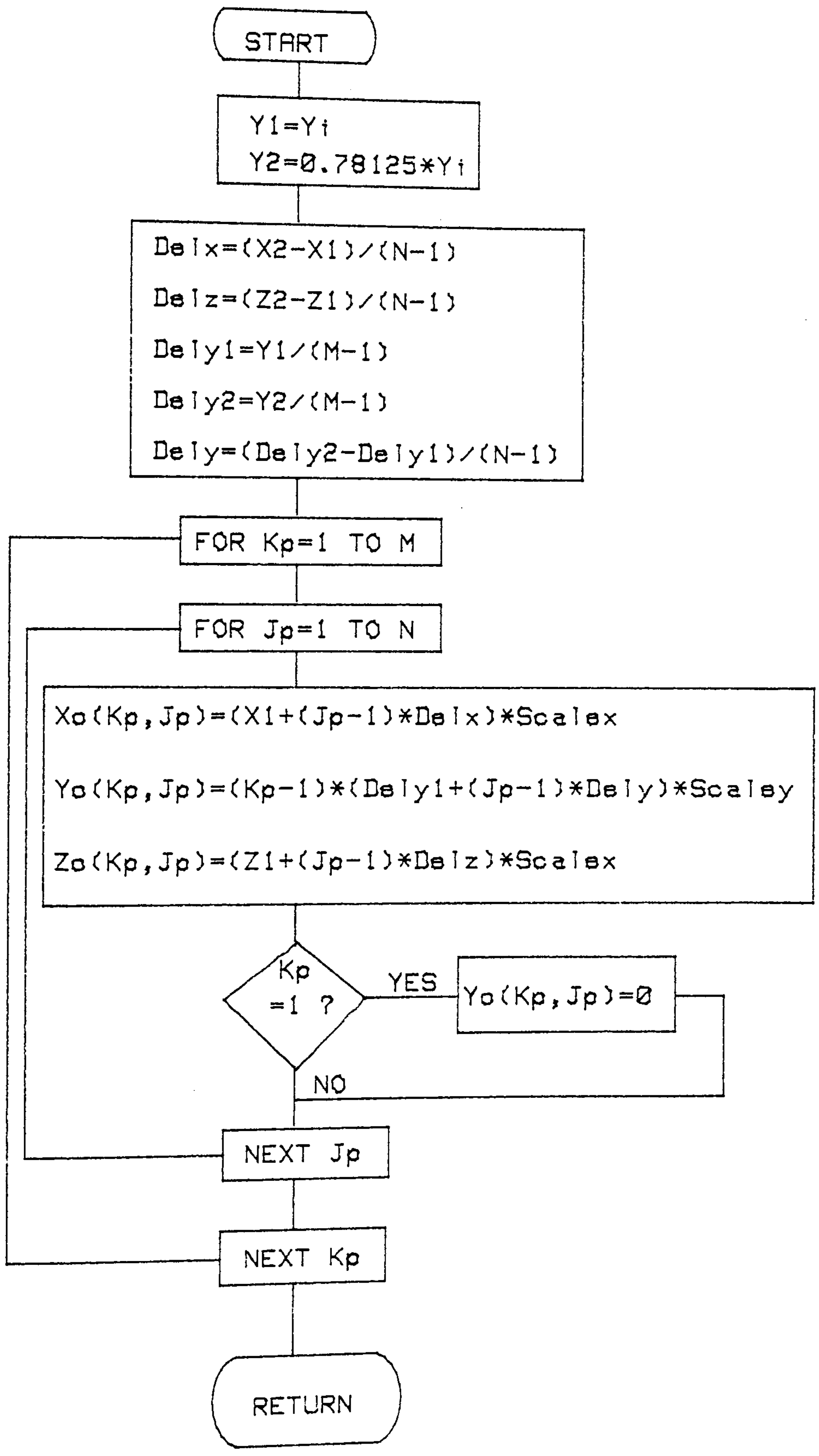


Fig. 6.26 Typical Bezier net for impeller blade

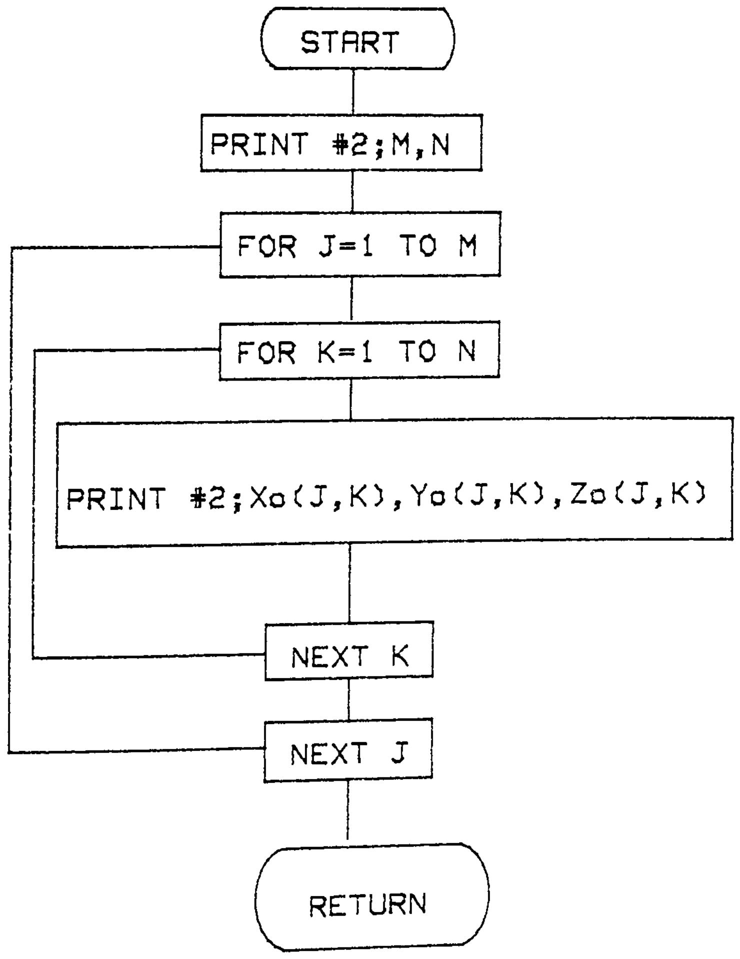
Fig. 6.27. Flowchart for subroutine DIM-BLADE



6.7.2.3 Subroutine DOCUMENT

As the name implies , this routine merely records the number of control points in u and v directons as well as all the Be'zier net coordinates in an orderly manner into the database.

Fig. 6.28. Flowchart for subroutine DOCUMENT



6.8 CLOSING REMARKS

In this chapter , the principle of inter-relationship between sub-parts of a total object via some matching conditions has been applied to the interpretation of engineering drawings. A program which interpretes engineering drawings of plate and shell structures from the MEDIA drafting program and prepares the Be'zier net data for use in the geometric modeller described in Chapter 5 has been discussed. The user can access the program by using function key #6 in the drafting package. Since the MEDIA drafting program cannot descriminate between the data of one view from that of another view , several subprograms were written in order to isolate the data of each view. This step can be ommitted for drafting packages that can store the geometric data of different views separately.

The module for shell structures has been extended to interprete fan impellers which are assemblage of plate and shell structures.

CHAPTER 7

TRANSLATION OF DRAFTING/MODELLER GEOMETRY INTO FINITE-ELEMENT ANALYSIS INPUT DATA.

7.1 INTRODUCTION

Real life objects to be designed are three-dimensional by nature , whereas the traditional tools used by engineers in design (drawing sheets , board and computer screens) are necessarily two-dimensional. This contradiction is implicit in traditional drafting. The basic elements comprising any drawing outline are points , lines and curves.

There is therefore no direct link between drafting , which is two-dimensional and finite element analysis of real life objects which are oriented in space. However , the development of three-dimensional meshed-surface modellers have offered a suitable link between drafting and three-dimensional finite element analysis.

Such modellers can also provide data on the surface properties of components. The ready availability of component areas and properties can be a definite asset to component manufacturers using thin plate and shell structures such as in aerospace and automobile industries where complex geometries are encountered.

In this Chapter , the linking of drafting and finite-element analysis via meshed-surface modelling (Be'zier technique) is considered for general thin plate and shell structures and also for fan impellers in particular. Surface properties computation from the modeller geometric data is also treated to reinforce the usefulness of the geometric modeller in integrated CAD system.

7.2 MESH GENERATION —— A GAP BETWEEN CAD AND FEM

The process of mesh generation if seen in a new light , will make possible a closing of the gap between the rather separate fields of computer-aided drafting and finite element analysis. In the past the process of mesh generation formed part of the data-preparation stage for finite-element analysis. In an extended form it may now serve as an interface between Drafting and FEM by adapting the description of an object as required by one form to the form needed by the other. At the present state of the art , an automated mesh generation could prove most valuable to an integrated computer-aided design system.

Within the context of bridging the gap between CAD and FEM there are two distinct methods for mesh generation of a geometric model and these are described in this section.

7.2.1 MESH GENERATION AFTER GEOMETRIC MODELLING IS COMPLETE

In this case , the mesh generator receives the geometric model (usually 3-D solid) from a geometric modeller and rapidly generates the finite element meshes and prepares input data for a finite element analysis program. A general survey of commercial packages currently available may be found in Fredrikson et. al. [51]. In this section , we review the pioneering work of Wördenweber [52] and one of the key commercially available packages , FEMGEN [53].

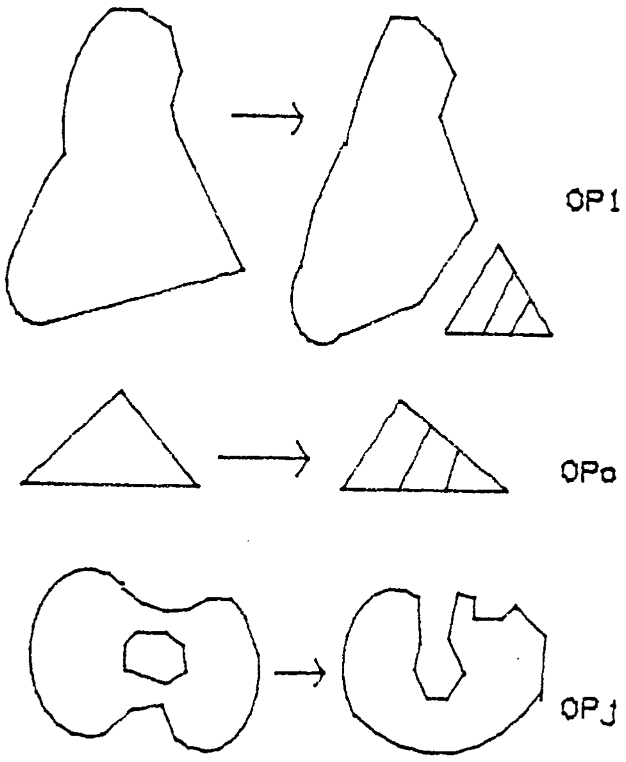
Wördenweber investigated methods for automatic generation of finite element meshes from shapes already defined in a geometric modeller. He applied the theory of decompositon of shapes into triangular or tetrahedra for two and three-dimensional , planar and curvilinear objects. In decomposing the object , he used the Euler formula which gives the minimum number of triangles in polygons and polyhedra. A set of operations were formulated which operate on and decompose a topological polygon or polyhedra. For the case of a polygon , Wördenweber used a set of operations :

Operation OP_1 subracts one triangle and vertex.

Operation $OP_\emptyset$ removes the last triangle ; it thus decreases by one the multiplicity and subtracts three vertices.

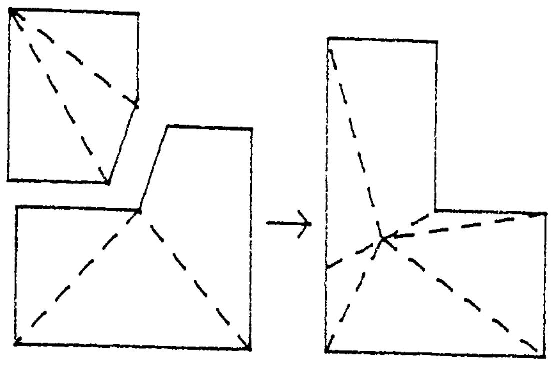
Operation OP_j deletes a hole by joining an interior to an exterior web. It reduces the genus and adds two vertices.

The set of operations shown below is both necessary and sufficient for the decomposition of topological polyhedra.



These topological operations guarantee closure. A set of simple tests were arranged to ensure conformity.

When the domain has been fully decomposed into triangular meshes , neighbouring meshes are then assembled together. Together the decomposition and assembly will now generate a mesh for a given polygon.



Similar operations were included for curved surfaces and solid objects. Wördenweber's work is considered one of the pioneering works in bridging the gap between a geometric modeller and finite element analysis. However , the major drawback in his work is the use of simplex elements i.e three-sided triangular elements , say, whilst quadrilateral elements were not considered.

The general purpose finite element mesh generator FEMGEN has become established as a bridge between CAD and FEM systems. FEMGEN is interfaced to about five commercially available CAD systems (e.g. ROMULUS) on one side and about fifteen commercially available FEM systems(e.g. NASTRAN) on the other side. A model from a 3-D modeller is received by FEMGEN through its interface and such a model is known as a "body". The first task on receiving geometry from a CAD system is to remove detail in the geometric model which is of no relevance to the intended finite element analysis or may unnecessarily complicate the finite element model. The next task is to divide the model into "Femgen bodies" which requires some knowledge of the character of FEMGEN bodies in general and knowledge of the character of the mesh of finite elements which would be suitable for the analysis of the relevant load case or cases. The bodies are defined by splitting existing lines , sketching new lines , defining some FEMGEN surfaces (plane , sphere , cylinder and cone) all with the graphical cursor and defining bodies between pairs of surfaces. After adjusting the divisions on the sides of the surfaces as judged necessary , each body may then be test-meshed. Node symbols are then added to the picture. Then if all the test meshes indicate that the character and quality of the mesh should be acceptable , the user may instruct FEMGEN to generate the whole mesh , which will be almost entirely properly connected , each body with its neighbour.

Both of these methods generate mesh on an object which has already been represented by a separate modeller.

7.2.2 <u>MESH GENERATION BY GEOMETRIC MODELLER</u>

The idea introduced by Coons [5] and Be'zier [7] for modelling engineering objects makes it possible to approximate and describe complex shapes as a collection of finite surface "patches". This approach has stimulated a great deal of research in modelling shapes because it naturally marries CAD to FEM since the surface is actually based on a mesh of curves. This aspect of object modelling has been fully treated in Chapter 5. However , a further interface program is needed to translate the geometry from the "patch" representation into the required finite element idealisation. The treatment of this aspect is discussed in section 7.4.

There is no known published literature on how the "patch" representation is translated into the required finite element idealisation , however , there is scarce literature which shows that such a geometric modelling technique is useful for bridging the gap between CAD and FEM.

Golden [54] used B-spline functions for the geometric representation of models and obtained the finite element meshes from the geometric constructions. However , there was no indication as to how the finite element meshes are obtained from the geometric constructions.

Came [55] in discussing the design procedure for centrifugal compressors at the National Gas Turbine Establishment showed that once the geometric definition of the compressor is complete , the geometric data can be translated from the "patch" representation into the required finite element idealisation.

7.3 INTEGRATION OF CAD AND FEM VIA BE'ZIER SURFACE REPRESENTATION

The Be'zier surface representation falls into the category of section 7.2.2 and plays a dual role of geometric modelling and pseudo idealization for finite element analysis. This class of geometric modellers are considered best for linking CAD and FEM because they remove the need for a complete redefinition of the object for analysis and invariably save computer time and memory. The integrated geometric modelling (Be'zier surface representation) and finite-element analysis in a CAD system can then be represented by Fig. 7.1.

Stress engineers , evident from published literatures , have accepted this geometric modelling technique for geometric representation of simple and complex engineering shells. One of such publications is that by Gallaghar [56].

In the present work , CAD and FEM are integrated via Be'zier surface representation. This has been specifically applied to fan impellers. This method is not only useful for general surface models but also in the Boundary-representation (refer to Chapter 3) of solid objects.

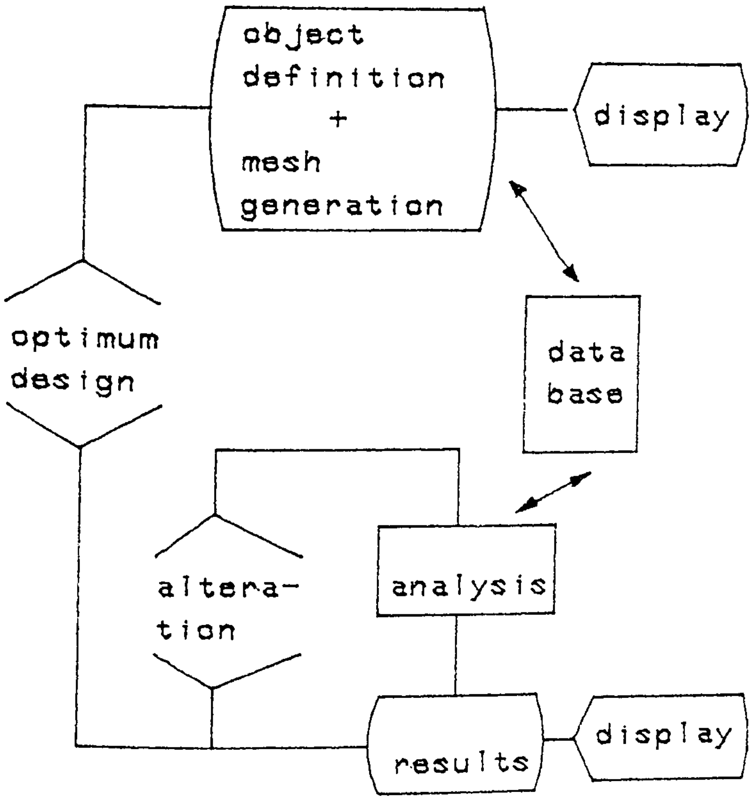


Fig. 7.1 Pseudo mesh generation for finite-element analysis incorporated in geometric modelling (Bezier technique)

7.4 THEORETICAL BACKGROUND

The basis of the translation of the geometry from the "patch" representation into the finite-element idealization is the use of the expressions derived to store the topology of the corner nodes of Be'zier patches.

The way the surface modeller , SUFACE (refer to Chapter 5) records the coordinate values of the surface grids is shown in Fig. 7.2a. As can be seen , the coordinates are recorded when the curves are drawn in the v-direction for constant u-values. These coordinates recorded in a "particular" manner are the information available for the translation process. From these , (see fig. 7.2a) , we seek to identify each Be'zier mesh topology (see fig. 7.2b).

In the surface modeller SUFACE , the number of Be'zier meshes in the u and v directions specified by the user are used to obtain the four corners topology of the Be'zier net. This is very useful for obtaining the topology of each grid in turn.

The author has devised the following set of formulae to meet the needs of the present scheme:

$$N_1 = i$$

$$N_2 = N_1 + N_y$$

$$N_3 = N_1 + N_x*(N_y + 1) \qquad 7.1$$

$$N_4 = N_3 + N_y$$

where i is th number that identifies the first node of the first grid , N_x is the number of subdivisions along the u-direction and N_y is the subdivision along the v-direction for a surface patch. The elegance of these formulae is that they are recursive and therefore can be used to assemble together different patches having different numbers of subdivisions. This application is evident in nodal renumbering scheme discussed in section 7.5.

Consider the patch shown in Fig. 7.2a where N_x=3 , N_y=4 and i=1. From the formulae ,

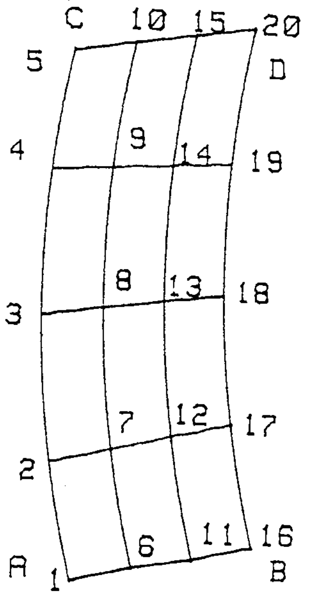
$$N_1 = 1$$

$$N_2 = 1 + 4 = 5$$

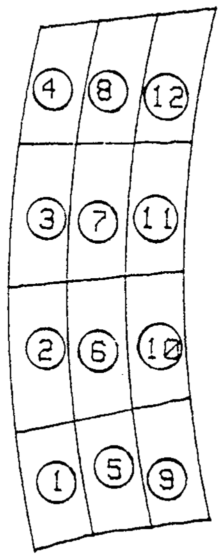
$$N_3 = 1 + 3*(4+1) = 16$$

$$N_4 = 16 + 4 = 20$$

which give the corner node numbers at A,B,C and D respectively.



(a) Nodal numbering for Be´zier meshes



(b) Grids identi-fication

Fig. 7.2 Surface grid topology

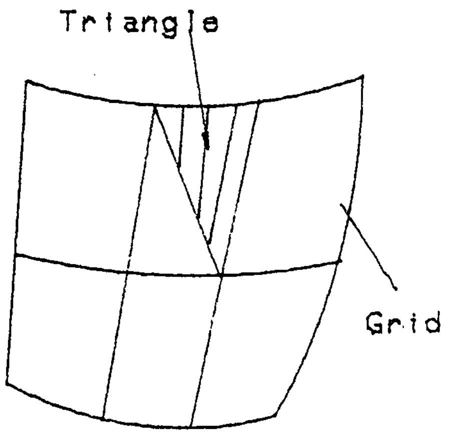
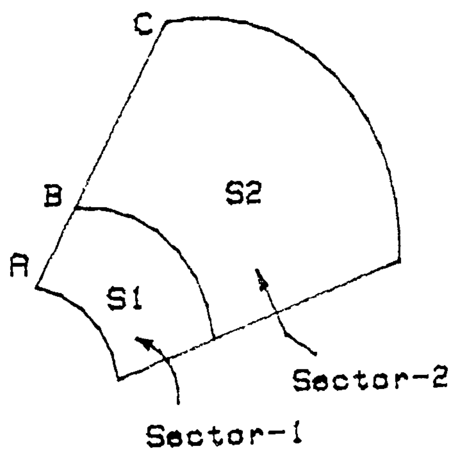


Fig. 7.3 Dividing a grid into two triangles

Given the number of subdivision in the u and v directions , it is therefore possible to obtain the topology of the four-sided Be'zier meshes and upgrade them to eight-node quadrilateral finite elements and six-node triangular finite elements.

In the present scheme the nodal points of the mesh are created and numbered from the lower left-hand corner , vertically and from column to column . From this information the element nodal connections can be established using an appropriate rule. It can be seen from figure (7.3) how a quadrilateral is formed and the shaded area shows how this region is divided to give two triangular elements.

In the case of the actual model of a fan impeller , the inner radii of the backsheet and cone may or may not be equal and the blade is not radial.



Consider the case where the inner radius of the cone is greater than that of the backsheet. In this case the backsheet has to be divided into two regions known as Sector-1 and Sector-2 , where Sector-2 is common to the backsheet and conesheet. Denoting Sector-1 as S1 and Sector-2 as S2 , the following Boolean operators can be applied :

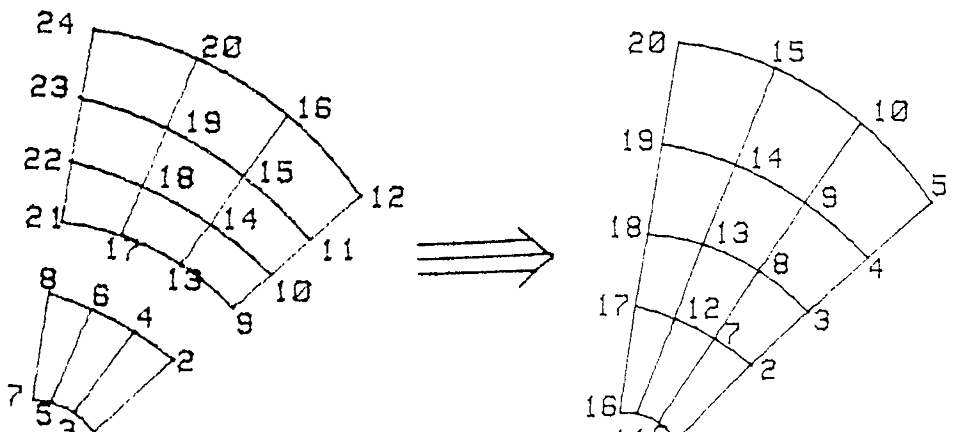
B = S1 U S2

C = B ∩ S1

where B refers to the region of the backsheet and C refers to the region of the cone.

The practical implication of this boolean operation is that the meshes on S1 and S2 have to be differently generated and then merged together to obtain that of the backsheet.

Another interesting point to note is that once Sector-1 is completely numbered , the first point in Sector-2 has a number having unit value greater than the last point of Sector-1. Attempting to merge S1 and S2 will result in the problem of renumbering S2 since it shares a common boundary with S1. Nodes 2 & 9 , 4 & 13 , 6 & 17 and 8 & 21 are common. The first step therefore is to renumber S1 and S2 as shown below.



The final stage of the translation of the Be'zier patches into finite elements involves "tying" together automatically the different finite element meshes. An algorithm which performs this function finds the coincident nodes (those with the same x,y and z coordinates within the surface patch) using the following principle:

For example, two points are coincident if

```
    ABS(X1-X2) < Del
and ABS(Y1-Y2) < Del
and ABS(Z1-Z2) < Del
```

where Del is a positive very small number, say 1E-3.

The higher numbered node is first deleted from the node list. The node list is then renumbered from the first coincident node to the last node number in a sequential order to avoid having unsequenced node numbers. The element connection array will also be updated according to the new node numbering sequence.

The program structure is given in section 7.5 where further information is given on the translation processes.

7.5 THE PROGRAM STRUCTURE

There are ample rewards to be gained once the surface design (Ref. Section 5.7) is completed. The geometric data for input to the finite element analysis package is derived from the database of the surface modeller.

The program PATCH which is the driving unit for the translation of CAD data for input to FEM for plate and shell structures as well as the fan impeller has been written as a subprogram and can be accessed using the ON KEY function of the desk top computer (Hewlett Packard HP9845B). It is linked to other subprograms using the LINK statement available in the HP9845B computer.

The program operates by considering each Be'zier patch in turn , starting from the left-hand corner of each Be'zier patch , and moving vertically through a column and then from column to column. The following steps indicate how the program is organised :

1. The corner node numbers of each patch are read from an array B.

2. The Be'zier meshes of the model are interpreted using the corner node numbers of step 1 and some constructed loops.

3. The interpreted four-sided Be'zier meshes are then upgraded to 6-sided triangular or 8-sided quadrilateral finite elements.

4. The node co-ordinates and element connections are found.

5. Steps 2-4 are repeated for the remaining patches until the whole array B has been scanned.

6. The total finite element meshes are then merged and the data of step 4 are updated .

7. By using the key #Ø, the user accesses the menu so that other data such as material property , boundary and loading conditions are interactively added.

8. The total finite element data is then stored in a specified file.

The generated data can be used in the SMILOF or IMPSMF finite element analysis programs.

The Be'zier patches are dealt with in turn , starting from the left-hand column of Be'zier meshes and working vertically , then progressing from column to column. Two loops are established which control this operation , namely IB and J.

Following the flowchart , it is observed that the first subroutine to be encountered is SIDE. This routine determines the finite element nodal co-ordinates for the nodes along the left-hand vertical edge of each Be'zier mesh as the program works through columns of Be'zier meshes. Mid-side nodes are interpolated and as the subroutine accesses another subroutine PO to store the nodal coordinates , a node number is allocated to each set of nodal coordinates. A similar routine MID is encountered which determines the finite element nodal co-ordinates for mid-side nodes between adjacent vertical edges of the Be'zier meshes. When triangular elements are considered , then the subroutine MID determines the nodal co-ordinate at the centroid of each Be'zier mesh.

Consider a Be'zier patch of twelve Be'zier meshes as shown in figure (7.4). As previously stated , the program operates on each Be'zier mesh in turn , generating the element nodal coordinates. In the present example Be'zier mesh 1 would be operated on first , generating nodal co-ordinates 1-3 and 1Ø-12. In order to achieve this an expression must be available which can define the node numbers along the vertical edge and mid-side nodes between adjacent vertical edges within the Be'zier mesh under consideration. This expression has been developed through the controlling variables IB and J mentioned already.

The main control variables are

Mls , Mlf , T , A , B , C , D , Ndel , NØ1 and NØ2

The corner node numbers and subdivision in the vertical direction of the Be'zier patch are used to construct a nested loop system which can be used in identifying each Be'zier mesh topology and geometry. The outer loop , denoted by J , is used to traverse the Be'zier

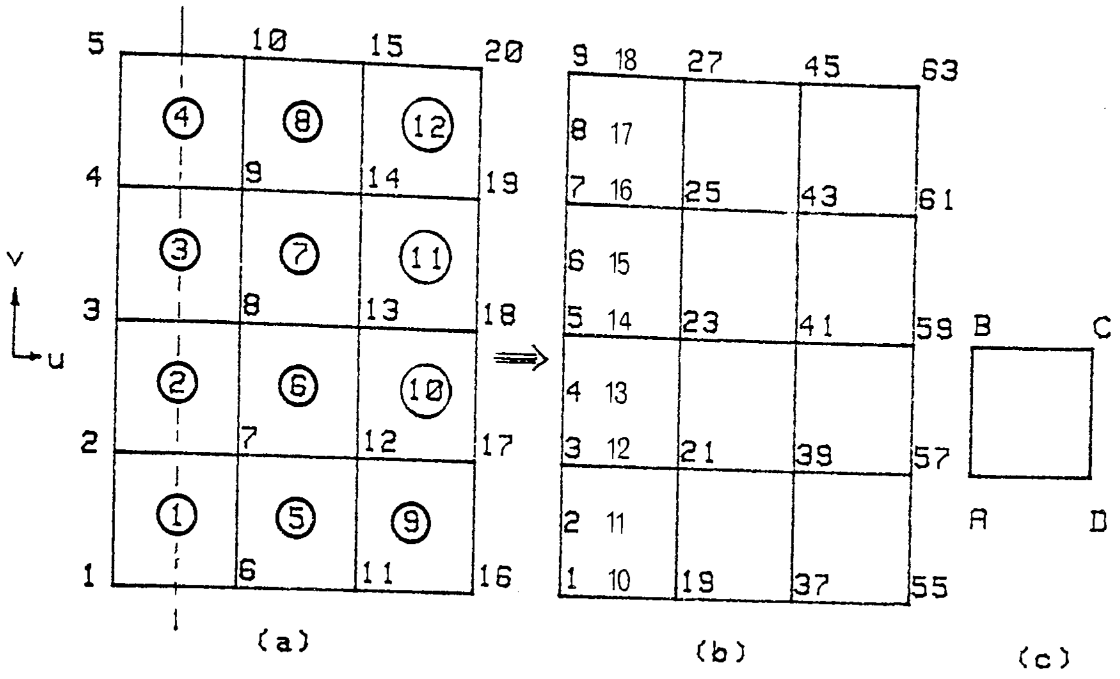


Fig. 7.4 Basic numbering of a Be'zier surface patch.

patch horizontally using the specified patch subdivision. The inner loops each denoted by K are used to increment the vertical node numbering and are limited by the number of the Be'zier patch subdivision. One of the inner loops deals with edge nodes while the other deals with mid-side nodes between adjacent edges. For example in Fig. 7.4a , the outer loop , denoted by J deals with Be'zier patch node numbers 1 , 6 , 11 and 16. One of the inner loops then deals with the edge nodes 1 and 2 , 2 and 3 , 3 and 4 and then 4 and 5 in that order and access is made to the subroutine SIDE to interpolate the mid-side nodes .This subroutine then accesses another subroutine PO for storing the coordinates and ensuring that duplication of nodal coordinates is avoided. Therefore after the first cycle of this loop , the coordinates 1--9 of Fig. 7.4b are obtained. An overall numbering scheme is also incorporated in the subroutine PO. The other inner loop considers mid-side nodes along the dashed line (between vertical edges) in Fig. 7.4a and again access is made to subroutine MID to interpolate nodal coordinates along this dashed line. After the cycles of both inner loops are completed , the outer loop automatically sets the next nodes to be considered as 6--1Ø, then 11--15 and finally 16--2Ø. However , when the last edge (nodes 16--2Ø) are considered , the inner loop which determines the mid-side nodes between adjacent edges does not function.

Using these variables it is possible to define the element nodes within any of the Be'zier mesh , thus

```
FOR J = NØ1 TO NØ3 STEP Ndel

    M1f = J

    M1s = J + Ndel - 1
```

where NØ1 is the first corner node number of a Be'zier patch and NØ3 is the third corner node number (refer to equation 7.1). For this example NØ1 = 1 and NØ3 = 16. Ndel is the number of subdivision in the vertical direction plus one. In this case Ndel = 5. For the first cycle of the outer loop , J = 1. It follows that M1f = 1 and M1s = 5.

The first inner loop then operates within the limits of M1f and M1s (refer to Fig. 7.4a) :

```
FOR K = M1f TO M1s-1

    T = K + 1
```

For K = M1f = 1 then T = 2. Therefore the co-ordinates of nodes 1 and 2 are used to access subroutine SIDE which interpolates these values to obtain the mid-side node co-ordinates. For K = M1f + 1 = 2 then T = 3 and the process continues until K = M1s - 1 = 4 in which case T = 5.

The second inner loop then commences within the same limits

```
FOR K = Mlf TO Mls - 1

    T = K + 1
    A = K
    B = T
    C = B + Ndel
    D = A + Ndel
```

```
When K = Mlf = 1 then
    A = 1
    B = 2
    C = 2 + 5 = 7
    D = 1 + 5 = 6
```

Therefore the corner nodes of the first Be'zier mesh are identified.The subroutine MID when accessed , uses adjacent nodes A , B and C , D to obtain mid-side nodes and the node at the centroid of each Be'zier mesh is easily obtained from these information for use when triangular elements are dealt with.

When the loop cycle for the control variable J is completed , then all the Be'zier meshes would have been identified and the finite element nodes obtained. Other patches are then considered with the help of the outermost loop designated by IB.

The next subroutine to be accessed is NODAL. In this subroutine , the total node number is obtained and the number for each node is read into an array NODE. These are used later for merging all finite element meshes obtained from the processes described in this section.

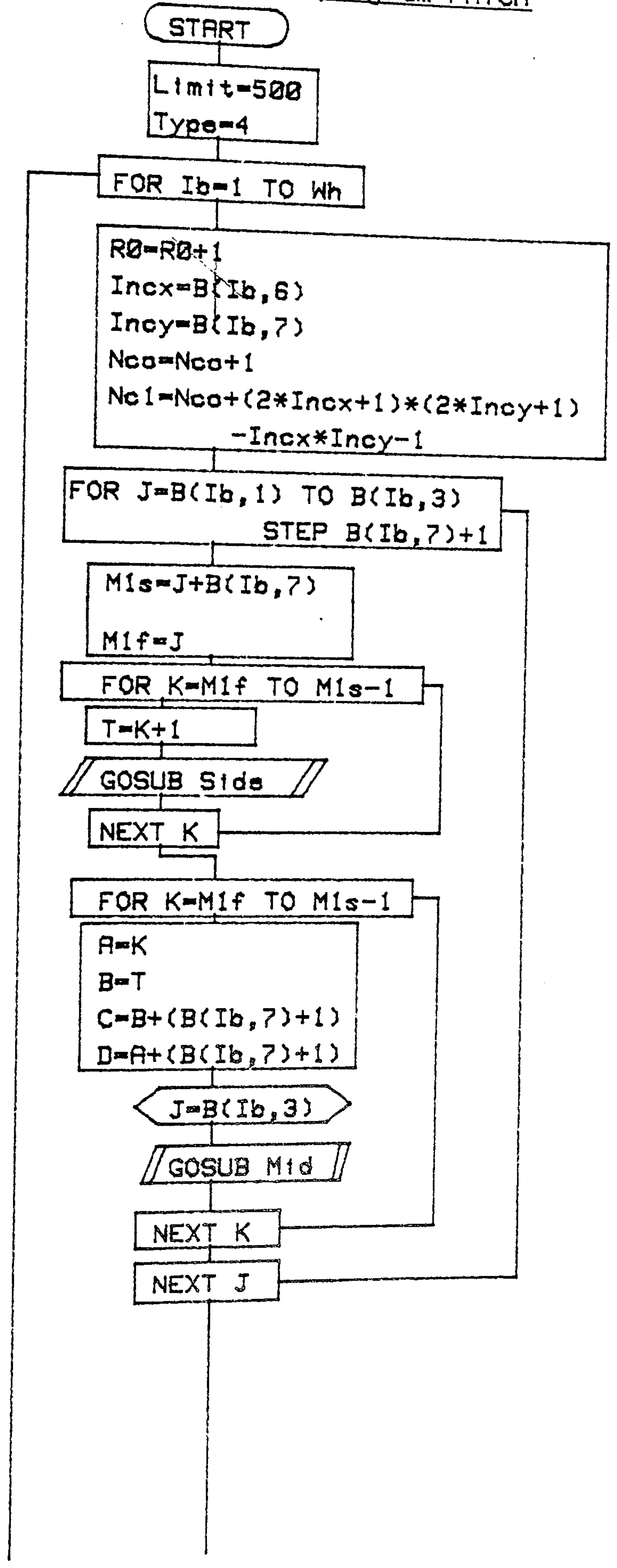
It was observed that the way that element connections are numbered vary with the application programs SMILOF and IMPSMF. While the element connection proceed in a clockwise manner in IMPSMF , it proceeds in an anti-clockwise manner in SMILOF. Therefore , two subroutines have been written for element connections and using the variable S , the program branches to the appropriate subroutine to obtain the element connections for either SMILOF or IMPSMF. Details and flowharts of the subroutines(known as MESH3Q and MESH4 for triangular and quadrilateral elements respectively) that establish the element connections for the application program SMILOF are presented in this section while those of IMPSMF(known as MESS3 and MESS4 for triangular and quadrilateral elements respectively) are omitted since the operations are similar.

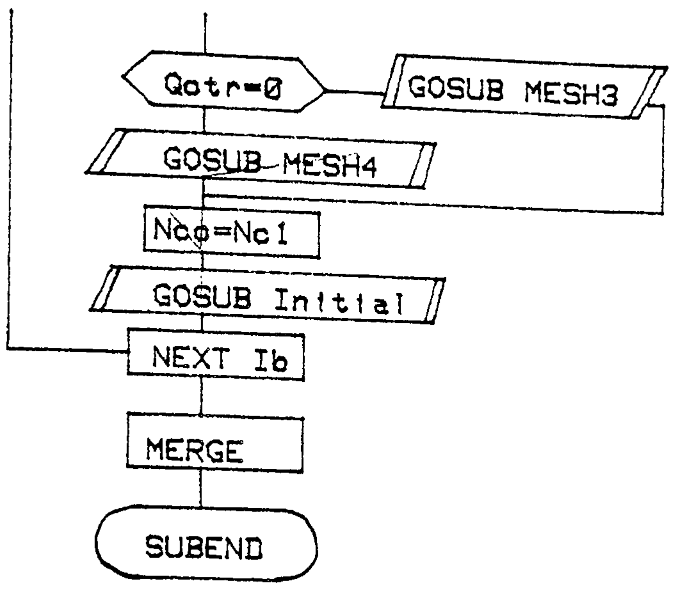
The final routine to be encountered is the merge algorithm which was saved in a data file MERGE .It can be accessed using the LINK statement of the HP9845. When linked , the subprogram MERGE is called which merges the total finite element meshes.

Before the subprogram exits , the element nodal connections are arranged in a format acceptable to the application programs SMILOF and IMPSMF.

The flowchart for the main subprogram PATCH as well as the flowcharts of subroutines and subprogram refered to in this section are also found in this chapter.

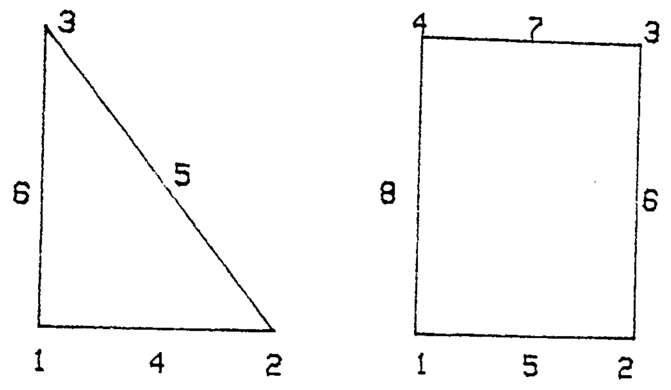
Fig.7.5 Flowchart for subprogram PATCH




Qotr=0
GOSUB MESH3
GOSUB MESH4
Nco=Nc1
GOSUB Initial
NEXT Ib
MERGE
SUBEND

7.5.1 Subroutines MESH4 AND MESH3Q

These subprograms operate in the same fashion and determine the element connections for the quadrilateral and triangular finite elements respectively. The elements are defined by six nodes in the triangular element and eight nodes in the quadrilateral element. The numbering sequence is shown below.



The numbering sequence is important and is used to identify the shape of the element , via its nodal coordinates.

(a) "MESH4"

The element connections for quadrilateral elements are determined by this subroutine.The recast form of the recursive formulae of equation 7.1 is used automatically to determine the first node number of the finite element for each Be'zier patch. This node number becomes N1t which is updated each time a patch is treated. A set of expressions which yield the element connections for each finite element is given as follows:

$$N1 = N1t$$

$$N2 = N1 + Inc$$

$$N3 = N2 + 2$$

$$N4 = N1 + 2$$

$$N5 = N1 + 2*(Inc + 1) - JØ \qquad \ldots\ldots \; 7.2$$

$$N6 = N3 - 1$$

$$N7 = N5 +1$$

$$N8 = N1 + 1$$

where Inc is three times the number of subdivision in v-direction plus two.

These element connections are then held in the array ICON for use in the merge algorithm.

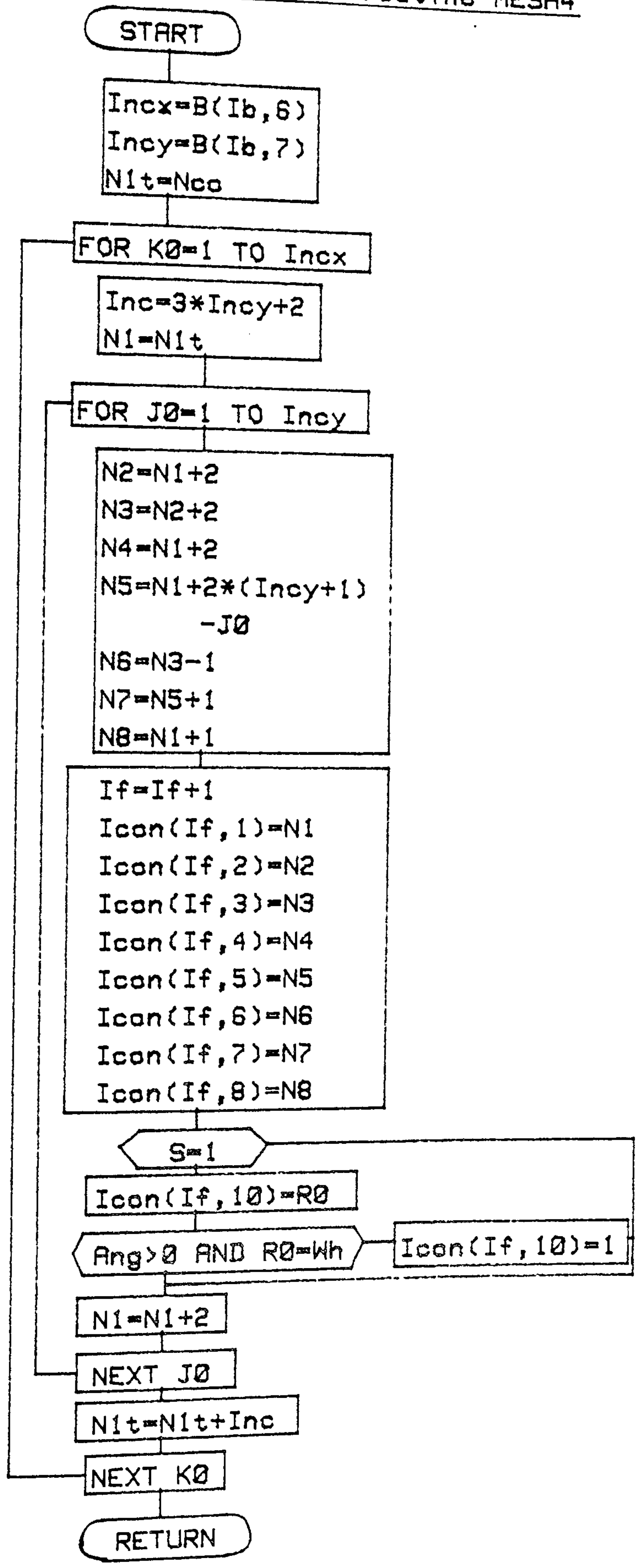
The steps involved in this subprogram shown in the corresponding flowchart are as follows:

1. The parameter Inc is determined from the number of subdivision in the v-direction.

2. Two counter loops KØ and JØ are constructed to control the scheme for the element connection using the expressions in equation 7.2.

3. The element nodal connections are held in array ICON and the numbering scheme for element connection updated.

Fig. 7.6 Flowchart or subroutine MESH4



(b) <u>"MESH3Q"</u>

As has been already mentioned , this routine operates in the same fashion as the subprogram MESH4. The element connections for one triangle of a quadrilateral element is given as

N1 = N1t

N2 = N1 + 2*(2*Incy + 1)

N3 = N2 + 2

N4 = N1 + 2*Incy + 1

N5 = N2 + 1

N6 = N1 + 1 ..... 7.3

where Inc equals four times the number of mesh in v-direction plus 2. For the other triangle of a Be'zier mesh ,

N7 = N1

N8 = N3

N9 = N1 + 2

N1Ø = N6

N11 = N1Ø + 1

N12 = N1 +1 ..... 7.4

The following steps correspond to the flowchart for the subroutine MESH3Q.

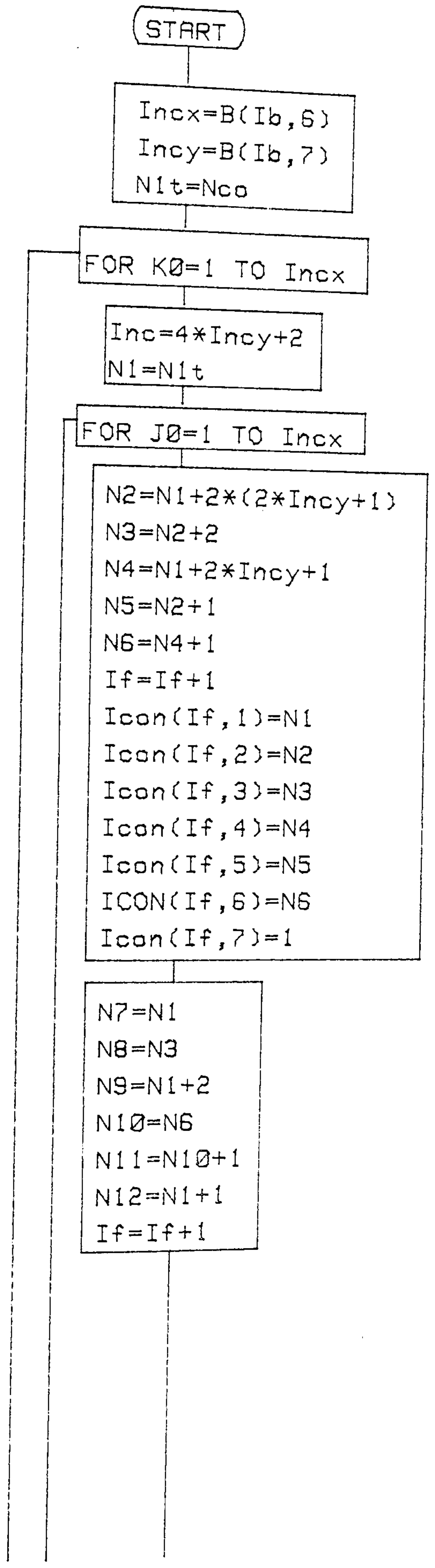
1. The variables INCX and INCY are initialised as limits for loops KØ and JØ respectively.

2. Variable INC which is part of the numbering scheme is set as a function of variable INCY.

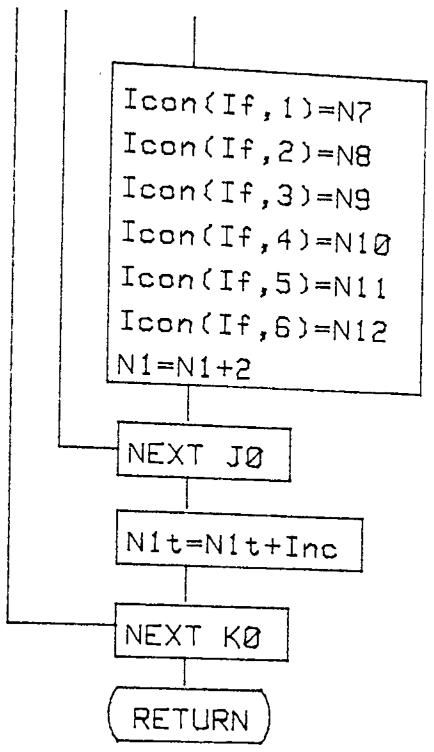
3. Each Be'zier mesh breaks down into two triangular elements , thus two sets of element nodal connections need to be computed. Variables N1....N6 are used to compute the nodal connections of one triangle while variables N7...N12 are used for the other triangle.

4. The element nodal connections are held in array ICON.

Fig. 7.7 Flowchart for subroutine MESH3Q



```
START
Incx=B(Ib,6)
Incy=B(Ib,7)
N1t=Nco
FOR K0=1 TO Incx
    Inc=4*Incy+2
    N1=N1t
    FOR J0=1 TO Incx
        N2=N1+2*(2*Incy+1)
        N3=N2+2
        N4=N1+2*Incy+1
        N5=N2+1
        N6=N4+1
        If=If+1
        Icon(If,1)=N1
        Icon(If,2)=N2
        Icon(If,3)=N3
        Icon(If,4)=N4
        Icon(If,5)=N5
        ICON(If,6)=N6
        Icon(If,7)=1
        N7=N1
        N8=N3
        N9=N1+2
        N10=N6
        N11=N10+1
        N12=N1+1
        If=If+1
```


Icon(If,1)=N7
Icon(If,2)=N8
Icon(If,3)=N9
Icon(If,4)=N10
Icon(If,5)=N11
Icon(If,6)=N12
N1=N1+2
NEXT J0
N1t=N1t+Inc
NEXT K0
RETURN

7.5.2 Subprogram "MERGE"

As the name implies , this subprogram merges all the finite elements into the total mesh using the principles established in section 7.4. Two loops Ip and K are used to compare each node's x , y , z coordinate values with other nodes , where Ip is used to identify the node under consideration and K for all other higher nodal values. Each time a pair of coincident nodes is encountered , the node having the lower node number held in array NODE is stored in an array NEW while the one with higher node number is stored in an array IOLD. These two arrays have a counter ICO which counts the number of times coincident nodes are encountered. The value of the higher node number initially stored in array NODE is then replaced by a value of -1 to show that it is a candidate to be deleted. Another loop I_p is then used to search through the arrays NEW and IOLD and all nodes having value of -1 in the array NODE are deleted. A counter ISUB is used to count how many times coincident nodes are encountered. When all nodes have been examined the old numbering is changed to a new numbering sequence and the element connection changed to correspond with the new node numbering.

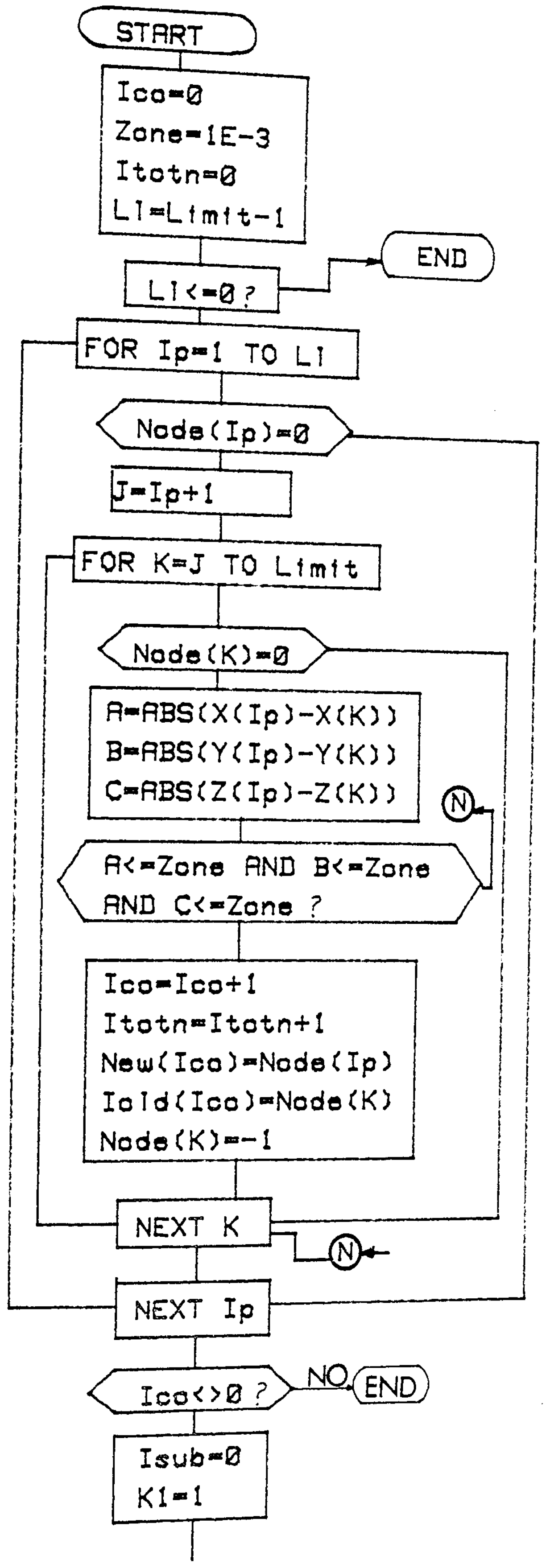
A summary of the sequence of operations in this subprogram which corresponds to the flowchart MERGE is given as follows :

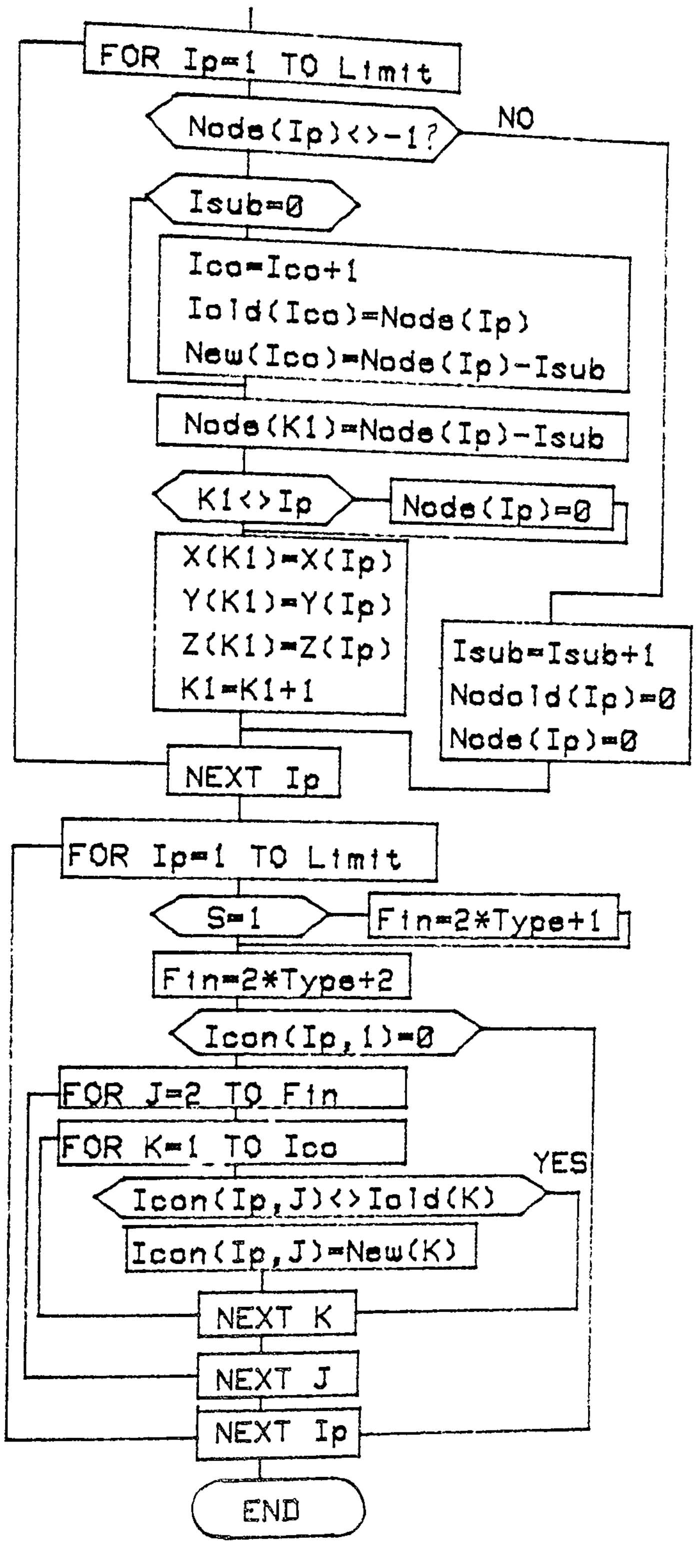
1. All coincident nodes are found while the lower and higher node numbers at such points are held in array NEW and IOLD.

2. The node numbering sequence is then changed using the data of step 1 and the total number of coincident nodes.

3. With this new numbering sequence , a new element connection is obtained.

Fig. 7.8 Flowchart for subprogram MERGE




FOR Ip=1 TO Limit
Node(Ip)<>-1?
NO
Isub=0
Ico=Ico+1
Iold(Ico)=Node(Ip)
New(Ico)=Node(Ip)-Isub
Node(K1)=Node(Ip)-Isub
K1<>Ip
Node(Ip)=0
X(K1)=X(Ip)
Y(K1)=Y(Ip)
Z(K1)=Z(Ip)
K1=K1+1
Isub=Isub+1
Nodold(Ip)=0
Node(Ip)=0
NEXT Ip
FOR Ip=1 TO Limit
S=1
Fin=2*Type+1
Fin=2*Type+2
Icon(Ip,1)=0
FOR J=2 TO Fin
FOR K=1 TO Ico
YES
Icon(Ip,J)<>Iold(K)
Icon(Ip,J)=New(K)
NEXT K
NEXT J
NEXT Ip
END

7.6 MATERIAL PROPERTY SPECIFICATION , LOAD AND BOUNDARY CONDITIONS

The data input (see fig. 7.9) for any finite-element analysis program is made up of geometric , material and loading data. So far , only the geometric data preparation has been discussed and this is the most difficult task for complex structures. The present trend is to duplicate the finite element model data in a database and a temporary file. Depending on the specific format of the finite-element analysis program , the user then specifies interactively the loadings , boundary conditions , thickness and material properties. Wu and Abel [57] adopted this method of adding these data after the geometric data have been derived from a modeller.

In the present research , the same trend of separating the geometric data from the complementary finite element analysis input data is adopted because of its advantages. Once the finite element model has been fully described and edited ,the user can then access a menu specially designed for the specification of the material properties as well as the load and boundary conditions. This makes it flexible to use the same geometric data for different finite-element analysis programs. The user only has to acquaint himself with the specific format of the a particular application program and where necessary include his own editing program to specify the material property specification as well as loading and boundary conditions.

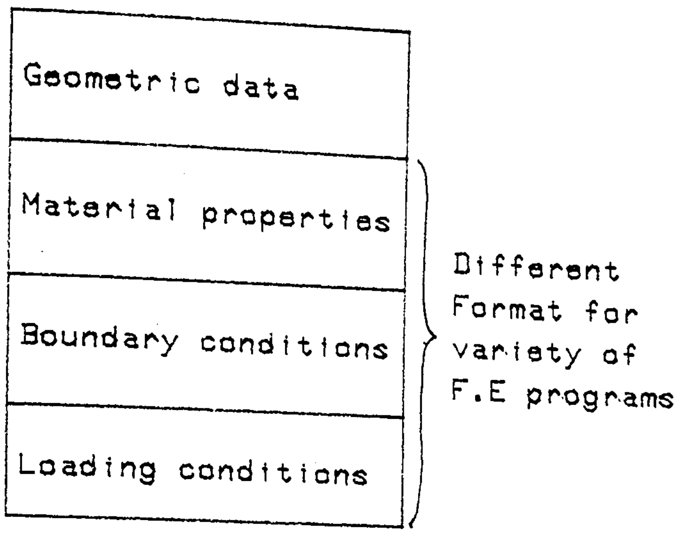


Fig. 7.9 Complete F.E data input

1. NUMBER OF JOB TO BE SOLVED
2. TYPE OF ELEMENT
3. NUMBER OF SETS OF FORCES
4. PRINCIPAL STRESSES & STRAINS
5. NO. OF NODES WHERE SKEWED BOUNDARY CONDITION ARE APPLIED
6. NO. OF METERIAL
7. OUTPUT
8. GRAVITY
9. NORMAL DISTRIBUTED PRESSURE
10. ROTATIONAL SPEED:RPM
11. NO. OF LIKE ELEMENTS,THICKNESS, STRINGS OF LIKE NODES
12. NO. OF SPECIFIED NODES
13. NO. OF SPECIFIED MIDSIDE NODES
14. ELASTIC CONSTANTS
15. CHECKING DATA FOR LINKING TO F.E.M
16. STORING ACCEPTED INPUT DATA FOR LINKING TO F.E.M
17. EXIT

FIG. 7.10 Menu for supplementary finite element analysis input data

The advantages of having the geometric data initially separated from the complementary finite-element analysis data and the use of interactive method in specifying such data are as follows :

(i) The common geometric data held in the databse and the temporary file can be used repeatedly for different analysis programs provided a small editing program is available for the addition of the material properties , boundary and loading conditions.

(ii) Preprocessing time is greatly shortened.

(iii) The opportunity for making errors is significantly reduced since a visual check can be made through the graphic devices.

(iv) Since all errors can be deleted prior to dumping the input data to a finite element analysis program, cost normally associated with running such program is substantially reduced.

In the context of the work reported in this thesis , a menu (see fig. 7.10) is available for the user to specify the material properties , boundary and loading conditions for the stress analysis program IMPSMF [13] . The menu is accessed through the overlay key #0. Numbers corresponding to the menu items are keyed to the numeric pad on the keyboard , which provides an important alternative way of selecting an option.

7.7 COMPUTATION OF AREA PROPERTIES OF SURFACES

The automatic computation of the geometrical and inertial properties for geometrically complex surfaces has become an important function (module) in integrated CAD systems. The importance of the module in a CAD system arises from the fact that these properties such as surface area , moments of inertia and principal axis direction are important in the calculation of certain stresses in a variety of engineering structures.

A number of schemes for the calculation of these properties have been devised for 2-D general shapes . Wilson and Farrior [58] have developed a scheme to achieve this when the boundary is described in terms of straight lines and circular arcs. Miles and Tough [59] described a method for approximating the boundary by a set of piecewise continous cubics with prescribed endpoint gradients and then using Green's theorem to evaluate the relevant integrals. However , none of these schemes is integrated in a CAD system.

Lee and Requicha [60] have given the requirement of integrating a geometrical and inertial properties module in a CAD / CAM system : that the methods for calculating these properties may be associated "naturally" with the representation schemes. The representation schemes (surface and solid) are those referred to in section 3. They discussed the association of the different solid representation schemes and the "natural" computational methods.

In another publication , Lee and Requicha [61] outlined algorithms which operate on solids represented in CSG (refer to section 3.4.1). The algorithms generate a collection of cells whose union approximates the solid , and compute the geometrical and inertial properties of the solid by adding the contributions of the individual cells.

Boyse [62] has considered the surface area of a 3-D solid as composed of sets of elements that completeley cover the surface of the solid. The area is computed through an integration method. He also considered the computation of the mass properties through numerical integration. The solid is considered broken down into each piece of long , thin bar which approximates to a parallelepiped with a square cross section. In his treatment , the mass properties for each bar is computed and then summed to obtain the mass properties for the solid.

In the present scheme , the area properties of each Be'zier mesh are computed and then summed up to obtain the overall area properties of the surface.

7.7.1 THEORETICAL BACKGROUND OF THE ALGORITHM

The basis of the scheme is the graphical method of determining moments of area in reference [63].

Consider a Be'zier surface (SEE fig. 7.11) covered with quadrilateral Be'zier meshes. Suppose it is required to determine the second moment of area of the surface patch about the centroidal axis X —— X. A convenient axis W —— W is assumed outside the patch parallel to X —— X. Advantage is taken of the fact that the Be'zier surface is already divided into Be'zier meshes , each having an area ΔA at a distance y from the axis W —— W . Since in the direction of v each Be'zier has the same thickness Δy , the areas will be proportional to their widths x.

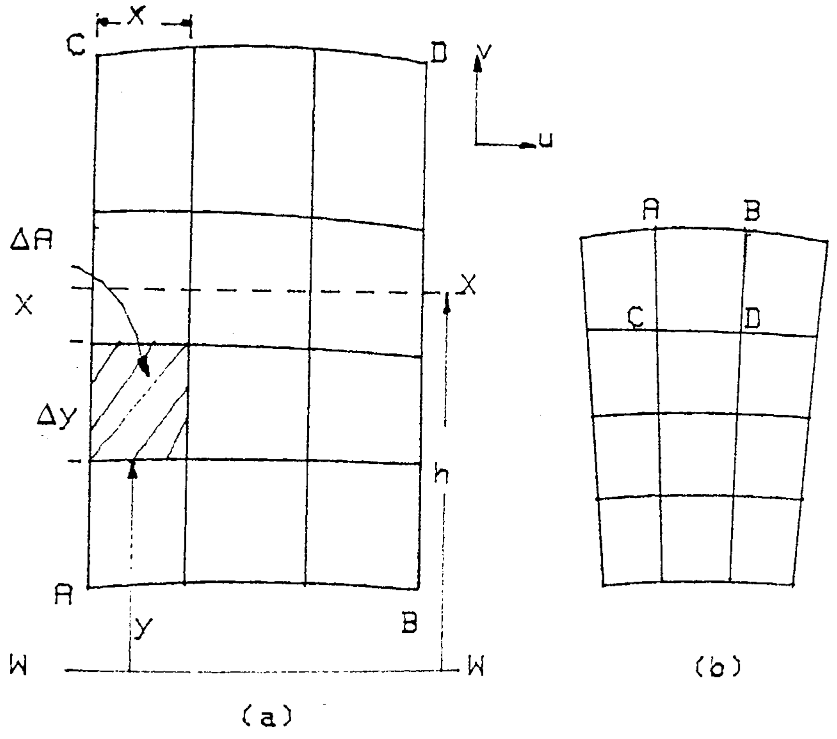


Fig. 7.11 Graphical determination of geometrical and inertial properties of a Be'zier surface.

The area A , centroidal coordinate $\bar{g}$, first and second area moments of area I_x and I_{xx} respectively about the x axis , I_y and I_{yy} respectively about the y axis , the product area moments I_{xy} are given by the following standard expressions :

$A = \sum \Delta A$

$\bar{X} = \sum x . \Delta A \;/\; A$, similarly for $\bar{Y}$ and $\bar{Z}$

$I_x = \sum x . \Delta A$, similarly for I_y and I_z

$I_{xy} = \sum x^2 . \Delta A$, similarly for I_{yy} and I_{zz}

$I_{xy} = \sum xy \Delta A$, similarly for I_{xy} and I_{yz}

$I_{AA} = I_x + A . \bar{X}^2$ (by parallel axes theorem)

When the opposite sides of a Be'zier patch are parallel (such as AC and BD in Fig. 7.11) then the Be'zier meshes are considered as rectangles and it is an easy matter to evaluate the properties. However , consider the case of a sector of the typical backsheet of a fan impeller in Fig. 7.11b. The Be'zier mesh ABCD may be considered as having AB and CB as being parallel and so the formulae for trapezoid [64] applies. Approximating the arc along AB or CD as straight lines does not lead to a marked loss in accuacy when the Be'zier meshes are many. However , for few Be'zier meshes on a sector , there is a marked loss of accuracy.

7.7.2 THE PROGRAM STRUCTURE

GEOM-DATA is written as a subprogram which can be accessed by the geometric modeller of section 5. The way the program operates is by considering each column (in v-direction) of the Be'zier meshes and then from row to row (in the u-direction). A summary of how the program is organised is as follows :

1. The corner coordinates of each Be'zier mesh are obtained.

2. Each Be'zier mesh is considered as a trapezium and the length of the parallel sides and the distance between them is determined.

3. For each Be'zier mesh , the area and the distance between the grid's centroid and the chosen axis W —— W are determined.

4. Using these data for all Be'zier meshes , the centroid of the whole structure is determined.

5. By repeating steps 1 —— 2 and using appropriate expressions , the second moments of area about the centroid of each Be'zier mesh are obtained. For each Be'zier mesh , the relative distance between the centroid of the whole structure and that between the centroid of each Be'zier mesh is obtained. Consequently , the second moments of area , principal second moments of area about X , Y and Z and the principal axis direction are determined.

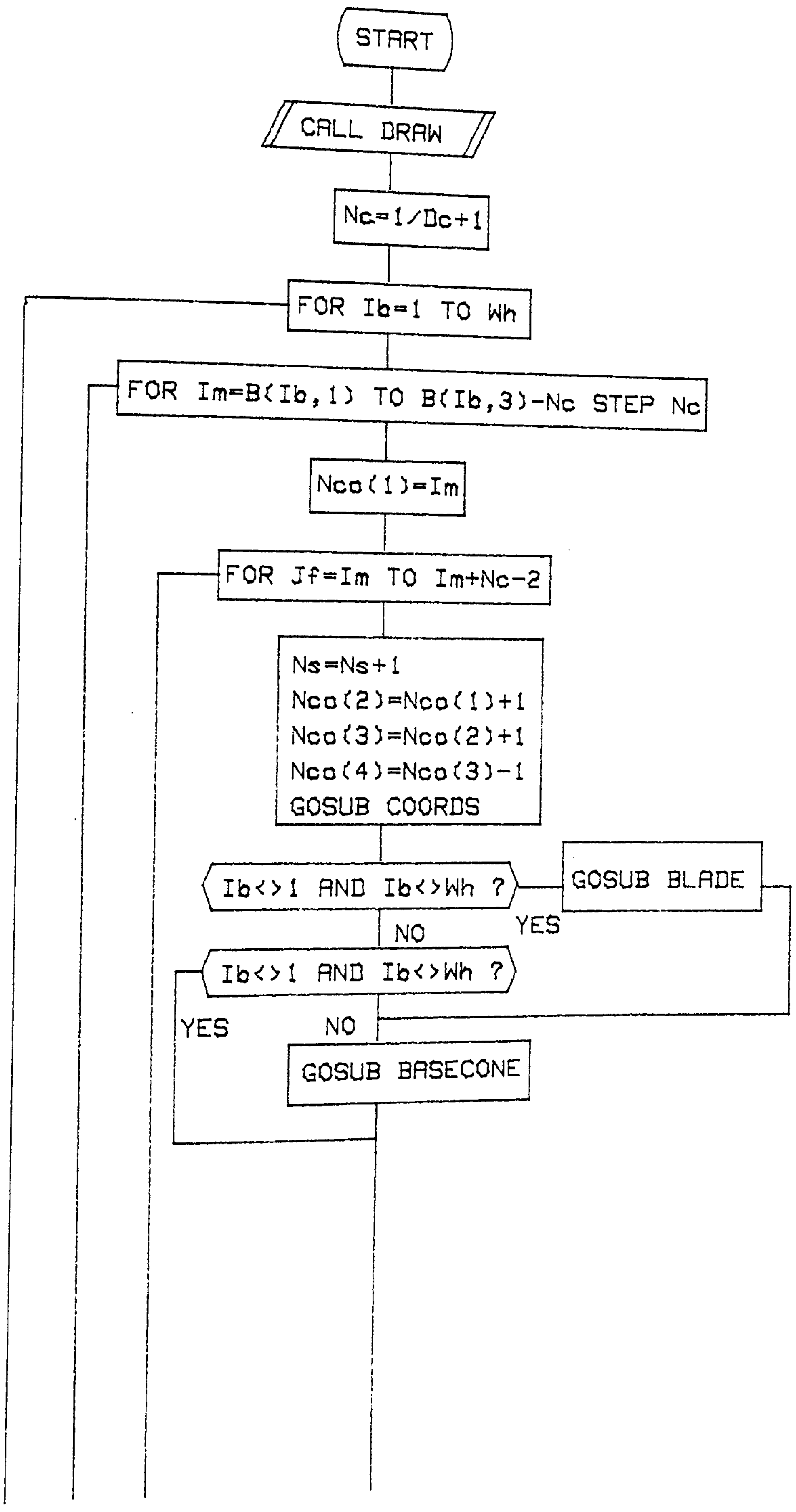
The flowchart shows that the program has three main loops I_B , I_M and J_F. The outer loop control selects which sub-structure to consider. The other two loops control the selection of each column (in v-direction) of the Be'zier meshes and then from row to row (in the u-direction).

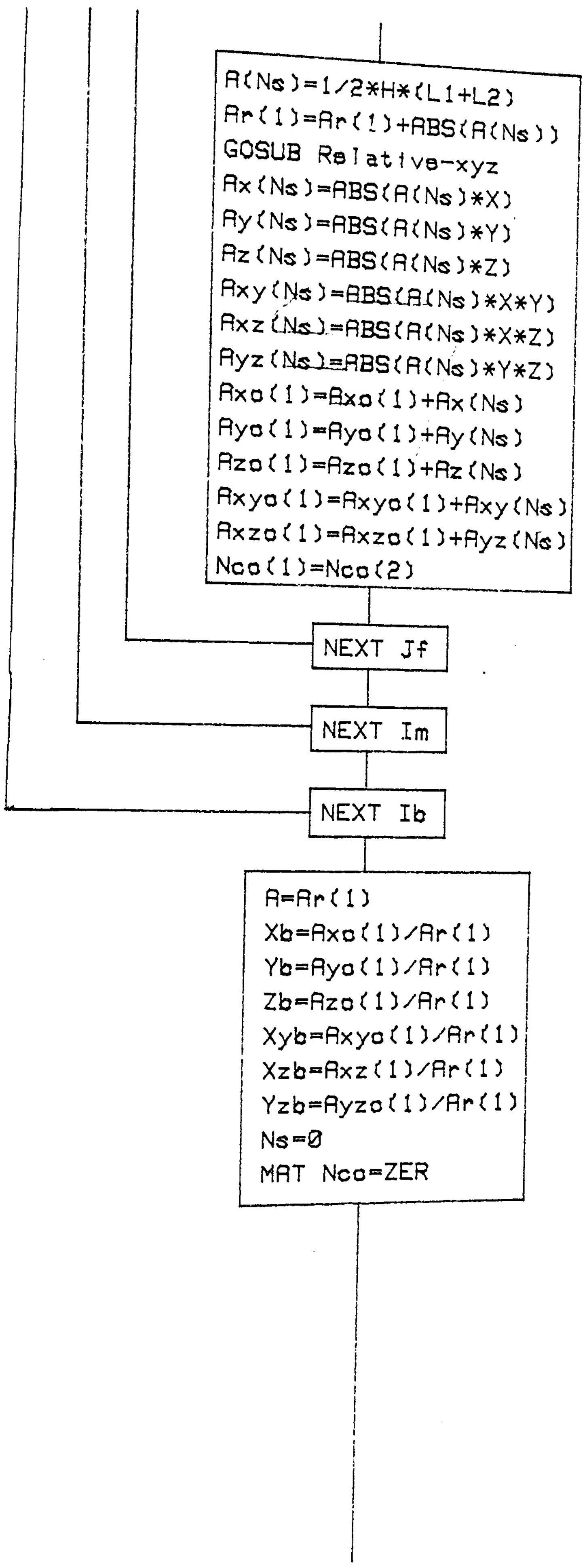
The four corner coordinates of each Be'zier mesh are obtained and the subroutine BLADE is accessed when the blade is considered , while the subroutine BASECONE is accessed when either the backsheet or the conesheet is considered. These subroutines can be accessed for any geometric shape. Their functions are identical and they evaluate the lengths of the parallel sides of each Be'zier mesh (considered as trapezoid) and the distance between them. Using these data , the area of each grid is calculated. The subroutine RELATIVE--XYZ is accessed to determine the distance of the centroid of each area from the chosen axis W --- W. The product of this distance and the area is taken and summed up for all the grids and the centroidal position for the whole structure is evaluated using the appropriate expressions in section 7.7.1.

Since the evaluation of the area properties of the whole structure is only possible when the centroidal position has been found (applies when graphical method is adopted) , loops similar to the ones just mentioned are constructed to consider each grid once more. This time the second moment of area of each Be'zier mesh about its own centroidal axis is evaluated. Moreover , the distance H_X

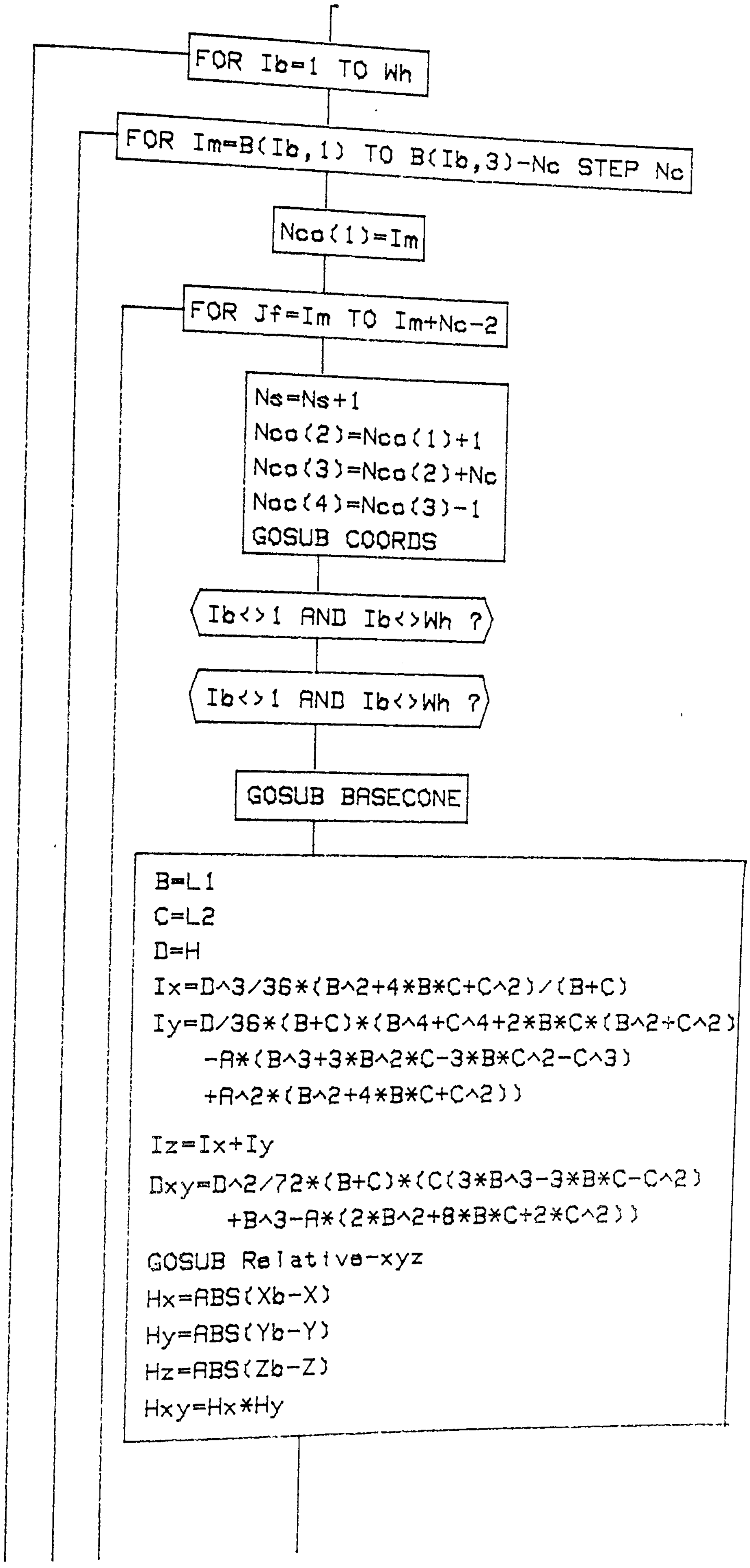
between each centroid axis and the chosen axis W — W is obtained for each grid and the product of its squared value with that of the corresponding area is taken. This value is added to the second moment of area about the centroid for each Be'zier mesh (parallel axes theorem). The principal second moments of area as well as the principal axes direction are then evaluated.

Fig 7.12 Flowchart for subprogram GEOM-DATA

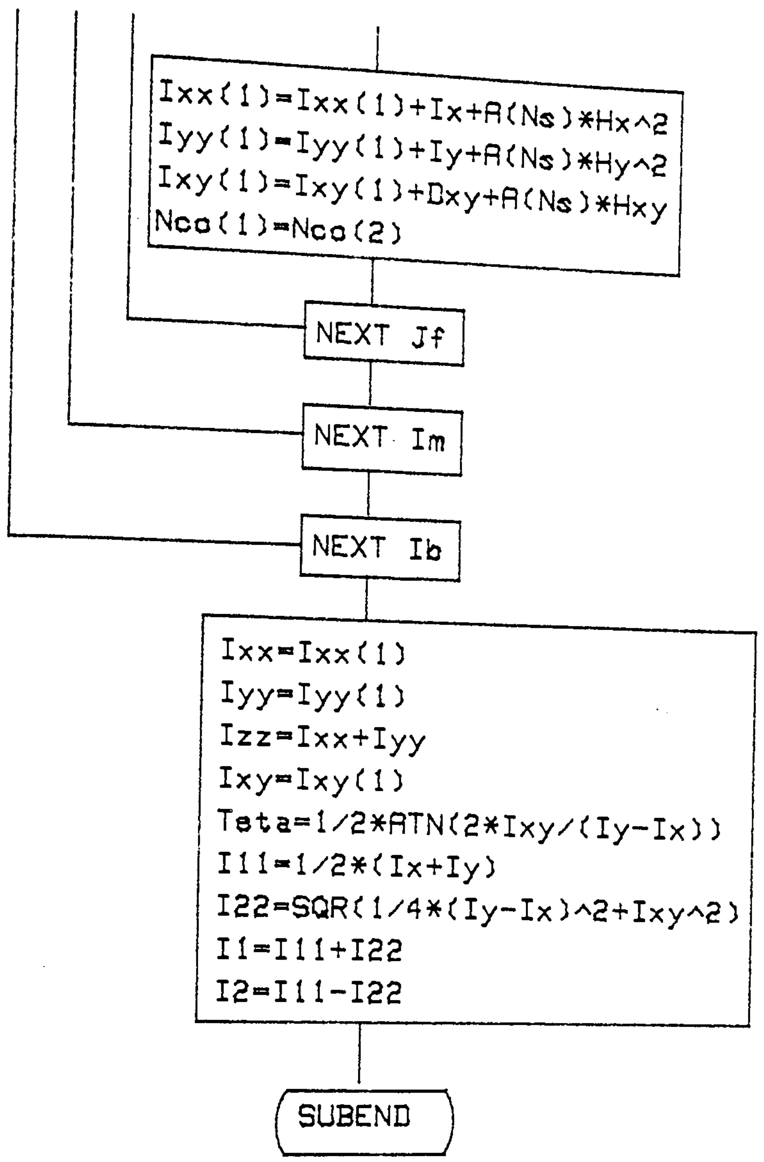




```
A(Ns)=1/2*H*(L1+L2)
Ar(1)=Ar(1)+ABS(A(Ns))
GOSUB Relative-xyz
Ax(Ns)=ABS(A(Ns)*X)
Ay(Ns)=ABS(A(Ns)*Y)
Az(Ns)=ABS(A(Ns)*Z)
Axy(Ns)=ABS(A(Ns)*X*Y)
Axz(Ns)=ABS(A(Ns)*X*Z)
Ayz(Ns)=ABS(A(Ns)*Y*Z)
Axo(1)=Axo(1)+Ax(Ns)
Ayo(1)=Ayo(1)+Ay(Ns)
Azo(1)=Azo(1)+Az(Ns)
Axyo(1)=Axyo(1)+Axy(Ns)
Axzo(1)=Axzo(1)+Ayz(Ns)
Nco(1)=Nco(2)

NEXT Jf

NEXT Im

NEXT Ib

A=Ar(1)
Xb=Axo(1)/Ar(1)
Yb=Ayo(1)/Ar(1)
Zb=Azo(1)/Ar(1)
Xyb=Axyo(1)/Ar(1)
Xzb=Axz(1)/Ar(1)
Yzb=Ayzo(1)/Ar(1)
Ns=Ø
MAT Nco=ZER
```

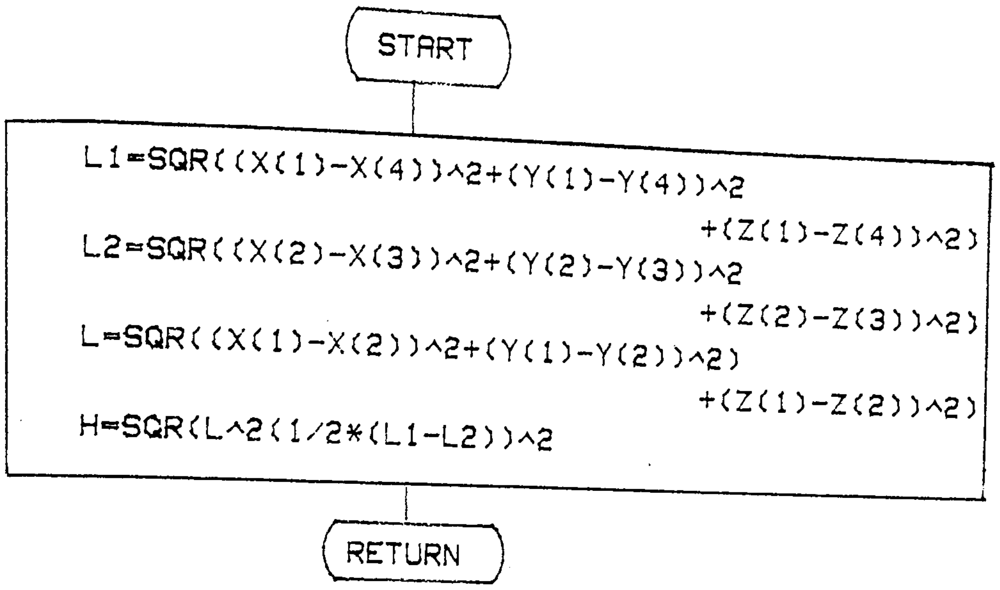


```
FOR Ib=1 TO Wh
    FOR Im=B(Ib,1) TO B(Ib,3)-Nc STEP Nc
        Nco(1)=Im
        FOR Jf=Im TO Im+Nc-2
            Ns=Ns+1
            Nco(2)=Nco(1)+1
            Nco(3)=Nco(2)+Nc
            Noc(4)=Nco(3)-1
            GOSUB COORDS
            Ib<>1 AND Ib<>Wh ?
            Ib<>1 AND Ib<>Wh ?
            GOSUB BASECONE
            B=L1
            C=L2
            D=H
            Ix=D^3/36*(B^2+4*B*C+C^2)/(B+C)
            Iy=D/36*(B+C)*(B^4+C^4+2*B*C*(B^2+C^2)
                -A*(B^3+3*B^2*C-3*B*C^2-C^3)
                +A^2*(B^2+4*B*C+C^2))
            Iz=Ix+Iy
            Dxy=D^2/72*(B+C)*(C(3*B^3-3*B*C-C^2)
                +B^3-A*(2*B^2+8*B*C+2*C^2))
            GOSUB Relative-xyz
            Hx=ABS(Xb-X)
            Hy=ABS(Yb-Y)
            Hz=ABS(Zb-Z)
            Hxy=Hx*Hy
```



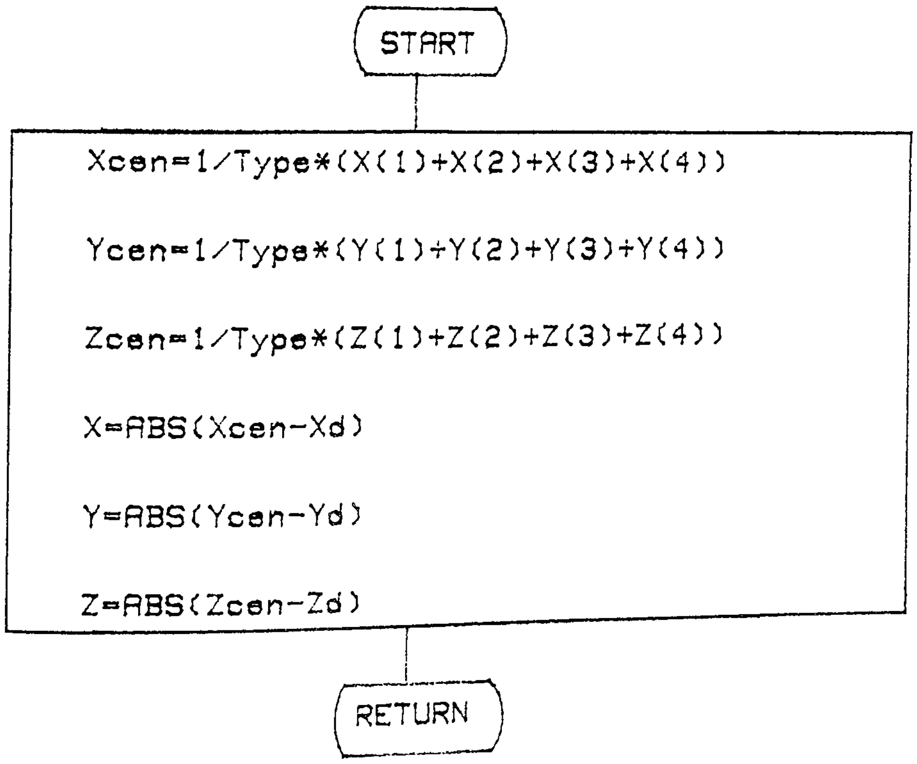
```
Ixx(1)=Ixx(1)+Ix+A(Ns)*Hx^2
Iyy(1)=Iyy(1)+Iy+A(Ns)*Hy^2
Ixy(1)=Ixy(1)+Dxy+A(Ns)*Hxy
Nco(1)=Nco(2)

NEXT Jf

NEXT Im

NEXT Ib

Ixx=Ixx(1)
Iyy=Iyy(1)
Izz=Ixx+Iyy
Ixy=Ixy(1)
Teta=1/2*ATN(2*Ixy/(Iy-Ix))
I11=1/2*(Ix+Iy)
I22=SQR(1/4*(Iy-Ix)^2+Ixy^2)
I1=I11+I22
I2=I11-I22

SUBEND
```

Fig. 7.13 Flowchart for subroutine BASECONE



```
L1=SQR((X(1)-X(4))^2+(Y(1)-Y(4))^2
                                  +(Z(1)-Z(4))^2)
L2=SQR((X(2)-X(3))^2+(Y(2)-Y(3))^2
                                  +(Z(2)-Z(3))^2)
L=SQR((X(1)-X(2))^2+(Y(1)-Y(2))^2)
                                  +(Z(1)-Z(2))^2)
H=SQR(L^2(1/2*(L1-L2))^2
```

Fig. 7.14 Flowchart for subroutine RELATIVE-XYZ



```
Xcen=1/Type*(X(1)+X(2)+X(3)+X(4))

Ycen=1/Type*(Y(1)+Y(2)+Y(3)+Y(4))

Zcen=1/Type*(Z(1)+Z(2)+Z(3)+Z(4))

X=ABS(Xcen-Xd)

Y=ABS(Ycen-Yd)

Z=ABS(Zcen-Zd)
```

7.8 CLOSING REMARKS

In this chapter , the translation of the geometric data resulting from the drafting/modeller processes into finite element input data has been discussed.

The translation module has been designed to transform and prepare input data (in the right format) for the application programs SMILOF and IMPSMF. In preparing the input data for the program IMPSMF , advantage was taken of sectorial symmetry for models such as fan impellers. Moreover , for the actual fan , the merging of parts of the sector for the backsheet was found to be useful in obtaining accurate nodal numbering and element connections.

In the next chapter , different examples are given to confirm the viability of using surface modelling as a bridge between drafting and finite element analysis of plate/shell type structures.

CHAPTER 8

ILLUSTRATIVE EXAMPLES

In order to verify that the geometric modeller described in Chapter 5 successfully links drafting and finite-element analysis , a number of examples were tested. It is not intended in this section to demonstrate the capabilities of the type of element used for stress analysis , rather emphasis is placed on the accuracy with which the geometric modeller transfers the geometric data (i.e dimensions) from the drafting package to the three-dimensional form suitable for input to the finite-element stress analysis program. All examples discussed here have been investigated by Jweeg [13] using the semiloof element and the stress analysis program SMILOF and IMPSMF ; moreover , he carried out some experimental work to confirm the finite element analysis results. In his work , input data were manually prepared and so it is worthwile , first , to compare accuracy of input geometry from the modeller discussed in chapter 5 and then the finite-element results using such input data. If there is no loss of accuracy in geometric data resulting from the drafting / modeller processes then it is imperative that the finite element result will be exactly the same as that obtained by Jweeg. Descrepancies can only arise from loss of accuracy in geometric data.

In all the examples discussed in this chapter , the following steps are essential in order to pass data from the drafting package to the finite element analysis program :

(i) The orthographic views of the model under consideration are drawn using the MEDIA drafting package.

(ii) The interpreting module , when accessed , interpretes the engineering drawing and prepares appropriate Be'zier nets which are essential for the geometric modeller discussed in chapter 5. Various shapes such as those illustrated in section (6.3.2) can be handled.

(iii) The modeller then uses the Be'zier net data as a basis for modelling the object and displaying it in three-dimensional form.

(iv) The translating module further transforms the geometric model data to obtain the nodal coordinates and element connections. It is possible to reduce the dimensions of a structure if only part of it is essential for analysis. Sectorial symmetry is exploited for repeating structures. Additional parameters are automatically calculated by the translator when sectorial symmetry exists.

(v) Finally , material properties and loading conditions are interactively added to the data of (iv) above fully to define the finite element input data. This is then dumped onto an input file for the finite element analysis program SMILOF or IMPSMF.

Several examples are presented in the rest of this chapter to demonstrate the capabilities of the programs developed in chapters 5 , 6 and 7. More examples are given in Appendix A in order to reinforce the capability of the interpreting module (see chapter 6) in passing geometric data from the drafting package , MEDIA to the surface modeller described in chapter 5. In all the examples , the middle surfaces of thin plate/shell structures are considered and hence their thicknesses are not shown in the orthographic views.

8.1. SQUARE PLATE CONVERGENCE STUDY

The convergence study of a thin square plate has been carried out and this is the simplest of all the examples presented in this chapter. Because the aim is to demonstrate the fact that the drafting / modeller / finite-element analysis interfacing works , emphasis in on the different steps involved in the interfacing processes. Therefore only clamped edges boundary condition and central concentrated load is treated.

8.1.1 Othographic views of a thin square plate

The "line" geometric definition function of the drafting package was used to draw the views as shown in Fig. 8.1.

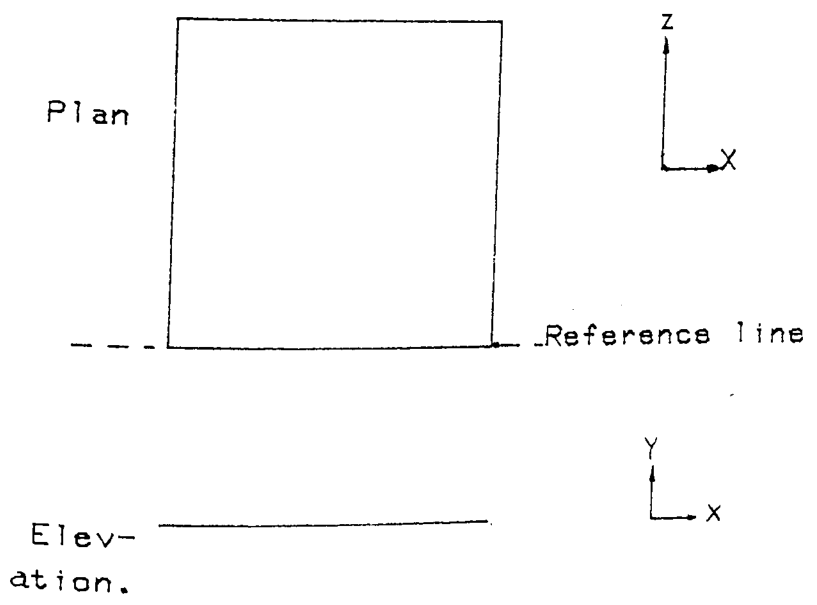


Fig. 8.1 Orthographic views of a thin square plate.

8.1.2 Be'zier net resulting from the interpreting module.

The Be'zier net for a flat plate shown in fig. 8.2 is very similar to the final geometric model. This data is the input to the geometric modeller.

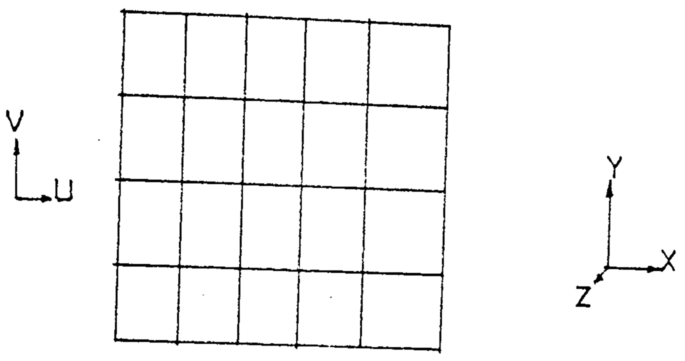


Fig. 8.2 Be'zier net for a thin square plate.

8.1.3 Geometric model resulting from the processes in the geometric modeller

Any number of meshes can be chosen for the u and v directions. Facilities are available to reduce the model size when symmetry conditions are encountered. For example , for a square plate , only one quarter of the model needed to be considered. Therefore , the model was reduced accordingly as shown in Fig. 8.4.

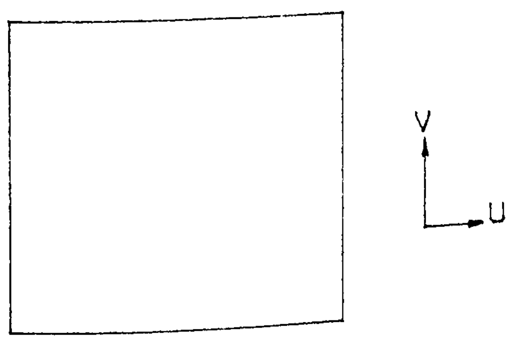


Fig. 8.3 Geometric model of a thin

8.1.4 Translating geometric model data into finite element data.

Given any number of meshes in the u and v directions , the translating module transforms the geometric model data into finite element analysis data (refer to chapter 7). For example , for the square plate , four uniform finite element meshes were considered :(1X1) , (2X2) , (3X3) and (4X4) (see fig. 8.4).

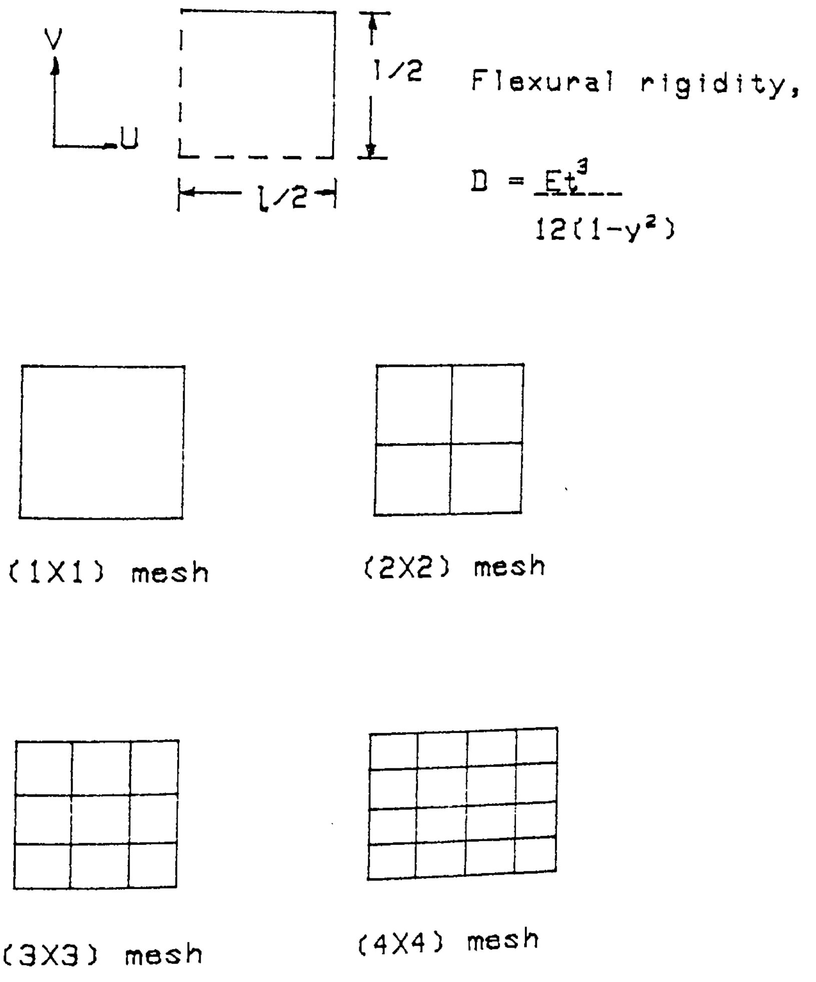


Fig. 8.4 Different meshes for convergence study.

8.1.5 Accuracy of geometric data from drafting / modeller processes

For flat plates , it is observed that there is no loss of accuracy in the dimensions when the Be'zier nets of the othographic views are constructed and submitted to the geometric modeller for display of the model. For example , a square plate having sides of length 200m was used for test and the modeller exactly reproduced the dimensions without loss of accuracy. The dimensions of the plate in the drafting module and after it has been modelled for input to finite element analysis are shown in Table 8.2.

8.1.6 Results from finite element analysis

The interfacing processes are complete when the data in section 8.1.4 are dumped onto a finite element analysis program. In order to study the rate of convergence , the deflection results were plotted and also compared with the results obtained from Jweeg's work. The results were also compared with the exact solution given by Timoshenko , Ref.[66] as shown in Fig. 8.2. The numeric results presented in Table 8.1 shows a rapid rate of convergence to the exact solution given by Timoshenko for displacements. Moreover , it clearly shows that the new approach of interfacing drafting and finite element analysis via Be'zier surface technique is not only feasible but also approaches the exact solution. Fig. 8.5 shows a comparison with the result of Jweeg ; since the modeller transfers the exact dimensions to the finite element program , the result agrees perfectly with that obtained by Jweeg. Moreover , it serves to confirm the fact that the semiloof element used in his work approaches the exact solution with a minimum number of degrees of freedom.

Mesh in symmetric quarter		(1X)	(2X2)	(3X3)	(4X4)	Exact
Concentrated load (P)	Central deflection $10^6 X \frac{D}{PL^2}$	7018	6071	5888	5800	5600

Table(8.1) Clamped square plate under concentratd load (P).

Process	Length l (mm)	Width l (mm)	Loss of accuracy (%) in modeller
Drafting	200	200	0
Modeller & F.E.	200	200	0

Table 8.2 Geometric accuracy of geometric modeller for square plate

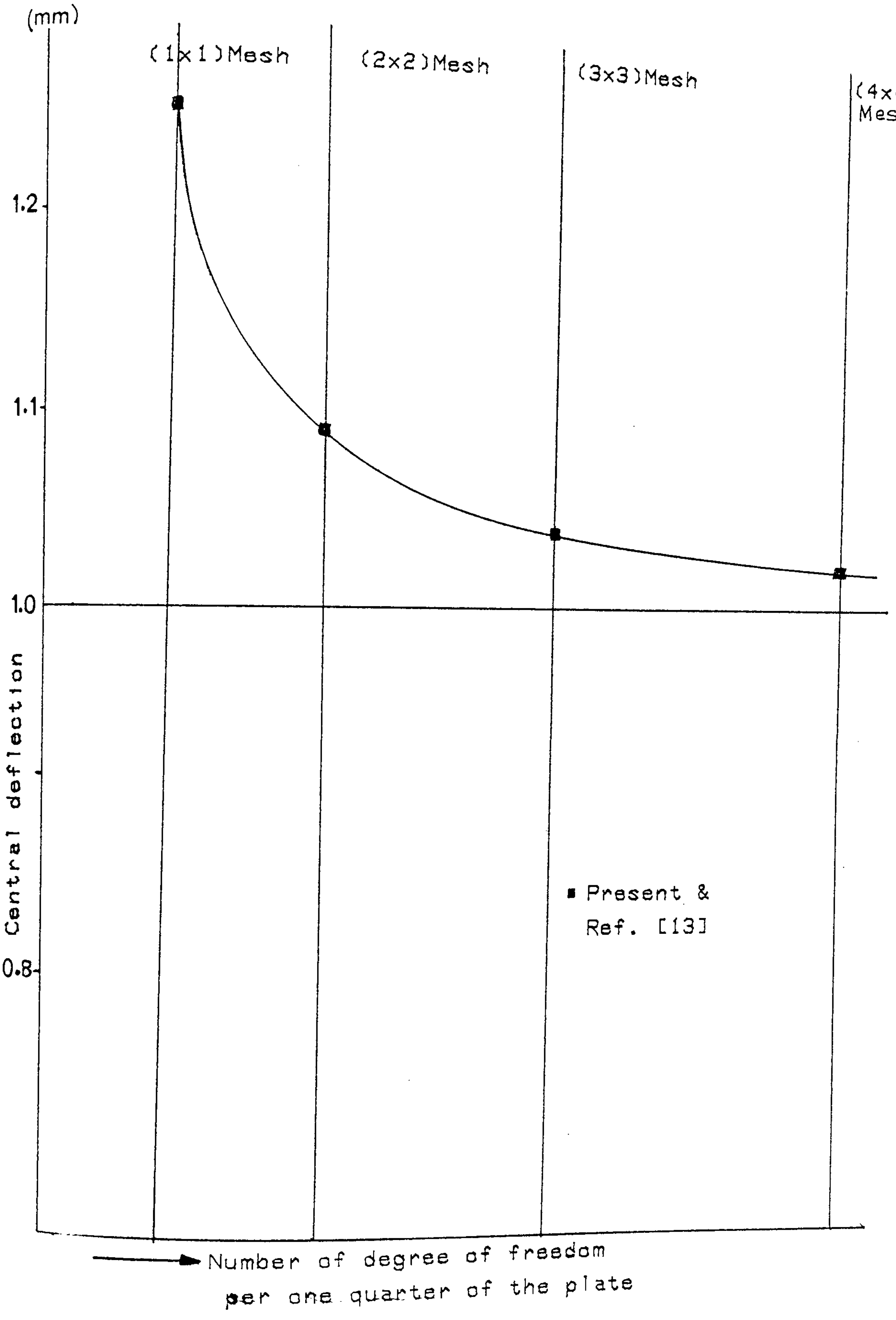


Fig. 8.5 Convergence of deflections for a square plate under concentrated load at centre

8.2 CYLINDRICAL SHELL ROOF

Cylindrical shell roof loaded by its own weight was tested to show that the developed programs can handle not only engineering structures composed of flat plates having straight edges but also shells having curved surfaces. For the cylindrical shell roof tested , the edges are straight and supported by diaphragms which are assumed to be infnitely rigid in their plane and infinitely flexible out of it. This is one of the most typical examples used as a performance test for shell finite element programs

8.2.1 Orthographic views of cylindrical shell roof

The "three points on circumference" geometric definition function of the drafting package was found to be best for drawing one of the orthographic views of the cylindrical shell roof shown in Fig. 8.6.

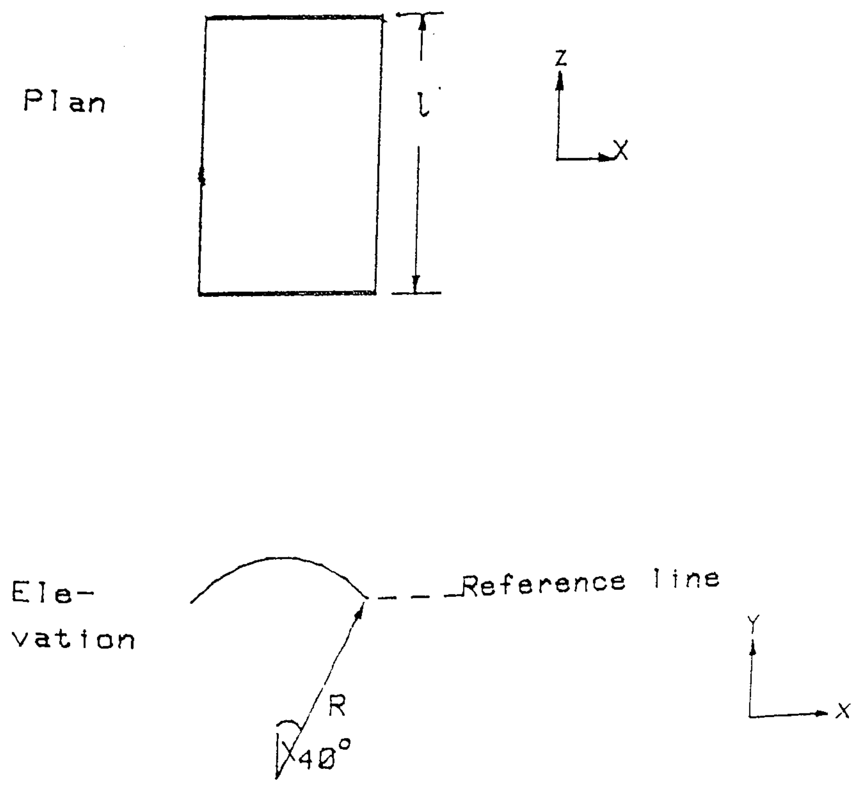


Fig. 8.6 Orthographic views of

8.2.2 Be'zier net resulting from interpreting module

The length of the roof in the orthographic view aligns with the z-direction when interpretation of views is complete as shown in Fig. 8.7.

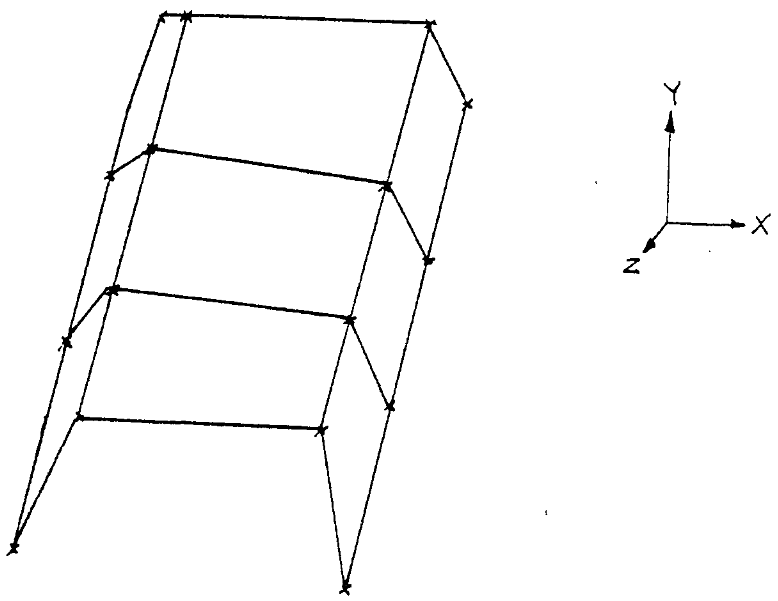
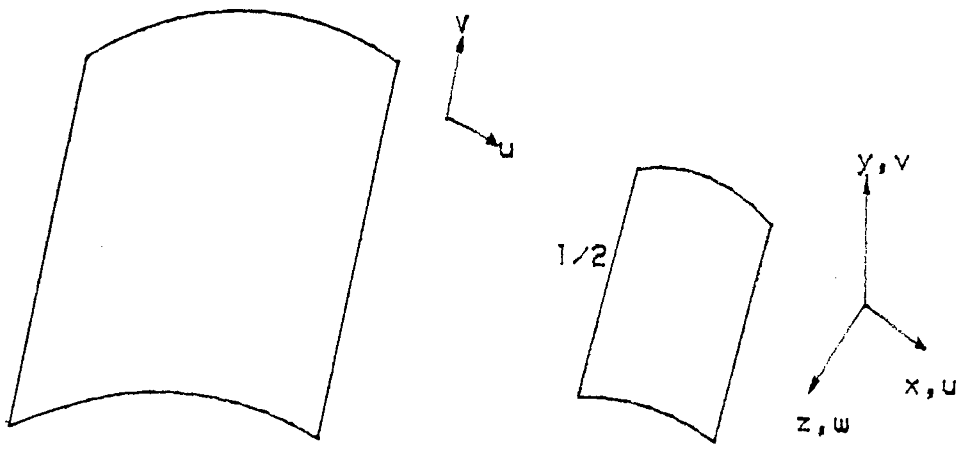


Fig. 8.7 Be'zier net for a cylindrical shell roof

8.2.3 Geometric model resulting from the geometric modeller

The geometric model of the cylindrical roof is shown in Fig. 8.8 and once more only a quarter of the model is considered because of symmetry.



(a) Model (b) A quarter of model considered due to symmetry

Fig. 8.8 Geometric model of a cylindrical shell roof.

8.2.4 Translating the geometric model data into finite element data

Four types (shown in fig. 8.9) of mesh were considered for the geometric model of fig. 8.8 by specifying the number of meshes along the u and v directions of the model when the geomtric modeller was accessed. The translation module prepares input data for the finite element analysis program. The cylindrical roof is not considered as a periodic structure , therefore the translation module , in preparing the finite element analysis input , omits data such as the array containing the numbers of nodes having similar behaviour , node number and angle of skewed nodes (refer to [13]).

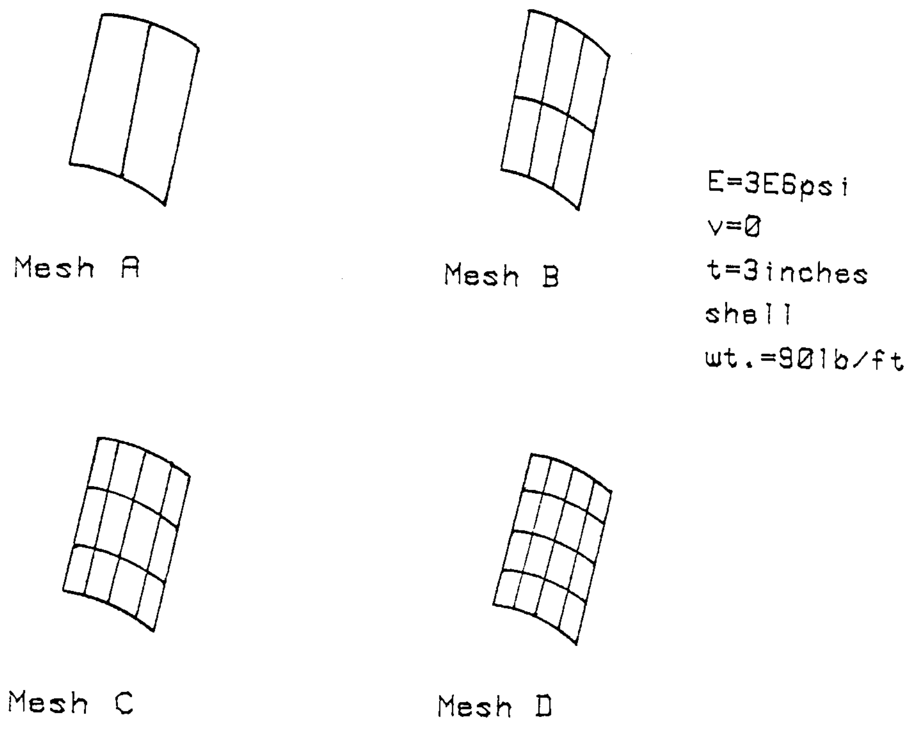


Fig. 8.9 Cylindrical shell roof under its own weight.

8.2.5 Accuracy of geometric data from drafing / modeller processes

The accuracy with which the geometric modeller translates the geometric data form the drafing package to the finite element analysis program for the cylindrical roof was examined by considering the height of the nodal points along the circumference. These heights correspond to the z-values of the nodal coordinates and they are compared with the actual values anticipated at such nodal points. As an example , the z-coordinate values for a (4X4) mesh are compared with expected values in Table 8.3. The modelled surface and the actual surface are shown in Fig. 8.1Ø. and it is observed that the geometric modeller discussed in chapter 5 mimics the actual surface almost exactly. Therefore , it is anticipated that the finite element result for the cylindrical roof will more or less be the same as those in reference [13] since the same elements and program were used.

Nodal points	Drafting (Actual)	Modeller & F.E.	Loss of accuracy(%) in modeller
A	25	25	Ø
B	24.62	24.28	1.4
C	23.5	22.87	2.7
D	21.65	21.Ø7	2.6
E	19.15	19.15	Ø

Table 8.3 Geometric accuracy of geometric modeller for cylindrical roof. (Refer to Fig. 8.10)

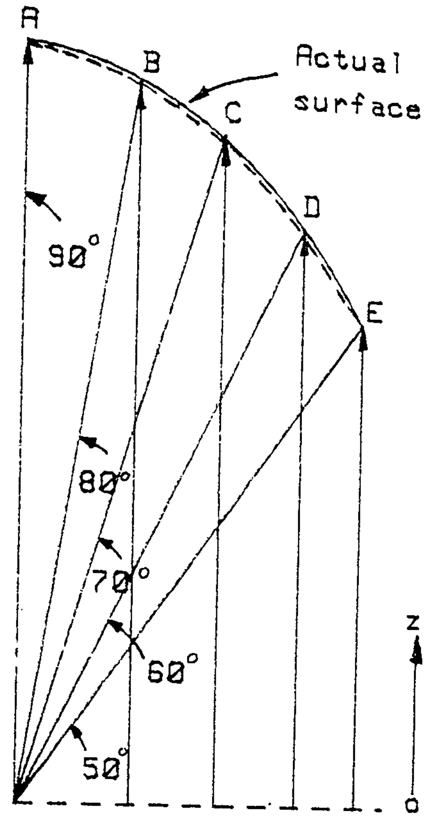


Fig. 8.10 Geometric accuracy of geometric modeller

8.2.6 Results from finite element analysis

The comparison between the results of the finite element approach using the Be'zier bicubic surface for idealization of finite elements for meshes A,B and C (see fig. 8.9) and of the analysis of Jweeg [13] as well as the analysis of Scordelis and Lo [67] , is given in Figs. 8.11- 8.13. The displacements membrane forces and bending moments are considered in these figures. It is observed that there exists good agreement between these sets of result. This emphasises the fact that the use of a meshed-surface modeller to prepare for the finite element analysis of shell structures from a drafting geometric data yields accurate results. This point is obvious from Table 8.3. Since the modeller transfers and transforms 2-D drafting geometric data accurately to the analysis module. Concerning the Semiloof element used in the analysis program, it is observed that a high rate of convergence to the exact solution exists. Acccurate results are observed even with the coarsest mesh such as mesh A.

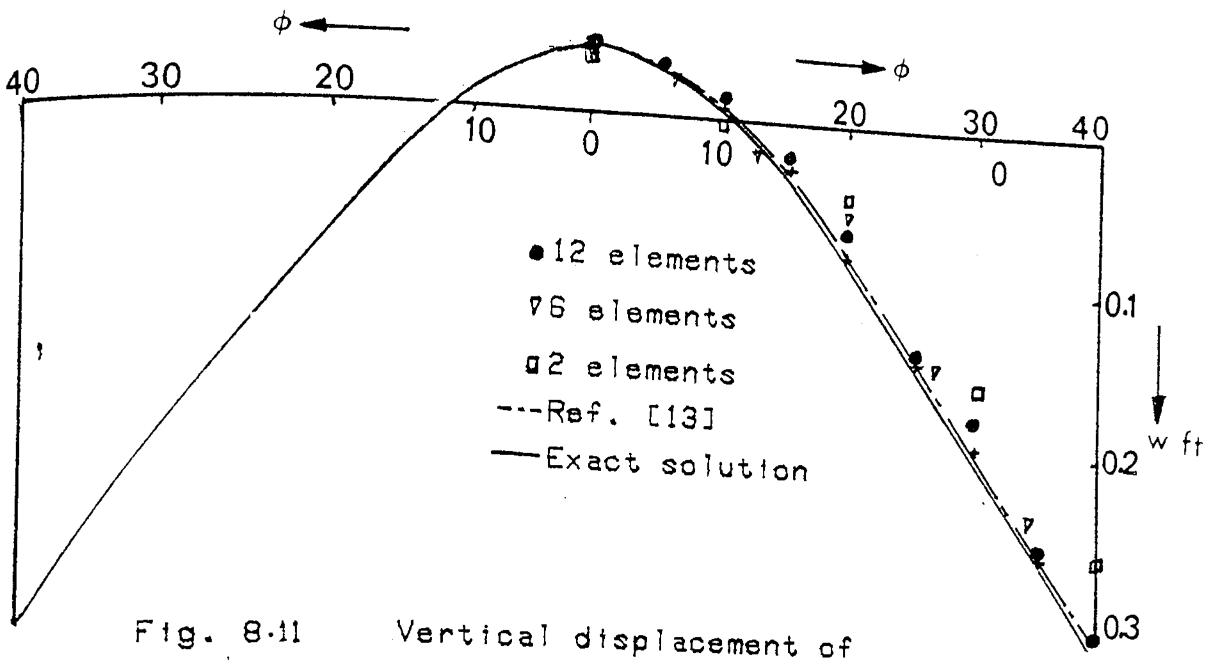


Fig. 8.11 Vertical displacement of central section

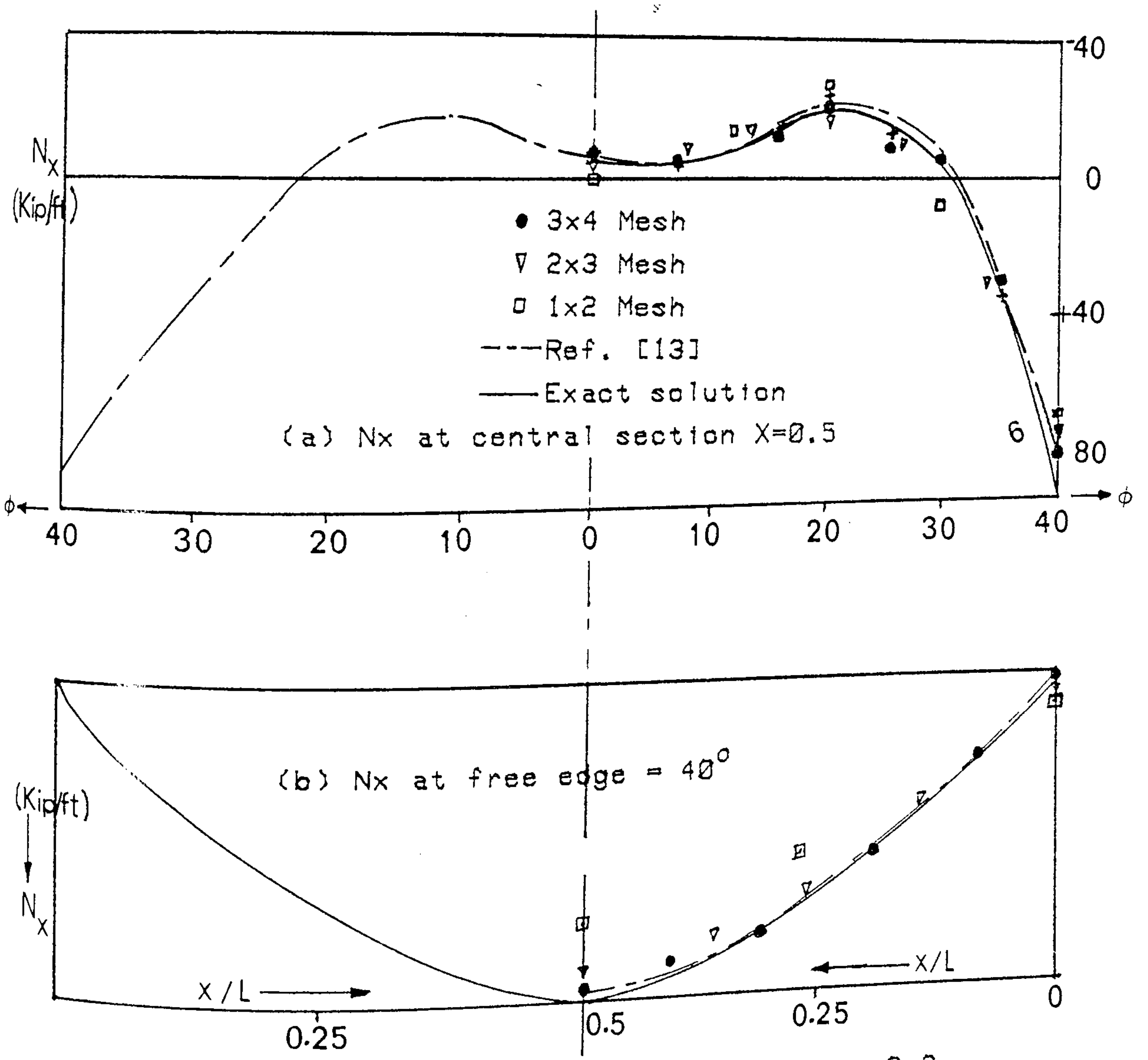
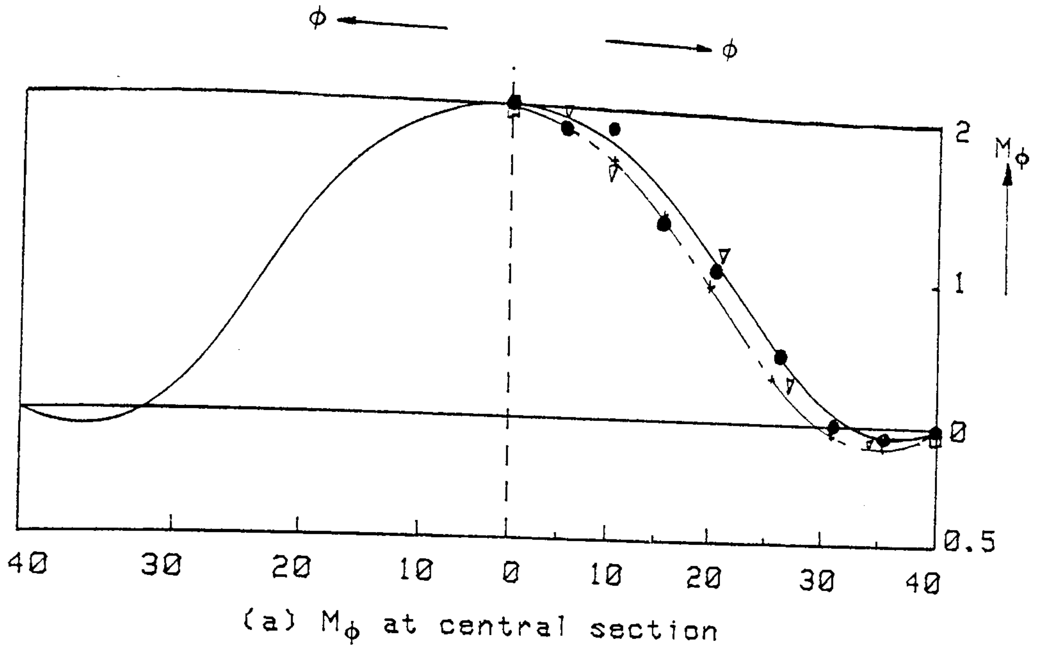
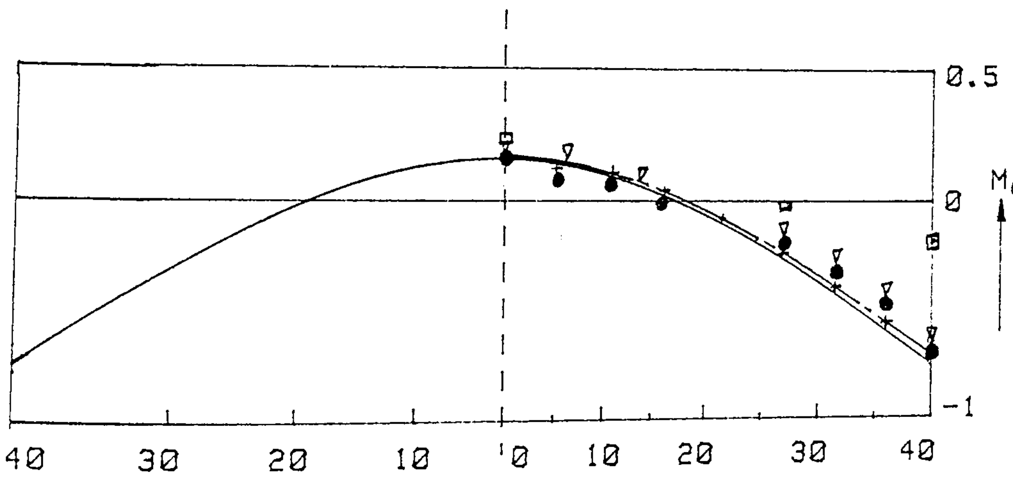


Fig. 8.12 A cylindrical shell roof of Fig. 8.8 finite element and exact solution



(a) M_ϕ at central section



(b) M_l at central section

● (3x4) Mesh
∇ (2x3) Mesh
⊡ (1x2) Mesh
—+—·-Ref.[13]
——— Exact solution

Fig. 8.13 A cylindrical shell roof of Fig. 8.8 finite element and exact solutions

8.3. SIMPLIFIED FAN IMPELLER .

In order to confirm the technique of exploiting the sectorial symmetry described in Chapter 6 , and the fact that the modeller can handle intersecting structures, it was decided to study the flow of data of a simplified radial fan impeller from drafting, via modeller to the finite element analysis program. Attention was given to the accuracy with which the geometric modeller passes on dimensions of the impeller on to the analysis module. Finite element results were studied in detail and compared with results obtained by Jweeg.

8.3.1 Orthographic view of Simplified Fan Impeller .

The orthographic views for the simplified fan impeller drawn in the drafting module is shown in Fig. 8.14. These views are interpreted by the interpreting module to prepare the Be'zier nets discussed in Section 8.3.2.

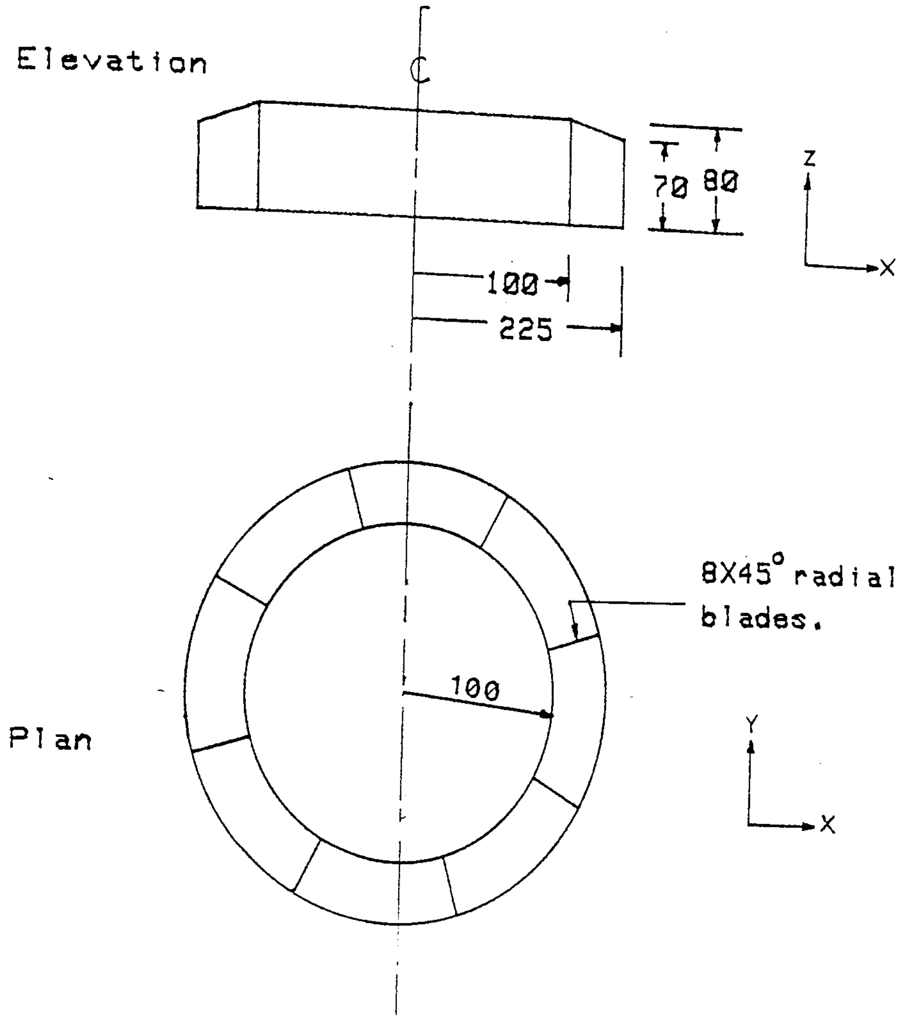


Fig. 8.14 Orthographic views of simplified fan impeller

8.3.2 Be'zier net of Simplified Fan Impeller .

Sectorial symmetry is exploited in rendering the Be'zier net, therefore only a sector is considered once the interpretation of the orthographic views is completed. This technique is extremely useful both in surface rendering by the geometric modeller and in finite element stress analysis. In the case of analysis, this technique conserves computer memory while in the case of surface rendering by the geometric modeller, the sector can be replicated several times to define the complete model.

It was observed that if any of the blades aligns with the X-axis, the finite analysis program IMPSMF which was used for the analysis, returns error message.

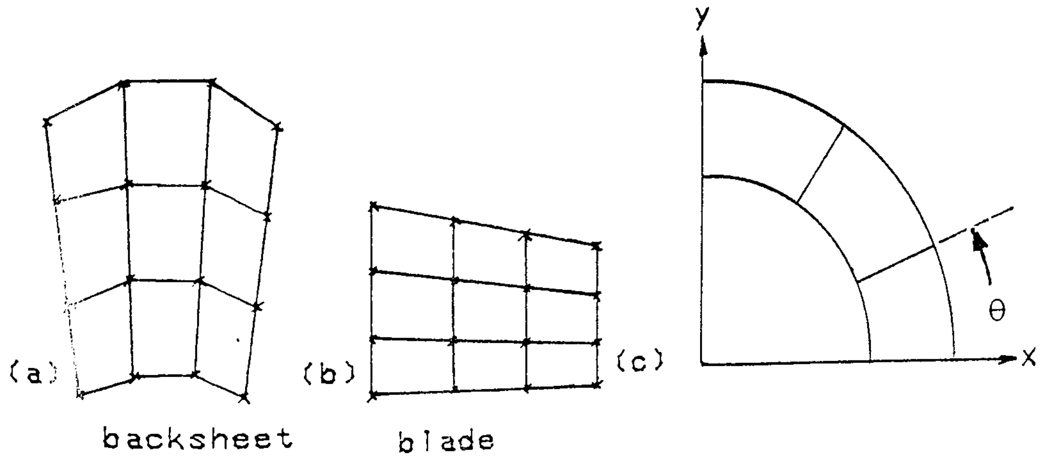


Fig. 8.15 Be'zier nets for simplified fan impeller

This was because the program does not accept blades which align with the X-axis. For this reason, the user is requested to input an initial angle (see Fig. 8.15c) with which the blade nearest to the

X-axis should be positioned. All other blades are then shifted by the same angular displacement.

8.3.3 Geometric model resulting from the geometric modeller .

The geometric modeller utilizes the Be'zier nets (see fig. 8.15) of the interpreted orthographic views (see fig. 8.14) for the construction and display of the geometric model shown in Fig. 8.16.

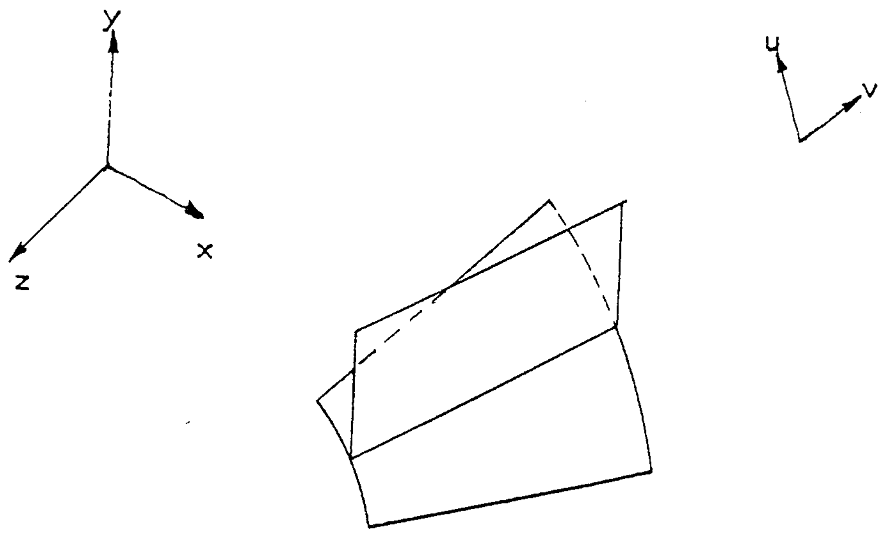


Fig. 8.16 Geometric model of simplified fan impeller

8.3.4 Translating geometric model data into finite element data.

Unlike the models already considered in the earlier sections, the translation of the geometric model data into finite element data is quite complicated because of the way that the analysis program IMPSMF accepts input data for fan impellers. The nodal coordinates and element connections of one half of the impeller sector is first dealt with, then the blade and finally the second half of the impeller sector.

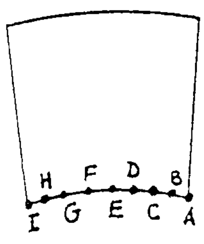
Specifications of the boundary conditions are assigned according to the instructions given in Ref. [13]. In order to be able to compare results with those of Jweeg, quadrilateral elements were chosen.

The translating module automatically establishes the numbers of nodes having similar behaviour, node number and angle of skewed nodes. However, it omits the semi-cone angle since this is not required for the simplified impeller.

8.3.5 Accuracy of geometric data from drafting/modeller processes .

In the case of the simplified fan impeller, it is only necessary to examine the accuracy with which the dimensions of the backsheet are transfered from the drafting, via modeller to the analysis module since we have observed that no loss accuracy occurs for flat plates (which represents the blade) .

A typical result of (4 X 2) elements on the backsheet along the circumference of the inner radius is presented here so as to see clearly the accuracy of the geometric modeller. It is observed that the modeller models the geometry accurately with an error of only 1%.



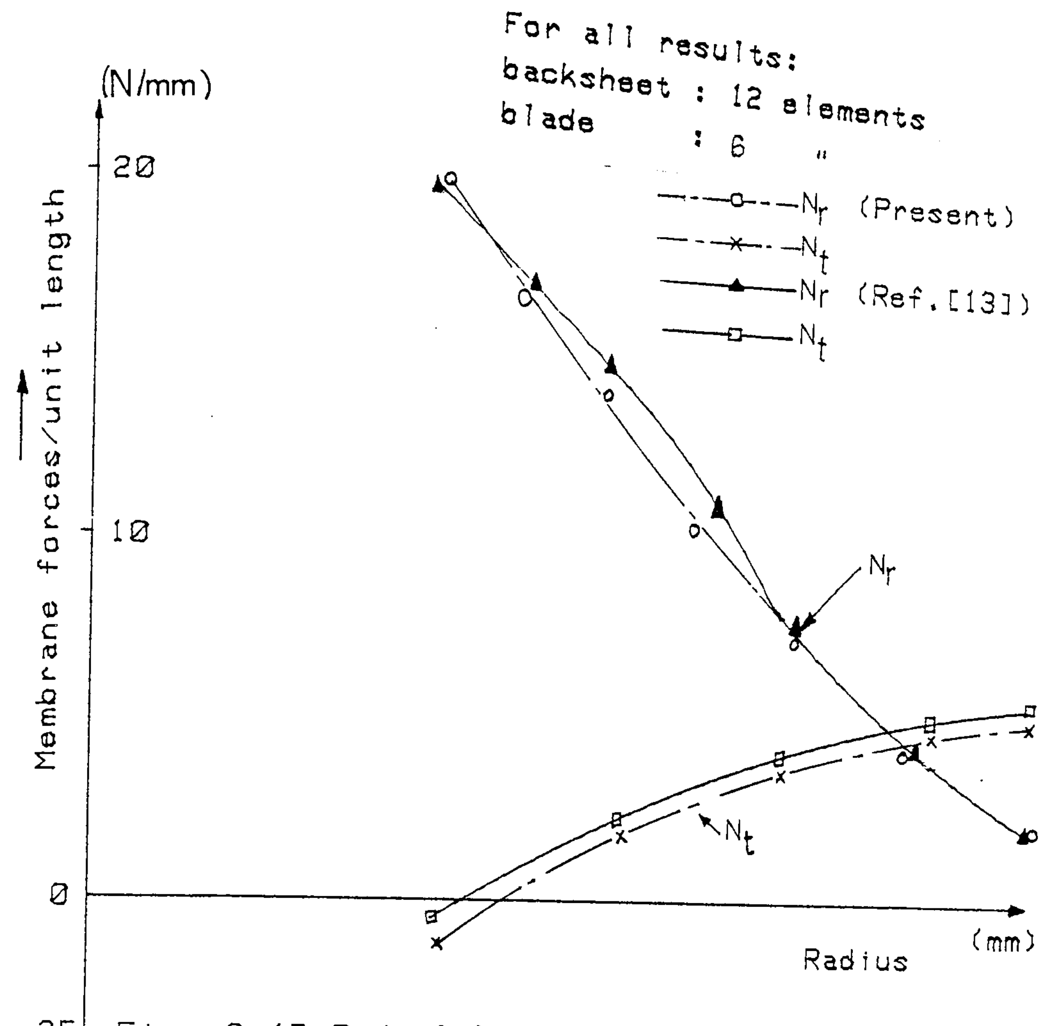
Nodal point	Drafting (Actual) Radius(mm)	Modeller & F.E. Radius(mm)	Modeller Error(%)
A	100	100	0
B	100	99	1
C	100	99	1
D	100	99	1
E	100	100	0
F	100	99	1
G	100	99	1
H	100	99	1
I	100	100	0

Table 8.4 Study of accuracy of geometric modeller for simplified fan impeller

8.3.6 Finite element results for Simplified Fan Impeller.

The results of the membrane force and bending moment per unit length obtained by Jweeg [13] was used as a reference solution in order to compare the results obtained from the drafting / modeller / finite element analysis processes. The results drawn on Figs. 8.17 to 8.2Ø have shown good agreement for the two selected sections on the backsheet i.e at $\theta=22.5^{o}$ and 37.5^{o} .

It is observed that the membrane force and bending moment per unit length drawn on Figs. 8.17 to 8.2Ø, decreases as the radius increases and approaches zero at the free end of the impeller. Moreover, as expected, the maximum bending moment occurs at the inner edge near the intesection of the backsheet with the blade. The high value in this region is due to the action of the blade.



25 Fig. 8.17 Radial & tangential membrane stress resultants for the line =22.5°: (backsheet) simplified fan impeller

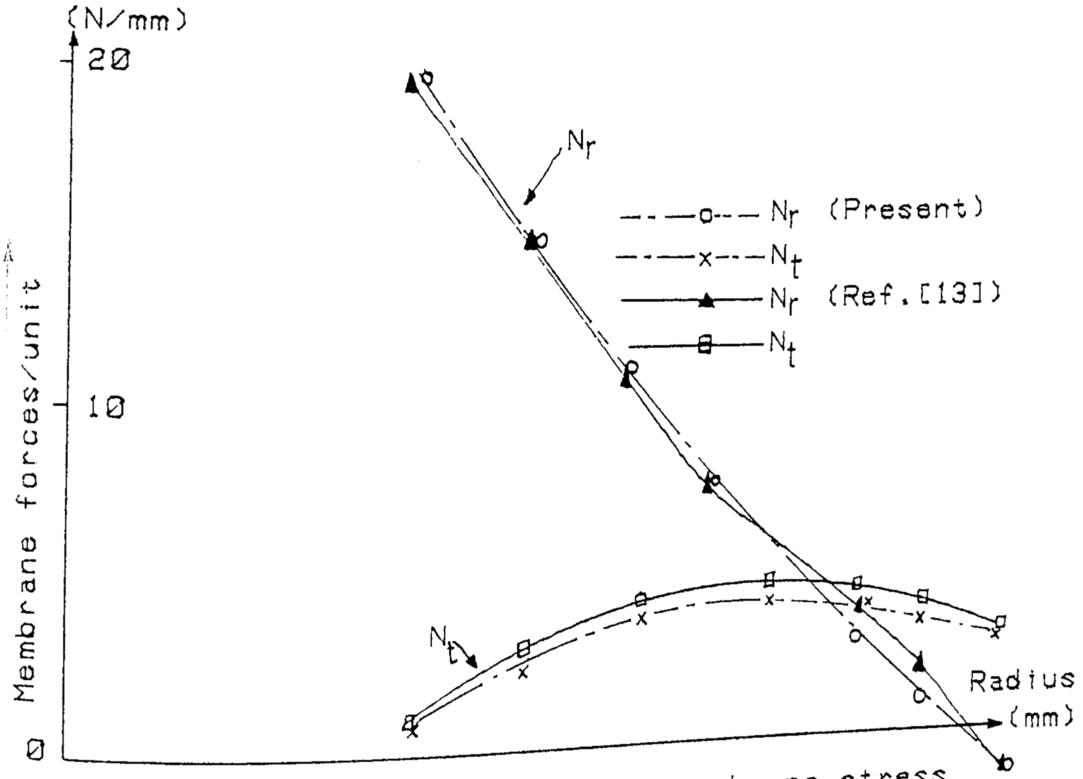


Fig. 8.18 Radial & tangential membrane stress resultants for the line =37.5°: (backsheet) simplified fan impeller

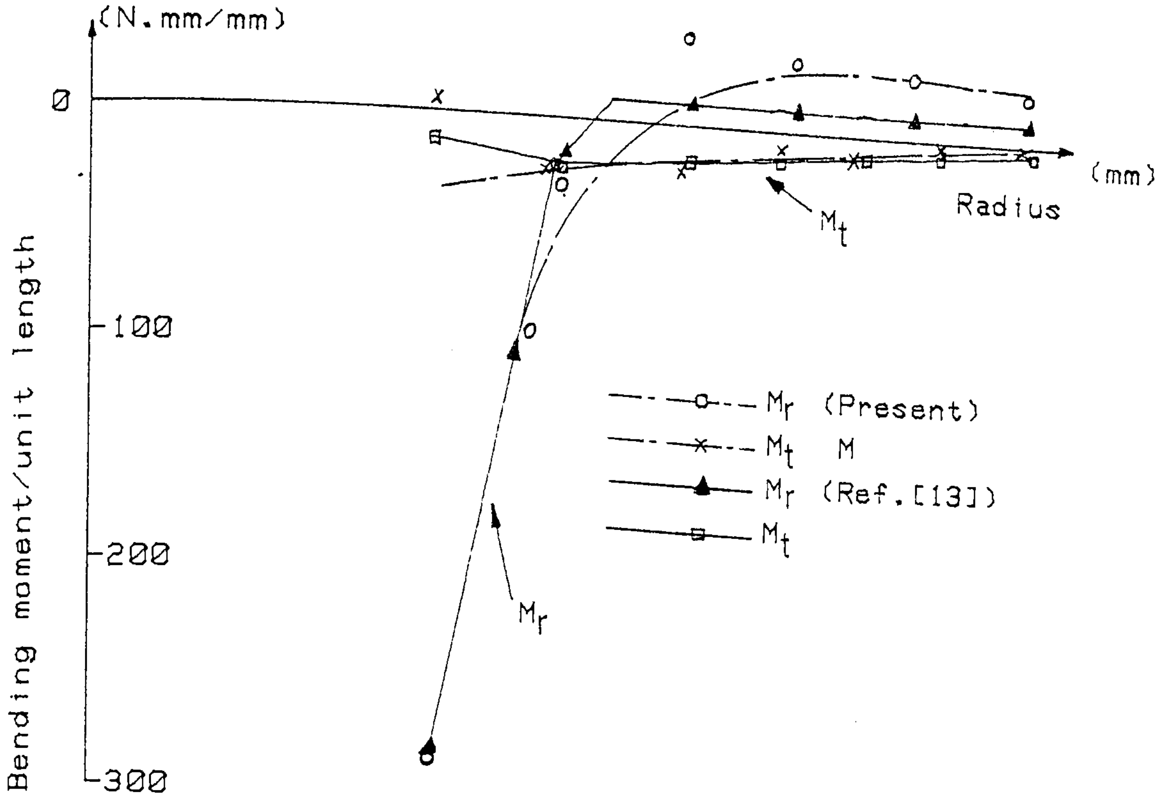


Fig. 8.19 Radial & tangential bending moments along the line =22.5°:(backsheet) simplified fan impeller.

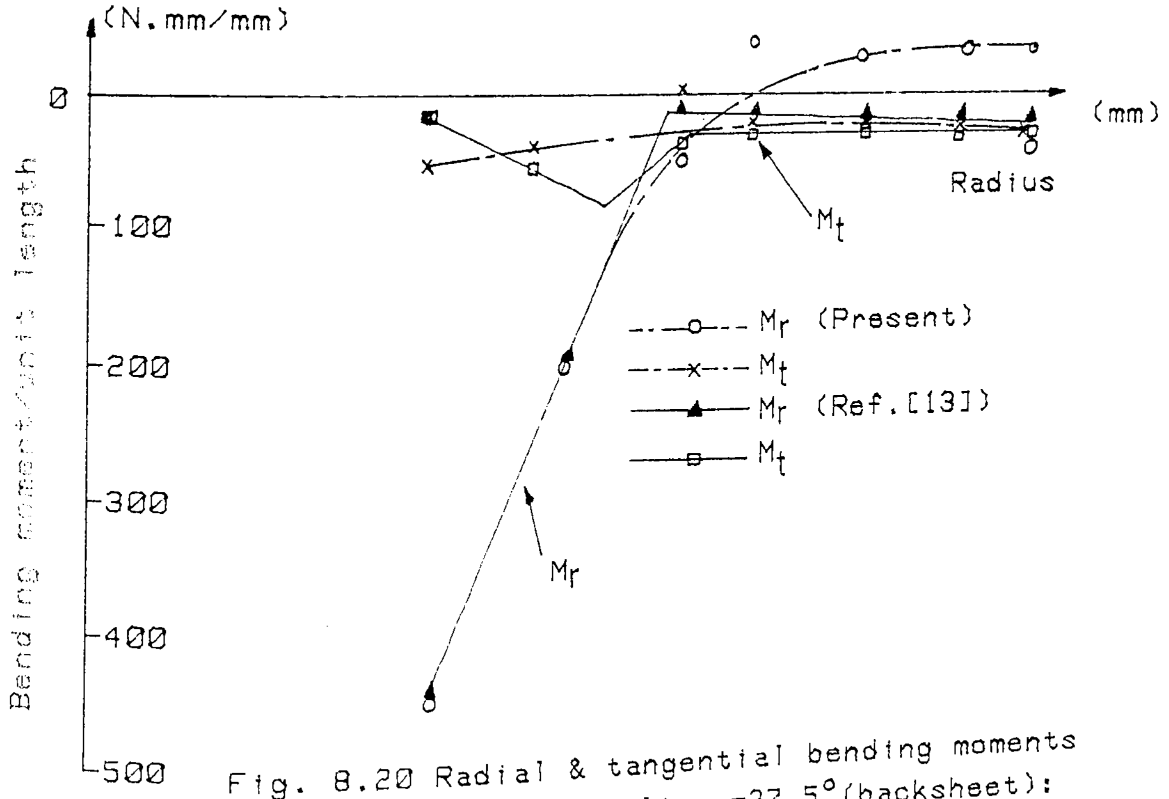


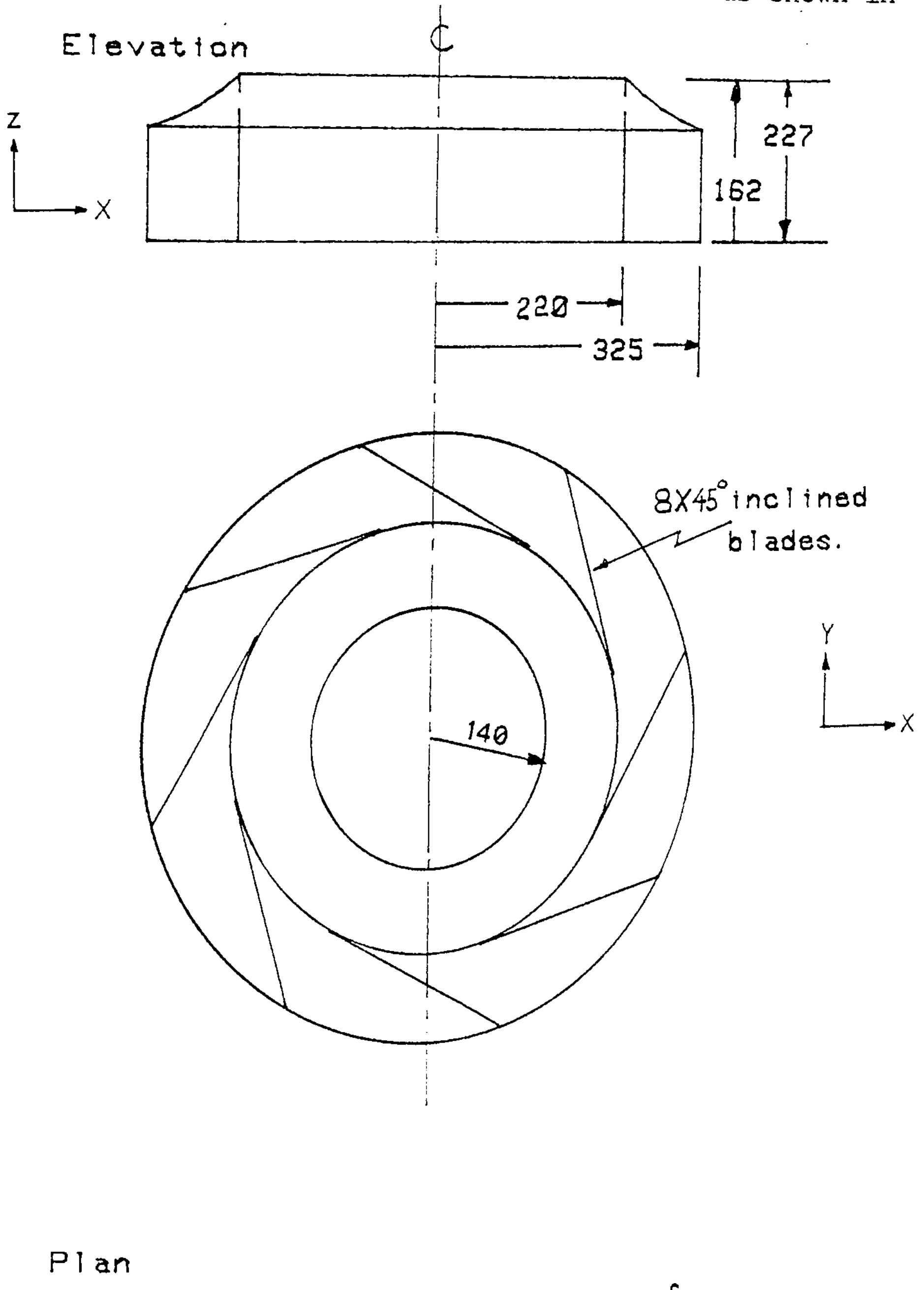
Fig. 8.20 Radial & tangential bending moments along the line =37.5°(backsheet): simplified fan impeller.

8.4 ACTUAL FAN IMPELLER

The final problem to be examined in this work is the actual fan impeller.

8.4.1 Orthographic views of actual fan impeller

The model chosen was typical of one in commercial production. It was a 650mm diameter lamminar , backward bladed impeller with dimensions of the backsheet , blades and the conesheet as shown in fig. 8.21.



aphic views of

8.4.2 Be'zier net prepared by interpretion module

After the engineering drawing has been interpreted by the interpretation module , sectorial symmetry is then exploited in preparing the Be'zier nets for the components of the fan impeller. Therefore , only a sector is utilized. The Be'zier nets for the backsheet , blades and conesheet are shown in fig. 8.22.

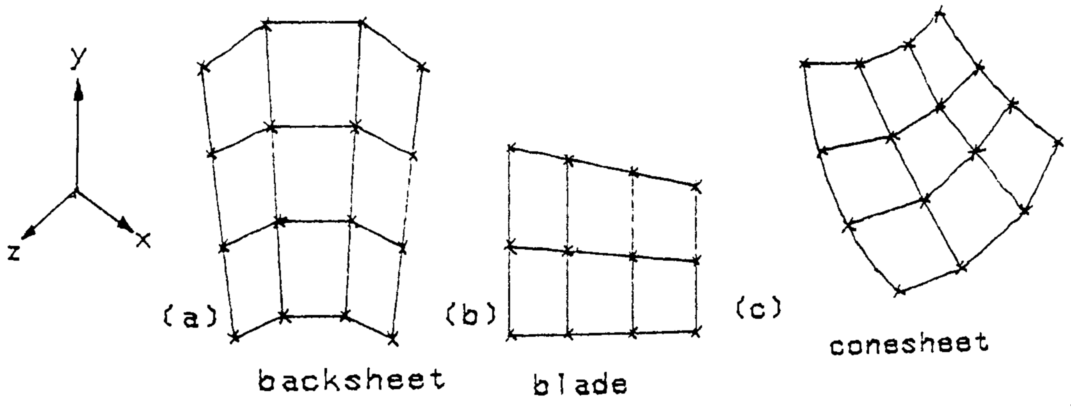


Fig. 8.22 Be'zier nets for actual fan impeller

8.4.3 Geometric model resulting from the geometric modeller

A sector of the geometric model of the actual fan is shown in fig. 8. 23. The whole fan can be obtained by replicating the sector several times.

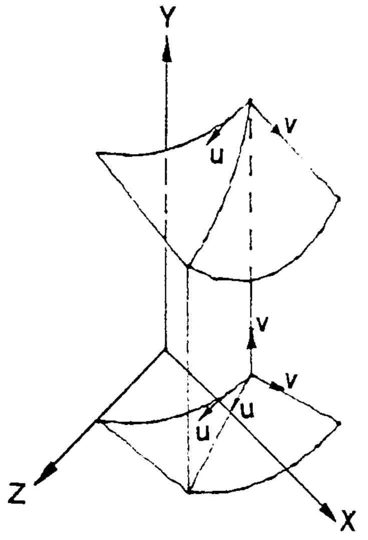


Fig. 8.23 Geometric model of actual fan impeller

8.4.4 Translating geometric model data into finite element data

Since the actual fan impeller was sectorially symmetric , only a sector of the impeller was analysed. This sector comprised part of the backsheet , one complete blade and a part of the conesheet. A triangular element was used.

Given the number of meshes along the u and v directions , the modeller displays the meshed surface and the translating module automatically computes the following data : nodal coordinates , elements connections , the semi-cone angle , the array containing the numbers of nodes having similar behaviour , the node number and angle of skewed nodes.

In this analysis , the inner edges of the fan impeller were assumed to be built in and the displacements at the geometrically corresponding nodes on the sector boundaries a--a and b--b were prescribed to be the same.

8.4.5 Accuracy of geometric data from drafting/modeller processes

The discussion on the geometric accuracy of the modeller for flat plates and simplified fan impeller are enough evidences that the backsheet and the blades of the actual fan impeller are accurately modelled within the error of 1%. The conesheet goemetry is also modelled accurateley within the error of 1%.

8.4.6 Discussion of the finite element analysis results for the actual fan impeller

The results in Ref. [13] formed the basis of comparison with the results of the drafting/modeller/finite element processes. It was decided to present the results in graphical form so that it will be easier to observe the correlations between results.

The finite element results for the inside of the backsheet of the actual fan impeller along the line = 27.5^0 are shown in fig. 8.24. (Refer to appendix B where codes are given to designate the backsheet , blade and conesheet and where relevant expressions for stresses on the inside/outside surfaces are also given). It is observed that the maximum stress occurs near the blade/backsheet junction. This was anticipated because of the interaction of the blade with the backsheet.

For the blade results , the calculated stresses on the root and tip surfaces are presented in the form of longitudinal stresses. It is observed that the stress distribution at the backsheet/blade junction is higher than that at the conesheet/blade junction which implies that the backsheet applies a greater restraining moment than the conesheet.

For the conesheet finite element results , it is observed that the stresses are higher near the conesheet/blade junction which as explained for the backsheet , was due to the action of the blade on the conesheet.

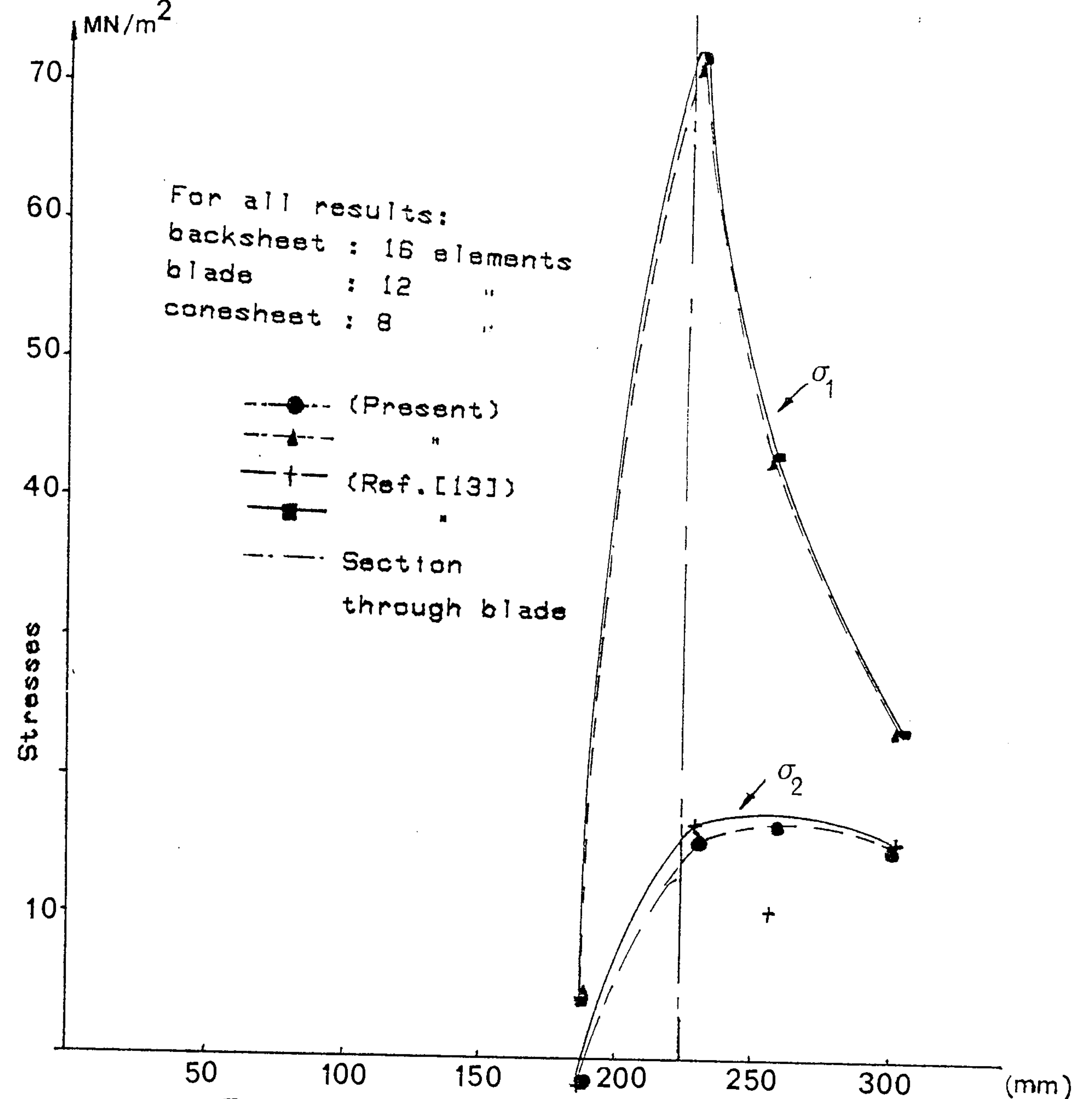


Fig. 8.24 Principal stresses on inside the backsheet(along centroids on line=27.5°) (actual fan impeller)

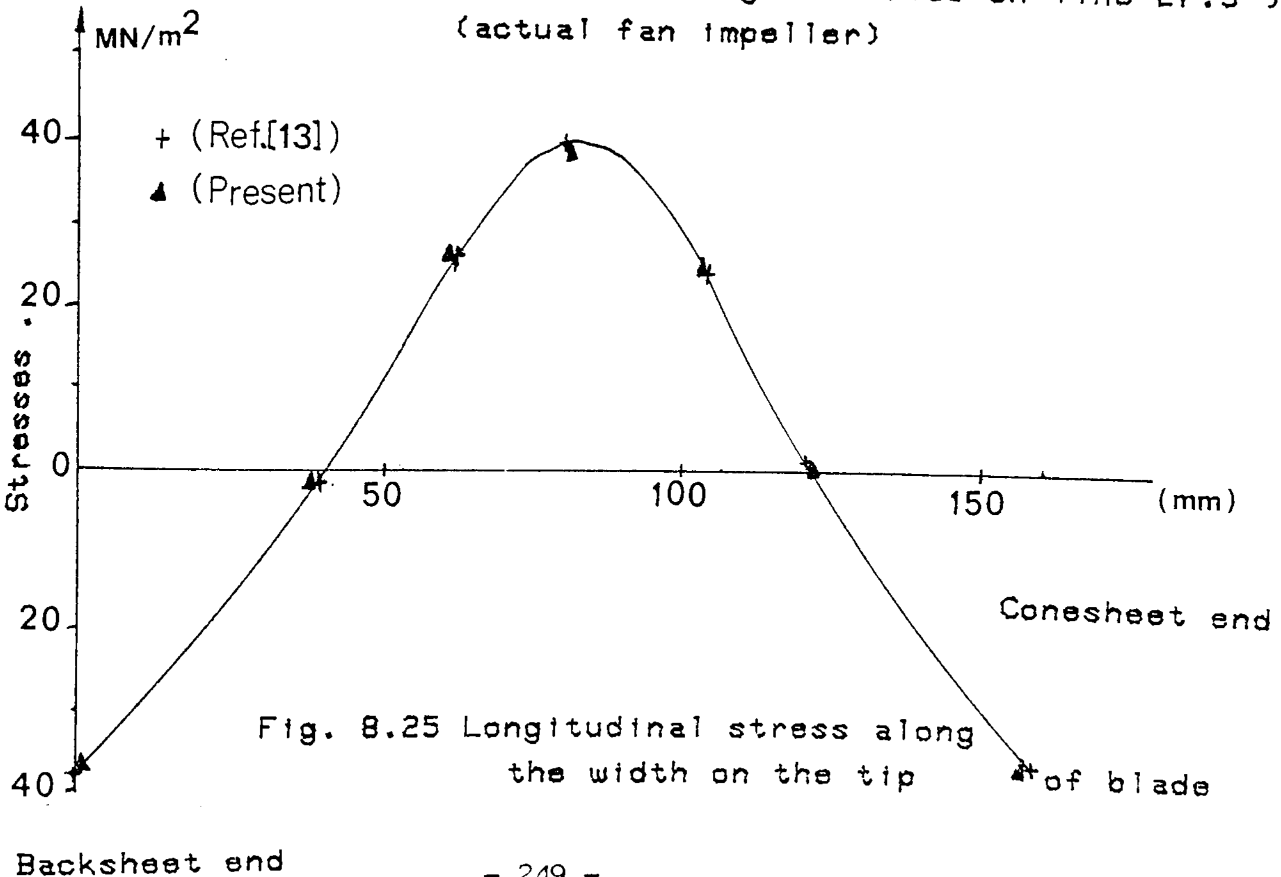


Fig. 8.25 Longitudinal stress along the width on the tip of blade

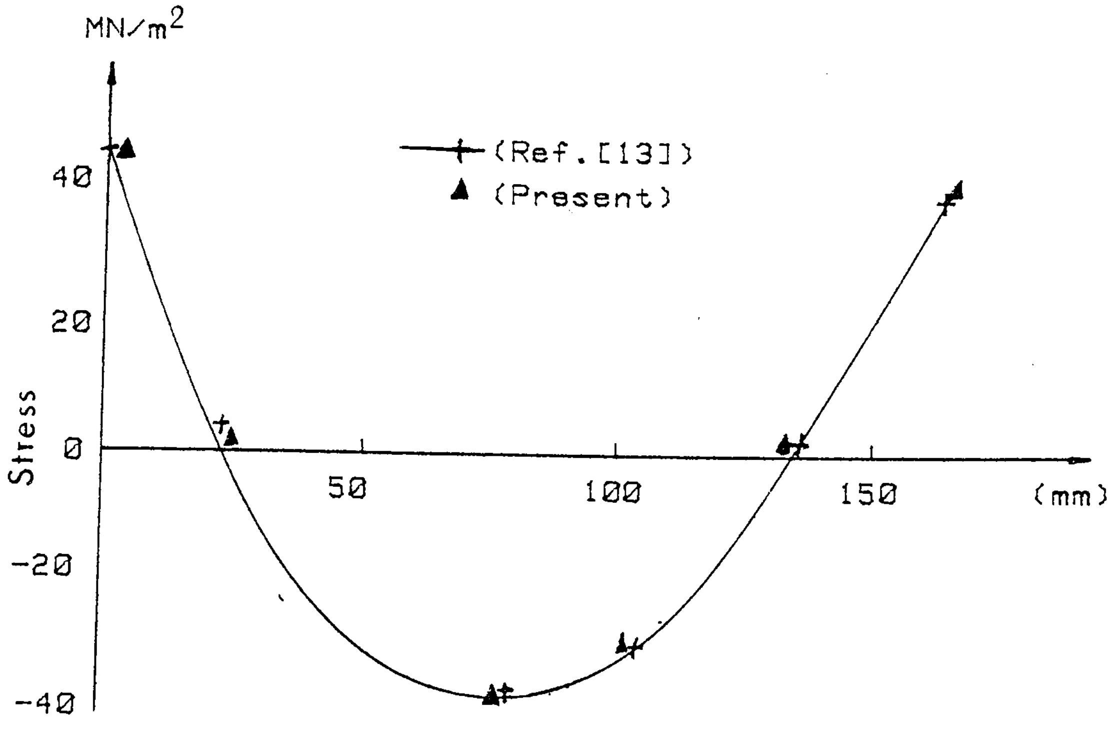


Fig. 8.26 Longitudinal stress along the width on the root of blade (actual fan)

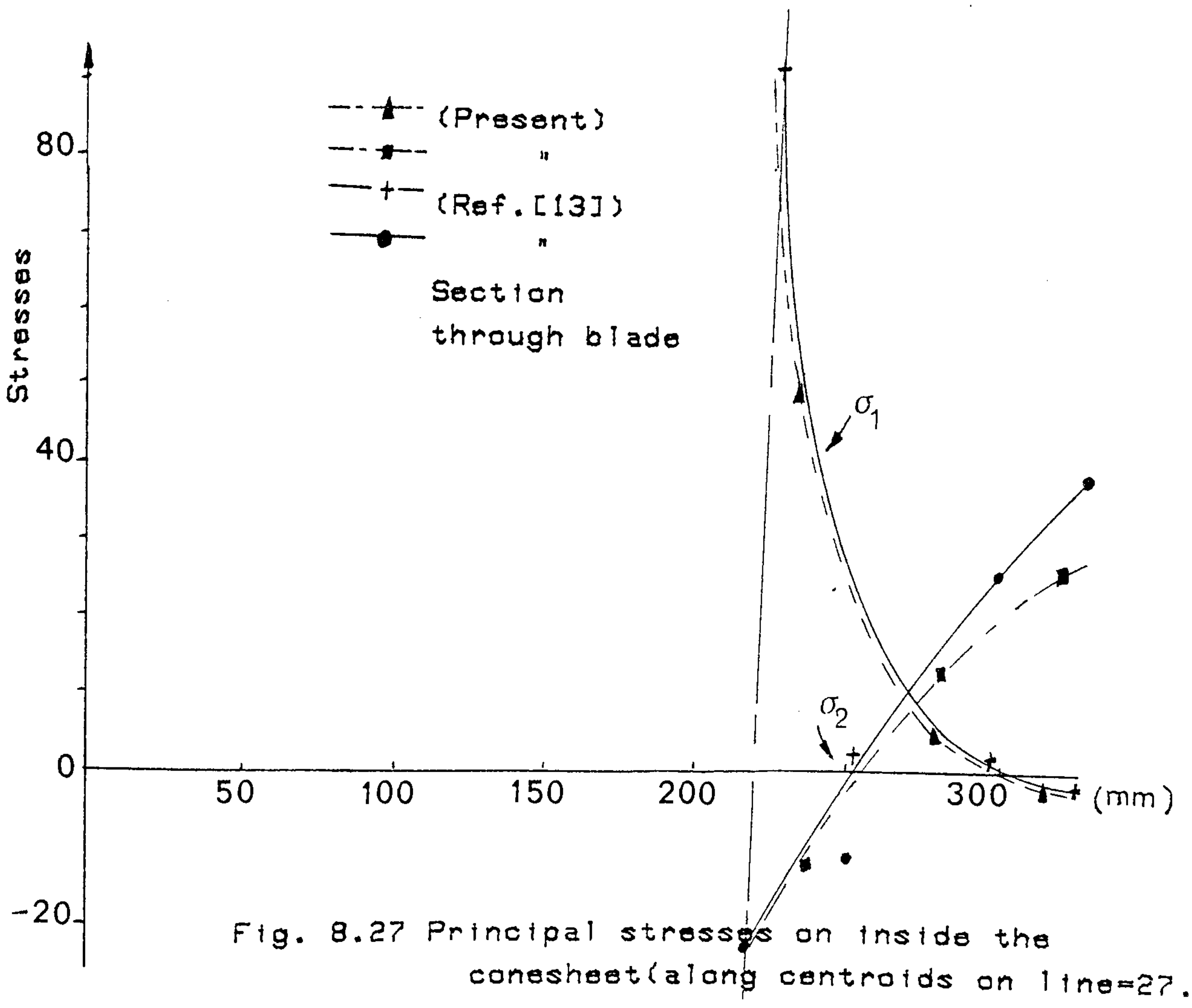


Fig. 8.27 Principal stresses on inside the conesheet(along centroids on line=27.5°)

8.5 CLOSING REMARKS

The Be'zier technique for surface representation has been used to bridge the gap between drafting and finite element analysis for shell type structures. It has been successfully applied to finite element data preparation for shell type structures and in particular , to a fan impeller. The finite element results showed good agreement with the results in Reference [13] where the input data were manually prepared. The correlation in results has confirmed the fact that Be'zier technique is suitable for idealization of models for finite element analysis for shell type structures.

Whilst developing the linking process for the actual fan impeller , some logic errors were observed in the application program IMPSMF (subprogram SKEWIM2) where the counter COUNT which is dependent on the number of skewed nodes on the backsheet and conesheet was wrongly represented;the same anomaly was observed for the counter JN. The linking processes for this model repeatedly crashed until these errors were removed.

CHAPTER 9

SUMMARY/GENERAL DISCUSSION , CONCLUSION AND RECOMENDATION FOR FURTHER WORK

The main reason for undertaking this research was to investigate the linking of drafting and finite element analysis and to see if a surface based geometric modeller could bridge the gap between these two. The program itself demonstrates that such a method is feasible. In this Chapter , the summary , conclusion and recommendations for further work are presented.

9.1 SUMMARY AND GENERAL DISCUSSION

9.1.1 Summary

The summary of the research reported in this thesis are as follows :

1. A computer program has been written which "understands" the meaning of a standard engineering drawing. The program prepares three-dimensional data from an existing two-dimensional general drafting package , MEDIA , for input to a surface-based geometric modeller. The program has been tested on a number of plate and shell structures.

2. A computer program SUFACE based on Bernstein-Be'zier polynomial patches for the definition of meshed-surfaces have been developed.

The input data to this program is the Be'zier net resulting from the interpretation of a standard engineering drawing discussed in section 9.1. The drafting / surface modeller interface has been tested on a number of plate and shell structures.

3. A translation module has been developed to translate the geometric data from the modeller , SUFACE , into a form suitable for input to the finite element analysis programs SMILOF and IMPSMF for general thin plate and shell structures.

4. The translation module was extended to take advantage of the sectorial symmetry of rotating fan impellers.

5. An application program GEOMDATA was developed which derives its input data from the geometric modeller , SUFACE , for the computation of geometric properties of any surface model.

6. The finite element analysis results from the drafing / modeller / finite element analysis interface for thin square plate , cylindrical shell roof were compared with results in Reference [13] and other researchers while the results for simplified and actual fan impellers were compared with results in Reference [13].

9.1.2 General discussion

The meshed-surface geometric modelling technique based on Bernstein-Be'zier polynomial patches which was used in this research work has been found to bridge the gap between the rather separate fields of drafting and finite-element analysis for plate/shell type structures. This technique is very useful for bridging the gap because the geometric modelling technique creates "pseudo meshes" which can be easily translated into finite element meshes. Time for input data preparation for finite element analysis is therefore greatly reduced. The linking of these separate fields has been made automatic by developing an interpretation module which acts as an interface between the drafting package and the geometric modeller developed in this research.

The Faculty of Engineering CAD laboratory has a commercial package , PIGS which is the interactive graphics suite for PAFEC finite element system. Finite element models are created using PIGS either by digitised input or interactive modelling ; both of which are manual. If detailed information about the model is available on engineering drawings , then digitised input is used , otherwise those nodes which form the basic outline of the model are interactively supplied. The library of PAFEC finite elements known as PAFBLOCK is the key to the generation of meshes by PIGS.
However , in the case of the work reported in this thesis , the interpretation of the engineering drawing is done by the developed programs and the meshes are automatically generated by the Be'zier surface technique. This new technique is very suitable in an integrated CAD system.

9.2 CONCLUSIONS

The following conclusions are drawn from the results of the research :

1. The computer program developed to interprete engineering drawings for the construction of plate and shell structures in three-dimensional space (a number of examples are given in chapter 8 and Appendix A) is reliable.

2. The stand-alone meshed-surface modeller , SUFACE , developed for three-dimensional surface construction is extremely versatile. It has two data-input points : one , from the engineering drawing interpretation module and the other , from manual input via computer key board. Using the second method , a user can very easily model very complicated engineering surfaces such as doubly-curved shell structures.

3. The use of Bernstein-Be'zier polynomial patches for the construction of thin plate and shell structures proved to be satisfactory for finite element idealization in an integrated CAD package for the HP9845B computer. Good agreeement was obtained between the drafting / modeller / finite element analysis interface results and results from finite element analysis when exact model geometry was manually input.

4. The membrane forces and bending moments obtained for the cylindrical shell roof and the membrane forces ,

bending moments as well as the principal stresses on the backsheet , blade and conesheet of the fan impeller obtained from drafting / modeller / finite element analysis interface agreed well with those obtained in Reference [13].

5. It is possible to instruct the modeller to prepare finite element analysis input data for only a part of a model if symmetry exists. For example , only a sector is considered in a fan model. The major advantage of exploiting sectorial symmetry is that computer memory is conserved during computation.

6. The accuracy of the finite element results depends on the number of elements for the model specified in the geometric modeller. The greater the number of elements the more accurate the results would be. However , memory capacity of the present facility limits the size of elements that can be handled for input to the application programs SMILOF and IMPSMF.

7. The application program GEOMDATA derives its input data from the geometric modeller , SUFACE and automatically computes the area properties of the model.

8. The different programs developed in this research work meet the requirements of different modules of a computer-aided design (CAD) system. Therefore , a significant outcome of the research is that an integrated CAD system has been developed with drafting (MEDIA) ,

modeller (SUFACE) , finite-element analysis (SMILOF/IMPSMF) , and area properties (GEOMDATA) as its different modules.

9.3 SUGGESTIONS FOR FURTHER WORK

The outcome of this research is the beginning of ample rewards to be reaped. The manner in which these rewards would be realised are matters discussed in this section :

1. Development of efficient algorithm for the computation of surface / surface and curve / surface intersection of plate and shell structures. This is obviously an area that is viable for further research and References [68] and [69] are recommended.

2. Developing a technique for the display of thickness of plate/shell structures by the geometric modeller discussed in chapter 5. At the moment , only the middle surfaces of models are displayed by the geometric modeller.

3. The key to a successful integrated CAD system is an efficient unified data-base. It is suggested that a unified data-base be developed to integrate the CAD functions (drafting , surface modeller , area properties and finite element analysis) of the present research.

4. The Hewlett Packard 9845B with only 187K bytes memory capacity and floppy disc of .5M bytes memory size is inadequate for a CAD

system. The execution time for the finite element analysis programs SMILOF and IMPSMF is prohibitive for substantial numbers of degrees of freedom. Therefore , it is suggested that firstly , the memory capacity of the computer be increased and secondly , a hard disc be used to improve execution time.

APPENDICES

APPENDIX A

FURTHER EXAMPLES OF DRAFTING/MODELLER /FINITE-ELEMENT ANALYSIS PROCESSES

Further examples are considered to reinforce the capability of the interpreting module for drafting/modeller interface. The geometric models and translated finite-element input data are also given.

A.1 Example 1

A.1.1 Orthographic view

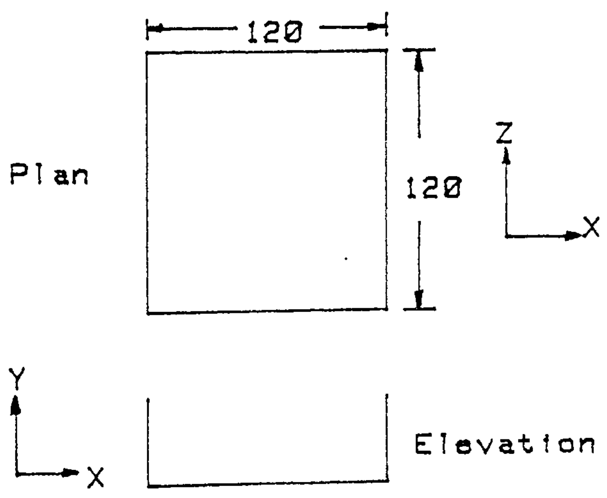


FIG. A·1 Orthographic view of Example 1

A.1.2 Be'zier net

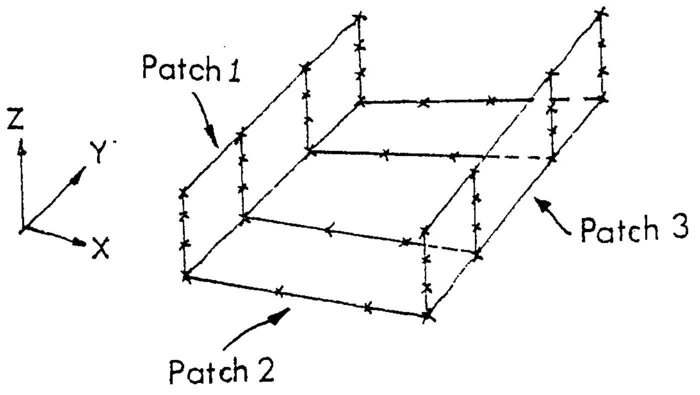


FIG. A.2 Be'zier net of Example 1

A.1.3 Geometric model

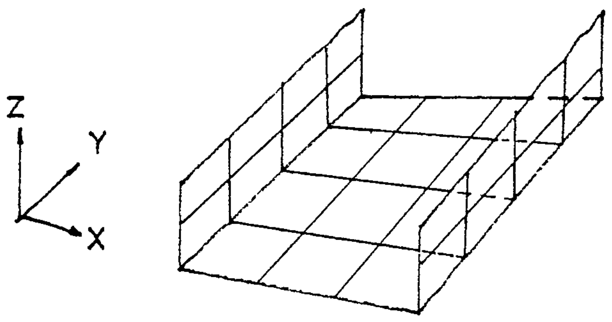
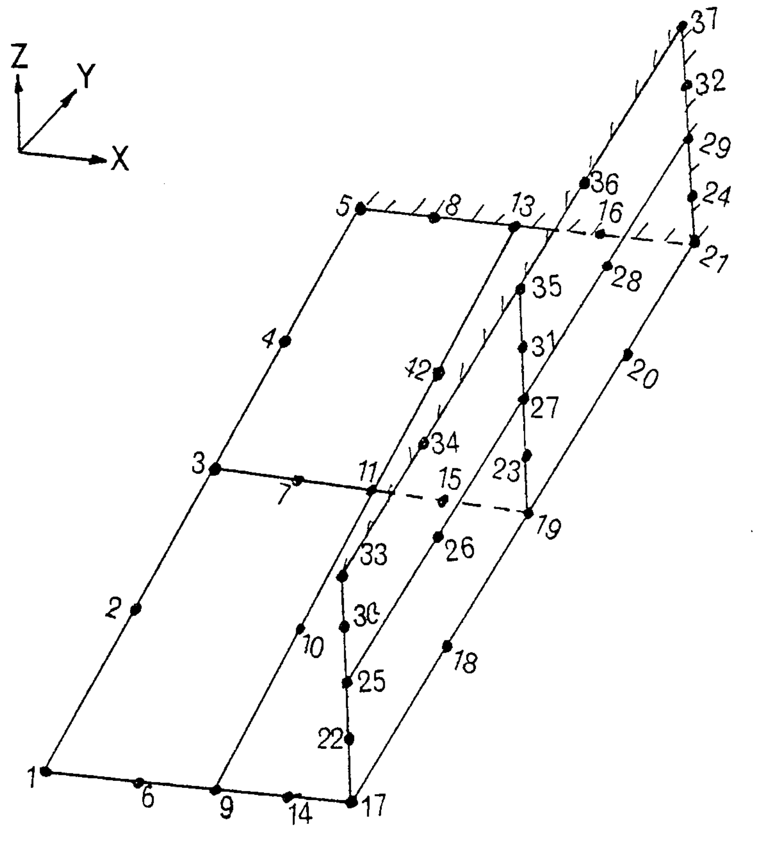


FIG. A.3 Geometric model of Example 1

A.1.4 Finite-element input data from the translating module

The translating module translates the geometric model data of section A.1.3 into the form suitable for the application program SMILOF for stress analysis :



(Enlarged)
FIG. A.4 Finite element model of Example 1

```
Type of element (Qort)= 1
No. of job to be solved (Njob)= 1
No. of elements (Nelemt)= 3
No. of nodes (Nnode)= 37
No. of sets of forces (Nsetfs)= 1
Principal stresses (Princ)= 1
No. of nodes where skewed boundary conds.(Nskw)= 0
No. of materials (Nmat)= 1
Type of output (Prnt)= 3
Gravity effect (Ch1)= 0
Normal distributed pressure (Ch2)= 0

Rotational speed (Omg)= 0

No. of nodes having similar behaviour (Nk)= 0
 0  0  0
 0  15.2962  0
 0  30.5923  0
 0  45.8885  0
 0  61.1846  0
 15.2964  0  0
 15.2964  61.1846  0
 15.2964  30.5923  0
 30.5928  0  0
 30.5928  15.2962  0
 30.5928  30.5923  0
 30.5928  45.8885  0
 30.5928  61.1846  0
 45.8892  0  0
 45.8892  61.1846  0
 45.8892  30.5923  0
 61.1856  0  0
 61.1856  15.2962  0
 61.1856  30.5923  0
 61.1856  45.8885  0
 61.1856  61.1846  0
 61.1856  0  26.7683
 61.1856  61.1846  26.7683
 61.1856  30.5923  26.7683
 61.1856  0  53.5365
 61.1856  15.2962  53.5365
 61.1856  30.5923  53.5365
 61.1856  45.8885  53.5365
 61.1856  61.1846  53.5365
 61.1856  0  80.3048
 61.1856  61.1846  80.3048
 61.1856  30.5923  80.3048
 61.1856  0  107.073
 61.1856  15.2962  107.073
 61.1856  30.5923  107.073
 61.1856  45.8885  107.073
 61.1856  61.1846  107.073
```

```
Element connections Icon(*):
 1  9  11  3  6  10  7  2  1

 3  11  13  5  7  12  8  4  1

 9  17  19  11  14  18  15  10  1

 11  19  21  13  15  20  16  12  1

 17  25  27  19  22  26  23  18  1

 19  27  29  21  23  28  24  20  1

 25  33  35  27  30  34  31  26  1

 27  35  37  29  31  36  32  28  1


No.of like elems,thickness & string of like nodes:
 8  .5  1  2  3  4  5  6  7  8


No. of spec. nodes,like nodes,code,presc. load/disp & node string:
 24  13  1  0  0  0  5  8  13  16  21  24  29  32  33  34  35  36  37 7
0 0  0  6  9  14  17  22  25  30  3  5  0  0  0  2  3  4  1  4  0  0 -1


No. of spec. midnodes,like midnodes,code,presc. loof nodes& node string:
 12  12  3  0  2  4  6  8  14  16  22  24  30  32  34  36


Elastic constants (E,v,p): 210000  .29  .0000000078
            End of checking
```

A.2 Example 2

A.2.1 Orthographic view

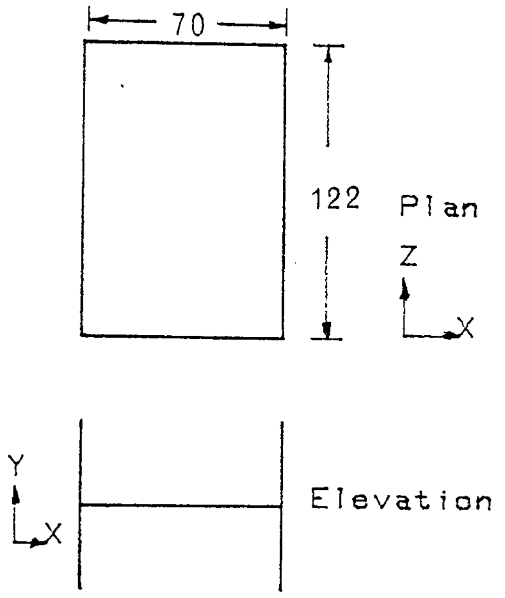


FIG. A.5 Orthographic view of Example 2

A.2.2 Be'zier net

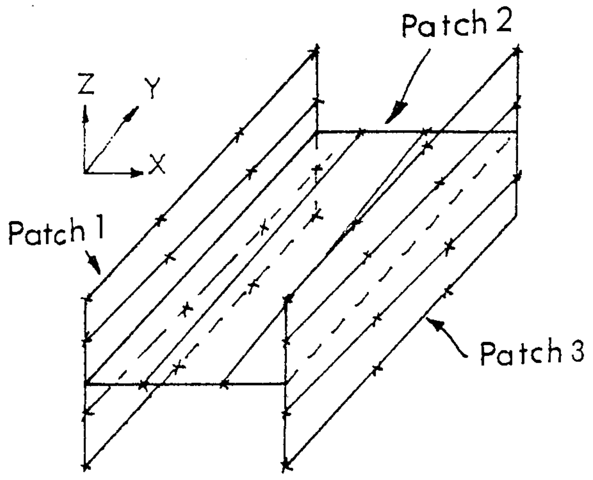


FIG. A.6 Be'zier nets of Example 2

A.2.3 Geometric model

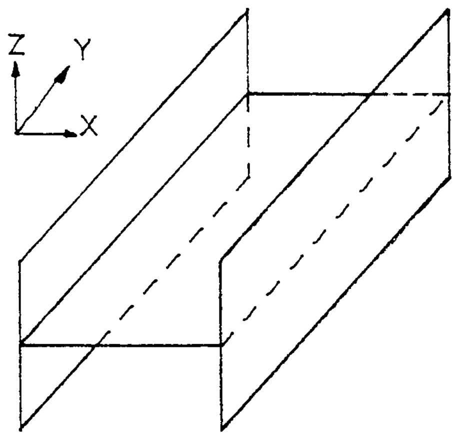
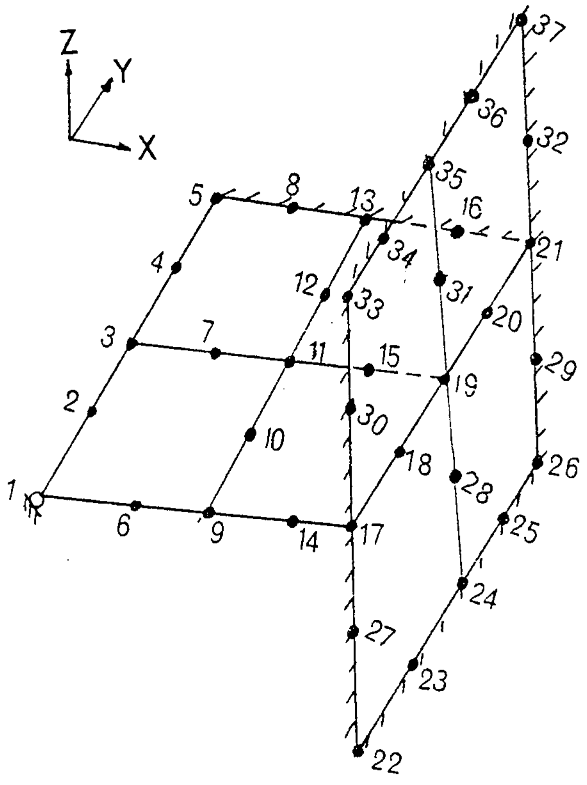


FIG. A. 7 Geometric model of Example 2

A.2.4 Finite-element input data from translator



```
Type of element (Qort)= 1
No. of job to be solved (Njob)= 1
No. of elements (Nelemt)= 8
No. of nodes (Nnode)= 37
No. of sets of forces (Nsetfs)= 1
Principal stresses (Princ)= 1
No. of nodes where skewed boundary conds.(Nskw)= 0
No. of materials (Nmat)= 1
Type of output (Prnt)= 3
Gravity effect (Ch1)= 0
Normal distributed pressure (Ch2)= 0

Rotational speed (Omg)= 0

No. of nodes having similar behaviour (Nk)= 0
 0  0  61.184
 0  24.8562  61.184
 0  49.7124  61.184
 0  74.5688  61.184
 0  99.4251  61.184
 17.2086  0  61.184
 17.2086  99.4251  61.184
 17.2086  49.7124  61.184
 34.4171  0  61.184
 34.4171  24.8562  61.184
 34.4171  49.7124  61.184
 34.4171  74.5688  61.184
 34.4171  99.4251  61.184
 51.6257  0  61.184
 51.6257  99.4251  61.184
 51.6257  49.7124  61.184
 68.8342  0  61.184
 68.8342  24.8562  61.184
 68.8342  49.7124  61.184
 68.8342  74.5688  61.184
 68.8342  99.4251  61.184
 68.8342  0  0
 68.8342  24.8562  0
 68.8342  49.7124  0
 68.8342  74.5688  0
 68.8342  99.4251  0
 68.8342  0  30.5916
 68.8342  99.4251  30.5916
 68.8342  49.7124  30.5916
 68.8342  0  91.7766
 68.8342  99.4251  91.7766
 68.8342  49.7124  91.7766
 68.8342  0  122.369
 68.8342  24.8562  122.369
 68.8342  49.7124  122.369
 68.8342  74.5688  122.369
 68.8342  99.4251  122.369
```

```
Element connections Icon(*):
 1   9   11   3   6   10   7   2   1

 3   11   13   5   7   12   8   4   1

 9   17   19   11   14   18   15   10   1

 11   19   21   13   15   20   16   12   1

 22   17   19   24   27   18   28   23   1

 24   19   21   26   28   20   29   25   1

 17   33   35   19   30   34   31   18   1

 19   35   37   21   31   36   32   20   1


 No.of like elems,thickness & string of like nodes:
  8   .5   1   2   3   4   5   6   7   8


 No. of spec. nodes,like nodes,code,presc. load/disp & node string:
  27   20   1   0   0   0   5   8   13   16   17   21   22   23   24   25   26   27   29   3
32 33   34   35   36   37   3   6   0   0   0   6   9   14   3   5   0   0   0   2   3   4   1
  0    0   -1    1


 No. of spec. midnodes,like midnodes,code,presc. loof nodes& node string:
  14   14   3   0   2   4   6   8   14   16   23   25   27   29   30   32   34   36


 Elastic constants (E,v,p): 210000  .29  .0000000078
            End of checking
```

A.3 Example 3 (No finite element input data included)

A.3.1 Orthographic view

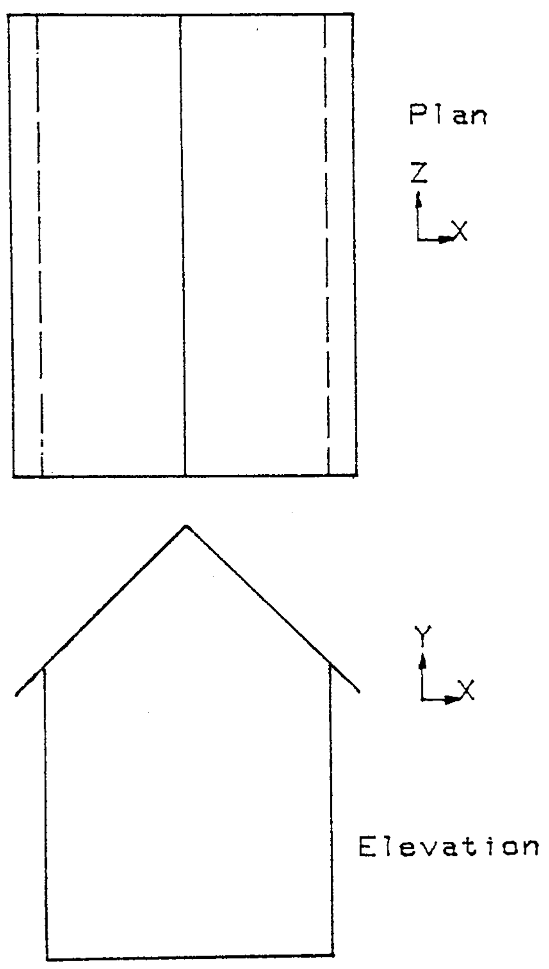


FIG. A.9 Orthographic view of Example 3

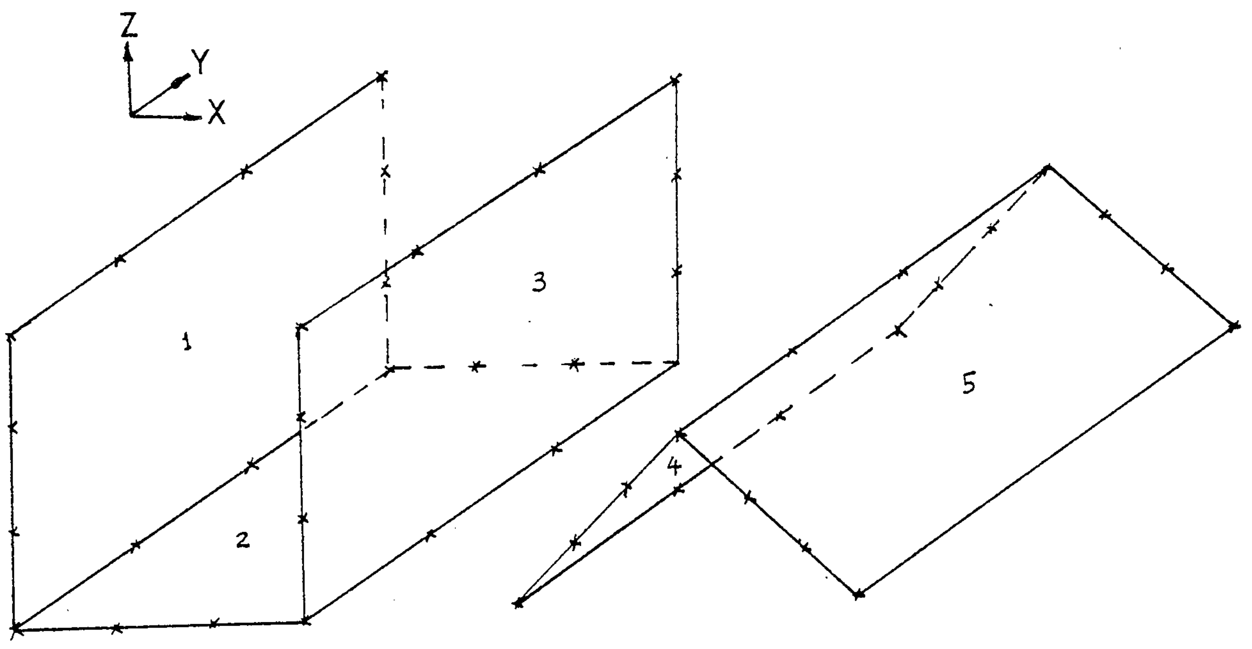


Fig. A.10 Bézier nets for Example 3

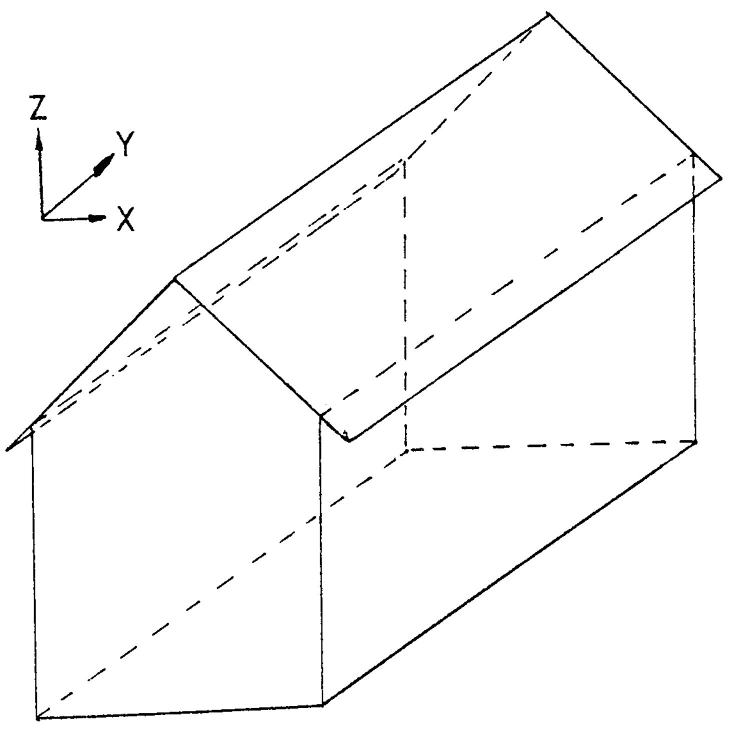
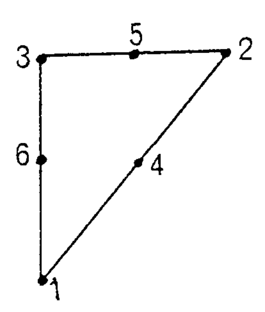


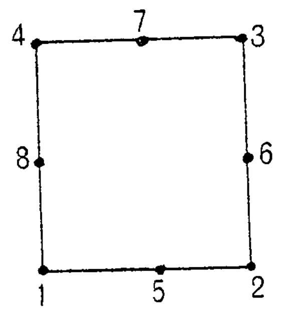
Fig. A.11 Geometric model of Example 3

APPENDIX B

B.1 IDENTIFICATION OF BACKSHEET, BLADE & CONESHEET OUTPUT RESULTS

The nodal connections and material number for each element are specified using the following sequence in the translation module :





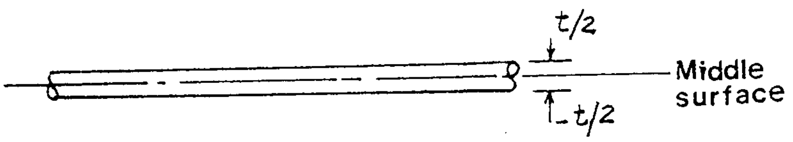
There are (7*Qort + 1) columns for the element and nodal connections matrix , where Qort = Ø or triangular element and Qort = 1 for quadrilateral element. Assuming M is the material number , then the translator records the nodal connections for the program IMPSMF as :

1 2 3 4 5 6 M	for triangular element , and
1 2 3 4 5 6 7 8 M	for quadrilateral element.

The element type (backsheet , blade or conesheet) is stored in the (7+2*Qort+1)th column. Therefore , using the codes in this column , it is easy to identify the output results that relate to any of the substructures of the fan impeller.

B.2 RELEVANT EXPRESSIONS FOR STRESSES

In this section of the appendix , the expressions for the stresses of a surface layer of shell at a distance z from the middle surface are presented. For the inner surface , the value of z equals t/2 while for the outer surface , z = -t/2 as shown below



The normal and shearing stresses are given as follows :

$$\sigma_x = \frac{N_x}{t} + 12.\frac{M_x}{t^3}.z$$

$$\sigma_y = \frac{N_y}{t} + 12.\frac{M_y}{t^3}.z$$

$$\sigma_{xy} = \frac{N_{xy}}{t} + 12.\frac{M_{xy}}{t^3}.z$$

REFERENCES

1.Moore , J.	"Eckard's Impeller - A ghost from ages past" , University of Cambridge , Department of Engineering , CUED / A - Turbo TR 83 , 1976.
2. Whitfield, A., Atkey, R. C. , and Wallace, F.J	"Computer aided design and testing of radial and mixed flow centrifugal impellers with straight and backward swept blades" , Paper C21/78 IMech.E.,1978.
3. Jansen, W.and Kirschner, A.M.	"Impeller blade design method for centrifugal compressor",NASA-SP3Ø4, pt.11, 1974, pp.537-563.
4.Smith, D.J.L & Merryweather, H.	"The use of analytic surfaces for the design of centrifugal impellers",International Journal for Num. Methods in Engineering, Vol. 7, 1973, pp.137-154.
5. Coons, S.A	"Surfaces for computer-aided design of space forms", Report MAC-TR-41, Massachusetts Inst. of Technology, MA, U.S.A., 1967.
6. Casey, M.V.	"A computational geometry for the blades and internal flow channels of centrifugal compressors", Transactions of the ASME, Vol.1Ø5, April 1983.
7. Be'zier, P.	"Mathematical and practical possibilities of UNISURF", Computer-Aided Geometric Design, Barnhill, R.E., and Riesenfeld, R.F., eds., Academic press, New York, pp.127-152, 1974.
8.Patton, R.G.	"Stresses and displacements in a rotating conical shells", J. App. Mech. A.S.M.E., 1943.
9. Glessner, J.W.	"A method of analysing the stresses in centrifugal impellers", Am. Soc. of Mech. Eng., Paper 54-A-167.
1Ø. Bell, R., & Benham, P.P.	"Theoretical and experimental stress analysis of centrifugal fan impellers", Journal of Strain Analysis,vol. 13, No. 3, 1978.
11. Haerle, H.	"The strength of rotating discs", Engineering Vol. CVI.1918, p.131
12. Bell, R.	"Theoretical and experimental stress analysis of centrifugal fan impellers" Ph.D Thesis, The Queen's University of Belfast, 1975.
13. Jweeg, M.	"Application of finite element stress analysis to rotating fan impellers", Ph. D Thesis ,

University of Aston in Birmingham,1983.

14. Irons, B. M. "The semiloof shell element", Conference on Finite Elements Applied to Thin Shells and Curved Menmbers, Chapter 11, University College, Cardiff, Wales, 1974, Edited by D.G. Ashwell and R.H. Gallagher.

15. Cooley, P. "MEDIA: 2-D Drafting package", Dept. of Mechanical Eng., University of Aston in Birmingham.

16. Sutherland, I. "SKETCHPAD: A Man-Machine Graphical Communication System", Proc. SJCC Vol.23, pp.329, 1963.

17. Markowsky, G. & Wesly, M.A. "Fleshing out wireframes",IBM J. Res. Dev., Vol.24, No.5, Sept.1980.

18. Forrest, A.R. "Mathematical principles for curve and surface", in Curved Surfaces in Engineering, Churchill College, Cambridge, pp.5-13,1972

19. Barnhill, R.E & Riesenfeld,R.F (eds) Computer Aided Geometric Design ,Academic Press,New York ,1974.

20. Gordon, W.J. & Schoenberg, I.J. (eds) "Distributive lattices and the approximation of multivariate functions",Proc. Symp. Approximation with Special Emphasis on Spline Functions, (Madison, Wisc. 1969). Academic Press, New York, 1969

21. Forrest, A.R "On Coons and other methods for the representation of curved surfaces", Computer graphics and image processing, Vol. 1, pp.341-359, 1972.

22. Roberts, L.G. "Machine perception of three-dimensional solids", Tech. Report 315, M.I.T. Lincoln Laboratory, Lexington, Mass., 1963.

23. Voelcker, H.B. & Requicha, A.G.G. "Solid modelling: a historical summary and contemporary assessment", IEEE Computer Graphics and Applications, pp.9-24, March 1982.

24. Requicha, A.G.G. "Representations for rigid solids: theory, methods and systems", Computing surveys, Vol.12, No.4, Dec., 1980.

25. Tilove, R.B "Set membership classification: a unified

approach to geometric intersection problems", IEE Transactions on Computers , Vol29, pp.874-883, 1980.

26. Tilove , R.B "Closure of boolean operations on geometric entities",Computer-Aided Design,Vol.12, pp.219-220,Sept.1980

27. Baer, A., Eastman, C & Henrion, M "Geometric modelling: a survey", Computer-Aided Design, Vol.11, No.5, September 1979.

28. Kimura, F "Geomap-III: designing solids with free-form surfaces", IEEE Computer Graphics & Applications, pp.58-72, June 1984.

29. Zienkiewicz, O.C. "The finite element method in engineering science", McGraw-Hill Book Company, New York, 1971

30. Richards, T.H. "Energy methods in stress analysis", Ellis-Horwood series in Engineering Science, London, 1977.

31.Herrmann, L.R. "Laplacian-isoparametric grid generation scheme", J. Engng. Mech. Div., ASCE, 102, pp.749-756, 1976.

32. Kamel, H.A & Shanta, P.J. "A solid mesh generation and result display package", J. Pressure Vessel Technology, ASME, 96, pp.207-312, 1974.

33. Zienkiewicz, O.C & Philips, D.V. "An automatic mesh generation scheme for plane and curved surfaces by isoparametric co-ordinates", Int. J. Num. Meth. Engng., Vol.3, pp.519-529.

34. Gordon, W.J. & Hall, C.A. "Construction of curvilinear co-ordinate systems and applications to mesh generation", Int. J. Num. Meth.Engng., Vol.7, pp.461-477, 1973.

35. Javaherian, Dowling, & Lyons, L.P.R. "Nonlinear finite element analysis of shell structures using the semiloof shell element", Computers and Structures, Vol.12, pp.147-159, 1980.

36. Martins, R.A.F. "Analysis of plates and arbitrary shells by the use of the semiloof shell element", Ph.D Thesis, University of Wales, 1976.

37. Forrest, R.A. "Computational Geometry", Proceedings of the Royal Society, Londom, A321, pp. 187-195

38. Be'zier, P. "Numerical control in automobile design and manufacture of curved surfaces", Proc. Curved

Surfaces in Engineering Conference, Cambridge, pp.44-88, 1972.

39. Lorentz, G.G. "Bernstein polynomials", University of Toronto Press, Toronto, 1953.

40. Forrest, R.A. "Interactive interpolation and approximation by Be'zier polynomials", The Computer Journal, Vol.15, No.1, pp.71-79 ,1972.

41. Sabin, M.A "Numerical Master Geometry", Proc. Curved Surfaces in Engineering Conference, Cambrigde, pp.23-25, 1972.

42. Gould, S.S. "Surface programs for numerical control", Proc. Curved Surfaces in Engineering Conference, Cambridge, pp.14-18, 1972.

43. Be'zier, P. "Numerical Control—Mathematics and Application", John Wiley, London, 1972.

44. Sutherland, I.E. "Three-dimensional data input by tablet", Proceedings of the IEEE, Vol.62, pp.453-461, 1974.

45. Idesawa, M. "A system to generate a solid figure from a three view", Bull. JSME Vol.16, pp.216-225, Feb.1973.

46. Preiss, K & Kaplanasky, E Spolving CAD/CAM problems by heuristic programming", Computers in Mechanical Engineering, pp.56-6Ø, Sept.1983

47. Adelfeld, B. "On automatic reconstruction of 3D structures from 2D representations", Computer-Aided Design, Vol.15, No.2, pp.59-64, Madrch 1983.

48. Lafue, G. "Recognition of three-dimensional objects from orthographic views", Proceedings 3rd Annual Conference on Computer Graphics, Interactive Techniques, and Image Processing, ACM/SIGGRAPH, pp.1Ø3-1Ø8, July 1976.

49. Newman, W.M. & Sproull, R.F. "Principles of Interactive Computer Graphics, 2nd Edition, McGraw Hill International Book Co., 1981.

5Ø. Foley, J.D. & Fundamentals of Interactive Computer Graphics, Addison Wesley Publishing Co.

51. Fredrikson, B Mackerle, J., Persson, B.G.A. "Finite-element programs in integrated software for structural mechanics and CAD", Computer- Aided Design, Vol.13, No.1, pp.27-39, Jan.1981.

52. Wordenweber, B "Automatic mesh generation of 2 & 3 dimensional curvilinear manifolds", Ph.D dissertation, Computer Laboratory, University of Cambrigde, Nov.1981.

53. Butlin, G. "CAD/FEM interfacing", C175/83, IMechE , pp.95-14, 1983.

54. Golden, M.E. "Geometric structural modelling: a promising basis for finite element analysis", Computers and Structures, Vol.1Ø, pp.347-35Ø ,1979.

55. Came, P.M. "The development, application & experimental evaluation of a design procedure for centrifugal compressor", Proc. Inst. Mech. Engr. Vol.192, pp.49-67,

56. Ashwell D.G. & Gallagher, R.H. Finite Elements for Thin Shells and Curved Members, (Eds.), John Wiley & Sons, 1976.

57. Wu, S. & Abel, J.F. "Representation & discretization of arbitrary surfaces for finite element shell analysis", Int. J. for Nu. Methods in Engineering, Vol.14, pp.813-836, 1979.

58. Wilson, H.B. & Farrior, D.S "Computation of geometrical and inerrial properties for general areas and volumes of revolution", Computer Aided Design, Vol.8, No.4, pp.257-263, Oct.1976.

59. Miles, R.G. & Tough, J.G. "A method for the computation of inertial properties for general areas", Computer-Aided Design ,Vol.15, No.4, pp.196-2ØØ, July 1983.

6Ø. Lee, Y.T.& Requicha, A.G.G. "Algorithms for computing the volume and other integral properties of solids. I: Known methods and open issues", Communication of the ACM, Vol.25, No.9, Sept.1982.

61. Lee, Y.T. & Requicha, A.G.G "Algorithms for cmpuing volume and other integral properties of solids. II: A family of algorithms based on representation conversion and cellular approximation", Communications of the ACM, Vol.25, No.9, Sept.1982.

62. Boyse,J.W. & Gilchrist, J.E. "GMSolids: interactive modelling for design and analysis of solids", IEEE Computer Graphics and Applications, Vol.2, No.2, pp.27-4Ø, March 1982.

63. Ryder, G.H. Strength of Materials , Macmillan and Co. Ltd, 1969.

64. Roark, R.J. & Formulas for Stress and Strain, 5th Edition,

Young, W.C.	McGraw-Hill Publishing Co. Ltd, New York, 1975.
65. Fenner, R.T.	Finite Element Methods for Engineers , McMillan Press ,1975.
66. Timoshenko, S.P. & Woinowsky-Krieger, S.	"Theory of plates and shells", 2nd Ed., McGraw-Hill Book Co., New York, 1959.
67. Scordellis A.C & Lo, K.S.	"Computer analysis of cylindrical shells", J. Am. Inst. Vol.61, May 1964.
68. Hanna, S.L., Abel, J.F. & Greenberg, D.P.	"Intersection of parametric surfaces by means of look-up tables", Computer Graphics & Applications, Vol. , pp.39-48, 1983.
69. Faux, I.D., & Pratt, M.J.	Computational Geometry for Design and Manufacture, John Wiley & Sons, New York, 1979.