SHIP HULL REPRESENTATION

BY

NON-UNIFORM RATIONAL B-SPLINE

SURFACE PATCHES

Thesis submitted for the degree of Master of Science in Engineering at the University of Glasgow

by

Manuel Filipe Simões Franco Ventura

Department of Naval Architecture and Ocean Engineering

University of Glasgow

March 1996

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

Summary

The purpose of this work is to propose a new method for representing the ship hull shape with mathematic surfaces so that geometric data can be generated for any point on the hull where required to assist the production process.

An extensive survey of previous work is presented covering both the use of parametric curves and surfaces to model the ship hull and also the most relevant software systems developed for that purpose. The main methods and algorithms available for the generation and edition of curves and surfaces are presented and compared taking into consideration the intended application. From the analysis of the formulations available it was concluded that the most adequate one, which however had not yet been extensively used to model ship hulls was the Non-Uniform Rational B-Splines (NURBS), due to the potential of their capability to represent exactly conic curves and surfaces. Therefore these surfaces were selected as the basis of the method developed in this thesis.

A procedure is proposed for the representation of a given hull form in a two step approach, creating first a wireframe model over which the surface patches are generated. Both curves and surfaces are based on the NURBS formulation. To create the wireframe model, first a set of longitudinal boundary lines is selected, dividing the surface into areas of similar shape. Then, these lines are fitted by curves and faired to some extent. Next, transverse sections are defined and split by the boundary lines. Surface patches are then generated over the transverse section curves within the limits of each patch. Finally, to obtain the traditional representation of the ship surface by transverse sections, buttocks and waterlines, contour lines are generated for constant values of x, y and z coordinates.

A computer system has been developed incorporating an interface that allows the visualization of the curves and surfaces being modeled. The system incorporates

several algorithms for generation and edition of curves and surfaces, in addition to the main contribution of this thesis which is the use of NURBS to represent the ship hull surface. The system also incorporates curve and surface analysis tools and some basic fairing algorithms so that during the several steps of the creation of the model, the fairness of the curves and surfaces can be evaluated and improved to some extent.

The procedure is tested and compared with an existing commercial system through some application examples, of a complete hull and in more detail in the bow region, showing that good results can be obtained with the system presented here.

Table of Contents

1. Introduction **1**

2. Representation of Ship Hull Surface **5**

2.1 Wireframe representations 9

2.2 Surface representations 14

2.3 Summary of reviewed approaches 17

2.4 Commercial systems for hull surface design 20

3. Parametric curves and surfaces **23**

3.1 Cubic splines 25

3.2 Bézier curves and surfaces 28

3.3 Coons surface patches 30

3.4 B-Spline curves and surfaces 34

3.5 Beta-splines 40

3.6 Rational B-Spline 41

3.7 Conclusion 50

4. Hull Representation by NURBS Surface Patches **51**

4.1 Filtering digitised input points 53

4.2 Interpolation and approximation of ship lines by B-spline curves 55

4.2.1 Parameterisation 55

4.2.2 Curve interpolation 57

4.2.3 Curve approximation 64

4.3 Curve shape control 65

4.3.1 Moving the control points 66

4.3.2 Changing the knot vector 66

4.3.3 Changing the weights 68

4.4 Basic algorithms 70
4.4.1 Degree raising 70
4.4.2 Knot insertion 71
4.4.3 Knot removal 72

4.5 Curve editing techniques 74
4.5.1 Curve refinement 74
4.5.2 Curve splitting 74
4.5.3 Curve joining 76
4.5.4 Creation of knuckles 77
4.5.5 Creation of straight segments 78
4.5.6 Creation of conic segments 79

4.6 Generation of the surface patches 82
4.6.1 Lofting 83
4.6.2 Ruled surfaces 85
4.6.3 Extrusion 86
4.6.4 Sweep surfaces 87
4.6.5 Blending surface 93

4.7 Representation of conical surface elements 93

4.8 Curve and surface analysis 96
4.8.1 Zero order interrogation tools 96
4.8.2 First order interrogation tools 96
4.8.3 Second order interrogation tools 98
4.8.4 Third order interrogation tools 104
4.8.5 Graphic representation of surface characteristics 105

4.9 Curve and surface fairing 106
4.9.1 Curve fairing 107
4.9.2 Surface fairing 110

4.10 Surface/surface intersection 111
4.10.1 Simplified intersections of free surface with sets of orthogonal planes 112
4.10.2 Simplified intersection of a free surface with a conic surface 114

4.11 Area of a surface patch 117

5. Application examples **120**

5.1 Description of system 121

5.2 Modelling a tanker of about 80,000 TDW 124
5.2.1 Bow 125
5.2.2 Forebody 129
5.2.3 Midships 130
5.2.4 Aftbody 131
5.2.5 Stern 132
5.2.6 Conclusions 134

5.3 Modelling of the bow of a tanker of about 160,000 TDW 136
5.3.1 Boundary and knuckle curves 136
5.3.2 Transverse sections 137
5.3.3 Surface patch generation 138
5.3.4 Hawse pipe generation and intersection with the hull 142
5.3.5 Conclusions 142

5.4 Comparison with a commercial package 143
5.4.1 Curves 143
5.4.2 Surfaces 144
5.4.3 Application examples 145

5.5 General remarks about modelling procedures 149

6. Conclusions **150**

6.1 Overall conclusions 150

6.2 Specific conclusions 153

6.3 Future developments 155
6.3.1 Mathematical algorithms 155
6.3.2 Features specific to ship modelling 160

References **161**

List of Figures

3.1 Wooden spline with ducks 25

3.2 Cubic spline with different end conditions 27

3.3 Bézier curve 29

3.4 Coons bilinear surface patch 31

3.5 Boundary curves and cross tangents on Coons patch 32

3.6 B-Spline curves of order 2, 3 and 4 36

3.7 Basis functions for $X = \{0,0,0,1,2,3,3,3\}$ 39

3.8 Basis functions for $X = \{0,0,0,0.5,1,3,3,3\}$ 39

3.9 Basis functions for $X = \{0,0,0,1.5,1.5,3,3,3\}$ 39

3.10 Geometric construction of a NURBS curve 42

3.11 Effect of changing the weight of control point 3 43

3.12 Linear precision property of NURBS curve 43

3.13 Convex hull property in NURBS curve of order 4 44

3.14 Representation of conics by NURBS curves 46

3.15 Circle represented by a 9 control point NURBS curve 46

3.16 Circle represented by a 7 control point NURBs curve 47

3.17 Circle represented by a 7 control point NURBs curve 48

3.18 Ellipse represented by NURBS curve 48

4.1 Types of curves used on ship lines 51

4.2 Boundary lines 53

4.3 Filtering of polygonal line with several tolerance values 54

4.4 Influence of type of parameterisation on curve shape 56

4.5 B-spline fitted to data points using the nodes (Method I) 58

4.6 B-spline fitted by the Least Square (Method II) 58

4.7 Two adjacent Bézier cubic segments 59

4.8 B-spline fitted with piecewise cubic Bézier segments (Method III) 61

4.9 Midship section interpolated with Method I 62

4.10 Midship section interpolated with Method II 62

4.11 Midship section interpolated with Method III 63

4.12 Midship section with extra points P1' and P8' interpolated with Method I 63

4.13 Data points approximated by a B-spline curve with 6 control points using the Least Square method 64

4.14 Curve smoothed by least square approximation 65

4.15 Effect of moving a single control point 66

4.16 Effect of the iteration on the knot vector 68

4.17 Changing the weight on a single control point 68

4.18 Changing the order of a NURBS curve 71

4.19 Knot removal on a curve 72

4.20 Knot removal on a surface 73

4.21 Curve refinement 74

4.22 Curve splitting 76

4.23 Curve joining 76

4.24 Creation of knuckle in point P_3 77

4.25 Comparison of basis functions 78

4.26 Creation of straight segments 79

4.27 Creation of conic segment 80

4.28 Circular arc generation 81

4.29 Generation of circular arc greater than 180 degrees 82

4.30 Surface generated by lofting of section curves 84

4.31 Ruled surface 86

4.32 Extruded surface 86

4.33 Sweep surface 87

4.34 Frenet frame 88

4.35 Reference frames for sweeping surface 89

4.36 Rotation about the X axis 91

4.37 Rotation about the Y axis 92

4.38 Surface of revolution obtained by sweeping 92

4.39 Cylindrical surface generated from two circular sections 94

4.40 Conical surface generated from two circular sections, one of zero radius 95

4.41 Circular ring surface 95

4.42 Circular surface 96

4.43 Contours of isophotes 97

4.44 Line curvature displayed by a porcupine representation and by the curvature plot 99

4.45 Normal curvature 101

4.46 Rectangular patch and degenerate patch 102

4.47 Triangulation of tensor product patch 105

4.48 Fairing of cubic B-spline curve 108

4.49 Fairing using piecewise circular arcs 109

4.50 Fairing using piecewise circular arcs 109

4.51 Contour segment in triangular element 113

4.52 Transverse sections of a surface obtained by contouring 114

4.53 Line/Triangle intersection 115

4.54 Surface/Cylinder intersection 117

4.55 Area of rectangular surface element 118

4.56 Area of triangular element 119

5.1 Layer structure of the software system developed 122

5.2 Digitised lines 124

5.3 Input half-sections, knuckle and profile 125

5.4 Bow represented by single surface patch 126

5.5 B-spline curves fitted to input data 127

5.6 Bow represented with single surface PS/SB 127

5.7 Bow represented by 4 surface patches 128

5.8 Exploded view of bow patches 128

5.9 Lines defining the forebody 129

5.10 Forebody surface patch 129

5.11 Lines defining the midship 130

5.12 Midship surface patch 130

5.13 Lines defining the aftbody 131

5.14 Curves defining the aftbody 131

5.15 Aftbody surface patch 132

5.16 Input stern lines 132

5.17 Stern curves 133

5.18 Stern surface patches 133

5.19 Exploded view of stern patches 135

5.20 Model of complete ship 135

5.21 Boundary and knuckle curves, forward 137

5.22 Digitised transverse sections (forward) 137

5.23 Initial curves used to model the forebody surface 138

5.24 Auxiliary surface 138

5.25 New sections obtained by surface intersection 139

5.26 Arc of 180 deg. Represented by NURBS curve 139

5.27 Bulb surface mirrored about centreline 140

5.28 Bow modelled with 6 surface patches 138

5.29 Intersection of hawse pipe with hull 142

5.30 Forebody of tanker 146

5.31 Aftbody of tanker 146

5.32 Full hull of tanker modelled with Autoship 147

5.33 Forebody of tanker modelled with a single surface 148

5.34 Aftbody and stern panel of tanker modelled with two surfaces 148

6.1 Affine invariant angle knot spacing 156

6.2 Triangle area for parameterisation 156

6.3 Surface filleting 158

6.4 Surface chamfering 158

6.5 Trimmed surface 159

List of Tables

2.1 Survey of hull representation methods....7

2.2 Survey of hull representation methods....8

2.3 Survey of hull representation methods....9

2.4 Summary of reviewed approaches to hull representation....19

2.5 Summary of reviewed approaches to hull representation....20

2.6 Survey of hull representation PC software....21

2.7 Survey of hull representation PC software....22

4.1 Evaluation of error in area computation....120

Acknowledgements

This work was carried out at the Department of Naval Architecture and Ocean Engineering (NAOE) of the Glasgow University and at the Instituto Superior Técnico, Universidade Técnica de Lisboa.

I wish to express my sincere gratitude to Professor Douglas Faulkner for all the cooperation of the NAOE department and to Professor C. Guedes Soares and to Mr. Ian Winkle for their valuable guidance, help, advice and support throughout the project.

I also would like to thank the Junta Nacional de Investigação Científica e Tecnologia - Programa Ciência, who provided me with the necessary financial support during my stay in Glasgow.

Manuel F. Ventura
March 1996

Author's statement:

All the material presented in this thesis is original, except where reference is made to other sources.

Notation

k	Order of the B-spline curve
n	Number of control points
C_i, $C_{i,j}$	Control points (vertices) of curves and surfaces
$B_{i,j}$	Bernstein basis functions
$N_{i,k}$	B-spline basis functions
P_i	Curve points matrix
X_i, Y_i	Knot vectors in u and v parametric directions
ς_ι	Nodes
D_i	Data points matrix
t	Curve parameter
β_1	Bias parameter
β_2	Tension parameter
l, m	Order of B-spline surface in u and v parametric directions
$S_{i,j}$	Surface points matrix
u, v	Surface parameters
w_i	Weight of control point (in NURBs formulations)
κ	Normal curvature
K	Gaussian curvature
H	Mean Curvature
$\mathcal{K}_{min}$, $\mathcal{K}_{max}$	Principal curvatures
$\mathcal{K}_{abs}$	Absolute curvature
k_c	Conic shape invariance of conic curves

1. Introduction

The traditional design process of a ship hull form, works mainly on a lines drawing on which three sets of lines representing the intersections of the hull surface with a series of planes parallel to the three orthogonal projection planes are shown. The drawing is organised in three views: the elevation or profile, generally called the sheer plan, a view looking down, called the half-breadths plan and the last one, showing the transverse sections, called the body plan. Each of the views represents a reference plane on which the several lines of the hull form have been projected. Normally, and taking into account that almost all ships are symmetrical about the longitudinal centreline plane, only half of the hull is represented [1]. Other important lines also represented on the lines plan are the tangency lines like the flat-of-side and the flat-of-bottom, the knuckle lines, representing discontinuities in the hull and the diagonals, representing intersection lines of the hull with arbitrary transversely inclined longitudinal plans.

Starting with a set of sections and waterlines defined by points, the hull surface design method is an iterative process in which the designer fits a line to each set of points, using a wooden spline, correcting the initial points as necessary to obtain a fair shape. After fairing a set of lines, the changes obtained are transported to the other plans. The process is repeated until all the lines represented have an acceptable degree of fairness and compatibility in the three views.

Some of the disadvantages of the traditional approach are the non existence of other tools than the designer's experience to assist the fairing work, and the accumulation of errors that the process generates. The evaluation of the fairness of the curves depends only of the visual inspection and the only fairing tools available are the wooden splines, on which the control parameter is the flexibility. The main source of errors are the readings on scaled drawings and the transport of data between different views, made in each iteration of the process.

The need of mathematical models for the description of ship forms was raised with the introduction in the 50's of numerical controlled machines in shipyard operation. The machines offered great improvements in efficiency but required accurate coordinate description of each plate shape. With the increase of computer power availability, in the 60's and 70's a growing offering of CAD systems and research in the field has occurred.

After the 80's, the major CAD systems were established, the methods of surface description were considered sufficient and R&D started moving to areas like solid-modelling applied to structure generation, piping and outfitting, engineering information management, data exchange with existing engineering systems, user interface, compatibility, portability, etc.

The rising market of personal computers in the late 80's, however, has generated a new wave of small systems dedicated to Naval Architecture. With the importance now given to the ease of use and thanks to the increasing computing power available, the methods of surface modelling have been re-evaluated and updated, and even some of the established software systems are improving the modules covering the creation of the hull shape, making them more interactive and porting them to more user friendly environments.

The earlier software systems dedicated to model the ship hull surface used the wireframe representation, reproducing with true 3D lines the traditional approach based on 2D projections of the 3D lines on the reference planes. Although these systems have supported the needs of ship designers and builders for long time, they are limited by a main drawback - to obtain data related to any points on the surface, not contained on the model basic lines, new lines must be computed. This is obtained by an interpolation process, in which intersections must be first computed with all the model lines in the opposite direction, and then fitting a curve to the obtained intersection points. The approximations done through this process do not guarantee that the points finally obtained do belong to the surface.

•

Surface representations, on the other hand, supply data for all the points required and guarantee that every point has a unique representation

The motivation for this work came from the recent developments in rational B-Spline curves and surfaces, that have been established as an industrial standard in many CAD systems and are already included in standard graphics libraries like PHIGS PLUS and OpenGL and in standards for data exchange like IGES (Initial Graphics Exchange Specification) and STEP (Standard for the Exchange of Product Model Data, ISO 10303).

Rational B-Splines are well suited for interactive work and are powerful enough to represent almost every kind of curves and surfaces, even those that previously required an algebraic representation, such as the conics. This flexibility allows the software systems to be more compact, as they deal with a more reduced set of geometric entities, and simultaneously more efficient and reliable, due to the generalisation and simplification of the methods and algorithms developed to operate on those entities.

Mathematical models of ship hull are used mainly for two purposes:

- **generation of new ship forms**, either from scratch, from a set of design parameters, from systematic series or from parent ships;
- **representation of existing ship forms**, defined already in a body plan, obtained whether in a rough form from the basic design, or well defined from an existing vessel.

The first situation is typical of the basic ship design stage and the second one is necessary when accuracy of hull surface description is required for production, either in ship building or repair.

In the generation of new ship forms the amount of data available is small and the constraints are few allowing a range of solutions. The methods used must be more interactive to allow a trial-and-error approach and to provide the designer with a quick feedback of the operations made.

When representing existing forms, there is much more information available on a body plan, from which data such as boundary lines, knuckles, tangent directions, etc. can be obtained. However, as all this data must be fitted by the final model, it also represents a highly restrictive set of constraints. A system oriented for this purpose has less requirements of interactivity and so some steps of the process can be made more automatic.

The goal of this work is to compare and select mathematical algorithms of curve and surface design, in order to specify the methods and tools required to develop a process for representing the ship hull by means of parametric surface patches.

The thesis is structured as follows:

- **Chapter 2** presents a review of previous work regarding the geometric description of the ship hull. The review covers not only the relevant methods and algorithms proposed but also the dedicated software systems available.
- **Chapter 3** presents a review of the basic definitions and algorithms concerning parametric curves and surfaces. Cubic splines, Bézier, B-splines, Beta-splines, Coons patches and NURBS are the topics covered..
- **Chapter 4** describes the present approach, including the creation of the wireframe model, the generation of surface patches, the tools for curve and surface analysis and the concepts and some preliminary fairing algorithms.
- **Chapter 5** gives some examples of application of the presented algorithms to ship forms and compares the results with those obtained with a commercial package.
- **Chapter 6** summarises the contribution of this work and the conclusions reached, and identifies paths for future research in this area.

2. Representation of Ship Hull Surface

During the four basic stages of the ship design process, the requirements regarding the level of definition of the hull surface and the accuracy required are different [2]:

- **Concept design**, rough definition of main dimensions and hull form coefficients
- **Preliminary design**, confirmation of the main dimensions and development of a surface definition sufficient to allow hydrostatics computations, within an accuracy of about 3% of the displacement
- **Contract design**, improvement of the surface fairing to the accuracy required to produce a ship model for hydrodynamic tests
- **Detail design**, surface definition compatible with the production requirements, that is, with an accuracy of about 3 mm (1/8 inch).

The mathematical representation of the ship hull shape is an important tool at every stage of ship design. The mathematical model can be used not only for almost all the types of theoretical analysis such as stability, hydrostatics, hydrodynamics, resistance and structural, but also to generate information for production. The mathematical formulation must provide the accuracy required for the intended applications.

Previous research work can be found concerning each of the design stages, but most of it has been motivated mainly by three types of problems: the generation of forms from design parameters, the creation of new forms from a parent hull (concept and preliminary design) and the fairing of the hull surface for production (detail design). The work concerning contract design is sometimes left to the responsibility of the hydrodynamic tank office.

One of the first to apply mathematical shapes systematically in ship design was the Swedish naval architect Chapman, who around 1760 mentioned in his book "A

Treatise on Shipbuilding" the use of a family of parabolas for the representation of waterlines and other curves on the hull surface. In the beginning of this century (1915), David W. Taylor used mathematical expressions to represent hull shapes of his systematic series. Sections were represented by parabolas or hyperbolas, depending on the fullness, while fifth order polynomials were used on the waterlines and section area curve.

The introduction of plate cutting machines with automatic control in the early 50's preceded the use of computers in the Naval Architecture related fields. In Tables 2.1 to 2.3, a brief survey of some of the systems and procedures developed until the late 60's is presented, based on a compilation made by Nowacki [3]. In the present work, the attention is focused on the work developed since then to the present time. In the several approaches used during this period, different types of representations, mathematical basis and objectives can be found, sometimes simultaneously. In this work, the methodology used was to divide the works into two groups, the ones based in wireframe models and those using single or multiple surface patches. In each of these groups, the review proceeds by sub-dividing them according to the mathematical basis used, trying to identify the dominant trends. From this methodology it follows that the presentation order does not always match the chronological sequence and so in Tables 2.4 and 2.5 a summary of the approaches reviewed is presented in chronological order. Finally a brief review of some of the more recent and widely used systems running on PC computers is presented.

Table 2.1 Survey of Hull Representation Methods

Author	Institut./ Country	Year	Purpose	Input	Procedure	Function
D. Taylor	US Navy	1915	Creation & syst. variation	Hull parameters	Draught function	Polynom.
Weiblum	Univ. Berlin	1934	Syst. variation	Hull parameters		Polynom.
Benson	U.K	1940	Creation of Lines	Hull parameters		Polynom.
Lackenby	BSRA UK	1950	Syst. variation	Parent Hull	Affine distortion	
Thieme	Univ. Hamburg	1952	Creation	Parameters		Polynom.
Taggart	US	1955	Creation of lines	Hull parameters		Polynom.
Theilheimer & Starkwheather	US Navy	1957 1961	Interpolation and fairing	Offsets	Draught function	Discont. cubics
Rosing & Berghuis	Holland	1959	Fairing	Offsets	Draught function	
Pien	US Navy	1960	Approxim.	Offsets	Sectional method	Polynom.
Kerwin	MIT US	1960	Rough approxim.	Offsets	Sectional method	Legendre polynom.
Martin	NPL UK	1961	Rough approxim.	Offsets of S.A. curve		Chebysh. Polynom.
Lidbro	Sweden	1961	Interpolation	Offsets	Surface fitting	
	Bergens Norway	1961	Fairing	Offsets	Draught function	Polynom.

Table 2.2 Survey on Hull Representation Methods (cont.)

Author	Institut./ Country	Year	Purpose	Input	Procedure	Function
F. Taylor	UK	1962	Interpolation	Waterline offsets		Chebysh. polynom.
Miller & Kuo	Univ. Glasgow	1963	Interpolation	Offsets	Draught function	Polynom.
Berger & Webster	Todd Shipyard US	1963 1966	Fairing	Offsets	Surface fitting	Discont. cubics
Williams	SSET Sweden	1964	Creation of lines	Hull parameters	Draught function	Polynom.
Hamilton & Weiss	MIT US	1964	Creation of lines	Hull parameters	Surface fitting	Surface cubics
Bakker	NSMB Holland	1965	Fairing	Offsets	Sectional method	
Gospodnetie	NRC Canada	1965	Interpolation	Offsets	Sectional method	Elliptic integrals
Corin	US Navy	1966	Fairing	Offsets	Sectional method	Discont. cubics
Tuck & V. Kerkzek	US Navy	1968	Fairing	Offsets	Sectional method	Conform. mapping
Söding	Germany	1966	Creation of lines	Offsets	Section method	Discont. polynom.
Kantorowitz	DSRI Denmark	1967	Interpolation	Offsets	Surface fitting	Orthog. polygon.
Kaiser et al.	Germany	1968	Interpolation	Offsets	Surface fitting	Surface polynom.

Table 2.3 Survey on Hull Representation Methods (cont.)

Author	Institut./ Country	Year	Purpose	Input	Procedure	Function
AUTOKON	Norway		Fairing	Offsets	Section method	Spline polynom.
Hoshino, Kimura, Igarashi	Mitsubishi Japan	1966	Fairing	Offsets	Section method	Discont. cubics
Breitung	Tech.Univ. Berlin	1969	Fairing	Offsets	Surface method	Discont. cubics
Kwik	Univ. Hamburg Germany	1969	Creation of lines	Hull parameters	Section method	Polynom.
Buczkowski	Polland	1969	Fairing & creation	Offsets, parameters	Surface method	
VIKING	Sweden		Interpolation	Offsets	Surface fitting	Splines & conics
Kuiper	NSMB Holland	1970	Creation of lines	Hull parameters	Draught function	Polynom.

2.1 Wireframe representations

The earlier approaches to the design of the ship hull geometry were an attempt to reproduce the traditional process, using transverse sections, waterlines and buttocks to create a wireframe model of the hull shape.

The purpose of many early works was the generation of hull lines from design parameters, for preliminary design. Buczkowsky [4] used polynomials in his variational approach to the generation of ship lines. For Kuiper [5], the input consists on a set of required design parameters (contours, midship section, block coefficient, longitudinal centre of buoyancy, waterline area coefficient and half angle of entrance of the load waterline). The shape of each waterline is defined by a function of a set of

form parameters and of the draught, for which these functions are commonly designated by draught functions. The waterlines were represented by sixth degree polynomials and the accuracy obtained with this process is suitable to preliminary ship design. Kuo [6] presented a technique for generating a fair hull surface from a set of design parameters (main dimensions, block and midship section coefficients, LCB) and control curves (sectional area curve, bow and stern profiles and aft control section) and a tri-dimensional shape coefficient measuring the volume distribution in the vertical direction. The surface is represented by a polynomial equation whose coefficients are computed using functions of volume and volume moments. Another type of polynomial, a combination of cubic polynomials and circular arcs, was used in the New Lines System, a fairing program developed at the Mitsubishi Heavy Industries [7]. Being designed to supply information to Numerically Controlled (N.C.) cutting tools, the system in the final stage, converted the polynomials into equally spaced poly-arcs.

The US Navy developed in the early seventies, a system composed mainly of two modules, HULGEN and HULDEF, for the creation and fairing of ship lines [8]. HULGEN, the Ship Hull Form Generation program, was an interactive system to be used in the concept stage of design, and used design parameters to generate an initial form. In this system each station is obtained from two polynomial equations, one from the keel to the load waterline and the other from there to the main deck edge. To control the surface shape the user acts on eight control curves. Five of those curves represent offsets and the remaining three represent slope distributions. The offsets of the section area and the load waterline, are represented by polynomials of 7th order in ships without parallel body or of 9th order otherwise. The maindeck edge, the flat-of-bottom and the keel profile are represented by cubics with inner linear segments if required. The slopes of the deck edge, load waterline and flat-of-bottom are also represented by cubics with linear segments in between.

HULDEF, the Ship Hull Definition program, tried to solve the difficulties with longitudinal interpolation of the previous systems, oriented mainly to the definition of the surface by transverse frames or stations, by defining the hull shape with

longitudinal lines. To assume that all the stations are intersected by the same set of lines, the longitudinal lines selected were iso-girth lines, instead of the traditional waterlines, buttocks and diagonals. The only tools available for surface fairness assessment were the line plots and the evaluation of the first and second differences on equally spaced ordinates. The latter, called the Difference Method, is based on the assumption that for a smooth set of data, the nth. differences of a nth. order polynomial is constant. The third difference is not used in this method because, due to the piecewise nature of the polynomials used, there is a discontinuity in the third derivative at each data point.

The initial models based on polynomials, due to their dependence on the coordinate system, had several types of problems, such as the impossibility of including straight line segments in a curve or the axis orientation difficulties, which gave origin to the search for alternative models. Conformal mapping and parametric curves were two separate approaches developed.

Conformal mapping

Conformal mapping is the current designation of the one to one correspondence between two points on two distinct planes, expressed by a single analytical function. Normal utilisation consists in mapping free-form shapes into shapes whose equations and properties are known. In the case of ship surface, sections can be, for instance, conformally mapped into the unit circle on the complex plane. One of the first and best known applications of conformal mapping to hull geometry description, was the work of Lewis [9] on converting to general ship sections the results of studies of the water inertia in problems of ship motions obtained with semi-circular section models. The sections were completely defined by the local section area, depth and breadth. The capability of describing an entire section with one equation and the freedom of infinite slope values were the characteristics that brought popularity to the conformal mapping representations over polynomial ones, mainly in the field of hydrodynamics. Reed and Nowacki [10] addressed the problem of the creation of lines using conformal mapping from the unit circle. Due to the difficulties of this method to represent sections with flare, only the underwater part of the hull sections was represented. More information on earlier applications to ship design can be found in

Hoffman and Zielenski [11]. More recently, conformal mapping functions are adopted by Keane [12] in a method that allows the definition of flare and rise of floor in the sections.

Parametric polynomial curves

Parametric polynomial curves were the choice for most of the systems. The NASD, NKK Advanced Ship Design system [13], was oriented to the production stage. The hull was divided in sub-surfaces by boundary lines. The sub-surfaces were classified as plane, cylindrical or curved surfaces. The fairing process was applied to the curved surfaces using iterative cross fairing of three offset tables. These tables contained the half-breadths (y) of the sections at the waterlines, the heights above top of keel (z) of the intersections of the buttocks with the sections and the distances form amidships (x) of the intersection of the buttocks with the waterlines. The system used a type of plane cubic spline curves called APT TABCYL splines. Each curve was faired separately. Curves were first divided in segments and then a spline was fitted to each segment. For the curve fairing, the differences between the coefficients of the cubic terms of adjacent segments were computed. If in two consecutive segments these differences had opposite signs, then the point between the segments had to be corrected, replacing it with the value computed assuming its removal. Even recently, some forms of parametric polynomial curves are still used to model the hull form, as in the work of Tsujita and Eida [14], using a combination of sinusoids and ellipses.

Cubic splines

Meshes of cubic splines curves (refer to Section 3.1) were used by Rabien [15] to obtain new hull shapes from parent ships. The input consisted of a group of plane transverse sections and a set of arbitrary longitudinal lines. Both these types of lines were described by point coordinates and eventually by slopes and curvature conditions.

Bézier curves

The Hamburg Ship Model Basin developed a system [16] based, at first, on Bézier curves (refer to Section 3.2) for the fairing of ship models for tank testing. However,

due to the difficulties found in modelling the round aft extremities of some waterlines, a special type of parametric curve, Cornu's spiral, was developed and incorporated into the system. The main characteristics of these curves is their capability to represent circular arcs and straight lines exactly. Kouh and Chau [17] used rational Bézier curves to define the hull shape in preliminary design. First master curves, cross-sectional curves which are not necessarily planar, are specified, defining the main geometrical features. Next, longitudinal curves are fitted to the points corresponding to iso-parameteric values on the master curves. This mechanism is similar to lofting (refer to Section 4.7.1) although curves are generated instead of a surface. In order to avoid the global behaviour of Bézier curves, the curves used in this approach are composed by cubic segments with controlled boundary conditions.

B-spline curves

Also oriented for the production of ship models was the CAMILL (Computer Aided Milling) system described by Rogers [18]. The representation of the lines could be made using cubic splines, Bézier or B-spline curves (refer to Section 3.4) up to the 6th order. The system was provided with routines for interactive editing and automatic curve approximation using least square methods. As the system did not provide any tools to evaluate the fairness of the created lines, this had to be done by plotting them at a large scale. Nowacki [19] discussed hull form generation from form parameters using cubic B-splines. B-splines were also the basis of the B-LINES module of the BRITDES system [20]. This module was developed to generate new hull forms using interactive graphics. The input was given in the form of the profile, flat-of-side, flat-of-bottom and knuckle lines. Curvature plots were provided to evaluate the fairness of the lines.

Wireframe models, due to their simplicity and limited amount of required data, are quite suitable for the purpose of preliminary design and basic calculations. They cannot, however, provide either the type of information or the accuracy for production purposes obtained from the surface models.

2.2 Surface representations

With the increase of computational power, 3D surface modelling techniques became available in the early seventies and applications to hull representation in the late seventies. The most frequent approach to 3D modelling is a two step process. In the first step, forms are generated by plane curves under the control of a set of space curves, normally boundary lines and knuckles. The advantage here is the more intuitive control of the shape by the designer when dealing with curves, compared to the direct manipulation of surfaces, which leads to a quicker definition of the global shape. In the second step, one or several surfaces are fitted to the curve network. The surfaces generated contain the information required to obtain, without ambiguities, a better knowledge of the quality of the shape.

Coons patches

Coons patches (refer to Section 3.3) became widely used in the first surface approaches to hull representation. The Forward Design System (FDS) [21] uses bicubic patches and cubic splines to fit the boundary curves and to interpolate derivative and twist values at the patch corners. Munchmeyer [22] presented a model based on fifth degree Coons patches generated over a net of fifth degree B-spline curves. First a grid of curves was fitted to the sections and waterlines and then the intersection points were used as the corners of the surface patches. The interesting concept here is that the surfaces interpolate lines instead of just points, in order to keep the fairness of the lines. Stroobant and Mars [23] proposed a mixed approach with Coons patches and third degree B-spline tensor product surfaces, generating control lines approximated by B-spline curves. In Reese [24] the boundary lines are approximated by cubic B-splines and the patches are bicubic or biquintic. Although assuring continuity over the patches, these models required extensive data input from the user, some of which are of difficult determination, such as the cross derivative (twist) values on the corners of the patches. For this reason these systems had poor interactivity and the editing of the surface was difficult.

Algebraic conic surfaces

The generation of hull surfaces based on single curved conical surfaces was the goal of several authors motivated by simplicity of production. The STEERBEAR system [25], was based on a model of tiled parametric surfaces, each defined by two space curves (directrices) and some internal, non-intersecting curves (geratrices). These supporting curves were based on a concept, the designated Kock spline, which defined a type of curve constituted by straight segments and circular arcs, with controlled continuity between segments. Space curves were defined by two projections, i.e., by two planar curves. Lauritsen [26] described a hull form modelling system in which the hull surface was composed of several separate surfaces. First the boundary lines and boundary conditions were defined. The surfaces were then divided in plane, single curved and double curved surfaces. Each single curvature surface was created as a conical surface obtained from generator lines and two Bézier curves.

Bézier surfaces

The Unisurf system [27] was the first system based on the Bézier formulation of curves and surfaces (refer to Section 3.2) and, although designed for the automobile industry, it was also applicable to ship hull design. The main advantage over previous systems was the intuitive shape control obtained by the manipulation of the control polygon. Chaojun [28] divided the hull in 3 to 5 patches longitudinally and 2 or 3 layers vertically, and used bicubic Bézier surfaces. A set of compatibility equations must be solved in order to smooth the connections between adjacent patches. For Pommelet [29] the Bézier patches could be of arbitrary degree. Bézier surfaces, due to the lack of local control and the increase of the degree of the polynomials with the number of control points, have become less popular for use in the representation of ship hulls.

B-spline surfaces

B-splines curves and surfaces (refer to Section 3.4) have been widely used for the representation of ship hull geometry. The properties of B-splines, namely the local control and the possibility of introducing discontinuities by increasing multiplicity in control points have proved to be more suitable for this purpose than the Bézier

formulation. Most of the approaches presented have been based on multi-surface models. The concept of multi-surface patch models emerges from the fact that, despite the great convenience of a single surface model, it is difficult to cope with the co-existence in the ship hull surface of areas of great simplicity and a high degree of complexity (e.g. the bulb area), without losing the simplicity of modelling and processing that were its main advantages in the first place. Another advantage, is the possibility of modelling different areas of the hull separately, which is most useful in practical terms, since it allows the designer to follow the sequence that is more convenient for production requirements. In a surface patch model, care must be taken in order to ensure that patch boundary lines do not cross discontinuity lines.

Fog [30] represented the entire hull by a single tensor product B-spline surface. The surface is of the fourth order and the knot vectors used are non-uniform. Beyer [31] also used the B-spline curve on the design tool named Direct Curve Manipulation, DCM, developed for the interactive modelling of hulls. The curvature distribution along the curve could be displayed to assist the designer who could also select the type of continuity between curve segments (C^0, C^1 or C^2) The general surface modeller system, GENSURF [32] and its customised version for ships, HULLSURF [20], represent a hull surface by B-spline bi-cubic patches defined over boundaries approximated by B-spline curves, both using uniform knot vectors.

Jensen [33] has developed an automatic procedure for generating a single B-spline surface to represent a ship hull surface. The longitudinal contour of the centre-line plane and any knuckle lines are interpolated by a cubic spline fitted to selected points. Then, for each section, a user defined number of control points is obtained by least-square approximation. The grid composed by the section control points is then used to generate a tensor product B-spline surface. Single surface models like this are attractive for the simplicity obtained in having the surface described by a single element and by the reduced amount of data required. Furthermore, the surface is controlled by a external net of control points which allows a more intuitive control of any changes. Standersky [34] combines the interactive capabilities of the B-spline tensor product surface with the variational approach to the shape generation of ship

hulls. The methodology of this work presents three steps. First, a simple surface is generated from a set of basic design parameters such as the volume, centre of buoyancy, sectional curve areas and initial stability. The surface obtained is then edited manually to correct locally undesired shape features. Finally, an automatic distortion procedure is used to correct the volume and centre of buyancy changed during the interactive work.

More recently, Bardis and Vafiadou presented a model [35] that tries to combine the B-spline formulation with the local control of the patch boundaries obtained by concepts borrowed from the Beta-spline formulation. First, longitudinal boundary lines are approximated by B-spline curves interpolating selected points. Then, transverse sections and the longitudinal parametric first derivatives are approximated by B-spline curves fitted to section offset points and longitudinal tangent values, respectively. Finally B-spline surface face patches are generated between each pair of consecutive curves, using first derivative values from the tangent values on the boundaries multiplied by bias functions β_1, similar to those used in Beta-spline formulations.

Rational surfaces

The use of rational surfaces to represent ship shapes is due to the necessity to model exactly conic areas. The first attempts made by Kouh and Söding [36] used rational forms of cubic spline surfaces, fitted using continuity of position and slope at the boundaries which are described by a network of cubic splines, defined by given offsets and boundary conditions. Another form of rational cubic spline, defined in terms of non-negative tension parameters was presented by Clemens [37] applied to the generation and fairing of ship lines. This spline preserves local concavity/convexity and guarantees the continuity of the second derivatives, but only if free-end conditions are applied.

2.3 Summary of reviewed approaches

Coons surfaces were the first choice of most of the early works using surfaces to represent ship hulls. The difficulties in computing the twist vectors at the patch corners and in editing were their main drawbacks. Bézier surfaces became very

popular due to the easy and intuitive way by which the shape can be controlled, but they do not offer local control. B-spline surfaces, by retaining all the characteristics of the Bézier and adding local control, have proved to be a good tool for modelling the ship's hull surface, but in spite of all their advantages, they still present some drawbacks such as the impossibility to represent exactly conic surfaces such as the bilge areas of large ships. This limitation, motivated the utilisation of rational forms of the most used types of parametric surfaces such as cubic spline and Bézier and, more recently, B-spline. The rational form of B-splines, commonly designated by NURBS (Non-Uniform Rational B-Splines) is the state of the art in surface modelling and is already being used by some systems for hull surface modelling (Ref. Section 2.4).

In Table 2.4 the approaches to hull surface representation reviewed are summarised in chronological order and classified by the type of representation, mathematical basis and target design stage.

Table 2.4 Summary of reviewed approaches to hull shape representation

Author/ System	Year	Type of rep.	Obs.	Mathem. Basis	Target design stage	Fairness Eval.	Fairing Tools
Lewis	1929	W		M	P	N	N
Buczkowsky	1969	W		P	C	N	N
Kuiper	1969	W		P	C	N	N
UNISURF	1974	W+S		Be			
Kuo et al	1974	W		P	C	N	N
Reed et al	1974	W		CM	P		
Hattori et al/NKK	1977	W		Cs	D		
Tokumaru	1977	W		P	D		
Nowacki et al	1977	W		Bs	C		
HULGEN	1977	W		P	C		
HULDEF	1977	W		P	P		
Collatz et al.	1977	W		Be	T		
Rogers	1977	W		Cs,Be,Bs	T	N	N
Rabien	1979	W		Cs	P		
Yuille/FDS	1979	S		Cs+CP			
Munchmeyer et al	1979	S		Bs+CP			
Rogers et al/BSSD	1980	S	Mp	Bs			
Stroobant et al	1982	S		CP+Bs			Y
Fog	1984	S	Sp	Bs	P		
B-LINES	1985	W		Bs	C, P	Y	N
Reese	1985	S		Bs+CP			
Lauritsen	1985	S		Be			
Chaojun	1985	S		Be			

Table 2.5 Summary of reviewed approaches to hull shape representation

Author/ System	Year	Type of rep.	Obs.	Mathem. Basis	Target design stage	Fairness Eval.	Fairing Tools
Norskov et al	1985	S		P	P, D		
Pommelet	1985	S		Be			
HULLSURF	1985	W+S		Bs			
Kouh et al	1985	W+S		S			
Beier/DCM	1985	W		Bs	P	Y	
Keane	1987	W		CM	P		
Jensen et al	1988	W+S	Sp	Bs			
Standersky	1988	S	Sp	Bs	C		
Clemens	1991	W+S		S			
Bardis	1992	W+S	Mp	Bs			
Kouh et al	1993	W		Be	P		
Tsujita and Eida	1995	W		P	P		

Type of representation:
- W - Wireframe
- S - Surface
- W+S - Mixed

Observations:
- Sp - Single patch
- Mp - Multi-patch

Target Design Stage:
- C - Conceptual
- P - Preliminary
- T - Contract
- D - Detail

Mathematical basis:
- P - Polynomial functions
- M - Conformall mapping
- Cs - Cubic spline
- Be - Bézier
- Bs - B-spline
- S - other splines
- CP - Coon's patch

2.4 Commercial systems for hull surface design

Although it is not always possible to know details of the mathematical algorithms used in commercial systems, in Tables 2.6 and 2.7 a brief survey of some of the more common hull surface design systems available for PC computers, referring the type of representation and the mathematical basis used.

Table 2.6 Survey on Hull Representation PC Software

Package Name	Company/ Country	Version/ Year	Model Type	Mathematical Basis	Obs.
FastShip	Design Systems & Services, Inc. USA	1989	Surface	B-spline	Deformation of a single surface
MacSurf	Graphic Magic Ltd	Vs. 5.0 1993	Surface	NURBS	Multiple patches (Max. 50)
MAST System4	MAST Systems Ltd	1994			Stability calculations
HullForm	Blue Peter Marine Systems Australia	1994			Stability calculations
PC-SHCP	C. Tremblay & Associates Inc. Canada	Vs. 4.0	Wireframe	Polynomials	Stability calculations
FastYacht	Design Systems & Services, Inc. USA	1993	Surface	NURBS	Multiple patches
ShipHull 2000	NorthStar Software Inc. Canada	Vs. 2.0 1993	Wireframe		Stability calculations

Table 2.7 Survey on Hull Representation PC Software (cont.)

Package Name	Company/ Country	Version/ Year	Model Type	Mathematical Basis	Obs.
ShipCAM	Albacore Research Canada	Vs. 4.2 1994	Surface	B-spline 4th order	
FairLine FL/2B	AeroHydro Inc. USA	Vs. 2.85 1994	Surface		Single Surface
MultiSurf	AeroHydro Inc. USA	Vs. 1.62 1994	Surface	Cubic spline/ B-spline	Multiple patches
AutoShip	AutoShip Systems Corporation	Vs. 5.1 1994	Surface	NURBS	Multiple patches (Max.200)
Cadesnav	ENVC Portugal	1994	Wireframe	Cubic splines	Detail design
Defcar	NYSL Spain	1994	Surface		Detail design
Blines	BMT UK	1993	Wireframe	Cubic B-splines	Basic Design
HullSurf	BMT UK	1993	Surface	B-splines	Multiple patches (Max.5)

It can be seen that most of this systems are already using surface representations and that B-splines are still the preferred mathematical basis. At least three systems use already multiple patches of NURBS surfaces. In general, the available systems are more oriented to preliminary design than to detail design.

3. Parametric curves and surfaces

Space curves can be classified into non-parametric and parametric. Polynomials are the non-parametric curves more used in mathematical representation of shapes, either in explicit or implicit forms. Polynomials have, however, several limitations, such as the impossibility of imposing tangency conditions to vertical lines and the dependency of axis orientation, that make their application to ship hull modelling impractical. Parametric curves are more suitable to represent closed curves or other shapes that assign multiple values to the same value of the independent variable.

As a background to the following chapters, the basic theory of parametric curves and surfaces will be presented in this chapter, focusing on the formulations that, due to particular properties, have been used by the naval architects to describe the shape of ship hulls.

Curves are represented by such basic information as given by a set of points in space. Depending on the way in which the curve fits the points, we can have two types of problems. Given n data points $P_1(x,y,z)$, $P_2(x,y,z)$, ..., $P_n(x,y,z)$, if it is intended to determine a curve that passes through all these points, that is an *interpolation* problem. If it is intended to obtain a curve that approximates the points, although keeping a fair shape, that is an *approximation* problem.

In the following, one interpolating curve, the cubic spline, and one interpolating surface, the Coons patch will be described. The formulation of four approximation curves: Bézier, B-spline, Beta spline and rational B-spline will be presented, together with their respective extensions to surfaces.

The curves will be compared on the basis of the following properties:

- **Linear precision** - is the capability of the curve to reproduce straight line segments.

- **Convex hull property** - each point on the curve lies in a convex hull of a set of control points. The convex hull of a set of points is defined as the set that is formed by all convex combinations of a point set, that is, weighted sums of the type

$$p = \sum_{j=1}^{n} \alpha_j p_j \qquad (3.1)$$

in which p_j are points in a 3D space and α_j are weights, defined in such a way that

$$\sum_{j=1}^{n} \alpha_j = 1 \quad \text{and} \quad \alpha_1, \alpha_2, ..., \alpha_n \geq 0$$

- **Variation diminishing** - is the property of a curve that is not intersected by any straight line more often than the control polygon is.

The basic developments on surfaces in computer aided design were carried out by Coons [38], Bézier [27] and Casteljau. Bézier working at Renault and Casteljau working at Citroen, developed separately curves and surface concepts that although mathematically equivalent, became popular under the designation of Bézier because Calteljau's work, was kept unpublished until 1978. Both developed surfaces consisting of networks of rectangular patches, but while Coons patches exactly interpolate the boundaries, a Bézier surface matches some data and approximates the rest. In Coons patches the blending functions can be chosen very freely (Refer to Section 3.3) while on Bézier surfaces, the blending functions are based on Bernstein approximation.

As for the curves, B-splines surfaces are an extension of Bézier surfaces and Non-Uniform Rational B-spline (NURBS) surfaces are a generalisation of B-spline surfaces.

3.1 Cubic splines

The traditional method of drawing curved lines in naval architecture uses a long thin wooden or plastic strip called a *spline*. The spline shape is controlled through the positioning of lead weights, commonly referred as "ducks" (Fig. 3.1).

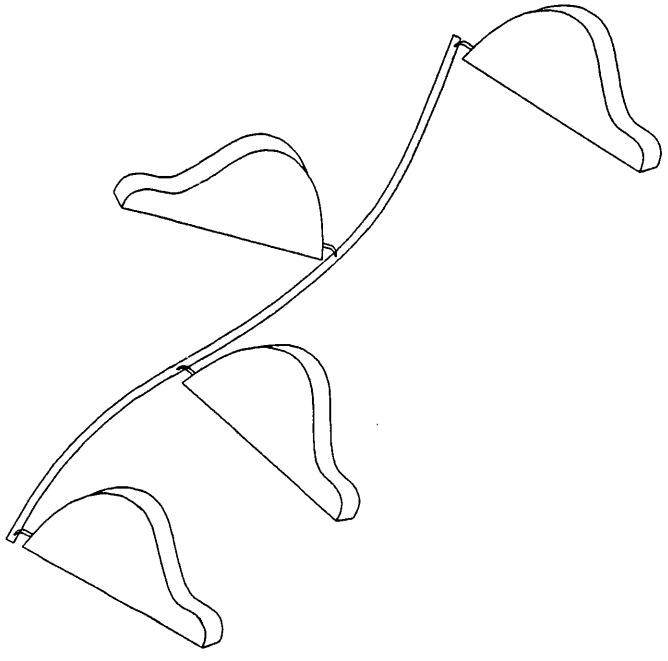

Figure 3.1 Wooden spline with ducks

The mathematical description of the spline curve can be obtained from the theory of the deflection of elastic beams. Considering the spline as a thin uniform elastic beam, and for small deflections, the Euler law relates the deflection of the axis of the beam $y(x)$, with the bending moment $M(x)$ by the expression:

$$y''(x) = \frac{M(x)}{EI} \tag{3.2}$$

in which E is the Young's modulus of the material and I is the moment of inertia of the beam section.

Assuming that the beam is simply supported at the weights, then the bending moment will vary linearly between them, i.e., $M(x) = Ax+B$. After replacing $M(x)$ and integrating twice, we obtain deflection:

$$y(x) = \iint \frac{M(x)}{EI} dx = \frac{1}{\mathrm{EI}} \iint (Ax + B) dx = Ax^3 + Bx^2 + Cx + D$$

The first conclusion that can be derived is that between each pair of adjacent weights (or data points) the beam assumes the form of a cubic polynomial segment. Joining each pair of adjacent segments in such a way that the slope and curvature are equal at their joint, a curve made of cubic segments is obtained, which is called a *cubic spline*. For each segment of the cubic spline, the curve can then be defined by

$$P(t) = At^3 + Bt^2 + Ct + D \tag{3.3}$$

in which t is the parameter normalised to the interval *[0,1]*. To obtain a piecewise curve, the segments must satisfy boundary conditions. The set of boundary conditions established as the basis for the Hermite interpolation is:

$$\begin{aligned} P(0) &= p_0 \\ P(1) &= p_1 \\ P'(0) &= T_0 \\ P'(1) &= T_1 \end{aligned} \tag{3.4}$$

in which p_0, p_1 are the first and last points of the segment and T_0, T_1 are the corresponding tangents to the curve. The derivative of *P(t)* is given by

$$P'(t) = 3At^2 + 2Bt + C \tag{3.5}$$

Replacing (3.5) in (3.4) and computing the constant values, (3.3) can be written, in matrix form, as follows

$$P(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} [H][G]$$

in which *[H]* is

$$[H] = \begin{bmatrix} +2 & -2 & +1 & +1 \\ -3 & +3 & -2 & -1 \\ 0 & 0 & +1 & 0 \\ +1 & 0 & 0 & 0 \end{bmatrix} \tag{3.6}$$

and *[G]* is the *geometric vector*

$$[G] = \begin{bmatrix} p_i \\ p_{i+1} \\ T_i \\ T_{i+1} \end{bmatrix}$$

Generalising, a spline of order k is a piecewise polynomial of degree $k-1$, with continuity of derivatives of order $k-1$ at the segment joints and is defined by [39]:

$$P_k(t) = \sum_{i=1}^{k} C_i t^{i-1} \quad \text{for } t_1 \le t \le t_2 \tag{3.7}$$

where t_1, t_2 are the values of the parameter at the beginning and the end of the segment and C_i are the coefficients determined by specifying k boundary conditions for the spline segment.

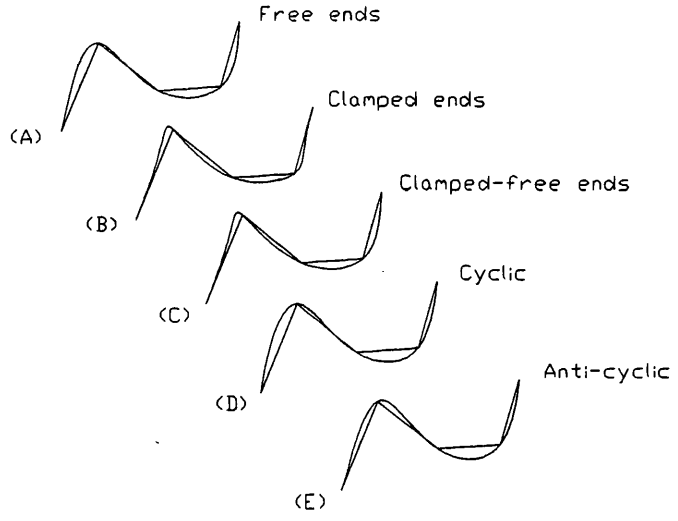

Figure 3.2 Cubic spline with different end conditions

Referring to Fig. 3.2, some common boundary conditions are:

- Natural spline:

$$P''''(t_1) = 0 \qquad P''''(t_n) = 0 \tag{3.8}$$

- Clamped spline:

$$P''(t_1) = D_1 \qquad P''(t_n) = D_n$$
$$\text{given} \quad D_1, D_n \text{ imposed derivatives} \qquad (3.9)$$
$$\text{given} \quad D_1, D_n \text{ imposed derivatives}$$

- Cyclic spline:

$$P'(t_1) = P'(t_n) \qquad P''(t_1) = P''(t_n) \qquad (3.10)$$

- Anti-cyclic spline:

$$P'(t_1) = -P'(t_n) \qquad P''(t_1) = -P''(t_n) \qquad (3.11)$$

The most useful splines are the cubic splines, since 3 is the lowest degree that allows the curve to have an inflection point and also to have non-planar points, that is, to be a true space curve. Cubic splines have C^2 continuity, that is, they are piecewise continuous in position, slope and curvature. They interpolate all the given data points.

The main disadvantages of the cubic splines are their impossibility to exactly represent conic curves and the possibility of having undesirable oscillations.

3.2 Bézier curves and surfaces

The curves generally known as Bézier curves were developed separately by Calsteljau and by Pierre Bézier in the early sixties.

A parametric Bézier curve is defined by [39]:

$$P(t) = \sum_{i=0}^{n} C_i B_{n,i}(t) \qquad for \;\; 0 \le t \le 1 \qquad (3.12)$$

where $B_{n,i}$ are the Bernstein basis functions of degree n. The Bernstein polynomials developed around 1912, are a constructive proof of the theorem proved by Weierstrass in 1885, which stated that any continuous univariate function can be approximated by polynomials up to a given tolerance. The Bernstein polynomials are defined by:

$$B_{n,i} = \frac{n!}{i!(n-i)!}(1-t)^{n-i}t^i$$
$$= \binom{n}{i}(1-t)^{n-i}t^i \qquad \text{for } i = 0,\ 1,\ \ldots,\ n \tag{3.13}$$

where t is the local coordinate of the current curve segment and $b_0, b_1, \ldots, b_n$ are called the Bézier points of the segment and geometrically form the vertices of the Bézier polygon.

Bernstein polynomials have the following properties:

- partition of unity $\sum_{i=0}^{n} B_{n,i}(t) = 1$
- positivity $B_{n,i}(t) \geq 0 \quad for\ t \in [0,1]$
- recursion $B_{n,i} = (1-t)B_{n-1,i}(t) + tB_{n-1,i-1}(t)$

Bézier curves became popular because their shape can be controlled in a very intuitive way by the manipulation of a 3D open control polygon (Fig. 3.3).

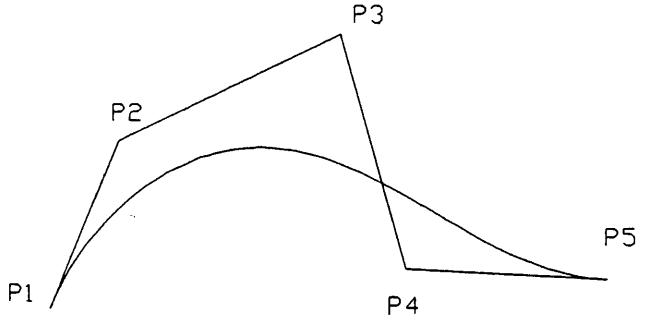

Figure 3.3 Bézier curve

The curve is tangent to the first and last segments of the polygon and interpolates the first and last vertices, but not the interior ones. The order is equal to the number of vertices of the polygon.

A Bézier curve is a smooth curve that is easy to compute and to control through the position of the polygon vertices. The increase of the degree of the polynomials with

the increase of data points and the global nature (if one polygon vertex is changed, all the curve is changed) are the main drawbacks of this family of curves.

A rational form of the Bézier curve is defined by [40]:

$$P(t) = \frac{\sum_{i=0}^{n} \beta_i C_i B_i^n(t)}{\sum_{i=0}^{n} \beta_i B_i^n(t)} \quad \text{for } \beta_i > 0 \tag{3.14}$$

in which β_i are the called *weights* of the Bézier points C_i. Rational Bézier curves are a generalisation of the non-rational form which is obtained if all $\beta_i = 1$. The main advantage of rational curves is their capability of representing conic shapes exactly.

A non-rational Bézier surface is defined by [39]:

$$S(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} C_{i,j} B_{n,i}(u) B_{m,j}(v) \tag{3.15}$$

where $B_{n,i}(u)$ and $B_{m,j}(w)$ are the Bernstein basis functions defined by:

$$B_{n,i} = \frac{n!}{i!(n-i)!} u^i (1-u)^{n-i} \tag{3.16}$$

3.3 Coons surface patches

Several authors have dealt with the problem of interpolating a smooth free-form surface through a topologically rectangular network of curves in 3D space.

The method developed by Steve Coons [38] interpolates a surface patch to fit four given boundary curves that form a continuous closed boundary to the patch. This designation of *blending function* comes from the fact that they *mix* or *blend* the shapes of the boundary curves to produce internal curves that define the surface. The main difference between this approach and the previous ones is that the boundary curves can be of any form. Regarding the blending functions *f(u)*, the only requirements are:

- $f(u) \in [0,1]$
- $\sum_{i=1}^{n} f_i(u) = 1$

Eligible functions include:

- Polynomials
- Step functions
- Piecewise linear functions
- Piecewise polynomials

In its simpler form, a bilinear blending function is used to obtain what is called a linear Coons surface as shown in Fig. 3.4.

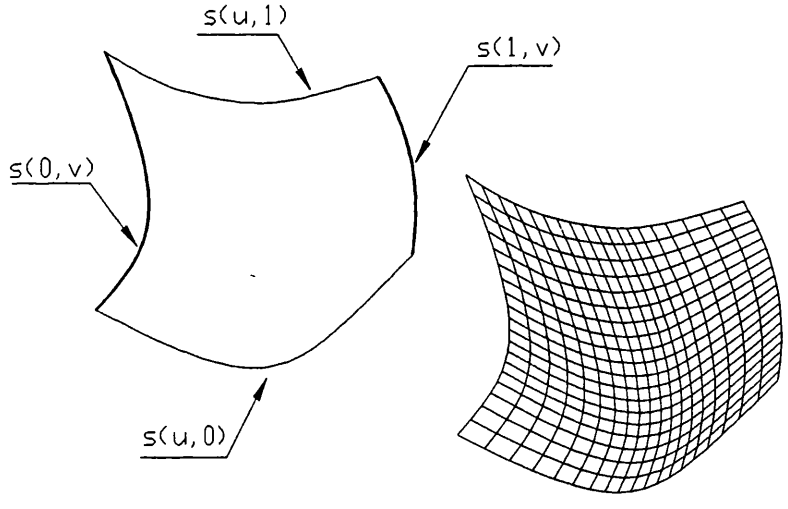

Figure 3.4 Coons bilinear surface patch

If the four boundary curves are *r(u,0)*, *r(u,1)*, *r(0,v)* and *r(1,v)*, the surface is defined by [39]:

$$S(u,v) = [1-u \quad u \quad 1] \left[\begin{array}{cc|c} -r(0,0) & -r(0,1) & r(0,v) \\ -r(1,0) & -r(1,1) & r(1,v) \\ \hline r(u,0) & r(u,1) & 0 \end{array}\right] \begin{bmatrix} 1-v \\ v \\ 1 \end{bmatrix} \tag{3.17}$$

in which the functions *(1-u)*, *u*, *(1-v)* and *v* are the blending functions.

A bi-linearly blended Coons surface is only C^0, that is, it interpolates only the boundaries. A bi-cubically blended Coons surface patch uses cubic splines for its boundary curves and cubic blending functions, and it is able to interpolate both position and tangent information. The bi-cubic Coons patch can be considered as the Boolean sum or blending of three patches, to meet given boundary conditions as illustrated in Fig. 3.5.

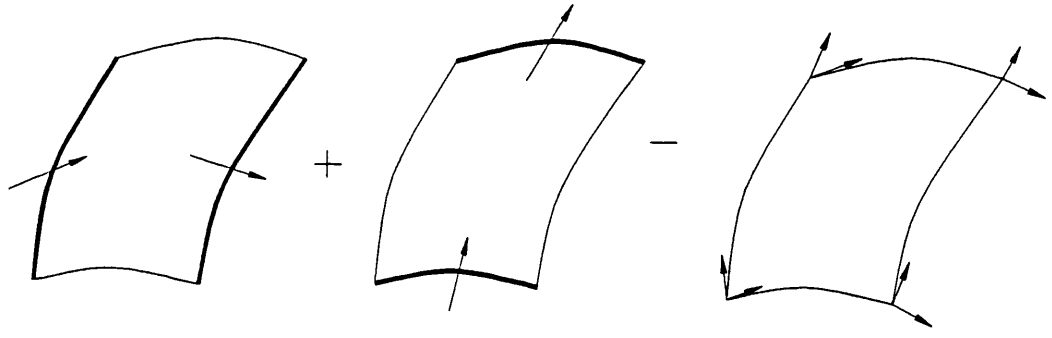

Figure 3.5 Boundary curves and cross tangents on Coons patch

A bi-cubic Coons patch is defined by [39]:

$$S(u,v) = S_1(u,v) + S_2(u,v) - S_3(u,v) \tag{3.18}$$

where

$$S_1(u,v) = [F_1(u) \quad F_2(u) \quad F_3(u) \quad F_4(u)] \begin{Bmatrix} s(0,v) \\ s(1,v) \\ s_u(0,v) \\ s_u(1,v) \end{Bmatrix}$$

$$S_2(u,v) = [F_1(v) \quad F_2(v) \quad F_3(v) \quad F_4(v)] \begin{Bmatrix} s(u,0) \\ s(u,1) \\ s_v(u,0) \\ s_{uv}(u,1) \end{Bmatrix}$$

and

$$S_3(u,v) = [F_1(u) \quad F_2(u) \quad F_3(u) \quad F_4(u)] \left[\begin{array}{cc|cc} s(0,0) & s(0,1) & s_v(0,0) & s_v(0,1) \\ s(1,0) & s(1,1) & s_v(1,0) & s_v(1,1) \\ \hline s_u(0,0) & s_u(0,1) & s_{uv}(0,0) & s_{uv}(0,1) \\ s_u(1,0) & s_u(1,1) & s_{uv}(1,0) & s_{uv}(1,1) \end{array}\right] \begin{bmatrix} F_1(v) \\ F_2(v) \\ F_3(v) \\ F_4(v) \end{bmatrix}$$

where:

$$s = s(u,v)$$

$$s_u = \frac{\partial s}{\partial u} s(u,v)$$

$$s_v = \frac{\partial s}{\partial v} s(u,v)$$

$$s_{uv} = \frac{\partial^2 s}{\partial u \, \partial v} s(u,v)$$

and $F_i(u), F_i(v)$ are the Hermite functions.

As can be seen from equation (3.18), the bi-cubic surface patch is defined by four cubic blending functions $F_i(u)$ and, at each of the four corners, the coordinates s, two derivative vectors, s_u and s_v , and the twist vectors (cross derivatives) s_{uv}. These derivatives and cross derivatives are functions of the parameters u and v and do not represent physical tangents on the surface boundaries.

When trying to generate a surface patch from a rectangular grid of points, one easy way of obtaining the derivatives s_u and s_v is to fit curves (cubic spline, B-spline, or others) to the rows and columns of points. The problem remaining is how to obtain the twist values, s_{uv} at the four corners of the patch. The twist vector is not a geometric invariant because it is a function of the parameterisation, and its value does not depend on the order in which the derivatives are taken, that is,

$$s_{uv} = \frac{\partial^2 s}{\partial u \, \partial v} s(u,v) = \frac{\partial^2 s}{\partial v \, \partial u} s(u,v)$$

The simplest solution seems to be to assign zero values to the surface twists, as suggested in [41]. But, if each boundary curve is not obtainable from the opposite one

by translation, the use of zero twist values will generate flat spots on the surface. Other ways of estimating the twist values can be found in the literature such as Adini's method [42]:

$$s_{uv}(u_i, v_j) = \frac{s_v(u_{i+1}, v_j) - s_v(u_{i-1}, v_j)}{u_{i+1} - u_{i-1}} + \frac{s_u(u_i, v_{j+1}) - s_u(u_i, v_{j-1})}{v_{j+1} - v_{j-1}} - \frac{s(u_{i+1}, v_{j+1}) - s(u_{i-1}, v_{j+1}) - s(u_{i+1}, v_{j-1}) + s(u_{i-1}, v_{j-1})}{(u_{i+1} - u_{i-1})(v_{j+1} - v_{j-1})}$$

Bessel's twist and the Selesnick method [81].

The original Coons patch concept can be adapted to other formulations by changing the nature of the boundary lines or the blending functions, using B-splines, for example [43].

Gordon, working for General Motors, has generalised the notion of a Coons patch to interpolate a full rectangular grid of lines instead of just the external boundaries [44], that is, many patches can be treated simultaneously, is a way similar to Bézier surfaces.

3.4 B-Spline curves and surfaces

Although B-spline curves were already studied in the nineteenth century by N. Lobatchevsky, their utilisation for data approximation and smoothing started in 1946, with Schoenberg, and were first introduced in CAD by J. Ferguson from Boeing Co. in 1963.

The first algorithms to compute the B-spline functions used divided differences. A B-spline of order k was then defined as the k-th divided difference of $g_k(s;t)$

$$M_{i,k}(t) = g_k(t_i,...,t_{i+k};t) \tag{3.19}$$

in which t_i and t_{i+1} are real numbers so that $t_i \le t_{i+1}$ for $i = -\infty,...,+\infty$ and

$$\begin{aligned} g_k(s;t) &= (s-t)_+^{k-1} \\ &= (s-t)^{k-1} \quad if\ s \ge t \\ &= 0 \qquad\quad if\ s < t \end{aligned} \tag{3.20}$$

The normalised spline $N_{i,k}(t)$ is given by

$$N_{i,k}(t) = (t_{i+k} - t_i) M_{i,k}(t)$$

This algorithm, however, is not satisfactory from the computational point of view, since it involves subtraction, an operation which can lead to great lost of accuracy if the difference between the two numbers is small. A detailed error analysis of the divide difference method for computing B-splines can be found in [45].

A recursive algorithm was later independently developed by Cox [45] and de Boor [46], which provides a numerically stable way to compute the spline basis functions. Gordon and Riesenfeld applied these basis functions to curve definition. So, a B-spline curve of order k (Fig. 3.6) is a piecewise polynomial of degree $k-1$ defined by [39]:

$$P_k(t) = \sum_{i=1}^{n+1} C_i . N_{i,k}(t) \tag{3.21}$$

in which C_i are the $n+1$ control points and $N_{i,k}$ are the B-spline basis functions of order k defined by the Cox-de Boor recursive expressions:

$$\begin{aligned} N_{i,1}(t) &= 1 \quad if \quad t_i \le t < t_{i+1} \\ &= 0 \quad otherwise \end{aligned} \tag{3.22}$$

$$N_{i,k}(t) = \frac{t - t_i}{t_{i+k} - t_i} N_{i,k-1}(t) + \frac{t_{i+k} - t}{t_{i+k} - t_{i+1}} N_{i+1,k-1}(t)$$

defined over the knot vector:

$$X = \{t_1, t_2, t_3, \ldots, t_m\} \tag{3.23}$$

The B-spline basis functions have partition of unity and positivity properties.

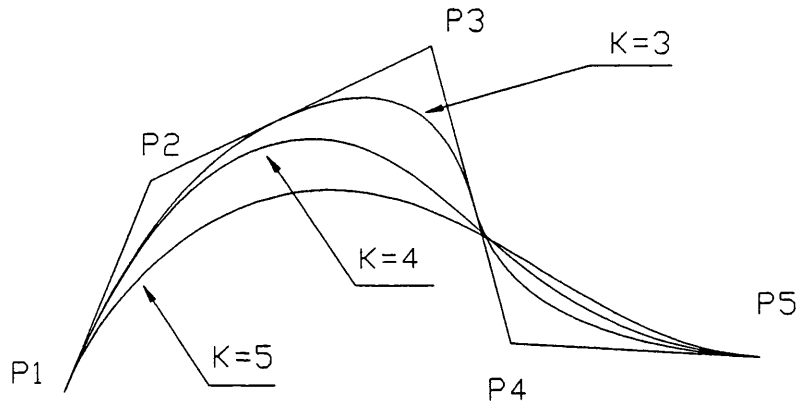

Figure 3.6 B-Spline curves of order 2, 3 and 4

The knot vector must satisfy the two following requirements:

- it must be non-decreasing, that is, $t_i \leq t_{i+1}$
- the knots can be repeated r times, and then they have *multiplicity r*, or be *simple* and have multiplicity one. The multiplicity must not be greater than the order of the spline.

The knot vectors can be classified as:

- *uniform*, if all the knots have an equal spacing, that is, $(t_i - t_{i-1}) = const, \forall i$, for example:

  $\{0.0 \;\; 0.5 \; 1.0 \;\; 1.5 \;\; 2.0\}$

  while if const = 1, it will be *periodic*, for example:

  $\{0 \;\; 1 \;\; 2 \;\; 3 \;\; 4 \;\; 5\}$

- *non-periodic or open uniform*, if it has internal knots equally spaced and knot values with multiplicity equal to the order of the spline at the extremities, for example:

$\{0\ 0\ 0\ 1\ 2\ 3\ 4\ 5\ 5\ 5\}$

- *non-uniform*, if the knots have multiplicity equal to the order of the spline at the extremities and are not necessarily equally spaced, as in the following:

  $\{0.0\ 0.0\ 0.0\ 1.0\ 1.4\ 2.0\ 2.3\ 3.0\ 3.0\ 3.0\}$

  or if it has internal knots with multiplicity greater than one, as for example:

  $\{0\ 0\ 0\ 1\ 2\ 3\ 4\ 4\ 4\ 5\ 6\ 7\ 7\ 7\}$

The knot vectors can be *normalised*, by imposing the condition that the parameter values must be in the interval [0, 1], which does not have any influence on the shape of the curve.

B-spline curves have the following properties:

- Linear precision
- Convex hull property, for no more than k control points.
- Variation diminishing
- Invariant under *affine transformations*

If the end knots don't have multiplicity equal to the order, the first and last control vertices will not be contained by the curve, and are called "phantom vertices".

When the order of a B-spline curve is equal to the number of control points the knot vector consists only of the end points both with multiplicity equal to the order of the spline, as for example:

$\{0\ 0\ 0\ 0\ 1\ 1\ 1\ 1\}$

and produces a degenerated spline basis equivalent to the Bernstein basis, i.e., the B-spline degenerates into a Bézier curve.

A B-spline surface can be defined as the tensor product of two B-spline curves [39]:

$$S(u,v) = \sum_{i=1}^{n+1} \sum_{j=1}^{m+1} C_{i,j} N_{i,k}(u) N_{j,l}(v) \tag{3.24}$$

where $C_{i,j}$ are the control points of the surface and $N_{i,k}$ and $N_{j,l}$ are the B-spline basis functions given by:

$$\begin{aligned} N_{i,k}(u) &= 1 \quad if \quad u_i \leq u < u_{i+1} \\ &= 0 \quad otherwise \\ N_{i,k}(u) &= \frac{u - u_i}{u_{i+k-1} - u_i} N_{i,k-1}(u) + \frac{u_{i+k} - u}{u_{i+k} - u_{i+1}} N_{i+1,k-1}(u) \end{aligned} \tag{3.25}$$

In Figures 3.7 to 3.9 it is possible to see the influence of the knot vector values in the shape of the basis functions. The uniform knot vector generates a basis function of symmetrical shape (Fig. 3.7). A non-uniform vector generates an asymmetric basis function (Fig. 3.8). A repeated value in the knot vector implies a discontinuity in the corresponding basis function (Fig. 3.9). In every case, the influence of each basis function extends for a number of intervals equal to the order of the curve.

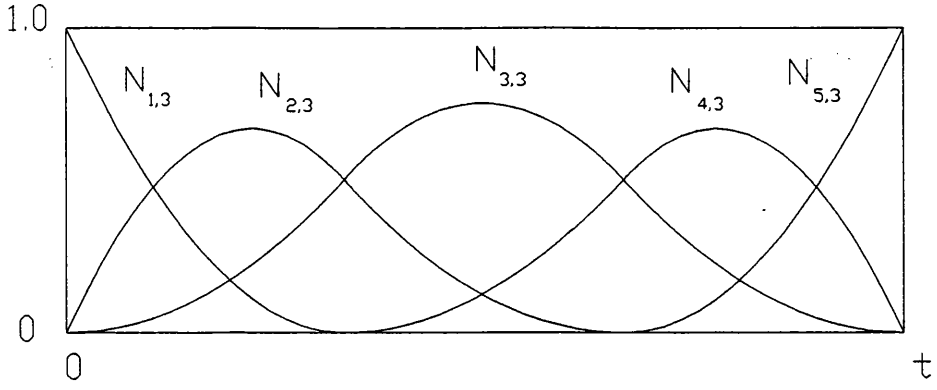

Figure 3.7 Basis functions for $X = \{0,0,0,1,2,3,3,3\}$

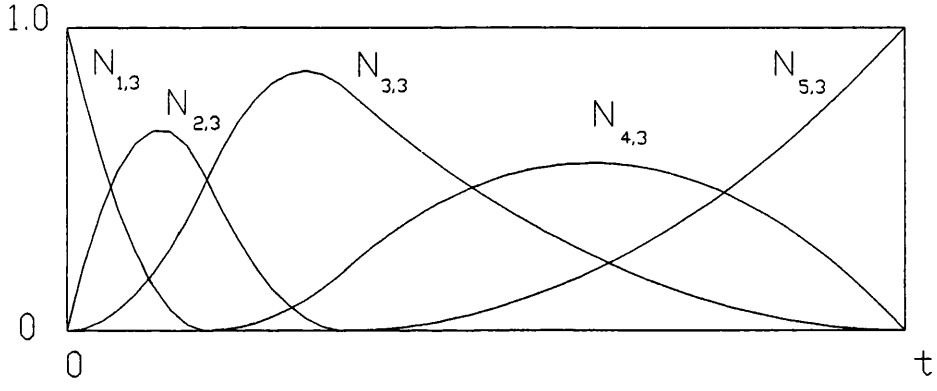

Figure 3.8 Basis functions for $X = \{0,0,0,0.5,1,3,3,3\}$

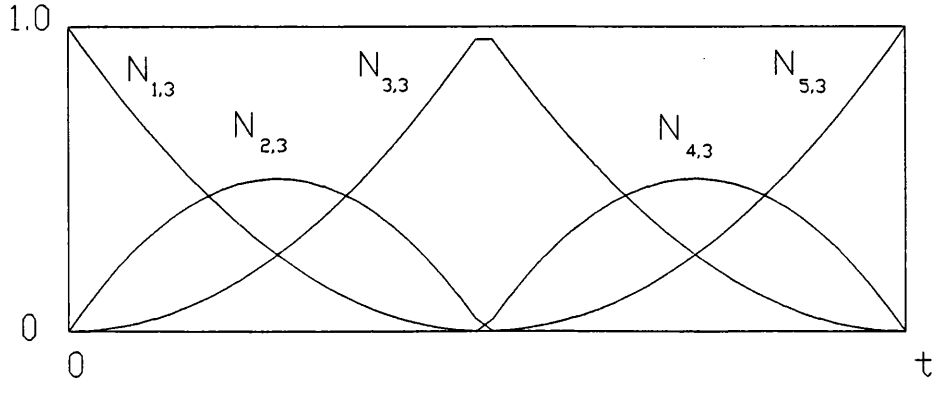

Figure 3.9 Basis functions for $X = \{0,0,0,1.5,1.5,3,3,3\}$

3.5 Beta-splines

One of the advantages of the B-splines is their capability of controlling the degree of continuity at the joints between curve segments without interfering with the order or the number of control vertices.

Cubic Beta-splines were introduced by Barsky, in 1981, as a generalisation of B-splines and are based on the notions of geometric continuity and on the mathematical modelling of tension.

The requirement of second degree *parametric continuity*, C^2, between curve segments in B-splines, is replaced by the requirements of the so called *geometric continuity*, G^2, of the unit tangent and curvature vectors. This introduces constrained discontinuities in the first and second parametric derivatives, which are expressed in terms of the shape parameters β_1 and β_2, called *bias* and *tension*, respectively. A cubic Beta-spline curve is defined by [47]:

$$P_i(u) = \sum_{r=-2}^{1} b_r(\beta_1, \beta_2; u) C_{i+r} \quad for\ 0 \le i < 1 \tag{3.26}$$

in which b_r are the basis functions

$$b_r(\beta_1, \beta_2; u) = \sum_{g=0}^{3} c_{gr}(\beta_1, \beta_2) u^g \quad for\ 0 \le u < 1\ and\ r = -2, -1, 0, 1 \tag{3.27}$$

$\beta_1=1$ indicates continuity of the first parametric derivative and $\beta_1=1$ with $\beta_2=0$ indicates continuity of the first and second parametric derivatives.

Parametric continuity reflects the smoothness of the parameterisation and not necessarily the smoothness of the curve, while the geometric continuity is a measure of the continuity that is not dependent of the type of parameterisation.

The shape parameters provide extra degrees of freedom, allowing the refinement of the shape of the curve without changing the control points. Increasing or decreasing the bias parameter β_1, forces the joint between two segments to move towards or apart from the control point, along a straight line. Increasing the parameter β_2 increases the tension in the curve segment, decreasing tendencies to oscillations and inflection points.

Beta-spline share with B-splines the properties of convex hull and local control.

3.6 Rational B-Spline

Rational curves were introduced in Computer Aided Geometric Design by Coons in 1975. Rational B-Splines were first studied by Versprille, in 1975 [48].

The term "rational" means that the functions are obtained by the "ratio" of two polynomial expressions. In the case of rational B-splines, they are defined by the expression [49]:

$$P(t) = \frac{\sum_{i=1}^{n+1} C_i w_i N_{i,k}(t)}{\sum_{i=1}^{n+1} w_i N_{i,k}(t)} \qquad (3.28)$$

where C_i are the defining polygon vertices, w_i are the weights and $N_{i,k}(t)$ are the B-spline basis functions.

Any point $C_i = [x, y, z]$ in 3D space can be interpreted as the projection from the origin to the hyperplane $w=w_i$, of a point $C_i^w = [wx, wy, wz, w]$ in 4D homogeneous coordinate space. A rational B-spline curve is the projection in 3D space of a polynomial (non-rational) B-spline curve defined in 4D space. If all the control points have $w=1$ then the shape of the rational B-spline curve coincides with the shape of a integral (non-rational) B-spline curve (Fig. 3.10).

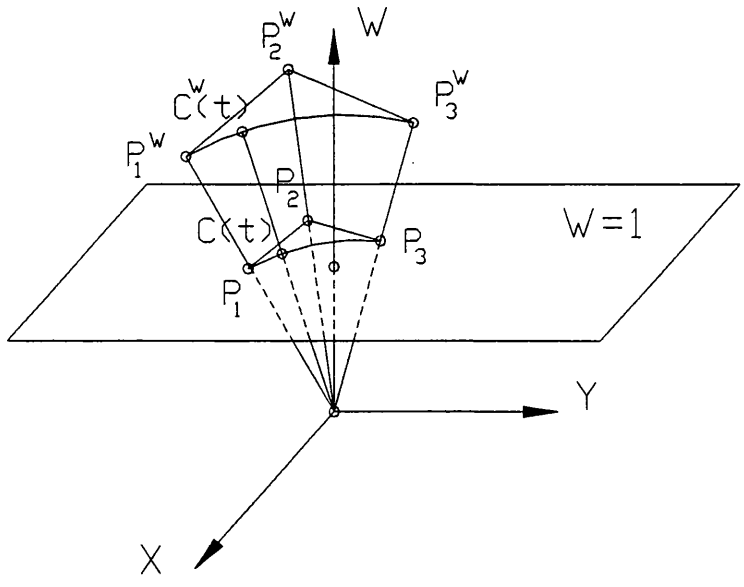

Figure 3.10 Geometric construction of a NURBS curve

Equation (3.25) can be written in the form [49]

$$P(t) = \sum_{i=1}^{n+1} C_i R_{i,k}(t) \tag{3.29}$$

where $R_{i,k}$ are the *rational basis functions*

$$R_{i,k}(t) = \frac{w_i N_{i,k}(t)}{\sum_{j=1}^{n+1} w_i N_{j,k}(t)} \tag{3.30}$$

The non-uniform version of this curve is commonly referred to as NURBS. If all the weights are equal, the curve reduces to a B-spline. Generally, to ensure the non-negativity of the basis functions, it is assumed that:

$$\begin{aligned} &w_1, w_n > 0 \\ &w_i \geq 0 \qquad i = 1,2,.....,n-1 \end{aligned} \tag{3.31}$$

When w_{i-1} and w_{i+1} are fixed, for weight values of $0 < w_i \leq 1$ the curve is pushed away from the control point. Increasing the value of the weight $w_i \geq 1$ has the effect of attracting the curve to the point and for $w_i = \infty$ the curve will interpolate the point. In practice that can be obtained with reasonably high values of w. For $w_i = 0$, the respective control point is ignored as shown in Fig. 3.11.

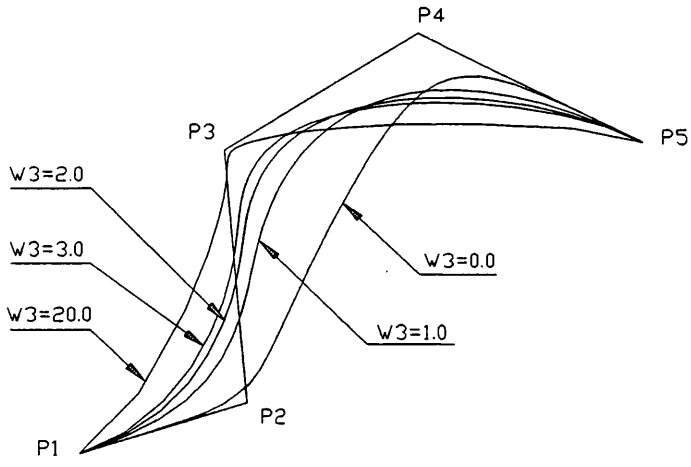

Figure 3.11 Effect of changing the weight of control point 3

A rational B-Spline has all the properties that apply to the non-rational formulation as illustrated in Figs. 3.12 and 3.13. Furthermore, they are *projective invariant*, that is, a projective transformation (parallel or perspective) can be applied to a curve by applying it to its control points, and they can represent exactly conic segments, including circular arcs.

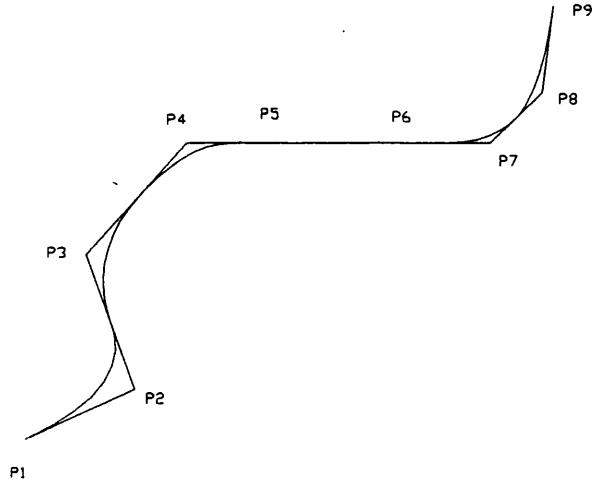

Figure 3.12 Linear precision property of NURBS curve

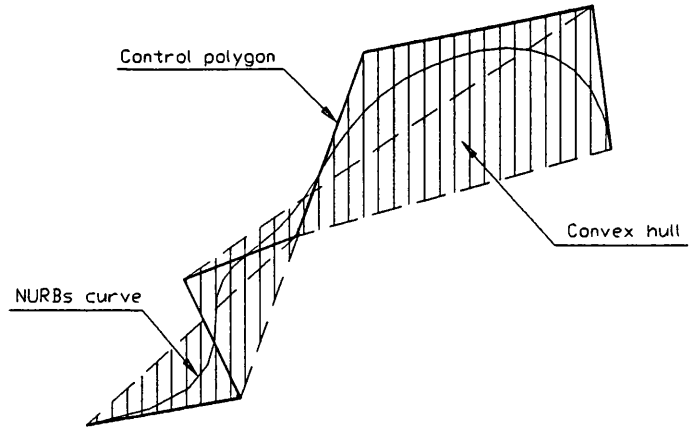

Figure 3.13 Convex hull property in NURBS curve of order 4

The NURBS formulation also has some drawbacks, such as:

- Requires more storage to define standard curves and surfaces, like circles or ellipses
- Some characteristic dimensions, like the radius of an arc or a circle, are not explicit
- Improper alterations of the weight can lead to unpredictable alterations of the shape

The derivatives of a rational B-spline curve can be obtained by differentiation of equation (3.28)

$$P'(t) = \sum_{i=1}^{n+1} C_i R'_{i,k}(t)$$

where $R'_{i,k}$ are the derivatives of the rational basis functions given by:

$$R'_{i,k}(t) = \frac{w_i N'_{i,k}(t)}{\sum_{j=1}^{n+1} w_i N_{j,k}(t)} - \frac{w_i N_{i,k}(t) \sum_{j=1}^{n+1} w_i N'_{j,k}(t)}{\left(\sum_{j=1}^{n+1} w_i N_{j,k}(t)\right)^2}$$

The first derivatives of the non-rational basis functions are given by

$$N'_{i,k}(t) = \frac{N_{i,k-1}(t) + (t - t_i)N'_{i,k-1}(t)}{t_{i+k-1} - t_i} + \frac{(t_{i+k} - t)N'_{i+1,k-1}(t) - N_{i+1,k-1}(t)}{t_{i+k} - t_{i+1}} \tag{3.32}$$

and, in general, the sth derivative of $N_{i,k}$ can be obtained from

$$N^s_{i,k}(t) = k\left(\frac{N^{s-1}_{i,k-1}}{t_{i+k} - t_i} - \frac{N^{s-1}_{i+1,k-1}}{t_{i+k+1} - t_{i+1}}\right) \tag{3.33}$$

As mentioned earlier, one of the advantages of NURBS curves is the capability to exactly represent conical curves, which is only possible by means of approximation, when dealing with other formulations.

Considering a curve of degree two with 3 control points as shown in Fig. 3.14, if it represents a conic, then, although the weights w_1, w_2 and w_3 can have different values, the ratio

$$k_c = \frac{w_1 . w_3}{4 . w_2^2} \tag{3.34}$$

called the *conic shape invariance*, remains constant for each type of conic, as follows:

$4k_c < 1.0 \quad \Rightarrow \quad ellipse$
$4k_c = 1.0 \quad \Rightarrow \quad parabola$
$4k_c > 1.0 \quad \Rightarrow \quad hyperbola$

For circular arcs, not only the condition $k_c > 1$ must be fulfilled, but also the triangle $\Delta[P_1P_2P_3]$ must be isosceles. The radius of the arc obtained is given by

$$R = \frac{\sqrt{(1 + 4b^2)}}{4b}$$

in which

$$b = \frac{\sqrt{k^2 - 1}}{2}$$

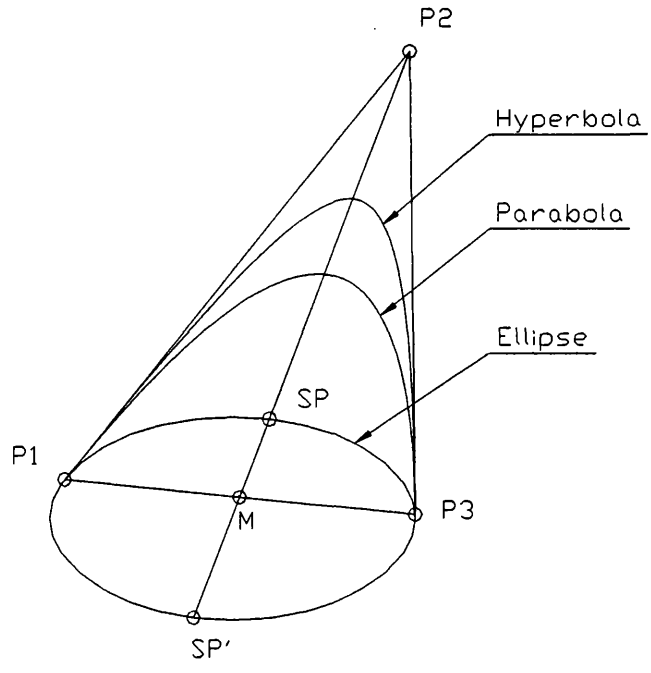

Figure 3.14 Representation of conics by NURBS curves

Full circles can be obtained by patching together circular arcs [49]. Using four 90° arcs, the circle is defined by nine control points as shown in Fig. 3.15.

$$X = \{0,0,0,0.25,0.25,0.5,0.5,0.75,0.75,1.0,1.0,1.0\}$$
$$W = \{1.0, \sqrt{2}/2, 1.0, \sqrt{2}/2, 1.0, \sqrt{2}/2, 1.0, \sqrt{2}/2, 1.0\}$$
$$P = \{P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_9\}$$

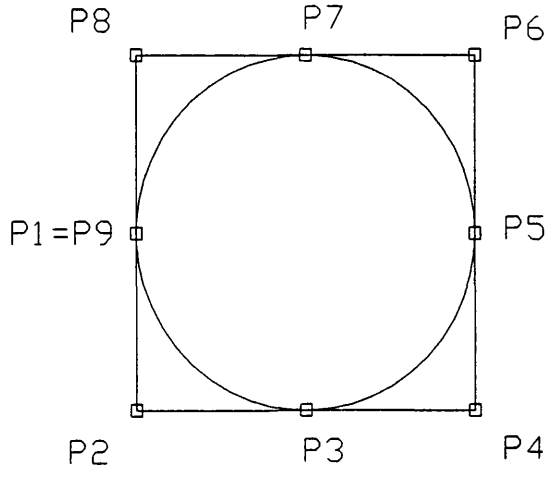

Figure 3.15 Circle represented by a 9 control point NURBS curve

This representation can be simplified by removing the repeated $\frac{1}{4}$ *and* $\frac{3}{4}$ knot values, resulting in a circle generated with just seven control points (Fig. 3.16) defined by

$$X = \{0,0,0,0.25,0.5,0.5,0.75,1.0,1.0,1.0\}$$
$$W = \{1.0,0.5,0.5,1.0,0.5,0.5,1.0\}$$
$$P = \{P_1, P_2, P_3, P_4, P_5, P_6, P_7\}$$

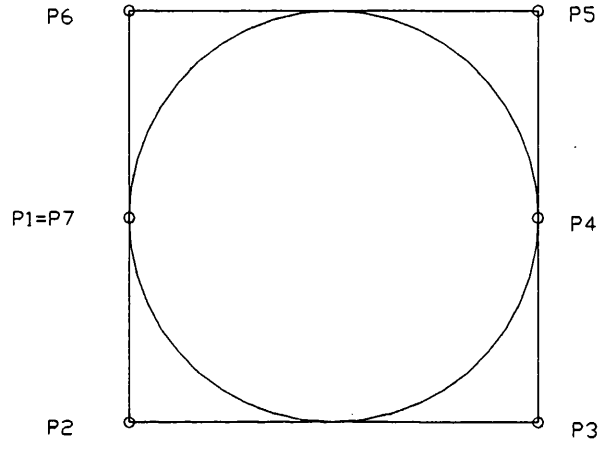

Figure 3.16 Circle represented by a 7 control point NURBs curve

Circles can also be obtained by patching three 120° arcs as shown in Fig. 3.17

$$X = \left\{0,0,0,\frac{1}{3},\frac{1}{3},\frac{2}{3},\frac{2}{3},1.0,1.0,1.0\right\}$$
$$W = \{1.0,0.5,1,0.5,1.0,0.5,1.0\}$$
$$P = \{P_1, P_2, P_3, P_4, P_5, P_6\}$$

In general the quality of the parameterisation increases with the number of arcs used so, in the present work, the circle representation using 9 control points will be used, in order to obtain better results when using circles as support curves for the generation of cylindrical surface patches.

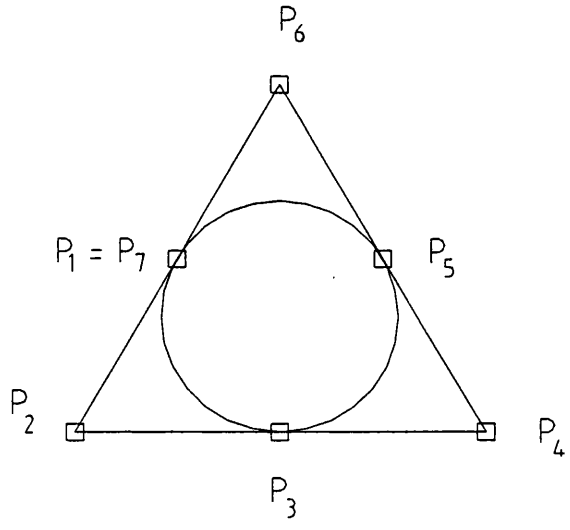

Figure 3.17 Circle represented by a 7 control point NURBs curve

The NURBS representation of a full ellipse can be obtained by applying an affine transformation to a circle, as the one represented by seven control points, keeping the weight distribution and the knot vector as shown in Fig. 3.18.

$$X = \{0.0,0.0,0.0,0.25,0.5,0.5,0.75,1.0,1.0,1.0\}$$
$$W = \{1.0,0.5,0.5,1.0,0.5,0.5,1.0\}$$
$$P = \{P_1, P_2, P_3, P_4, P_5, P_6, P_7\}$$

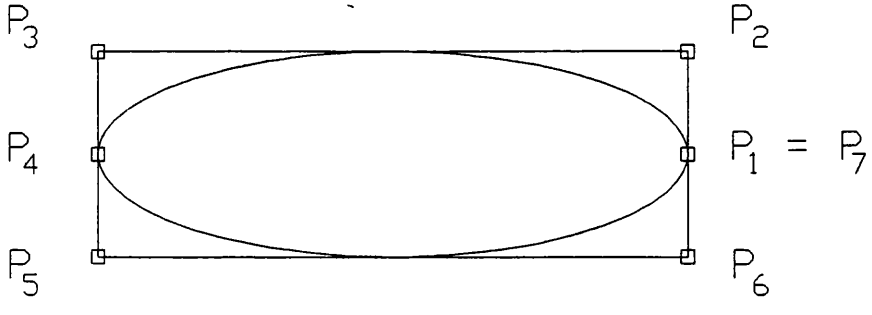

Figure 3.18 Ellipse represented by NURBS curve

A rational B-spline surface in four dimensional homogeneous coordinate space is given by the Cartesian product defined by [50]:

4. Hull Representation by NURBS Surface Patches

In the present method of mathematically modelling the ship hull surface, the first step is to create a wireframe model, following the traditional approach in design offices. The ship lines used are the sections, bow and stern profile contours, flat-of-bottom, flat-of-side, deck contour and knuckle lines, if any.

The description of ship lines needs a type of curve capable of representing free-form curve segments as well as straight lines, conic curves and even circular arcs (Fig. 4.1).

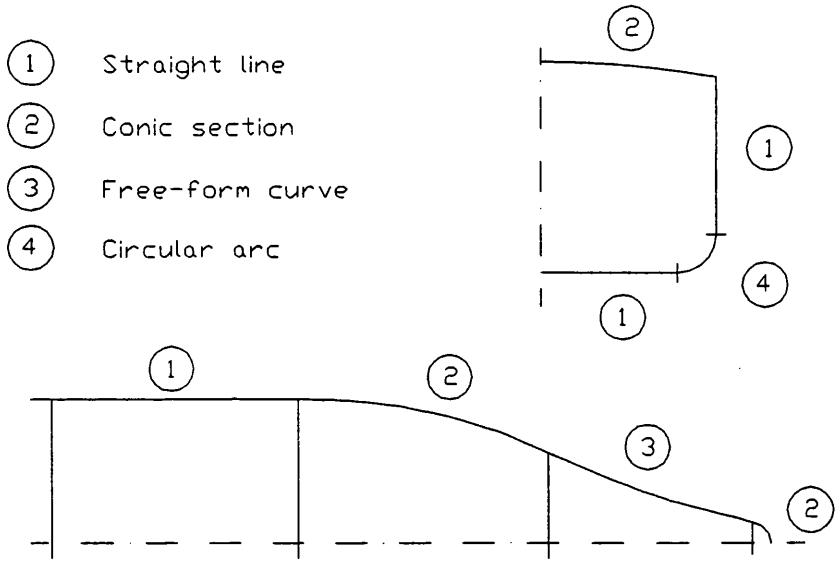

Figure 4.1 Types of curves used on ship lines [36]

For the sake of model simplicity, it is desirable to use a curve formulation that not only can represent accurately all the types of lines mentioned, but that may also be extended to surface generation.

The type of formulation selected for the representation of curves and surfaces selected in the present work is the Non-uniform Rational B-spline (NURBS).

The potential benefits of the extra control obtained in NURBS curves and surfaces are still a topic of research and so, in the present work, only the capability of representing conics exactly will be used. However, being a superset of Bézier and B-spline formulations, all the existing algorithms can, in principle, be adapted. In general, Bézier curves and surfaces can be converted into B-splines by knot insertion and B-spline algorithms can be converted by generalising point representation to four homogeneous coordinates *(x,y,z,w)* and using unit weights.

During the process of creation of the wireframe model, some specific lines of the body plan will be approximated by B-spline curves. The procedure to generate the curves will have the following steps:

- Manual or digitised input of points defining the selected curve
- Filtering of input points, if digitised
- Automatic curve fitting or approximation of the data points
- Editing of the curve for fine tuning of the shape, introduction of knuckles, etc..

The wireframe model must provide the support and boundary lines required to generate the surface patches. The transverse sections, the stem and stern contour will be used as the main support lines. The boundary lines normally defined in the body plans, are the flat-of bottom, the flat-of-side, the deck contour and the knuckle lines, if any (Fig. 4.2).

In general, the surface patches will be generated by lofting, although other techniques like ruling, extrusion and blending can also be used to deal with particular cases.

The algorithms and methods used to create the wireframe model, to generate and to interrogate the surface patches are described and discussed in the following Sections.

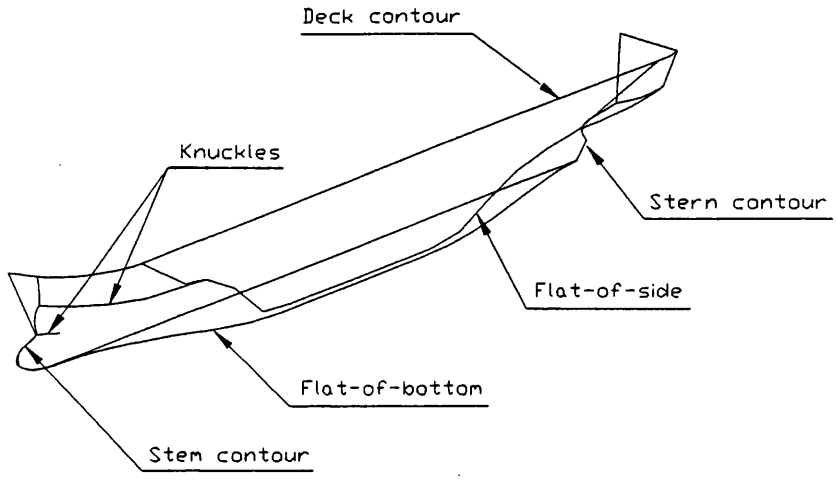

Figure 4.2 Boundary lines

4.1 Filtering digitised input points

The quality of digitised input points depends very much on the practice of the operator. Sometimes the information contained in the points is redundant, that is, too many points are picked and, in some cases, like straight line segments for instance, the probability is that the excess points are not collinear, introducing erroneous information. In general, the excess of information should be avoided because:

- it increases the size of the matrices to be inverted during the curve fitting and approximation processes (refer to Section 4.2), which can cause numerical problems
- it increases the size of the control grid of the surfaces generated, which increases storage requirements
- as in general the fairing of lines and surfaces is associated with the removal of redundant information, filtering input data will save time in the fairing processes

For the filtering of digitised input lines a simple filtering method was developed. The criterion used is that the new polygonal line defined by the output points should never be at a distance superior to a given tolerance δ from the polygonal of the input points. For each point P_i, a 3D straight line segment $r(x,y,z)$ is defined by the points $\overline{P_i P_{i+j}}$, with j=1,2,..n

$$r(x,y,z) \equiv \frac{x-x_i}{u} + \frac{y-y_i}{v} + \frac{z-z_i}{w}$$

in which u,v,w are the components of the unit vector $\vec{L}$

$$\vec{L} = \frac{\left(P_{i+j} - P_i\right)}{\left|\left(P_{i+j} - P_i\right)\right|}$$

The distance from the point P_{i+k}, with k=1,2,..j to r is computed by

$$d = \frac{\left|\vec{L} \times \left(P_{i+k} - P_i\right)\right|}{\left|\vec{L}\right|}$$

If $d < \delta$ for a given tolerance $\delta > 0$, then the point P_{i+1} can be removed, a new line segment is defined by the points $\overline{P}_i\overline{P}_{i+2}$ and the process is repeated until a point is not removed or the last point is reached. When any of these things happen, the starting point is incremented, $i=i+1$ and the process continues until all the points are tested.

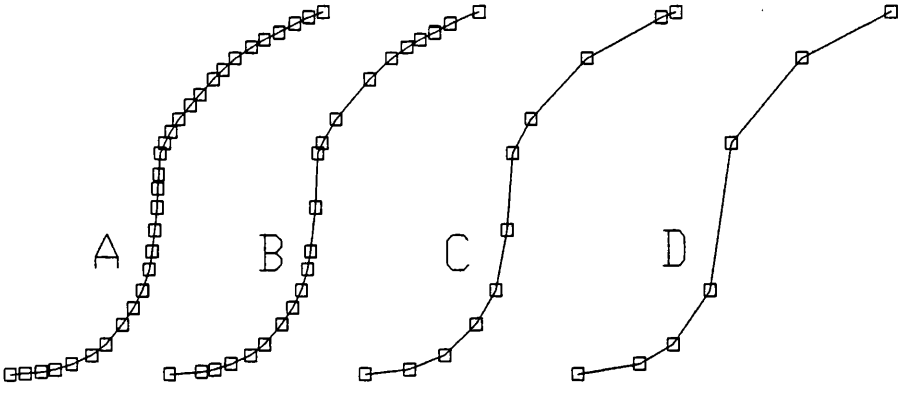

Figure 4.3 Filtering of polygonal line with several tolerance values

In (Fig. 4.3) the polygonal line (A) with 31 points, was reduced to 22 points when filtered to $\delta < 0.01$(B), to 11 points with $\delta < 0.05$ (C) and to 7 points with $\delta < 0.10$(D).

4.2 Interpolation and approximation of ship lines by B-spline curves

Due to their importance in several phases of the process, the curve interpolation and approximation algorithms used in the next Sections will be detailed and compared.

The general curve fitting problem can be described as follows: given a set of ordered data points, find the control points and the knot vector that generate a B-spline curve of order k that contains (interpolates), or passes near (approximates), those points.

The relation between the curve data points $[D]$ and the control points $[B]$ is, in matrix form,

$$[D]=[N][B] \tag{4.1}$$

in which $[N]$ are the spline basis functions (3.20). If the number of data points is equal to the number of control points required, then the matrix $[N]$ is square and so, the problem has a straightforward solution

$$[B]=[N]^{-1}[D] \quad \text{for } 2 \le k \le npts \tag{4.2}$$

Since the basis functions are defined over a knot vector, and computed for a defined set of parameter values t_j, the problem of curve fitting solving (4.2) reduces to:

- define a knot vector
- define the parameter value to be assigned to each data point.

4.2.1 Parameterisation

The knot vector and so the type of parameterisation has a big influence on the shape of the curve obtained. The uniform parameterisation has the disadvantage of not having a geometric measure, i.e., it does not have any relation to the actual location of the control points and so, if these have an uneven spacing the results obtained are very poor.

The most common parameterisation is the *normalised chord length* between control points, but as they are not yet available at this stage, data points will be used as a first approximation.

$$t_1 = 0$$
$$t_i = \frac{\sum_{j=2}^{i} |P_j - P_{j-1}|}{\sum_{j=2}^{j=n} |P_j - P_{j-1}|} \quad i = 2,3,\ldots,n \tag{4.3}$$

The *centripetal* parameterisation [51], obtained by (4.4) has proven to give better results then the chord-length, for $p=0.5$.

$$t_1 = 0$$
$$t_i = \frac{\sum_{j=2}^{i} |P_j - P_{j-1}|^p}{\sum_{j=2}^{j=n} |P_j - P_{j-1}|^p} \quad i = 2,3,\ldots,n \tag{4.4}$$

The influence of the parameterisation on the curve shape is much more important on curve fitting than when just generating curves with given control points.

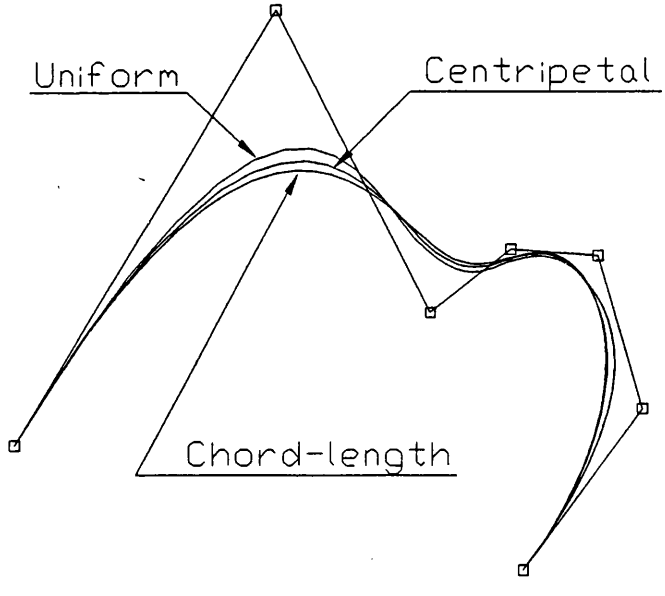

Figure 4.4 Influence of type of parameterisation on curve shape

In (Fig. 4.4) several curves with the same order and same control points, but with different types of parameterisation, can be seen.

In general, all the control points computed in interpolation and approximation processes will have, initially, equal weight values assigned (namely 1.0), that is, the rational curve obtained will be, in fact, equivalent to a non-rational B-spline curve.

4.2.2 Curve interpolation

To obtain the parameter values corresponding to the control points, one method is to use the notion of *nodes* ζ_i of the knot vector x_j. The nodes, also called *Greville abscissas* are an averaged summation of the knot values defined by [52],

$$\zeta_i = \frac{1}{k-1}\sum_{j=i+1}^{i+k-1} x_j \qquad i = 1,2,3,\ldots n \tag{4.5}$$

Assuming the nodes as the values of the parameter corresponding to the control points B_i, then,

$$P(\zeta_i) = B_i \qquad i = 1,2,3,\ldots,n \tag{4.6}$$

If the curve interpolates the data points $D_{i,3}$ at the same parameter values, the following equation is obtained:

$$[G][B] = [D] \tag{4.7}$$

in which $G_{i,j} = N_{j,k}(\zeta)$ $0 \le i, j \le n$ is called the *Gram matrix*. The control points are obtained by solving equation (4.7). This interpolation method will be referred in the following as *Method I*. In (Fig. 4.5) a set of 12 points fitted by this method is shown.

Instead of obtaining the parameters from the knot vector as before, another approach is to define first the parameters, by the chord-length or centripetal methods described before, and from these to generate a knot vector. The internal values can be obtained by an averaging process [53] as follows

$$u_i = \frac{1}{k-1}\sum_{k-1}^{i+k-2} t_j \qquad i = 1,2,\ldots,n-k$$

the knot will then be

$$X = \{0^1,\ldots,0^k, u_1,\ldots,u_{n-k},1^1,\ldots,1^k\}$$

The control points are obtained by solving the system (4.7). This interpolation will be referred to in the following as Method II. In (Fig. 4.6) the same set of data points used in (Fig. 4.5) is fitted using this method.

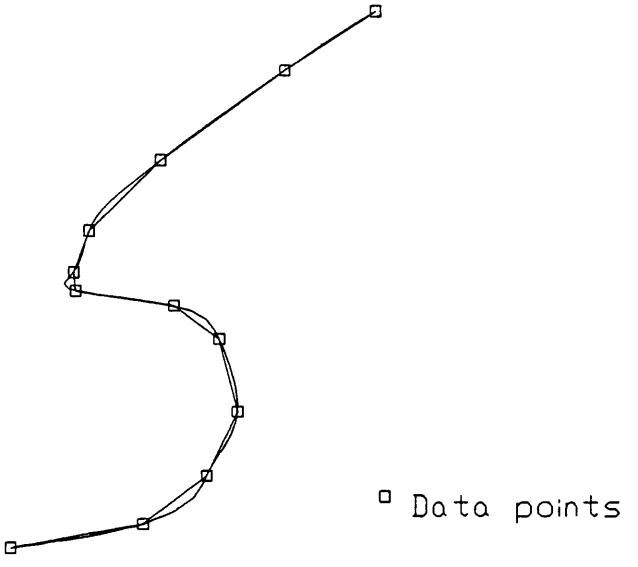

Figure 4.5 B-spline fitted to data points using the nodes (Method I)

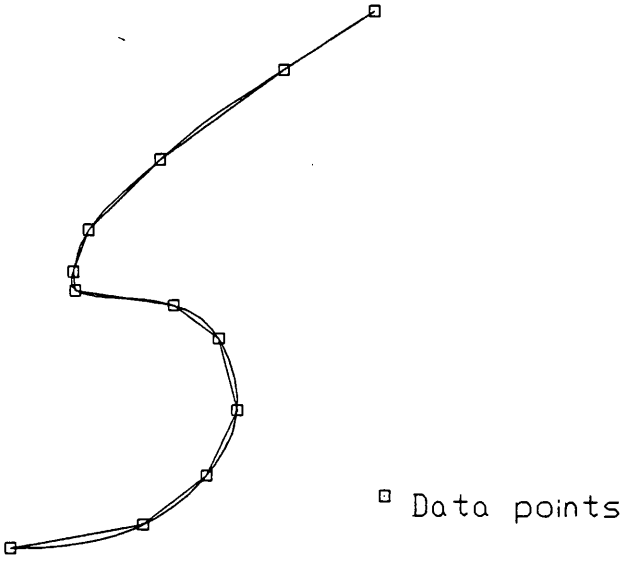

Figure 4.6 B-spline fitted by the Least Square (Method II)

Using interpolation methods, the number of control points obtained is equal to the number of interpolated data points which sometimes becomes a limitation, when there are too many data points resulting, for example, from a digitising process. To have fewer control points, it is necessary to obtain a sparser knot vector or, to use an uniform knot vector, which does not depend on the data points or control points. One technique is to perform a least square approximation as described by Rogers and Fog [54].

The above curve fitting methods are global, i.e. they use all the data points to obtain the interpolating curve. Piegl [55] presented a local algorithm that builds a piecewise curve with cubic Bézier segments, each defined by 4 consecutive control points, and imposes tangent values at the joints (Fig. 4.7).

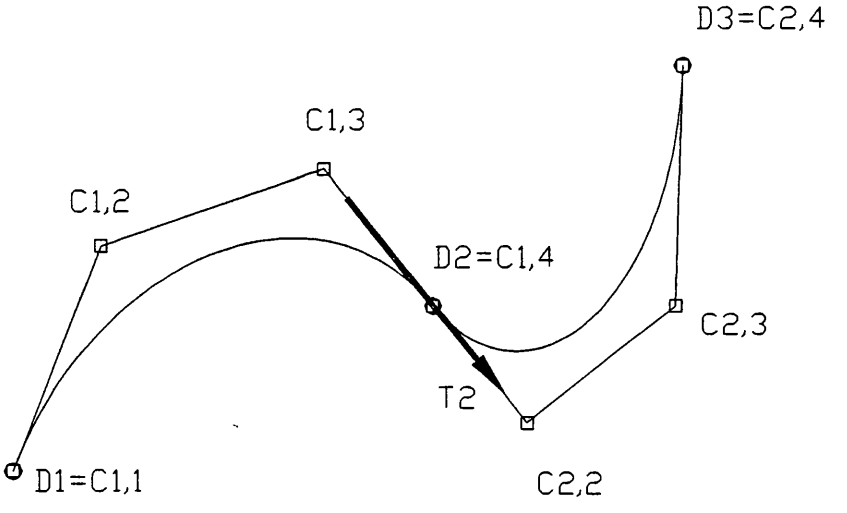

Figure 4.7 Two adjacent Bézier cubic segments

The tangents at the joints T_i are computed according to Akima [56] and are given by

$$T_i = (1-\alpha)v_i + \alpha\, v_{i+1} \qquad i = 2,\ldots,n-1$$

in which v_i are the vectors defined by the data points

$$v_i = D_i - D_{i-1}$$

and α is a parameter defined by

$$\alpha = \frac{|v_{i-1} \times v_i|}{|v_{i-1} \times v_i| + |v_{i+1} \times v_{i+2}|}$$

In the extremities where the vectors v_i cannot be computed, the following are assumed, as suggested by Piegl [57]

$$\begin{aligned} v_1 &= 2v_2 - v_1 \\ v_0 &= 2v_1 - v_2 \\ v_n &= 2v_{n-1} - v_{n-2} \\ v_{n+1} &= 2v_{n+2} - v_{n+1} \end{aligned}$$

For each Bézier segment $[u_i, u_{i+1}]$ the two extreme control points coincide with the data points

$$\begin{aligned} C_{i,1} &= D_i \\ C_{i,4} &= D_{i+1} \end{aligned}$$

and the two internal control points are

$$\begin{aligned} C_{i,2} &= C_{i,1} + \gamma T_i \\ C_{i,3} &= C_{i,4} - \gamma T_{i+1} \end{aligned}$$

with

$$\gamma = \frac{u_{i+1} - u_i}{3}$$

To convert the Bézier segments into NURBS representation, the control points will be

$$C_{1,1}, C_{1,2}, C_{1,3}, C_{2,2}, \ldots, C_{n-1,2}, C_{n-1,3}, C_{n-1,4}$$

and the knot vector

$$X = \{0,0,0,0, u_1, u_1, u_1, u_1, \ldots, u_{n-2}, u_{n-2}, u_{n-2}, u_{n-2}, 1,1,1,1\}$$

As mentioned in chapter 3, the control of the shape of a Bézier curve is very intuitive, but one of the drawbacks is that the order of the curve generated is equal to the number of control points. This feature implies that, for the number of points assumed

reasonable to describe, for instance, a ship section, the order of the curve obtained is so high that the curve becomes difficult to control. This algorithm, that will be referred in the following as Method III, manages to provide a good shape control keeping the order low (Fig. 4.8). Its main inconveniences are that the resulting curve is only C^1 continuous and that the number of control points generated, $2(n-2)+3$, is greater than the number of data points, which increases not only the storage requirements but also the complexity of any surfaces generated with the curve.

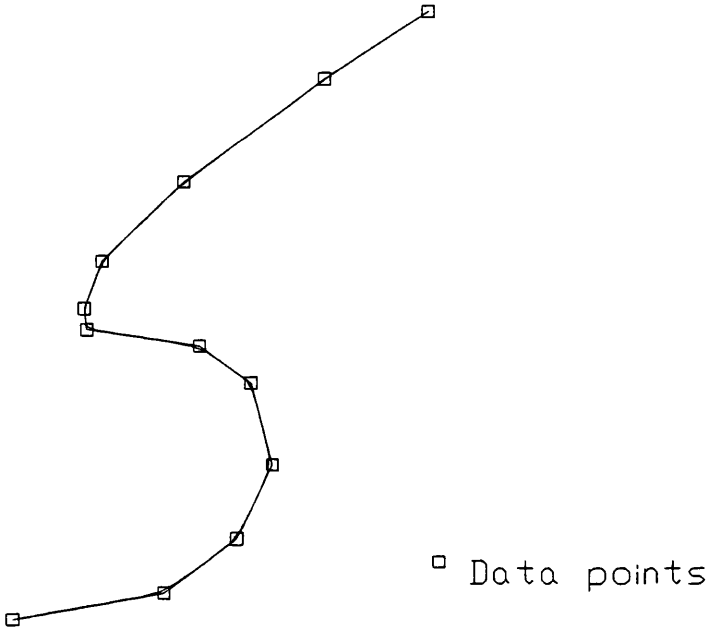

Figure 4.8 B-spline fitted with piecewise cubic Bézier segments (Method III)

In order to compare the efficiency of these three interpolation methods described, in practical conditions they were applied to a set of data points, with a shape typical of the mid-ship section of a large tanker (flat bottom and side, round bilge). The distribution of the points is similar to the one obtained digitising a drawing - larger point spacing on regular or straight zones and smaller spacing on curved zones. The three methods were compared using the same order (3) and the same type of parameterisation - centripetal - which generally leads to best results.

It is possible to see (Fig. 4.9) that Method I generates a curve that, although interpolating the data points, has unexpected behaviour in the transition zones between the straight lines and the circular arc (P_2-P_3 and P_7-P_8).

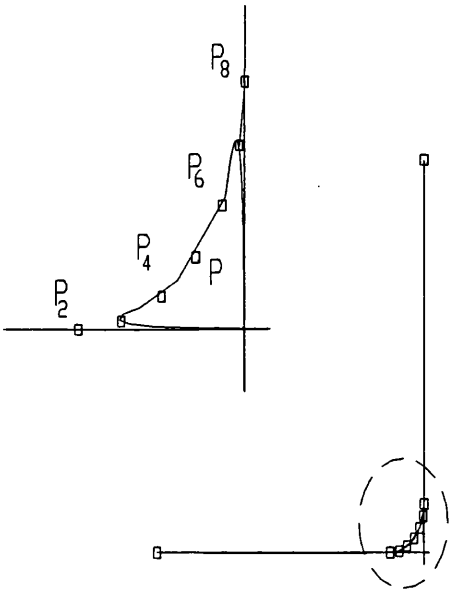

Figure 4.9 Midship section interpolated with Method I

The curve obtained from Method II has much better behaviour although it is possible to detect on the curvature porcupine representation (Fig. 4.10) that the transition zones still have irregularities.

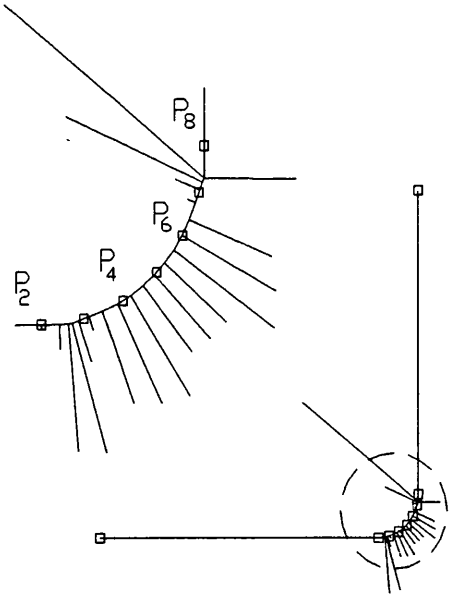

Figure 4.10 Midship section interpolated with Method II

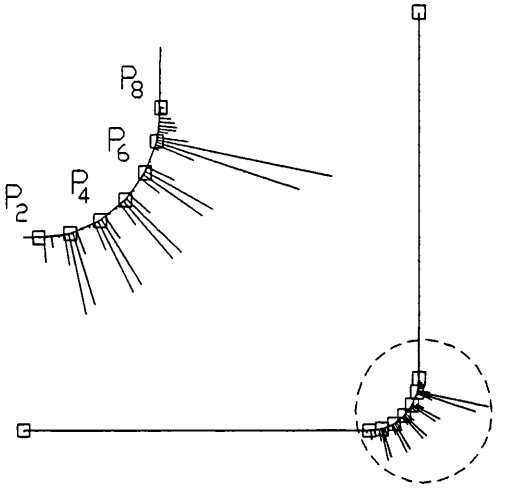

Figure 4.11 Midship section interpolated with Method III

Method I is more dependent on the distribution of the control points since the introduction of two more points in the transition segments ($P_{1'}$ and $P_{8'}$) improved significantly the behaviour of the curve (Fig. 4.12).

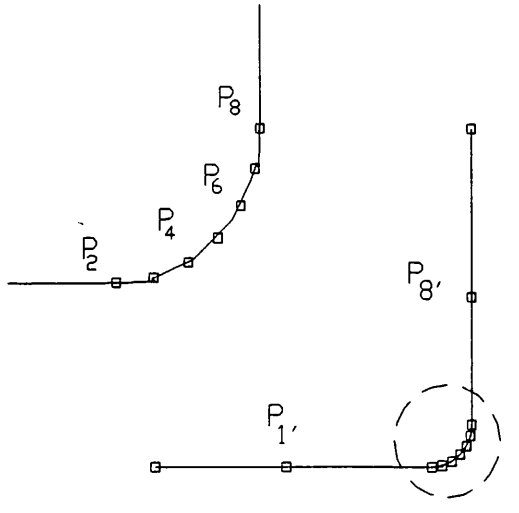

Figure 4.12 Midship section with extra points P1'and P8'

interpolated with Method I

4.2.3 Curve approximation

If, in order to obtain a more controllable curve, or to have a set of curves defined with the same number of control points, it is required to have the number of control points less than the number of data points, the matrix *[N]* is not square. In that case, knowing that the product of a matrix by its transpose is always square,

$$[N]^T[D]=[N]^T[N][B] \tag{4.8}$$

and now the control points can be obtained, solving the equation

$$[B]=\left[[N]^T[N]\right]^{-1}[N]^T[D] \tag{4.9}$$

The B-spline defined by these control points only approximates the given data points. In the example of (Fig. 4.13), twelve data points were approximated by a B-spline curve of order 3, with six control points. This approximation method can be used as a reasonable smoothing tool if the number of control points required is equal to the number of data points minus one (Fig. 4.14).

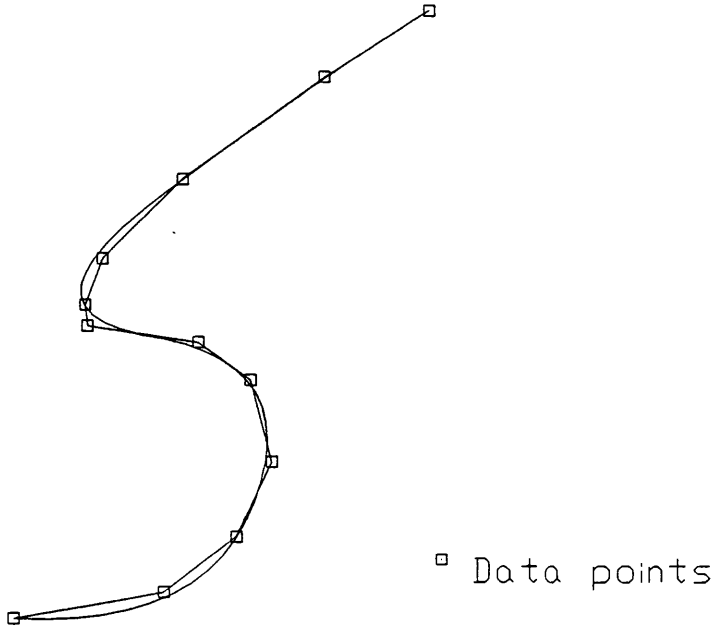

Figure 4.13 Data points approximated by a B-spline curve with 6 control points using the Least Square method

When the number of required control is equal to the number of data points minus one the resulting curve is a good smooth approximation (Fig. 4.14).

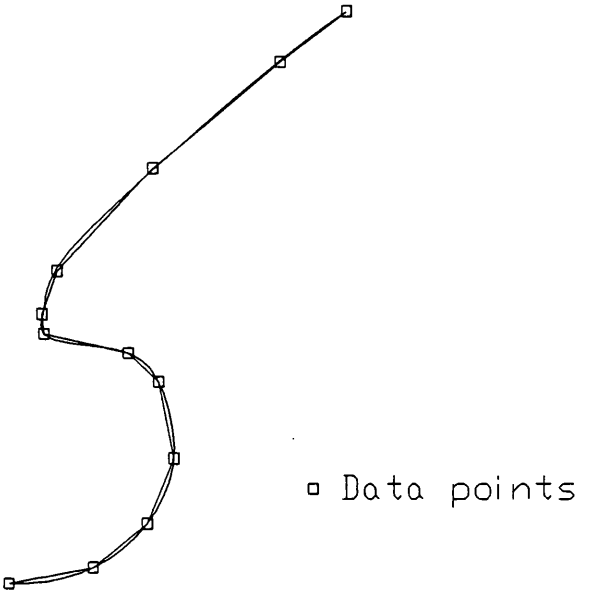

Figure 4.14 Curve smoothed by least square approximation

Being an approximation, the curve obtained does not guarantee the interpolation of all the data points. A drawback of this method, is that the designer cannot define which points should be interpolated, which should be approximated and what is the degree of approximation required for each point.

The above curve fitting and approximation procedures, are not capable of generating discontinuities in the first or second derivatives, which means that the control points generated have multiplicity equal to one. So, to have more precise control of the shape, such as introducing knuckle points or including conic segments with defined dimensions, manual editing of the curve will be required.

4.3 Curve shape control

Looking at the curve definition, there are three parameters through which it is possible to control the shape of a rational B-spline curve:

- changing the control points
- changing the knot vector
- changing the weights

4.3.1 Moving the control points

The control of the shape of a rational B-spline curve through the shape of the control polygon is very intuitive as can be seen in Fig. 4.15 which shows the successive shapes of the curve while the position of control point P_3 moves to $P_{3'}$.

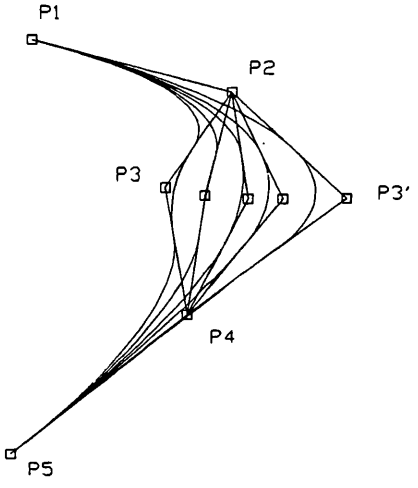

Figure 4.15 Effect of moving a single control point

4.3.2 Changing the knot vector

An improvement of the curve approximation can be obtained by adjusting the parameter values [58]. The method consists of minimising, through an iteration process, the error function defined by

$$E = \delta_x^2 + \delta_y^2 + \delta_z^2 \tag{4.10}$$

in which $\delta_x, \delta_y, \delta_z$ are functions of the distances between the data points P_i and the computed ones, P_{ci}

$$\delta_x = (P_{xc} - P_x) + \frac{\varphi P_x}{\varphi t}\Delta t$$

$$\delta_y = (P_{yc} - P_y) + \frac{\varphi P_y}{\varphi t}\Delta t \qquad (4.11)$$

$$\delta_z = (P_{zc} - P_z) + \frac{\varphi P_z}{\varphi t}\Delta t$$

The new parameter values are obtained by

$$t_i = t_i + \Delta t$$

in which

$$\Delta t = \frac{(P_x - P_{xc})(\varphi P_{xc} / \varphi t) + (P_y - P_{yc})(\varphi P_{yc} / \varphi t) + (P_z - P_{zc})(\varphi P_{zc} / \varphi t)}{(\varphi P_{XC}/\varphi t)^2 + (\varphi P_{YC}/\varphi t)^2 + (\varphi P_{ZC}/\varphi t)^2} \qquad (4.12)$$

The initial knot values are the normalised chord lengths of the data points, scaled by the maximum value of the open uniform knot vector x_{max}

$$t_i = \left(\frac{\sum_{j=2}^{i} (P_j - P_{j-1})}{\sum_{j=2}^{n} (P_j - P_{j-1})} \right) x_{max} \qquad (4.13)$$

The control points obtained by this procedures are located in a 3D space without any constraints. A better result can be obtained if the coordinates of the points computed are constrained to belong to given plane or curve and these values replaced in each step of the iteration process.

Experimenting with this method, it was noted that the adjustment of the knot vector, although improving the approximation to the data points, can sometimes increase the inflections (Fig. 4.16). In addition, the improvements obtained by methods such as this are apparent, because whenever the curve is edited for some reason, and they normally are, the knot vector is recomputed and the improvements are lost.

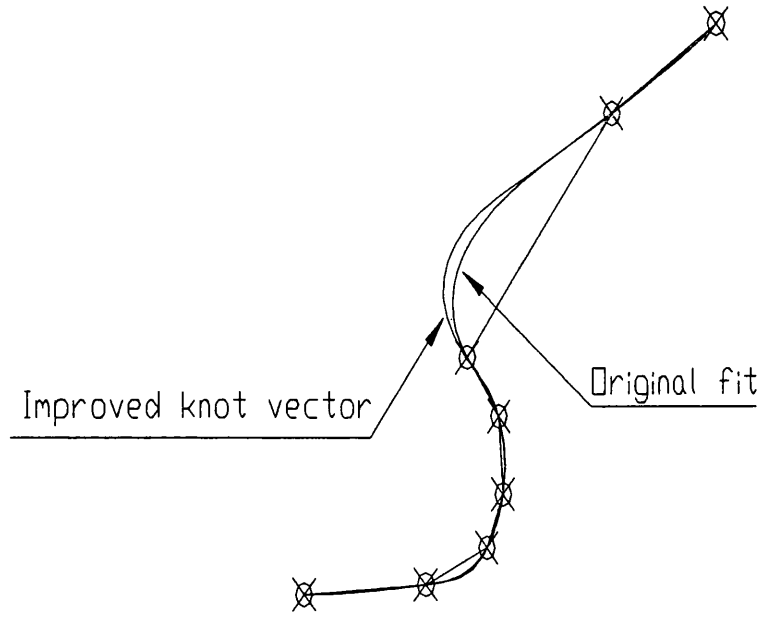

Figure 4.16 Effect of the iteration on the knot vector

4.3.3 Changing the weights

The increase of the weight value pushes the curve to the respective control point, although pulling it away from the other control points. In the limit, when the weight value is high enough, the curve tends to interpolate the control point. A weight of zero is equivalent to the deletion of the corresponding control point (Fig. 4.17).

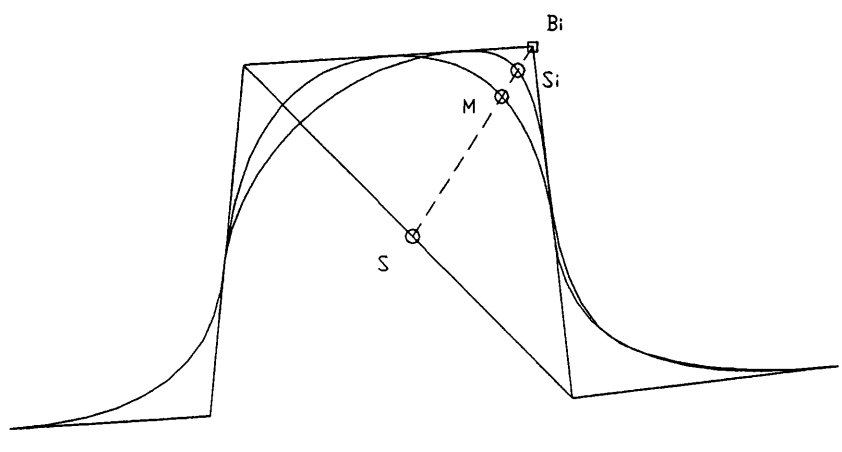

Figure 4.17 Changing the weight on a single control point

From the designer's point of view, the direct manipulation of the weight value is not intuitive, so a geometrical measurement must be defined and quantified [59]. In the following, the distance d along the line $\overline{SB_i}$ will be that geometrical measurement. In Fig. 4.17, different positions of a point S_i from curve C, corresponding to the control point B_i can be seen:

$$S = C(t, w_i = 0)$$
$$M = C(t, w_i = 1)$$
$$S_i = C(t, w_i)$$

Using u and v as local coordinates along $\overline{SB_i}$

$$M = (1-u)S + uB_i$$
$$S_i = (1-v)S + vB_i$$

the cross-ratio of the points B_i, S_i, M and S defines the identity

$$\frac{\frac{(1-u)}{u}}{\frac{1-v}{v}} = w_i$$

If the point S_i is moved to a new position S_i^1 along the line $\overline{SB_i}$, then

$$w_i^1 = \frac{\frac{(1-u)}{u}}{\frac{1-v^1}{v^1}} = \frac{\frac{B_iM}{SM}}{\frac{B_iS_i^1}{SS_i^1}} = \frac{\frac{B_iM}{SM}}{\frac{B_iS + d}{SS_i \pm d}}$$

where $d = |S_i - S_i^1|$, the new weight can be obtained by

$$w_i^1 = w_i^0 \left[1 \pm \frac{d}{R_{i,k}(t)(B_iS_i - d)} \right]$$

where the positive sign corresponds to pull the curve towards the control point and the negative sign to push the curve away from it.

4.4 Basic algorithms

In the previous paragraph algorithms to change the shape of curves were discussed. However, some operations applied on curves like making them compatible for surface generation, conversion of type, etc., require the existence of a set of algorithms that change some curve attributes without changing, or at least producing minimal alterations to the curve shape. The most common algorithms are knot insertion, knot removal and degree raising. They will be described in the following for curves, but they can be implemented similarly for surfaces.

4.4.1 Degree raising

The purpose of the degree raising algorithm [60] is to increase the degree of the B-spline without changing the shape of the curve. It is used mainly to make curves compatible.

Considering a given NURBS curve

$$P_k^w = \sum_{i=1}^{n+1} C_i^w N_{i,k}(t)$$

defined over a knot vector

$$X = \{0,\ldots,0,t_1,\ldots t_1,\ldots,t_{s,},\ldots,t_s,t_{n+k},\ldots t_{n+k}\}$$

where the first and last knots have multiplicity k and the inner knots have multiplicity $m_1, m_3, \ldots, m_s$, the corresponding curve with degree elevated will be given by

$$P_{k+1}^w = \sum_{i=1}^{n+s+1} C_i^w N_{i,k+1}(t)$$

defined over the knot vector

$$Y = \{0,\ldots,0,t_1,\ldots t_1,\ldots,t_{s,},\ldots,t_s,t_{n+k},\ldots t_{n+k}\}$$

where the first and last knots have multiplicity (k+1) and the inner knots have multiplicity $m_1+1, m_3+1, \ldots, m_s+1$. The new control points C_i^w are obtained by

$$C^w_{i,k-r+1} = \frac{(y_{j+r} - x_i)C^w_{i,k-r} + (x_{i+r+1} - y_{j+r})C^w_{i-1,k-r}}{x_{i+r+1} - x_i} + C^w_{i,k-r+r}$$

where

$$C^w_{i,k-r+1} = \frac{(y_{j+r+1} - x_i)C^w_{i,k-r} + (x_{i+r+1} - y_{j+r})C^w_{i-1,k-r}}{x_{i+r+1} - x_i}$$

In Fig. 4.18 the order of a curve initially $K=3$ was raised twice, showing the new distribution of control points.

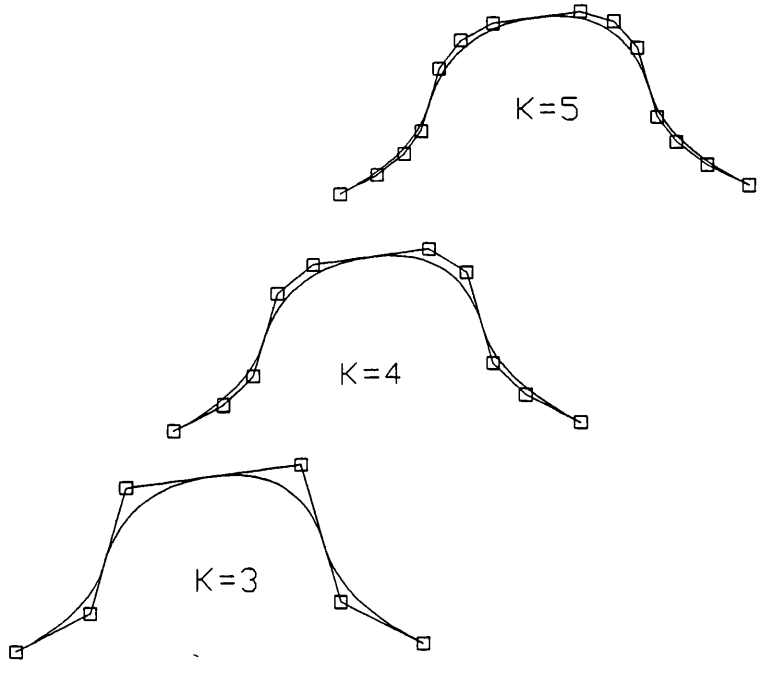

Figure 4.18 Changing the order of a NURBS curve

4.4.2 Knot insertion

Knot insertion algorithms allow the introduction of new values in the knot vector without changing the shape of the resulting curve.

Knot insertion is used in several different situations, such as:

- to make curves compatible
- to convert B-spline into Bézier representations

- to split curves
- to refine curves and surfaces as a pre-processing step for operations like shading, rendering, computation of intersections, etc.

The first knot insertion algorithm was presented by Chaikin for uniform B-splines of order three [61]. Later this algorithm was extended by Cohen, Lyche and Riesenfeld to B-splines of any order, the so called Oslo algorithm [62]. The general knot insertion algorithms for arbitrary order and knot vectors, appeared in 1980, from two different sources: one, known as the Bohem algorithm [63], inserts one single knot at a time and the other, the Oslo algorithm inserts a single control point.

4.4.3 Knot removal

Knot removal is the inverse operation of knot insertion and it tries to remove knots with a minimum change in the shape of the curve. The knot removal is an exact operation only when the knots removed were redundant to start with. So, and unlike knot insertion, knot removal is an approximation process, which means that the shape is actually changed to some extent, controlled by defining a tolerance value.

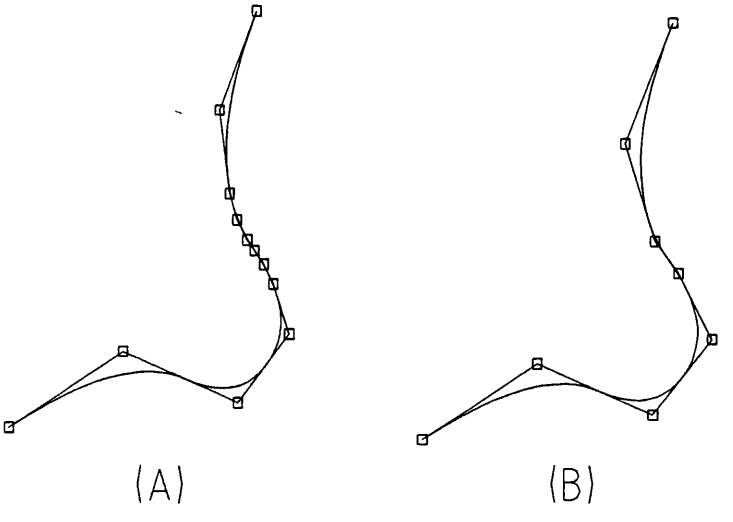

Figure 4.19 Knot removal on a curve

Efficient algorithms for knot removal for curves and surfaces were presented by Tiller [64]. In (Fig. 4.19) that algorithm was applied to a curve with 12 control points

(Fig. 4.19-A) from which 4 were successfully removed, for a tolerance value of 0.05 (Fig. 4.49-B).

Knot insertion is used in several applications to curve and surface modelling, such as converting B-spline to Bézier form and making curve segments or surface patches compatible for joining. After these operations are complete these extra knots should be removed as much as possible to reduce data complexity. Knot removal can also be used as a tools for obtaining simplified approximations to curves and surfaces and in fairing processes.

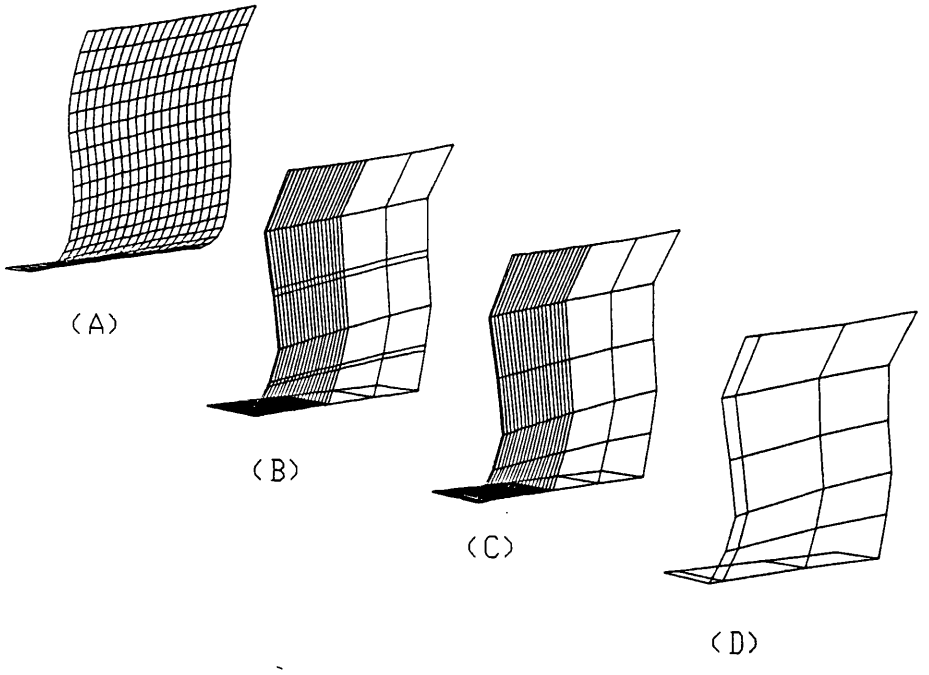

Figure 4.20 Knot removal on a surface

The same knot removal algorithm that is applied to curves can be applied to surfaces, considering each row or column of the control grid as a single curve. However, due to the rectangular nature of the control grid, one knot can be removed in the columns or row directions only if it is possible to remove it in each and every column or row. In Fig. 4.20-A is presented a surface with a control grid of (9x23) points (Fig. 4.20-B). Using a tolerance of 0.05, knot removal was applied to this surface, first in the V parametric direction, removing successfully 2 knots (Fig. 4.20-C) and then in the U direction, removing 19 knots (Fig. 4.20-D).

4.5 Curve editing techniques

From the designer's point of view, it is important to introduce or to change a defined shape characteristic. Having this in mind, and using the concepts and algorithms of Sections 4.3 and 4.4, a set of editing tools can be developed to change a curve to obtain specific shape changes.

4.5.1 Curve refinement

Since the easiest way to control the shape of a curve is through the shape of the control polygon, to obtain more control over a defined segment of the curve, it is necessary to refine the control polygon inserting one or more control points without changing the shape, using the knot insertion algorithm.

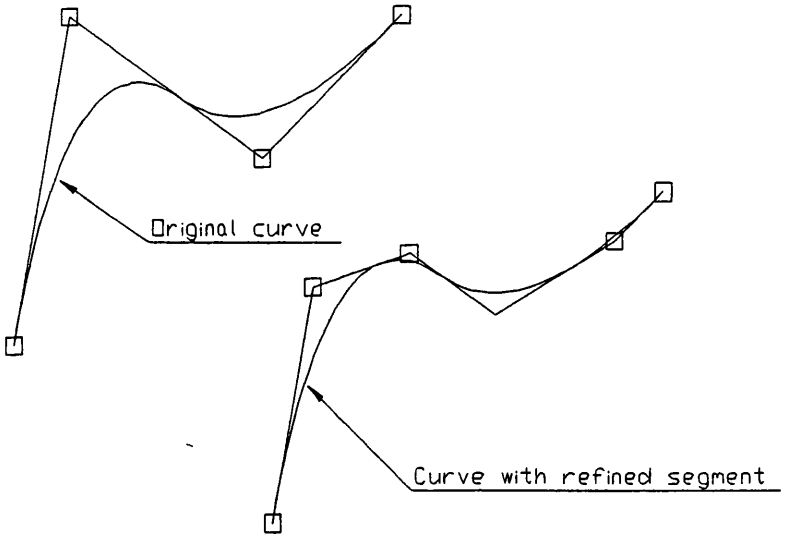

Figure 4.21 Curve refinement

In Fig. 4.21 it is possible to see the new distribution of control points resulting from the refinement of the original curve with one extra knot.

4.5.2 Curve splitting

When preparing curves to be used as support for surface generation, it is sometimes necessary to split a curve in two, in such a way that the resulting curves keep the shape of the original one.

The effect of splitting a curve is obtained by inserting a knot value $x \in [x_i, x_{i+1}[$ with a multiplicity *k-1* into the knot vector X.

$$X = \{\ldots x_i, x^1, \ldots, x^{k-1}, x_{i+1}, \ldots\}$$

The two resulting curves are defined as follows

$$B_I = \{B_1, \ldots, B_i\}$$
$$X_I = \{0^1, \ldots 0^k, \ldots, x_i, x^1, \ldots, x^k\}$$

$$B_{II} = \{B_i, \ldots, B_{n+k+1}\}$$
$$X_{II} = \{x^1, \ldots, x^k \ldots, x_{i+1}, \ldots, x_{m-k}, 1^1, \ldots, 1^k\}$$

In a practical application, instead of defining directly the parameter value where the splitting occur, the designer selects a point on the curve. The corresponding parameter value can be obtained using the Newton-Raphson iteration (4.14) along the parameter space.

$$t_{i+1} = t_i - \frac{f(t)}{f'(t)} \tag{4.14}$$

In which the derivative of the B-spline in rational form is given by

$$f'(t) = \sum_{i=1}^{n+1} C_i R'_{i,k}(t) \tag{4.15}$$

where $R'_{i,k}$ are the *rational basis functions*

$$R'_{i,k}(t) = \frac{w_i N'_{i,k}(t)}{\sum_{j=1}^{n+1} w_i N_{j,k}(t)} - \frac{w_i N_{i,k}(t) \sum_{j=1}^{n+1} w_i N'_{j,k}(t)}{\left(\sum_{j=1}^{n+1} w_i N_{j,k}(t)\right)^2}$$

In Fig. 4.22 it is possible to see the new control points $C_{1,i}$ and $C_{2,i}$ resulting from splitting the curve (A) by the line L, into the two curves shown in (B).

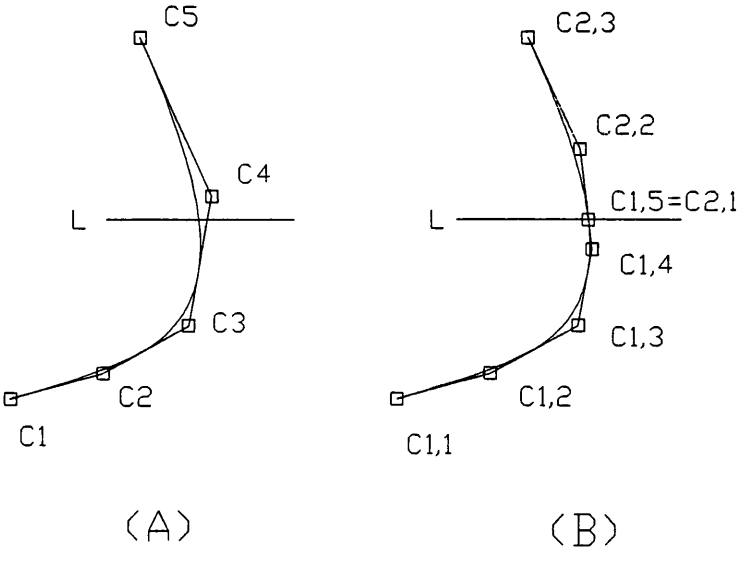

Figure 4.22 Curve splitting

4.5.3 Curve joining

Curve joining is the operational inverse of splitting and is used to merge two adjacent curves, in which the last point of the first curve coincides with the first point of the second point, keeping the shape.

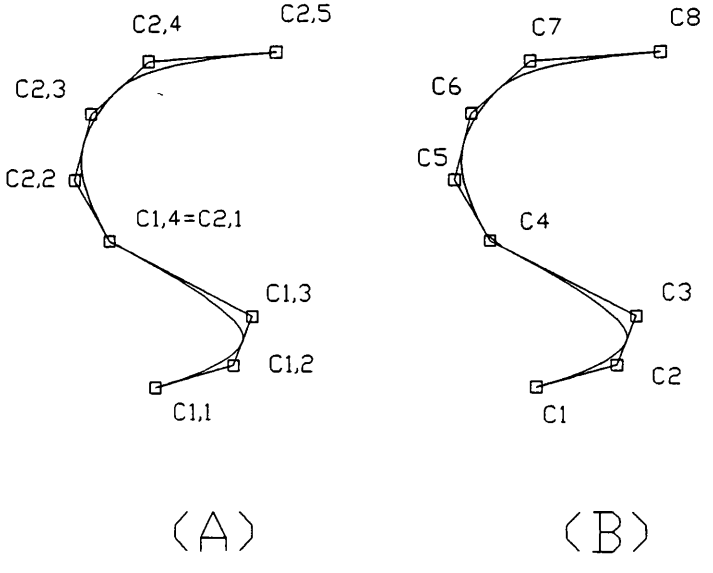

Figure 4.23 Curve joining

In Fig. 4.23 the two contiguous curves in (A) with the knot vectors

$$X_1 = \{0,0,0,1,2,2,2\}$$
$$X_2 = \{0,0,0,1,2,3,3,3\}$$

are joined resulting in the curve (B) with the knot vector

$$X = \{0,0,0,1,2,3\}$$

4.5.4 Creation of knuckles

In order to create knuckles, control points can be repeated with a multiplicity equal to *k-1*.

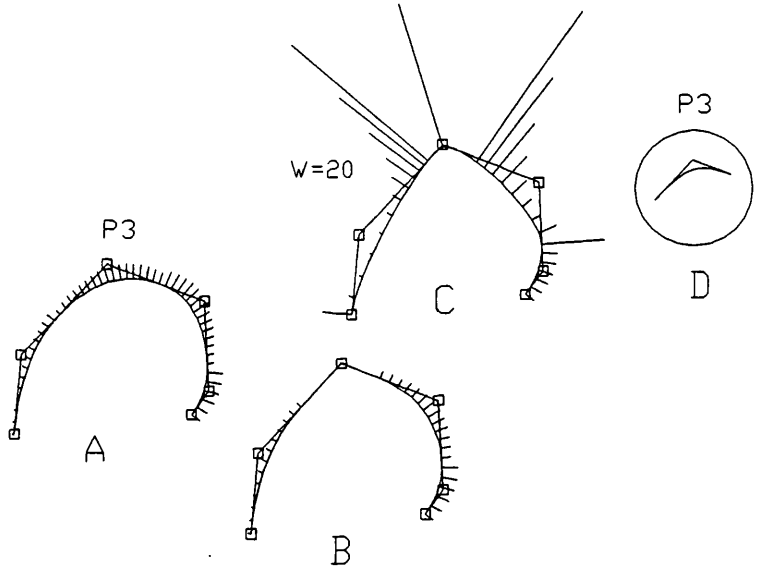

Figure 4.24 Creation of knuckle in point P3

In Fig. 4.24 curve (B) was obtained creating a knuckle on the point *P3* of curve (A).

The existence of multiple knots (and therefore multiple control points) can raise difficulties in subsequent curve fairing processes. One alternative solution, can be to split the curve by discontinuity points (refer to Section 4.5.2). The fairing algorithms can then be applied separately to each of the resulting segments.

The increase of the weight of the control point, seems to force the interpolation of the control point, giving origin to a knuckle (Fig. 4.24-C). But in fact, as it can be seen

from the curvature plot and looking into detail to the control point (Fig. 4.24-D) the result obtained is not a knuckle, but an arc in which curvature increases with the increase of the weight value.

Plotting the basis spline functions of the curves (Fig. 4.25) it is possible to see the increases of the maximum function value due to the repetition of the respective control point (Fig. 4.25-A,B), and the widening of the bell shape due to the increase of the corresponding control point weight (Fig. 4.25-C).

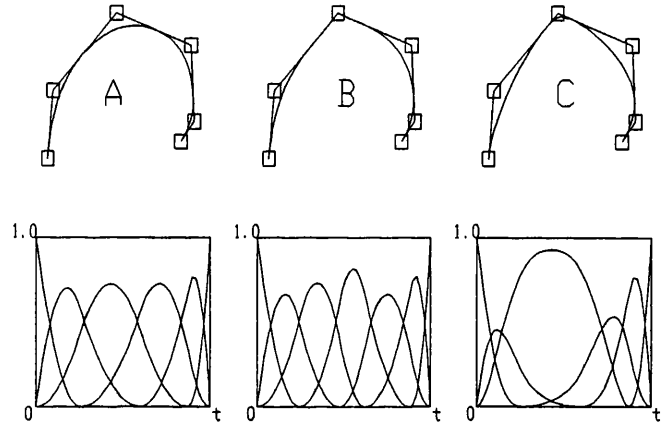

Figure 4.25 Comparison of basis functions

4.5.5 Creation of straight segments

Straight line segments can also be induced in a curve. First the two limiting control points are selected and a space straight line defined. Then, the internal control points can be made collinear to the straight line defined (Fig. 4.26). Since to obtain a straight segment at least three collinear points are necessary, if the selected points are consecutive, a third new point must be inserted, for instance at the mean parameter value.

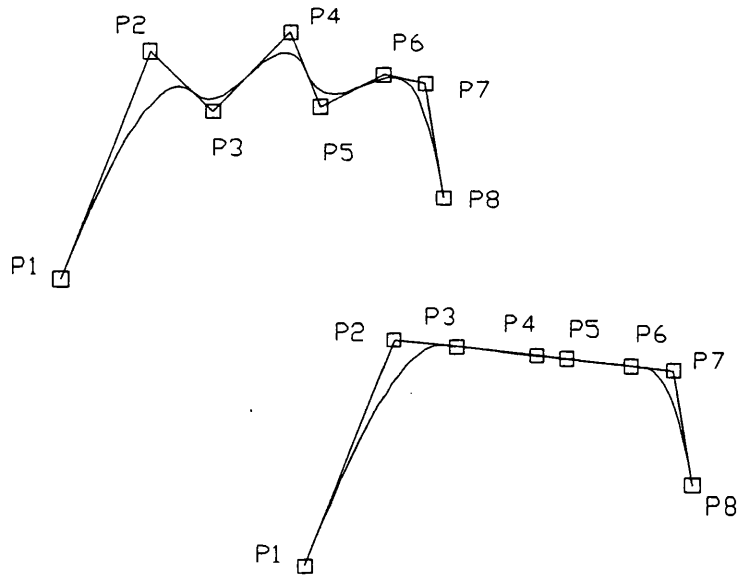

Figure 4.26 Creation of straight segments

4.5.6 Creation of conic segments

When editing a curve fitted to digitised points that contain conic segments of known characteristics, such as, for instance, a transversal section with a round or elliptical bilge, it is useful to have the capability to edit that curve segment and to convert it into a true conic. This capability, that is available due to the properties of the NURBS curves, allows not only an increased precision but also a substantial data reduction.

To create conic segments three control points are affected. The first and last of these points, are used to limit the extension of the conic segment and to define the boundary conditions. In the simpler case, by converting these points into knuckles, C^0 continuity is guaranteed between the conic segment and the adjacent ones. The position and weight of the middle point will be changed, according to the type of conic required (Fig. 4.27). Finally, the weight of the first or last points must be corrected to keep the conic shape invariance constant (Equation 3.34).

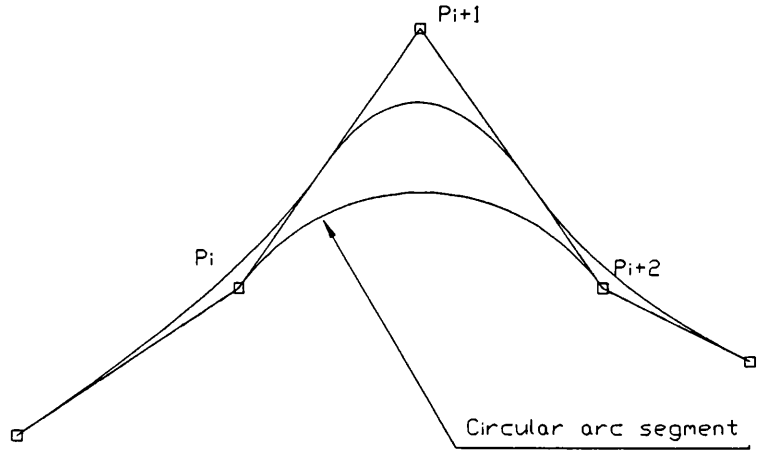

Figure 4.27 Creation of conic segment

In the particular case of circular arcs the procedure can be outlined as follows: after defined the required radius R (Fig. 4.28), the centre of the arc can be obtained by

$$C = M - (R - h_1)\vec{v}$$

in which M is the chord middle point and

$$h_1 = \sqrt{R^2 - \left(\frac{\overline{P_1P_3}}{2}\right)^2}$$

To obtain a circular arc the triangle $\Delta[P_1P_2P_3]$ must be isosceles and so the position of the point P_2 must be changed. The new position is obtained by

$$P_2' = M + h_2\vec{v}$$

The unit vector normal to the chord $\vec{v}$ can be computed by first computing the point N, from the projection of P_2 over the segment $\overline{P_1P_3}$

$$N = P_1 + dproj\frac{\vec{V}_2}{\|\vec{V}_2\|}$$

$$dproj = \frac{\vec{V}_1 \cdot \vec{V}_2}{\|\vec{V}_2\|}$$

$$\vec{V}_1 = P_2 - P_1$$

$$\vec{V}_2 = P_3 - P_1$$

The unit vector $\vec{v}$ is then

$$\vec{v} = \frac{P_2 - N}{\|P_2 - N\|}$$

The weight w_2 of the middle control point P_2 is obtained by

$$w_2 = \cos(\alpha)$$

in which α is the angle given by

$$\alpha = \tan^{-1}\left(\frac{\overline{P_3 M}}{\overline{MC}}\right)$$

From the known relation between the weight w_2 and the sides of the control polygon

$$w_2 = \frac{\overline{P_1 P_3}}{2\overline{P_3 P_2}}$$

the distance h_2 can be obtained

$$h_2 = \sqrt{\left(\overline{P_3 P_2}\right)^2 - \left(\frac{\overline{P_1 P_3}}{2}\right)^2}$$

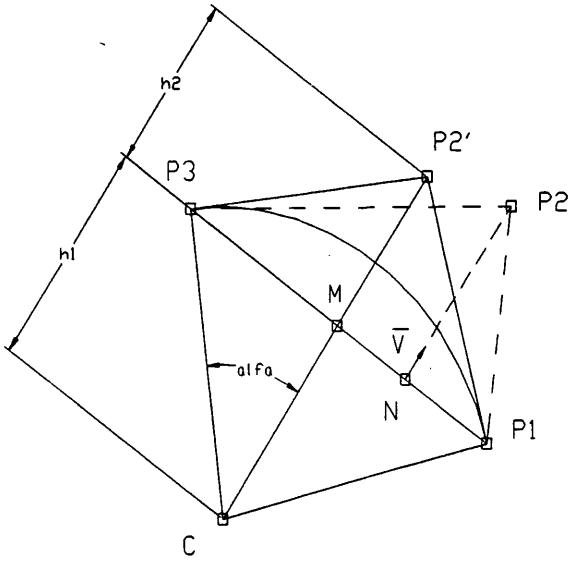

Figure 4.28 Circular arc generation

Finally, and since we are treating a segment of a curve, the weights of points P_1 and P_3 are not necessarily equal to 0 and 1 respectively so, to keep the conic shape invariance value constant the value of weight w_3 is recomputed

$$w_3 = w_1 . k_c$$

When the arc is equal to or greater than 180°, it must be reduced to lower values, for instance, by dividing it into two arcs.

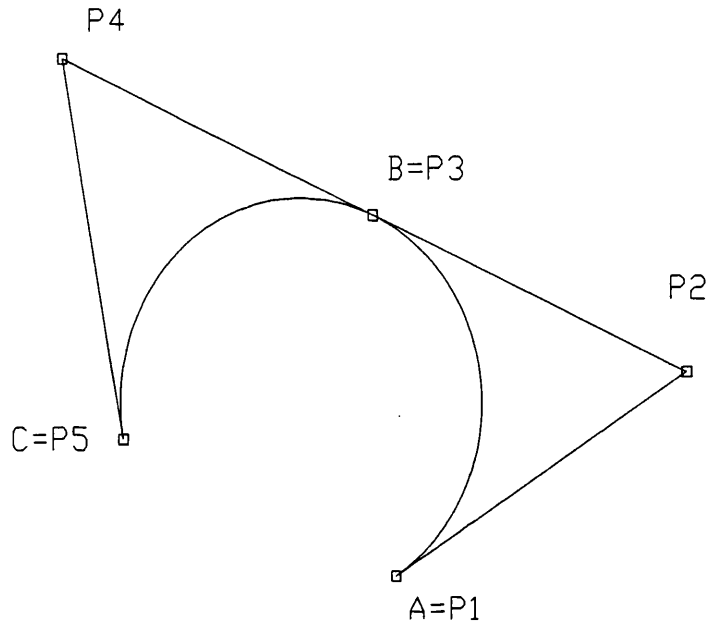

Figure 4.29 Generation of circular arc greater than 180 degrees

In Fig. 4.29 the arc $\angle$ABC was divided into the arc $\angle$AB generated by the control points $P_1P_2P_3$ and arc $\angle$BC generated by the points $P_3P_4P_5$. If necessary, the two conic segments can be joined back together using the algorithm described in Section 4.5.3.

4.6 Generation of the surface patches

After the definition of the support curves as described above, the surface patches can be generated. Several tools were developed for this purpose, taking into consideration the different types of constraints.

4.6.1 Lofting

In the present work, the surface patches will be defined mainly by a process called *lofting* or *skinning*. The lofting designation comes from the large halls (lofts) where ship hull forms were defined in the shipyards. Lofting can be defined as the generation of a smooth patch by fitting two parametric boundary curves that become embedded on two opposite sides of a parametric direction of the surface. The lofted surface $S(u,v)$ has the form

$$S(u,v) = b_1(u)\gamma_1(u) + b_2(v)\gamma_2(v) \tag{4.16}$$

where b_1 and b_2 are the boundary curves and γ_1 and γ_2 are the blending functions. Some authors [65], reserve the designation of skinning for the particular case in which the blending functions are B-spline functions. The CONSURF system [66,67,68], developed at the British Aircraft Corporation, was one of the first to use lofting of rational bi-cubic patches as a basic surface modelling tool.

In a more general description [69], lofting is the process of generating a surface that interpolates isoparametric points of a family of curves (cross sections or generator curves) in one of the surface's parametric directions (Fig. 4.30).

In some implementations of the lofting process, a *spine* curve is used to locate planar section curves in the 3D space. First, the section curves are translated so that the origin coincide with the target points along the spine curve; next, the section planes are rotated to a position normal to the spine curve and, finally, another rotation fixes the orientation of the sections in relation to the spine.

When applied to ship transverse sections, the lofting process is simplified since the base line can be used as a spine curve and so the curve sections - ship transverse sections - are already correctly positioned and oriented.

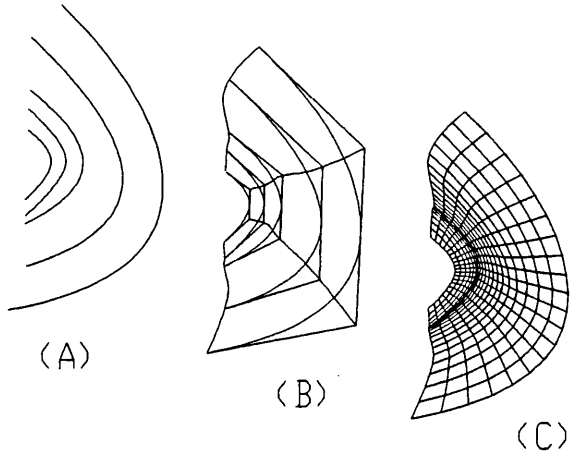

Figure 4.30 Surface generated by lofting of section curves

The lofting process can be divided into two main steps. First all the section curves are made compatible and next, isoparametric points of the curves are interpolated in the opposite parametric direction. To be compatible, the section curves must have the same order (degree), they must be defined over the same knot vector and they must have the same number of control points.

Generally the order used is the same (3rd. order is a good choice for the representation of ship sections), but if for some reason it differs from one curve to the other, the degree raising algorithm [70] will be used on the lower order curves. A common v direction knot vector will be obtained merging all the curve knot vectors - additional knots will be inserted [62] until each of the curves has the same knot vector.

After defining the order in the u parametric direction, for each row of the grid of control points, a curve will be fitted (Fig. 4.30-B) by the nodes method as described in Section 4.2.2, using centripetal knot vectors. The average of the knot vectors obtained in the curve fittings will be used as the u knot vector of the lofted surface (Fig. 4.30-C).

The quality of a surface generated by lofting depends of several factors:

- the existence of adjacent sections with knot vectors that have many different, although close, knot values - the final knot vector will have too many knots and will cause difficulties in the fairing process. A way of reducing this problem is to choose the defining points, whenever possible, over a common set of waterlines from bow to stern
- the spacing of the section curves - a very irregular spacing of the section curves can lead to very poor and sometimes unpredictable results
- the selection of the knot vector used for the curve interpolations in the parametric direction opposite to the curve sections
- the curve fitting algorithm.

The general lofting process described can be adapted to particular conditions, implementing several techniques to generate surface patches:

- Ruling
- Extrusion
- Sweeping
- Blending

4.6.2 Ruled surfaces

A ruled surface is generated by sliding a straight-line segment between two curves (Fig. 4.31). Mathematically it corresponds to linear interpolation between the two given rule curves and it can be defined by

$$S(u,v) = b_1(v)\gamma_1(u) + b_2(v)\gamma_2(u)$$

in which the blending functions are given by

$$\gamma_1 = (1-u)$$
$$\gamma_2 = u$$

and it can be seen as the simplest form of lofting.

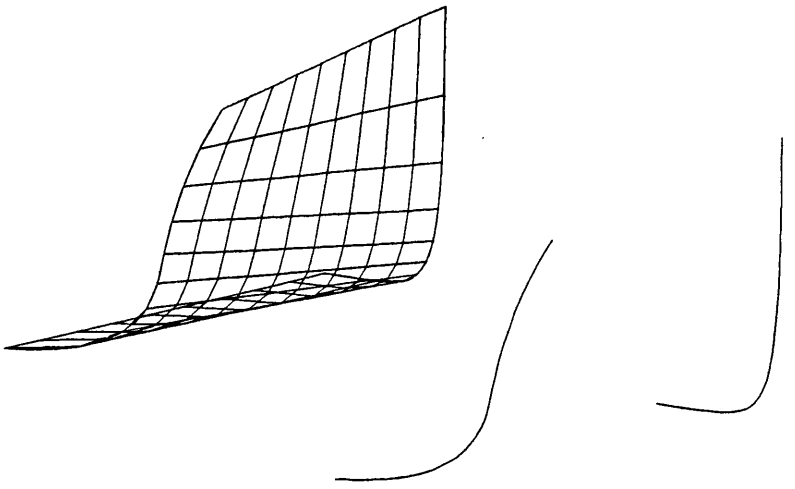

Figure 4.31 Ruled surface

Like general lofting, the ruling operation requires that the two curves are made compatible before the surface is generated.

4.6.3 Extrusion

Extruded surfaces are obtained from a profile curve and a vector (Fig. 4.32). The surface is generated by sweeping the profile curve along the vector direction.

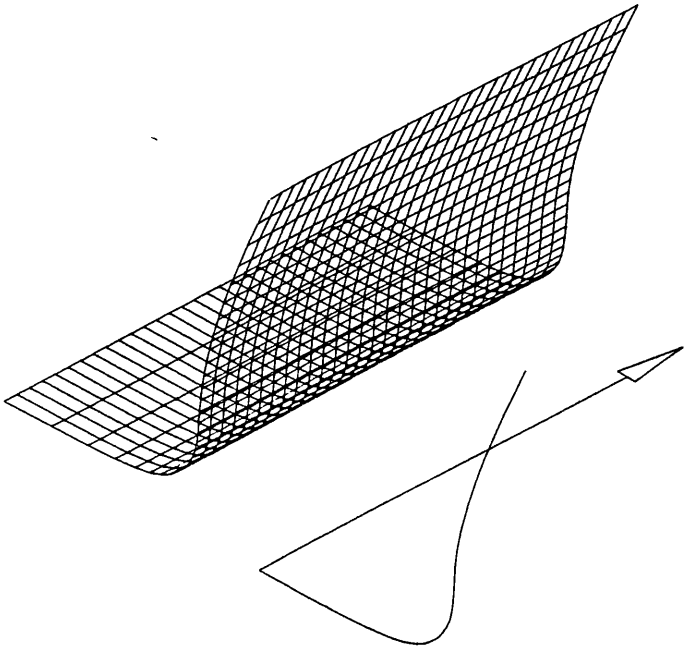

Figure 4.32 Extruded surface

Given one curve of order k defined by the control points B_{1i} and the knot vector x_j and the vector $\vec{v}$, a second curve can be defined using the control points obtained by

$$B_{2i} = B_{1i} + \vec{v}$$

the same order and the same knot vector. As the two curve are already compatible, by construction, and the surface is obtained by lofting, with order k in v direction and order 2 in u direction.

Extruded surfaces have constant shape and can be used, for instance, to model the ship parallel body.

4.6.4 Sweep surfaces

Sweep surfaces are obtained by sweeping a planar *profile* curve along a *trajectory* space curve (Fig. 4.33-A).

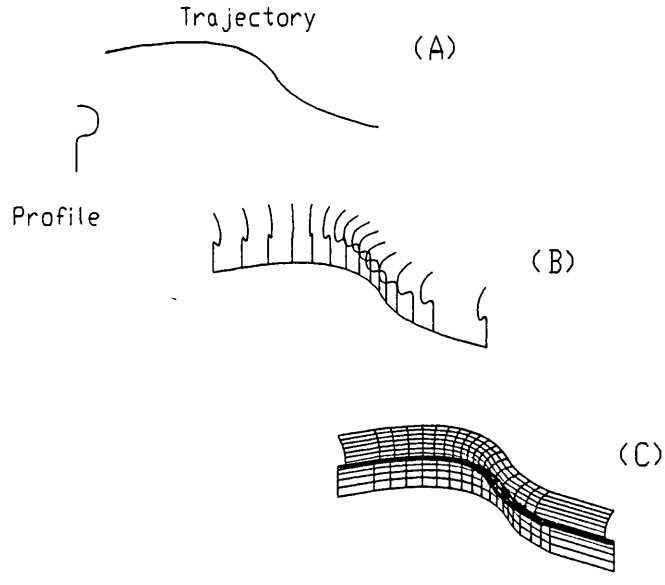

Figure 4.33 Sweep surface

The sweeping process can be seen as a particular case of lofting, in which all the sections are similar in shape.

The first step of the process (Fig. 4.33-B) is to generate copies of the profile curve, also called cross sections, and locate them in normal planes along the trajectory curve. Next, the surface patch is generated by lofting the cross-sections (Fig. 4.33-C). Surfaces of revolution can also be obtained by sweeping, using a circle as trajectory curve.

These cross-sections must be aligned properly and that requires the proper definition of local reference frames. One convenient and well known frame is the one due to Frenet, defined by the unit tangent T, the principal normal N and the binormal, B (Fig. 4.34).

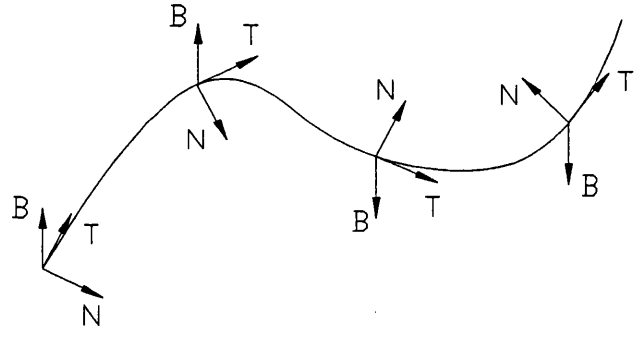

Figure 4.34 Frenet frame

The unit tangent vector is obtained from the first parametric derivative of the curve

$$T = \frac{D_1}{\|D_1\|}$$

The principal normal can be defined in the direction of the curvature, K

$$K = \frac{D_1 \times D_2 \times D_1}{\|D_1\|^4}$$

$$N = \frac{K}{\|K\|}$$

The binormal is obtained from the cross-product

$$B = T \times N$$

Although easy to compute analytically, the Frenet frame dependence on the curvature brings some inconvenience: it is undefined at points where the curvature is infinite, that is, in straight line segments, and changes suddenly in direction at inflection points (Fig. 4.34). This last problem is of particular importance when dealing with the generation of cross-sections in a surface sweeping process. One alternative approach [71] is to use the Frenet frame only to generate the first cross-section. The next frames can be obtained by computing T_I at each new location P_I. The new N_I and B_I vectors can be obtained by a rotation of the previous frame (T_0 - N_0 - B_0). The rotation axis A is obtained by

$$A = T_0 \times T_1$$

and the rotation, δ is

$$\delta = \cos^{-1}\left(\frac{T_0 \cdot T_1}{|T_0||T_1|}\right) \tag{4.17}$$

If the increment of the position vector is small enough, the orientation of the sections will be the same (Fig. 4.35) and the twist effect will be reduced,

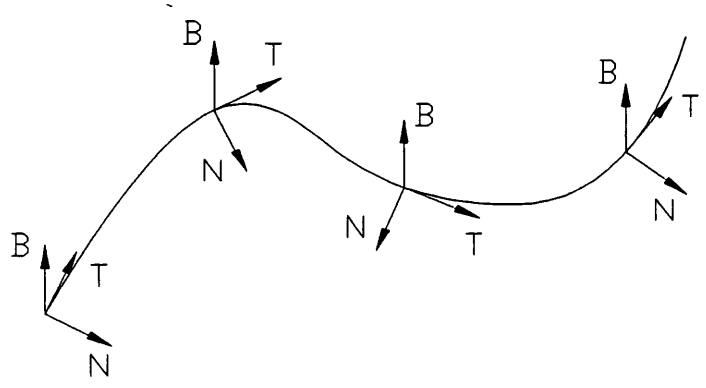

Figure 4.35 Reference frames for sweeping surface

To obtain the copies of the profile curve along the trajectory curve, the problem reduces now to a rotation about an arbitrary axis A defined by the direction cosines

(c_x, c_y, c_z) passing through a trajectory curve point (x_0, y_0, z_0). This transformation can be obtained by the following sequence of transformations:

- Translation from the current point to the point (x_0, y_0, z_0)
- Rotate first around the X axis and then around the Y axis in order to make the Z axis coincide with the arbitrary axis A
- Rotate around the Z axis by an angle of δ
- Inverse rotation
- Inverse translation

The complete transformation can then be represented by the composed transformation defined by

$$[M] = [T][R_x][R_y][R_\delta][R_y]^{-1}[R_x]^{-1}[T]^{-1}$$

where $[T]$ is the translation matrix defined by

$$[T] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -x_0 & -y_0 & -z_0 & 1 \end{bmatrix}$$

$[R_x]$ is the rotation about the X axis

$$[R_x] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha & 0 \\ 0 & -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

the angle of rotation α about the X axis (Fig. 4.36) is obtained from

$$\cos\alpha = \frac{c_z}{d}$$

$$\sin\alpha = \frac{c_y}{d}$$

in which $d = \sqrt{c_y^2 + c_z^2}$

$[R_y]$ is the rotation about the Y axis

$$[R_y] = \begin{bmatrix} \cos(-\beta) & 0 & -\sin(-\beta) & 0 \\ 0 & 0 & 1 & 0 \\ -\sin\beta & 0 & \cos(-\beta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

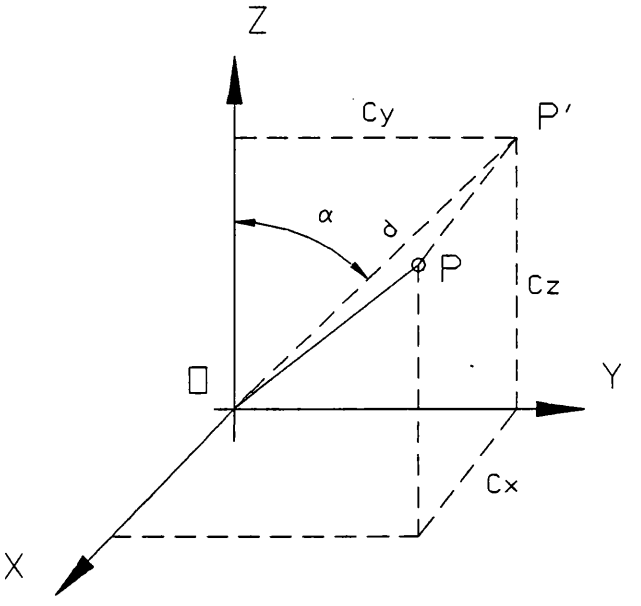

Figure 4.36 Rotation about the X axis

the angle of rotation β about the y axis (Fig. 4.37) is obtained from

$$\cos\beta = d$$
$$\sin\beta = c_x$$

and $[R_\delta]$ is the rotation about the A axis

$$[R_\delta] = \begin{bmatrix} \cos\delta & \sin\delta & 0 & 0 \\ -\sin\delta & \cos\delta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

in which δ is the angle obtained from (4.17).

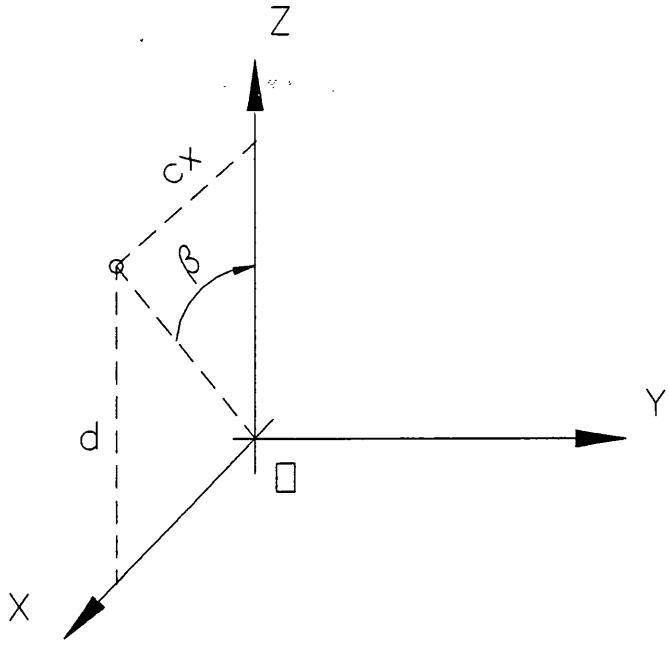

Figure 4.37 Rotation about the Y axis

Surfaces of revolution can be seen as a particular case of sweeping in which the trajectory is a circle (Fig. 4.38).

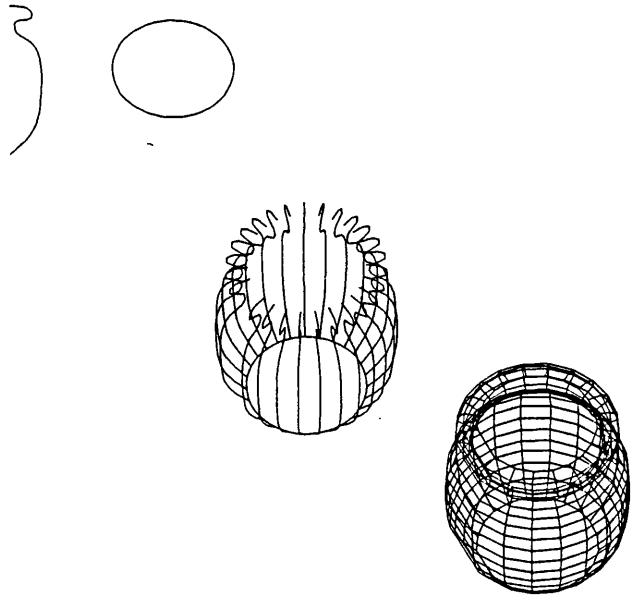

Figure 4.38 Surface of revolution obtained by sweeping

4.6.5 Blending surface

Surface patches interpolating four boundary curves are known as blending surfaces. This type of surface has been the object of the work of Coons and Gordon, hence they are also known as Coons-Gordon surfaces.

In order to keep the usage of the surface algorithms general, a common representation must be used for all the types of surfaces. The problem is then to represent a Coons-Gordon type of surface as a tensor-product NURBS surface.

Based on previous work [72] the surface can be obtained as a Boolean-sum

$$\begin{aligned} S(u,v) &= S_1(u,v) + S_2(u,v) - S_3(u,v) \\ &= \sum_{i=1}^{n}\sum_{j=1}^{m} N_{i,k}(u)N_{j,l}(v)C_{1_{i,j}} + \sum_{i=1}^{n}\sum_{j=1}^{m} N_{i,k}(u)N_{j,l}(v)C_{2_{i,j}} \\ &\quad -\sum_{i=1}^{n}\sum_{j=1}^{m} N_{i,k}(u)N_{j,l}(v)C_{3_{i,j}} \\ &= \sum_{i=1}^{n}\sum_{j=1}^{m} N_{i,k}(u)N_{j,l}(v)C_{i,j} \end{aligned}$$

where the net control points are obtained by

$$C_{i,j} = C_{1_{i,j}} + C_{2_{i,j}} - C_{3_{i,j}}$$

4.7 Representation of conical surface elements

To obtain a complete representation of the ship hull, there is the need to compute the intersection between the hull surface and other elements. These elements can be cylindrical or conical, like the hawse pipes and the side thrusters for example, but can also be other hull components, like the bulb.

Taking this into consideration, two ways of dealing with the problem can be foreseen: to develop specialised algorithms for the possible intersection cases, (cylinder/surface, cone/surface, etc.), or to develop one general purpose algorithm for surface/surface

intersection (SSI) that may deal with any kind of surfaces. In the present approach, NURBS surfaces are used to model the hull. Using also NURBS to model the cylinder and conical elements, a single SSI algorithm can be used to compute all the intersections required.

Hawse pipes and side-thruster tunnels can be made of a single cylindrical element or, more generally, composed by two elements, one cylindrical and one conical. The conical surfaces will be generated by a lofting method similar to the one used to generate the hull, but in this case circles will be used as section curves.

The problem of representing generic conic surfaces with NURBS can then be reduced to the lofting of conic curves as described in Section 4.7.1.

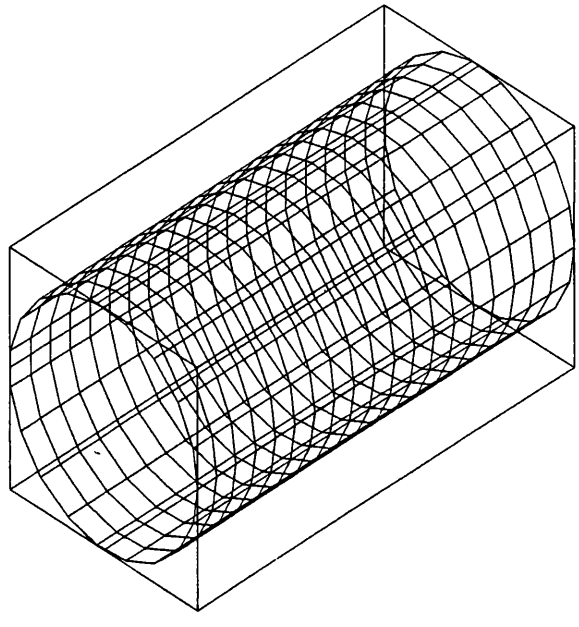

Figure 4.39 Cylindrical surface generated from two circular sections

In Fig. 4.39 the cylindrical surface was generated from two circular sections, with order 2 in the u parametric direction. If one of the bases of the cylinder degenerates to a circular section of zero radius, a conical surface is obtained (Fig. 4.40).

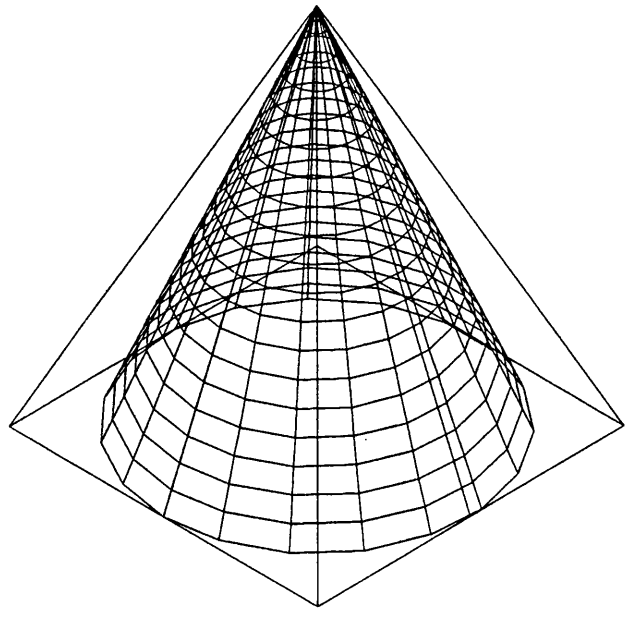

Figure 4.40 Conical surface generated from two circular sections, one of zero radius

Although the generation of conic surfaces can be seen just like a general application of lofting, the existence of repeated knots in the knot vectors of the conic section curves implies that the obtained surface is only C^1 in that parametric direction.

Ruling a surface between an inner and an outer co-planar circle, a circular ring can be generated (Fig. 4.41).

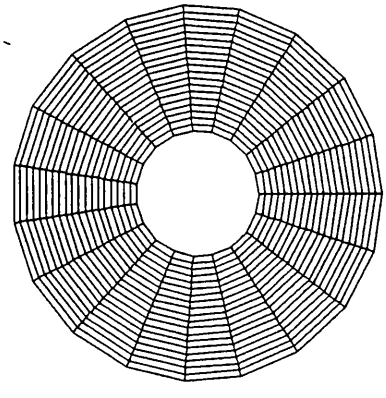

Figure 4.41 Circular ring surface

If the inner circle degenerates to a point by defining a radius of zero, a circular disk is obtained (Fig. 4.42).

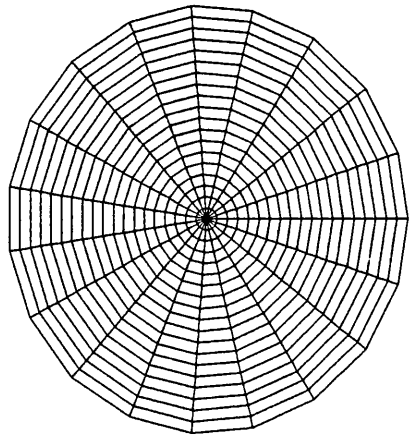

Figure 4.42 Circular surface

4.8 Curve and surface analysis

In this Section the main tools available for curve and surface analysis will be described. Surface models allow the use of much more sophisticated tools to evaluate the quality of the shape than the curvature plots normally used in the line models.

4.8.1 Zero order interrogation tools

The *wireframe* representation is obtained by displaying the boundary lines and a mesh of isoparametric curves in both parametric directions. This representation provides only a rough idea of the fairness of the surface.

When displaying a surface, it is more convenient to show it represented by *contour maps.* In the case of a ship hull, particular types of contour are obtained from the intersection with families of parallel planes at constant values of x, y, and z, which provide the traditional curves (sections, buttocks and waterlines) represented in the lines plan.

4.8.2 First order interrogation tools

Shading, isophotes and reflection lines are first order interrogation tools more commonly used in computer aided design.

- **Shading** gives a more realistic visualisation of the surface, by introducing a sense of volume. The shading commonly used in computer aided design results from a single source light, to avoid lengthy calculations.

- **Isophotes** are lines of constant light intensity on a surface, created by a parallel light source with a given direction, L. For a surface, an isophote is a line along which the quantity

$$n \cdot L = \cos \alpha$$

is constant, n being the normal unit vector and α is the angle of incidence, so that $0 \leq \alpha \leq 90$. If the surface is C^r continuous, then the isophotes are C^{r-1} continuous curves. Isophotes can be used to check the surface continuity across the boundaries of the patches. In the particular case of $n \cdot L = 0$, the corresponding isophotes are called *silhouettes*. In (Fig. 4.43) a light direction of $[1.0, 1.0, 1.0]$ generates on surface (A) the isophotes (B) and the silhouette (C).

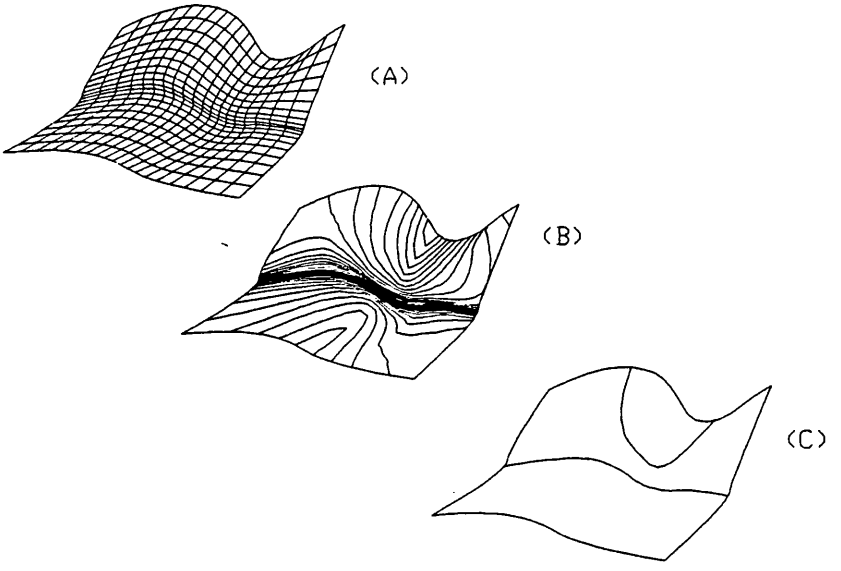

Figure 4.43 Contours of isophotes

- **Reflection lines** can be defined as the reflected image of a straight line light, as seen from a given point of view.

Reflection lines are used in many CAD systems as a general purpose tool for studying surface fairing. The method has its roots in the early automobile industry, when the irregularities in the surface of car hoods were detected by moving a car model into a dark room, provided with parallel light sources and analysing the irregularities of the reflection lines.

4.8.3 Second order interrogation tools

Curvature display is an efficient way to evaluate the fairness of both curves and surfaces.

The curvature κ of a space curve is positive by definition and is defined by [73]:

$$\kappa(t) = \frac{|x'(t) \times x''(t)|}{|x'(t)|^3} \tag{4.18}$$

In the particular case of a planar curve, and in order to be able to detect inflection points, a signed curvature may be defined by [73]:

$$\kappa(t) = \frac{\ddot{x}(t)\dot{y}(t) - \ddot{y}(t)\dot{x}(t)}{\left[(\dot{x}(t))^2 + (\dot{y}(t))^2\right]^{3/2}} \tag{4.19}$$

To ease the interpretation of the analysis results, it is normal practice to use graphical representations either by plotting the values of curvature or using the so called "porcupine" representation (Fig. 4.44).

In the curvature plot, the curvature values can be plotted against the parameter value, but as these values depend on the type of parameterisation, a better practice is to plot them against the curve length.

On the porcupine representation, the *curvature vectors*, with modulus proportional to the curvature values, normal to the curve at each point and by convention, oriented to the side opposite to the centre of curvature are plotted directly over the corresponding points on the curve.

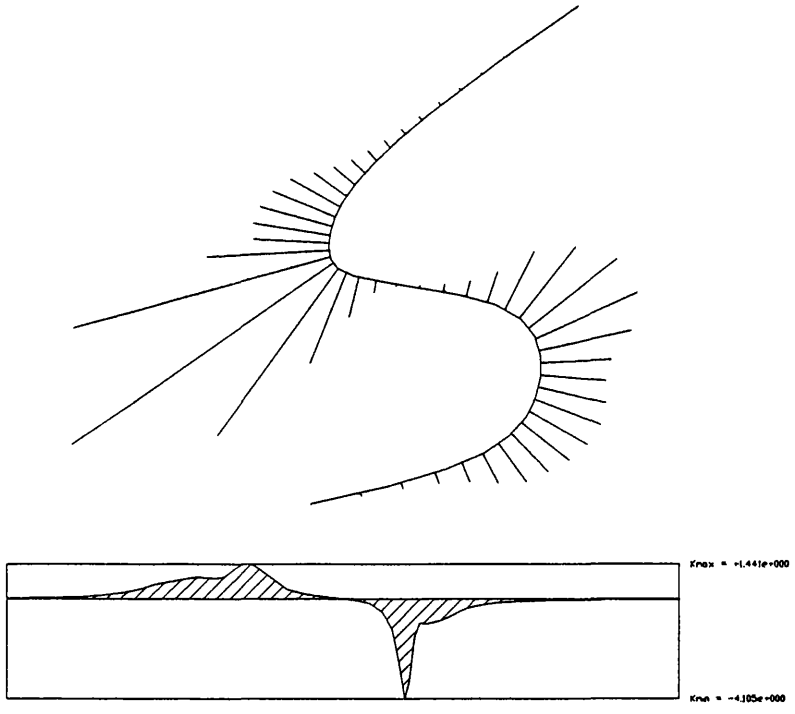

Figure 4.44 Line curvature displayed by a porcupine representation and by the curvature plot

The methods of evaluating the fairness of curves can also be used on surfaces, by studying the curvature of the iso-parametric lines of the grid.

A surface *r(u,v)* can be determined from two intrinsic quantities called the first and second fundamental forms. The *first fundamental form*, gives the infinitesimal arc length *ds* between two points *(u,v)* and *(u+du, v+dv)*, measured in the tangent plane of the surface at *(u,v)* and is defined by

$$ds^2 = dr.dr \tag{4.20}$$

The vector dr/dt at a point P of a surface is given by

$$\frac{dr}{dt} = r_u \frac{du}{dt} + r_v \frac{dv}{dt}$$

and so

$$dr = r_u du + r_v dv$$

replacing in (4.20)

$$\begin{aligned} ds^2 &= r_u \cdot r_u du^2 + 2r_u \cdot r_v dudv + r_v \cdot r_v dv^2 \\ &= Edu^2 + 2Fdudv + Gdv^2 \end{aligned} \tag{4.21}$$

where

$$E = r_u.r_u \quad F = r_u.r_v \quad G = r_v.r_v \tag{4.22}$$

The *second fundamental form*, gives twice the component of the displacement *dh* between *(u,v)* and *(u+du, v+dv)* perpendicular to the tangent plane at *(u,v):*

$$dh = Ldu^2 + 2Mdu.dv + Ndv^2 \tag{4.23}$$

where

$$L = n.r_u \quad M = n.r_{uv} \quad N = n.r_{vv} \tag{4.24}$$

and *n* is the surface unit normal at *(u,v)*

$$n = \frac{r_u \times r_v}{|r_u \times r_v|} \quad when \ |r_u \times r_v| \neq 0 \tag{4.25}$$

For surfaces several curvatures can be defined. The *normal curvature* of a surface *S* in a point *P* and in a given direction *t*, is the curvature of the normal section curve, i.e., the curvature of the intersection of the surface with a plane *N* in that direction containing the normal to the surface at the point (Fig. 4.45) and is defined [74] by

$$\kappa = -\frac{L(du)^2 + 2Mdudv + N(dv)^2}{E(du)^2 + 2Fdudv + G(dv)^2} \tag{4.26}$$

The sign convention used in equation (4.26) gives a positive curvature when the centre of curvature and the surface normal lie on opposite sides of the surface.

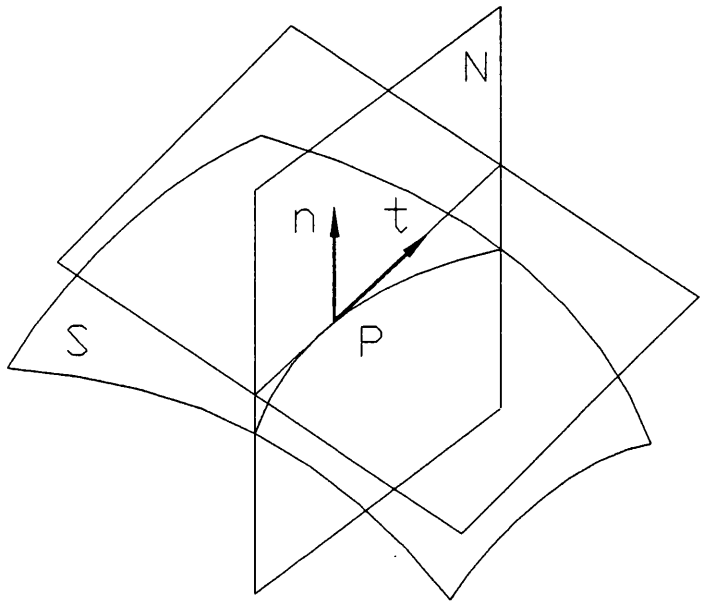

Figure 4.45 Normal curvature

The normal curvature has extreme values when $\partial\kappa / \partial du = 0$ and $\partial\kappa / \partial dv = 0$, i.e., when

$$\begin{aligned}(L + \kappa E)du + (M + \kappa F)dv = 0 \\ (M + \kappa F)du + (N + \kappa G)dv = 0\end{aligned} \tag{4.27}$$

In order to have a solution to equations (4.27), κ must satisfy the condition

$$\kappa^2 + 2H\kappa + K = 0 \tag{4.28}$$

in which H is the *mean curvature* and K is the *Gaussian curvature* defined by

$$\begin{aligned}H &= \frac{2FM - (EN + GL)}{2(EG - F^2)} \\ K &= \frac{LN - M^2}{EG - F^2}\end{aligned} \tag{4.29}$$

In situations (Fig. 4.46) where the rectangular patch (A) collapses to a triangular patch (B), that is, one of the edges reduces to a vertex of the patch, the surface patch is called degenerate. On degenerate surfaces $EG - F^2 = 0$ and so the curvatures can not be computed by (4.29).

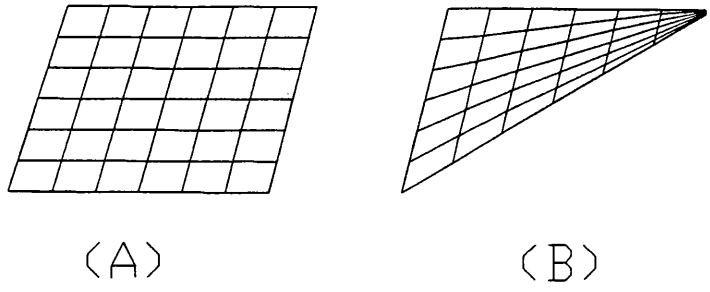

Figure 4.46 Rectangular patch and degenerate patch

In [75] the collapsed rectangular patched is mapped over a triangular domain to allow the computations of the curvatures in every surface point. The curvature of degenerate surface patches is beyond the scope of the present work and so in these cases, and only to guarantee the functionality of the curvature representation algorithms, it will be assumed that $H = K = 0$.

The solutions of (4.28) are called the *principal curvatures* and they represent the minimum and maximum values of the curvature at that point

$$\begin{aligned} \kappa_{\min} &= H - \sqrt{H^2 - K} \\ \kappa_{\max} &= H + \sqrt{H^2 - K} \end{aligned} \tag{4.30}$$

When at a certain point $H^2 = K$, $\kappa_{\min} = \kappa_{\max}$ and so κ at this point is constant in all directions. Such points are called *umbilic points*. If at a umbilic point $\kappa=0$, then at that point the surface is approximately plane. If $\kappa\neq0$, then the surface is approximately spherical.

The Gaussian (K), the mean (H) and also the absolute (κ_{abs}) curvatures can be expressed in terms of the principal curvatures

$$\begin{aligned} K &= \kappa_{\min}\kappa_{\max} \\ H &= \frac{1}{2}(\kappa_{\min} + \kappa_{\max}) \\ \kappa_{abs} &= |\kappa_{\min}| + |\kappa_{\max}| \end{aligned} \tag{4.31}$$

If the Gaussian curvature $K = 0$, that means that one of the principal curvatures $\kappa_{\min}$ or $\kappa_{\min}$ is zero. In this case, the surface is *developable*, i.e., can be unrolled into a

plane along the principal direction, without stretching or distortion. If $K > 0$ the principal curvatures have the same sign, whether positive, (the surface is convex at the point) or negative (the surface is concave). If $K < 0$, the principal curvature have opposite signs in the principal directions, which corresponds to a surface with a saddle shape.

The *curvature lines* are the curves on a surface whose tangent at every point is aligned with the principal direction. The *principal directions* of a certain point are those for which the normal curvature takes minimum or maximum values and are orthogonal, unless the point is an umbilic. The principal directions can be obtained from (4.27) solving the equations

$$\begin{aligned} du &= \beta\left(M + \kappa F\right) \\ dv &= -\beta\left(L + \kappa E\right) \end{aligned} \tag{4.32}$$

in which β is an arbitrary factor $\beta \neq 0$ obtained from (4.33) and $\kappa \in \{\kappa_{\min}, \kappa_{\max}\}$.

$$\beta = \pm\left[E\left(M + \kappa F\right)^2 - 2F\left(M + \kappa F\right)\left(L + \kappa E\right) + G\left(L + \kappa E\right)^2\right]^{-1/2} \tag{4.33}$$

The sign of β determines the direction in which the solution proceeds.

The normal curvature κ (4.26) can be decomposed into its normal and geodesic components

$$\kappa = \kappa_n + \kappa_g \tag{4.34}$$

The curves resulting from setting $\kappa_g = 0$ are called *geodesic curves.* Along these curves $\kappa = \kappa_n$, and so on every point of the geodesic the normal to the curve coincides with the normal to the surface. This curves represent arcs of the minimum distance between two given points on the surface. Remembering the first fundamental quadratic form of a surface (4.21), the distance between two point on the surface is given by

$$s = \int_{t_1}^{t_2} \sqrt{E\dot{u}^2 + 2F\dot{u}\dot{v} + G\dot{v}^2}\, dt \tag{4.35}$$

The geodesics are the curves that satisfy the equations

$$\begin{aligned} \ddot{u} + \Gamma_{11}^{1}\dot{u}^2 + 2\Gamma_{12}^{1}\dot{u}\dot{v} + \Gamma_{22}^{1}\dot{v}^2 = 0 \\ \ddot{v} + \Gamma_{11}^{2}\dot{u}^2 + 2\Gamma_{12}^{2}\dot{u}\dot{v} + \Gamma_{22}^{2}\dot{v}^2 = 0 \end{aligned} \tag{4.36}$$

where Γ_{jl}^{i} are the called *Christoffel symbols of the second kind*

$$\begin{aligned} \Gamma_{11}^{1} &= \frac{n.(r_{uu} \times r_v)}{|r_u \times r_v|} = \frac{\mathrm{GE_u} - 2FF_u + FE_v}{2(EG - F^2)} \\ \Gamma_{12}^{1} &= \frac{n.(r_{uv} \times r_v)}{|r_u \times r_v|} = \frac{\mathrm{EG} - FE}{2(EG - F^2)} \\ \Gamma_{22}^{1} &= \frac{n.(r_{vv} \times r_v)}{|r_u \times r_v|} = \frac{2\mathrm{GF_v} - GG_u - FG_v}{2(EG - F^2)} \end{aligned}$$

$$\begin{aligned} \Gamma_{11}^{2} &= \frac{n.(r \times r_{uu})}{|r_u \times r_v|} = \frac{\mathrm{GE_u} - 2FF_u + FE_v}{2(EG - F^2)} \\ \Gamma_{12}^{2} &= \frac{n.(r_u \times r_{uv})}{|r_u \times r_v|} = \frac{\mathrm{EG_u} - FE_v}{2(EG - F^2)} \\ \Gamma_{22}^{2} &= \frac{n.(r_u \times r_{vv})}{|r_u \times r_v|} = \frac{\mathrm{GE_u} - 2FF_u + FE_v}{2(EG - F^2)} \end{aligned} \tag{4.37}$$

and n is the surface unit normal at (u,v)

$$n = \frac{r_u \times r_v}{|r_u \times r_v|} \qquad when\ |r_u \times r| \neq 0 \tag{4.38}$$

4.8.4 Third order interrogation tools

Torsion is a measure of the deviation of a of space curve away from its osculating plane. The torsion of a parametric curve $r(t)$ is defined by

$$\tau(t) = -\frac{(r_t \times r_{tt}) \cdot r_{ttt}}{|r_t \times r_{tt}|^2}$$

Curves of order < 3 have zero torsion because r_{tt} is zero.

4.8.5 Graphic representation of surface characteristics

Given the tensor product form of the surfaces generated, for the computations described in the previous Section, the surface properties are computed over the vertices of a parametric grid, producing fields of scalar values, that are more understandable if represented in graphical form. The generation of contours of iso-lines and the display of coloured mappings are two of the most common forms of graphical representation of scalar fields.

The *contours* can be generated from a triangulation of the surface patch. Due to the rectangular nature of the tensor product patches, the triangulation of the surfaces can be obtained very easily, just by dividing by the diagonal each elementary rectangle into two triangles (Fig. 4.47).

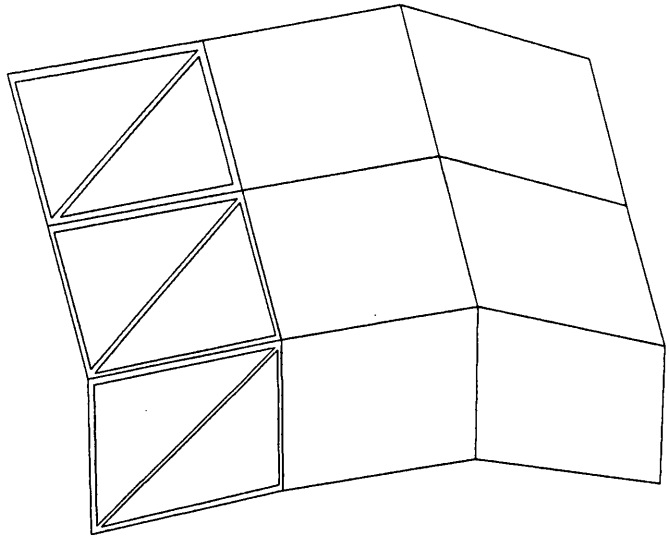

Figure 4.47 Triangulation of tensor product patch

In the present work, the contouring algorithm presented in [76], originally developed for 2D data points, was adapted to 3D and used with good results. This algorithm is applied over a triangulation of the surface and is based on the assumption that the value to contour has a linear variation inside each triangle.

Coloured mappings can be obtained assigning to the entire area of each rectangular element the average of the property values computed at the corners. The range of property values obtained for a surface patch is divided into a set of classes, to each a colour code is assigned. The number of colours depends on the accuracy required and on the type of display available. The colour codes are normally defined in a range going from the blue, for the lower negative values to the magenta, for the upper positive values.

When displaying Gauss curvature of samples of hull surfaces, it was noted that the range of values obtained can be very small and very close to zero and so it was more useful to divide the surface into just three classes allowing a qualitative division of the surface in developable areas ($k=0$), concave and convex areas ($k>0$) and saddle shaped areas ($k<0$).

4.9 Curve and surface fairing

Fairing is the process of eliminating undesirable shape features, such as local bumps and hollows, in order to obtain a smoother shape. The fairness of lines is a concept that has several definitions but is nevertheless still difficult to evaluate.

Atkins et al [77], presented six conditions as the requirements for a fair ship curve: continuity of the curve and its first and second derivatives, absence of extraneous inflection points, minimal deviation from the scaled offsets and good outlook to the eye.

For Nowacki [3], smoothness and fairness are two different concepts, being the former the absence of local bumps, in which a bump consists of two closely spaced inflection points, and the latter, the existence of continuous first and second derivatives.

Farin [73] defined a fair curve as that one whose curvature plot consists of relatively few monotone pieces.

So, the concept of a fair line or surface has both a mathematical and an aesthetic components. The aesthetic part is difficult to define and depends of the type of application in mind. Therefore, a fairing system must provide both an automated procedure based on a mathematical formulation of fairness and means to allow the designer to evaluate the results and to complete the process.

In general, the idea of unfairness of curves and surfaces is associated with excess of information and so many fairing methods try to remove it by removing knots or reducing the degree. Some authors, however, considering that unfair curves or surfaces are a result of over constrain, go in the opposite direction and try to relax the excess constraints by adding flexibility (degrees of freedom), for instance, by raising the degree, in spite of the problems normally associated with that.

4.9.1 Curve fairing

Kjellander [78], proposed a method for fairing cubic splines removing the offending knot and replacing it with a new one. The new knot is obtained by cubic Hermite interpolation using the position and derivative data at the two adjacent knots. The interpolated value is then used for a global cubic spline interpolation to all the position data. Due to this last interpolation, this is a global process. This method was later extended to parametric bicubic surfaces [79]. Farin [80] applied a similar method to cubic B-spline curves. First, the offending knot is removed and the control polygon is redefined. This algorithm is known as *knot removal.* Next, by *knot insertion*, a new knot is inserted in a way that the curve remains unchanged. Farin simplified this procedure so that if a curve is assumed to be incorrect at a knot value t_i, associated with the control point C_i, the new corrected control point is given directly by

$$C_i = \frac{(t_{i+2} - t_i)L_i + (t_i - t_{i-2})R_i}{t_{i+2} - t_{i-2}}$$

in which L_i and R_i are two auxiliary points obtained from

$$L_i = \frac{(t_{i+1} - t_{i-3})C_{i-1} + (t_{i+1} - t_i)C_{i-2}}{t_i - t_{i-3}}$$

$$R_i = \frac{(t_{i+3} - t_{i-1})C_{i+1} + (t_i - t_{i-1})C_{i+2}}{t_{i+3} - t_i}$$

Due to the nature of B-spline basis functions, this is a local method. In Fig. 4.48, an example of the results of this method is shown. An initial curve (Fig. 4.48-A) is faired twice, and the corresponding curvature plots are represented. The improvement from curve (B) to curve (C) is only visible on the curvature plot.

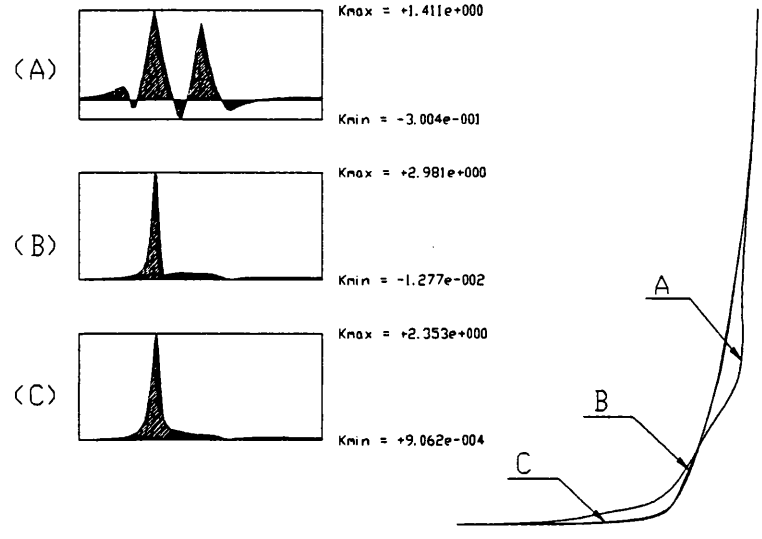

Figure 4.48 Fairing of cubic B-spline curve

A simple but sometimes useful fairing algorithm can be implemented using piecewise circular arcs. For every set of four control points, a circular arc is defined by the first, second and fourth points (Fig. 4.49). Then, the third point is moved from the initial position to the corresponding position on the arc (Fig. 4.50) and another set of four points is considered.

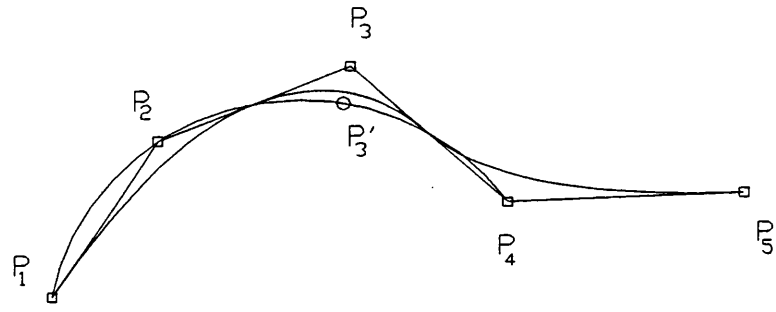

Figure 4.49 Fairing using piecewise circular arcs

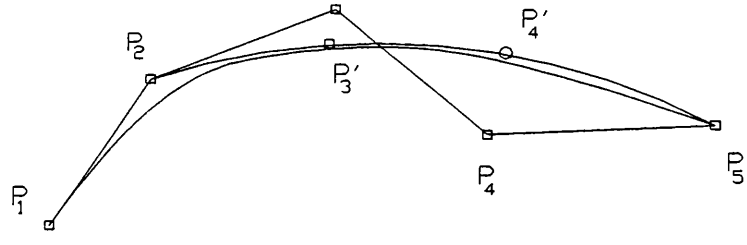

Figure 4.50 Fairing using piecewise circular arcs

To apply the above methods, the determination of the curve segment to be faired relies on qualitative interrogation tools such as those described in Section 4.8. An algorithm developed by Sapidis identifies automatically the offending knot, based on fairness indicators [81]. For each knot, a local indicator of fairness Z_i is defined

$$Z_i = \left|\kappa'(t_i^-) - \kappa'(t_i^+)\right| \qquad i = 3,\ldots,n \tag{4.39}$$

where $\kappa'(\tau)=d\kappa/ds$ is the derivative of the curvature with respect to the arc length.

As an approximation to the derivative with respect to the arc length, the derivative with respect to the parameter will be used:

$$\frac{\partial K}{\partial t} = \frac{\partial}{\partial t}\left(\frac{|r_{tu} \times r_{tt}|}{|r_t|^3}\right)$$

$$\frac{\partial K}{\partial t} = \frac{\frac{\partial}{\partial t}\left(|r_t \times r_{tt}|\right)|r_t|^3 - 3|r_t \times r_{tt}||r_t|^2 . |r_{tt}|}{|r_t|^6}$$

in which

$$\frac{\partial}{\partial t}\left(|r_t \times r_{tt}|\right) = X_t\left(Y_{tt} - Z_{uu}\right) + Y_t\left(Z_{ttt} - X_{ttt}\right) + Z_t\left(X_{ttt} - Y_{ttt}\right)$$

A global measure of the fairness of the entire curve is given the indicator ξ defined by

$$\xi = \sum_{i=3}^{n} z_i \qquad (4.40)$$

This method, due to the B-spline basis is a local method.

4.9.2 Surface fairing

Huanzong [82] proposed an algorithm based on the minimisation of the elastic strain energy of surfaces modelled as a non-rectangular mesh of elastic beams attached by elastic springs.

Optimisation techniques have been used successfully for surface fairing. The main issues are the definition of the objective function and of the constraints.

Ferguson [83] introduced the use of optimisation techniques for shape control in parametric curve and surface fitting problems. In the work of Andersson et al [84] is tackled the problem of creating convex surfaces with prescribed smoothness and distance to a set of given points less than a prescribed tolerance. This non-linear problem is solved by linearisation, e.g., by reducing it to a sequence of linear programming (LP) problems. Since, the matrices generated are of difficult solution by the standard LP method, the simplex method, a less explored method, the Karmarkar algorithm [85], is proposed.

In a method presented by Lott and Pullin [86] the control points of a B-spline surface are automatically corrected by a non-linear *constrained* optimisation algorithm. The objective function used as a measure of the fairness of the surface was derived in

analogy to the strain energy of a rectangular elastic plate and is a function of the principal curvatures κ_1 and κ_2

$$Q = \int_L \left(\kappa_1^2 + \kappa_2^2\right) du(dv) \tag{4.41}$$

The constraint used limits each point to be moved only a distance ε along the normal to the surface at the node values corresponding to the control point and is given by

$$E = \int_L \left[\left(x - x_0\right)^2 + \left(y - y_0\right)^2 + \left(z - z_0\right)^2\right] du(dv) < \varepsilon \tag{4.42}$$

Liu [87] took a similar approach but the objective is a quadratic function of both position and curvature.

$$\sigma = \sum_{i=1}^{n} w_{pi}\left(C(t_i) - D_i\right)^2 + w_m \int_{t_1}^{t_n} \left(\frac{\dot{C}(t) \times \ddot{C}(t)}{|C(t)|^3} \right)^2 dt \tag{4.43}$$

The control points are also constrained to move along straight lines, but now these lines can be selected by the user according to the circumstances. If necessary some control points can be fixed.

As the development of a surface fairing method such as these was not in the purpose of the present work, only a few very rough methods were implemented, using the curve fairing methods presented in the previous Section to fair separately the curves that form the rows and columns of the surface's control grid.

4.10 Surface/surface intersection

The intersection of two parametric surfaces, $s(u,v)$ and $r(w,t)$, is the locus of the points of both surfaces that satisfy the equation

$$s(u,v) = r(w,t) \tag{4.44}$$

Assuming that a point on the intersection is known, the intersection curve can be traced. Knowing that on the intersection curve the normal tangent must be the same on both surfaces

$$\frac{ds}{du}du+\frac{ds}{du}dv=\frac{dr}{dw}dw+\frac{dr}{dt}dt \tag{4.45}$$

The points on the intersection curve can be traced from the previous point and from the tangent values obtained by solving the following simultaneous differential equations

$$\begin{aligned}\frac{du}{ds}&=-\varphi_1(u,v,w,t)f_1\\ \frac{dv}{ds}&=-\varphi_1(u,v,w,t)g_1\\ \frac{dw}{ds}&=-\varphi_2(u,v,w,t)f_2\\ \frac{dt}{ds}&=-\varphi_2(u,v,w,t)g_2\end{aligned} \tag{4.46}$$

in which

$$\begin{aligned}\varphi_1(u,v,w,t)&=\pm\left[E_1f_1^2-2F_1f_1g_1+G_1g_1^2\right]^{-1/2}\\ \varphi_2(u,v,w,t)&=\pm\left[E_2f_2^2-2F_2f_2g_2+G_2g_2^2\right]^{-1/2}\end{aligned} \tag{4.47}$$

and

$$\begin{aligned}f_1&=\left(n_2.\frac{ds}{dv}\right) \qquad g_1=\left(n_2.\frac{ds}{du}\right)\\ f_2&=\left(n_1.\frac{dr}{dt}\right) \qquad g_2=\left(n_1.\frac{dr}{dw}\right)\end{aligned} \tag{4.48}$$

The above method is time consuming and to obtain quick results in two very common types of intersections used during hull modelling, simplified methods can be developed.

4.10.1 Simplified intersections of free surface with sets of orthogonal planes

During the modelling process it is necessary to obtain the intersections of the hull with sets of planes parallel to the orthogonal reference planes to obtain the transverse sections, waterlines and buttocks, that are the lines traditionally used to represent the shape of the hull through drawings.

A very quick way to obtain the intersections with families of orthogonal planes is to use the contouring techniques described in Section 4.8.5. First a set of surface points is computed over a rectangular parametric grid. Then, a scalar field is created associating each point *P(u,v)* with one of the respective coordinates (x,y,z) and the corresponding iso-line contours can be obtained. With this method, although only the vertices of each triangle are guaranteed to belong to the surface, it is easy to increase the accuracy, just by increasing the grid density. In Fig. 4.51 are represented the real surface intersection segment $\overline{P_1P_2'}$ and the corresponding contour segment $\overline{P_1P_2}$ obtained by this method, which is, in fact, the projection of $\overline{P_1P_2'}$ in the plane defined by the triangle *ABC*.

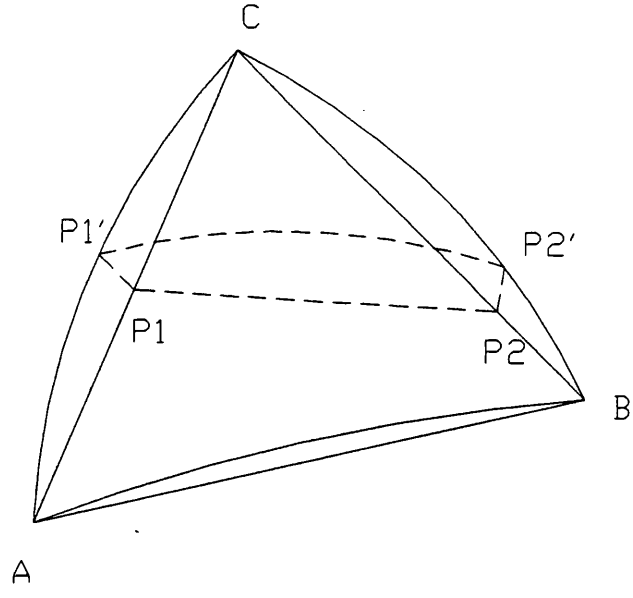

Figure 4.51 Contour segment in triangular element

The regular nature of the triangulation and the good results obtained did not seem to justify the extra computing time required by the modification presented in [88]. To compute the intersection of a contour with a triangle edge, the modified algorithm obtains the parameters *(u,v)* of the intersection point by linear interpolation over the parameter values on the vertices, computing then the corresponding surface point, instead of using linear interpolation directly between the coordinates of the two vertices.

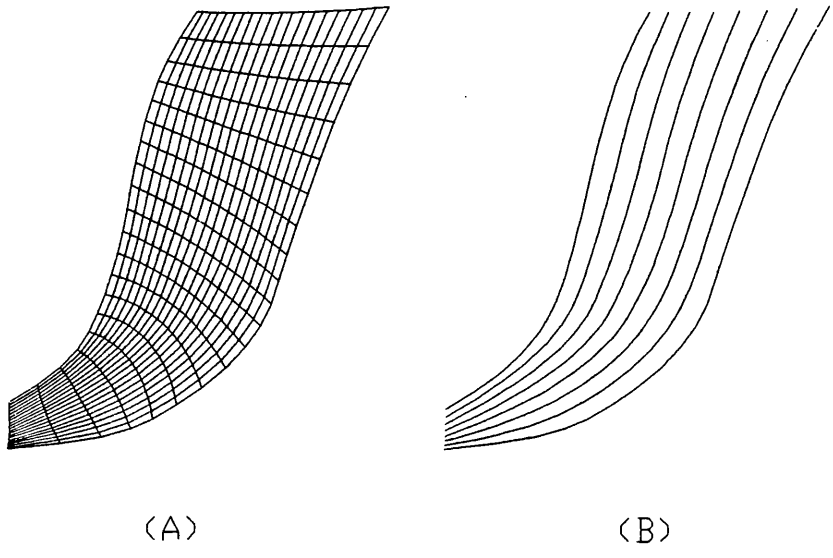

Figure 4.52 Transverse sections of a surface obtained by contouring

In Fig. 4.52 a set of transverse sections (Fig. 4.52-B) has been generated from a surface patch (Fig. 4.52-A) by contouring.

4.10.2 Simplified intersection of a free surface with a conic surface

During the modelling of a ship hull, several cases of a particular surface intersection case, the intersection of a general free surface with a conic, occur. Considering the possible simplifications a quicker algorithm can be developed.

Assuming that the conic surface is of order 2 in the other parametric direction, which is most common with hawse pipes, thruster tunnels, etc., that means that the generatrices are straight lines. The surface/surface intersection reduces in this case to a line/surface intersection. As in the present approach the hull surface is always generated as a Cartesian tensor product, it has an intrinsic rectangular structure that can be easily approximated by a triangular mesh. With this second simplification, the intersection problem reduces to the intersection of a straight line with a flat triangular surface (Fig. 4.53).

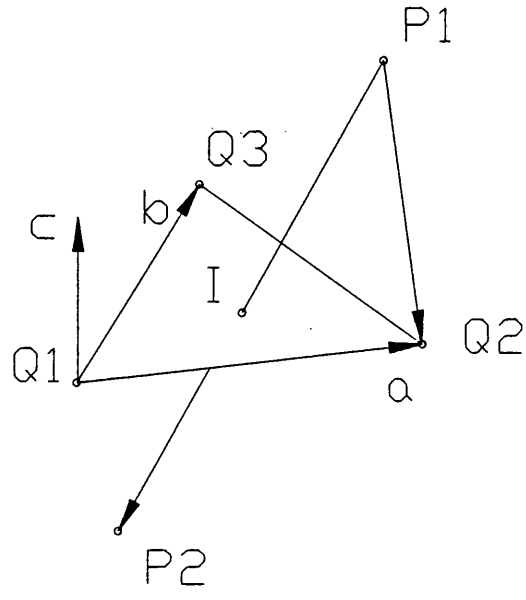

Figure 4.53 Line/Triangle intersection

For each triangular element $[Q_1Q_2Q_3]$, the vectors $\vec{a}, \vec{b}, \vec{c}, \vec{d}$ *and* $\vec{e}$ can be defined as follows

$$\begin{aligned} \vec{a} &= Q_2 - Q_1 \\ \vec{b} &= Q_3 - Q_1 \\ \vec{c} &= \frac{\vec{a} \times \vec{b}}{\left|\vec{a} \times \vec{b}\right|} \end{aligned} \tag{4.49}$$

$$\begin{aligned} \vec{d} &= P_2 - P_1 \\ \vec{e} &= Q_2 - P_1 \end{aligned} \tag{4.50}$$

The equation of the plane defined by the points $Q_1Q_2Q_3$ is given by

$$\vec{c} \cdot r = \vec{c} \cdot Q_1 \tag{4.51}$$

where $r(\lambda) = (x, y, z)$.

The parametric equation of the line P_1P_2 is given by

$$r(\lambda) = P_1 + \lambda.d$$

Replacing in (4.51)

$$\lambda_1 = \frac{\vec{c}.\vec{e}}{\vec{c}.\vec{d}}$$

To assure that there is an intersection between the line segment P_1P_2 and the triangle $\Delta Q_1Q_2Q_3$ the following condition must verify

$$0 \le \lambda \le 1$$

If that is verified, the parameter r is fixed and s is obtained by

$$s = s_1 + \lambda.\left(s_2 - s_1\right) \quad if\ 0 \le \lambda \le 1$$

The intersection point R can be obtained replacing λ in the line equation

$$R = P_1 + \lambda.d$$

$$\alpha = \frac{c \cdot (r \times b)}{c \cdot (a \times b)}$$

$$\beta = \frac{c \cdot (r \times a)}{c \cdot (b \times a)}$$

The line P_1P_2 intersects the triangle $\Delta Q_1Q_2Q_3$ if the following conditions are satisfied

$$\alpha \ge 0$$
$$\beta \ge 0$$
$$\alpha + \beta \le 1.0$$

The parameters corresponding to the intersection point I are

$$t = \alpha.t_1 + \beta.t_2 + (1 - \alpha - \beta)\, t_3$$
$$u = \alpha.u_1 + \beta.u_2 + (1 - \alpha - \beta)\, u_3$$

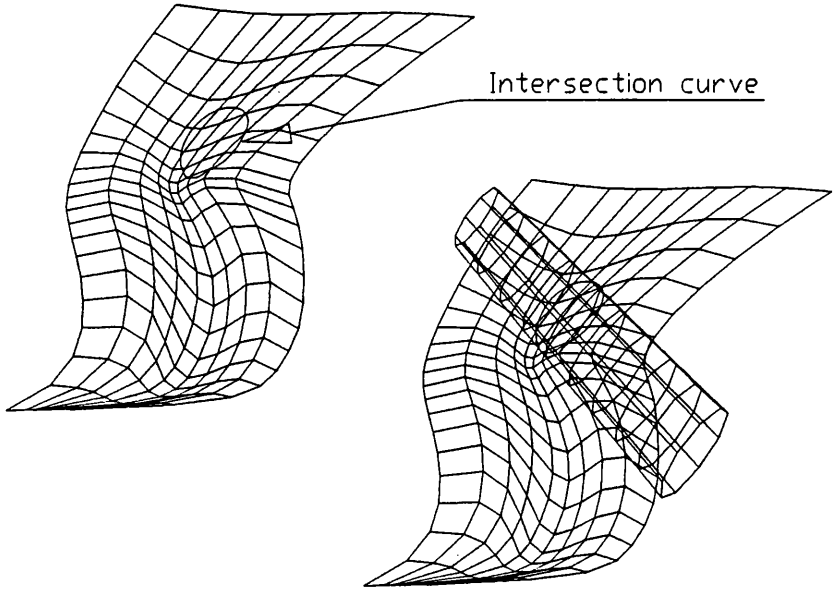

Figure 4.54 Surface/Cylinder intersection

The results of the presented simplification (Fig. 4.54) seem to have enough accuracy for the required application.

4.11 Area of a surface patch

The area of a rectangular element of a surface defined by four isolines (Fig. 4.55) can be computed by

$$\begin{aligned} dA &= |AB||AD|\sin\alpha \\ &= |AB \times AD| \end{aligned} \tag{4.52}$$

simplifying, it can be assumed that

$$AB \cong r_v dv$$
$$AD \cong r_u du$$

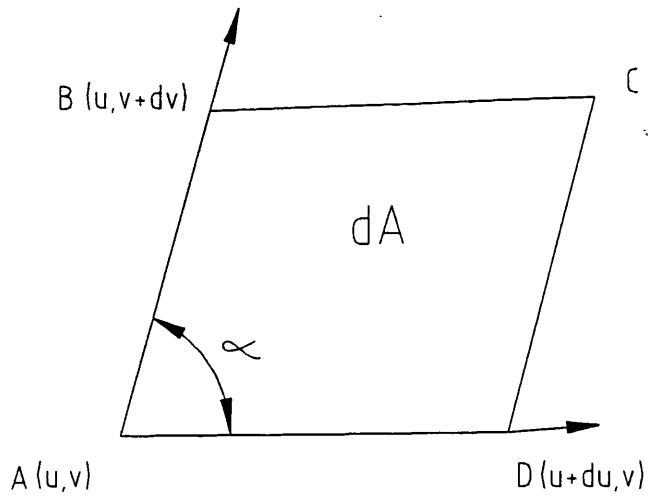

Figure 4.55 Area of rectangular surface element

replacing in (4.52) and from the first fundamental form constants (4.22)

$$dA = |r_u \times r_v| dudv$$
$$= \sqrt{EG - F^2}\, dudv$$

The area of a surface patch can finally be obtained by integrating

$$A = \int_{u1}^{u2} \int_{v1}^{v2} \sqrt{EG - F^2}\, dudv \qquad (4.53)$$

The computation of the area of a surface patch can be obtained very quickly by an approximation method avoiding the integration. Generating a triangulation over the surface, as described in Section 4.9.4, with a resolution corresponding to the accuracy required, the area can be approximated by the accumulation of the area of the triangular elements.

$$A_S = \sum_i A_{Ti}$$

The area of each triangular element (Fig. 4.56) can be computed by

$$A_T = \frac{1}{2} abs\left(\vec{n} \cdot \left(\vec{v}_1 \times \vec{v}_2\right)\right)$$

on which $\vec{v}_1$ and $\vec{v}_2$ are the vectors defined by the triangle vertices

$$\vec{v}_1 = P_{i+1,j} - P_{i,j}$$
$$\vec{v}_2 = P_{i,j+1} - P_{i,j}$$

and $\vec{n}$ is the normal unit vector of the plane defined by the triangle

$$\vec{n} = \frac{\vec{v}_1 \times \vec{v}_2}{|\vec{v}_1 \times \vec{v}_2|}$$

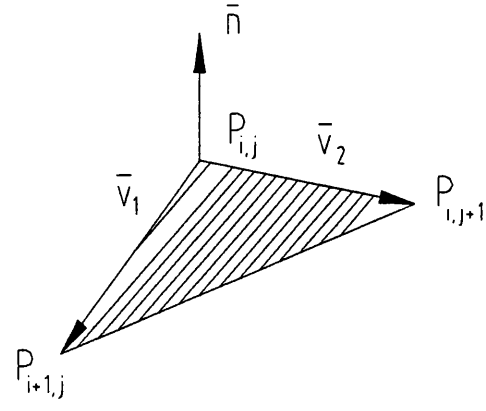

Figure 4.56 Area of triangular element

In order to check the degree of accuracy obtained with this approximation, the area of a cylindrical surface with a radius of 2.0 and a length of 6.0 was computed for several resolution values. In Table 4.1, the computed values are compared to the exact area value (75.398).

Table 4.1 Evaluation of error in area computation

Grid Resolution U direction	Grid Resolution V direction	Area	Error [%]
10	10	73.872	2.02
15	15	74.764	0.84
20	20	75.053	0.46
30	30	75.250	0.20

5. Application examples

The algorithms described in Chapter 4 were implemented as a modular system developed on a PC micro-computer. The system was developed over a commercial CAD system [89,90], whose graphics engine provides the elementary graphic primitives (points, lines, polylines, text, etc.), as well as the basic editing (delete, move, etc.) and viewing (pan, zoom, view points, etc.) capabilities. The modules are accessible from a hierarchical menu structure. The code was written in ANSI C.

The system was designed to create and edit NURBS curves and surfaces of arbitrary order. The parameterisation can be uniform, chord-length or centripetal, selected by the user. Cubic splines with selected boundary conditions and bi-linear or bi-cubic Coons surfaces can also be generated. For both curves and surfaces interactive editors were implemented by joining the basic algorithms under a common user interface, which allows the user to modify the shape and see the results in real time, while optionally checking the curvature distribution.

To be able to manage the input data and the resulting curves and surfaces necessary to model the hull, a simple database was implemented. The entities considered are:

- lines
- curves
- surfaces

The line entities, are the input data and consist on sets of ordered points defined by their 3D coordinates, with or without tangent vector information. The curve entities, store the NURBS curves generated from the data points, and are defined by the order, number of control points, the array of 4D control points, the number of knot values and the knot vector. The surface entities, store the NURBS surfaces created and are defined by the order, number of knots, number of control points and knot vectors for

each of the two parametric directions and an array storing the definition of the whole grid of 4D control points. The number of curves and surfaces is unlimited.

Input data (digitised or not) and generated curves are graphically represented by points connected by straight lines. Surface patches are represented by a grid of iso-parametric lines. The number of line segments and the resolution of the grid of iso-parametric lines used to display curves and surfaces respectively, are selected by the user.

In the following sections, the software system developed and some practical examples of the utilisation of the tools developed to model ship forms are presented. Finally, a comparison is made with a commercial package, AutoShip, that in its latest releases has also adopted the use of NURBS curves and surfaces to model the ship hull.

In general, only half the hull of the ship used was digitised and no special criterion were defined for that task, regarding the quantity and distribution of the input points.

5.1 Description of the system

The system was developed as a set of modules organised as software layers, so that each layer only uses the functions contained in the layers underneath.

In the figure 5.1 is presented the layer structure of the system. The contents of the layers are as follows:

Memory manag. Management of dynamic allocation of memory for single and multi-dimensional arrays.

General tools Set of general purpose tools, including manipulation of strings.

Matrix algebra Functions dealing with the required operations on matrices, such as matrix multiplication, identity matrix, matrix inversion, matrix inversion of diagonal banded matrices.

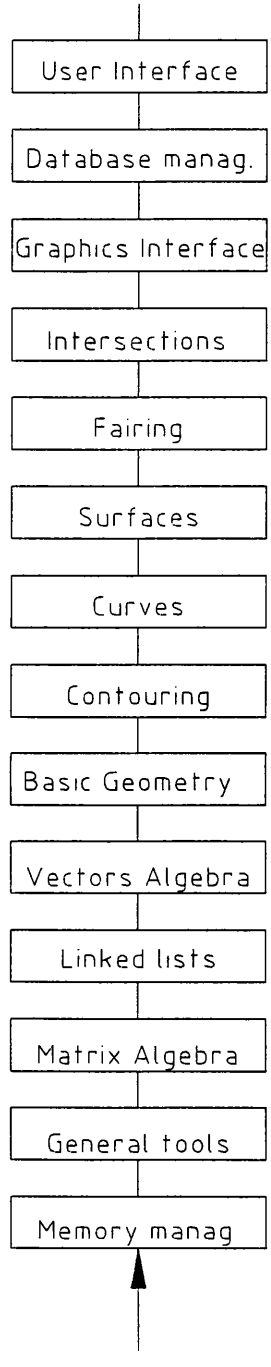

Figure 5.1 Layer structure of the software system developed

Linked lists Management of linked lists, to deal with the dynamic storage of entities of unknown number of elements.

Vectors algebra Operation with vectors such as dot products, external products, addition of a vector to a point, modulus of a vector, normalisation of a vector, and so on.

Basic geometry Functions providing capability to manipulate arrays of 2D or 3D points, transformation of coordinates, geometric transformations such as rotations about a specific or generic axis, translations and symmetries, and so on.

Contouring Functions to compute contours of iso-values of selected properties or coordinates, over a triangulated domain.

Curves Functions implementing the basic algorithms related to the computation of points, first and second derivatives of cubic spline, B-spline and NURBS curves. Computation of curve curvature. Basic algorithms for B-spline curves such as degree raising, knot insertion and knot removal.

Surfaces Functions implementing the basic algorithms related to the computation of points, first and second derivatives of Coons patches, B-spline and NURBS surfaces. Computation of several types of surface curvatures. Computation of isophotes. Basic algorithms for B-spline surfaces such as degree raising, knot insertion and knot removal.

Fairing Functions for the fairing of curves and surfaces.

Intersections Intersection of free form surfaces with conic surfaces.

Graphics interface Functions providing the capability to display graphical entities such as points, lines, polylines, and surface meshes. Interactive edition of curves and surfaces. Representation of the surface

curvatures distribution by means of coloured maps or contours.

Database manag. Tools for the management of a database to store the different types of entities describing the hull geometry of a ship.

User interface Functions providing the user interface, dealing with menus, dialogue boxes, validation of user input and so on.

5.2 Modelling of a tanker of about 80,000 TDW

The first approach to the utilisation of the system was influenced by the traditional approach and so sequences of points on transverse sections and longitudinal contours fore and aft, over a common set of waterlines were used as the initial input.

The preliminary bodyplan of a crude oil tanker was digitised, with a total of 24 lines, divided in 20 transverse sections, two knuckle lines, forward, and the bow and stern profile contours (Fig. 5.2). No special criteria was defined for the digitising task, so that the limitations of the fitting and approximation algorithms could be detected.

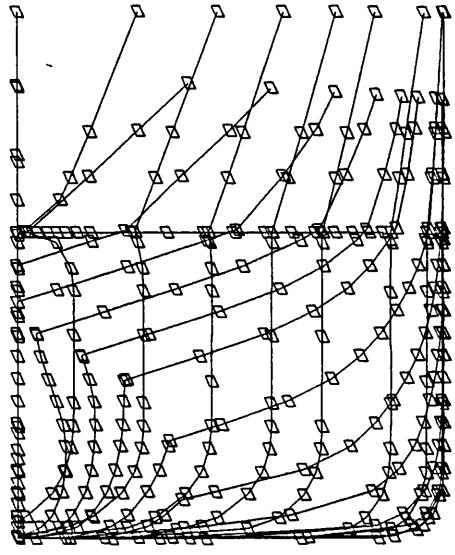

Figure 5.2 Digitised lines

The ship has the following main dimensions:

Length, overall	248.00 m
Length, between perpendiculars	240.00 m
Breadth, moulded	42.00 m
Depth, moulded	19.20 m
Draft, design	13.25 m

Taking into consideration the characteristics of the hull, it was decided, for modelling purposes, to divide it into five main zones, bow, forebody, midship, aftbody and stern, which were considered separately.

5.2.1 Bow

The bow zone was modelled from a set of 5 sections, a knuckle line and the bow profile (Fig. 5.3). First, the lines were approximated by B-spline curves of order three using the algorithm described in Section 4.3.3. The approximating and fitting algorithms are not capable of dealing directly with knuckles, and so, the profile and sections were edited individually in order to insert the knuckle points, using the knuckle line as guidance.

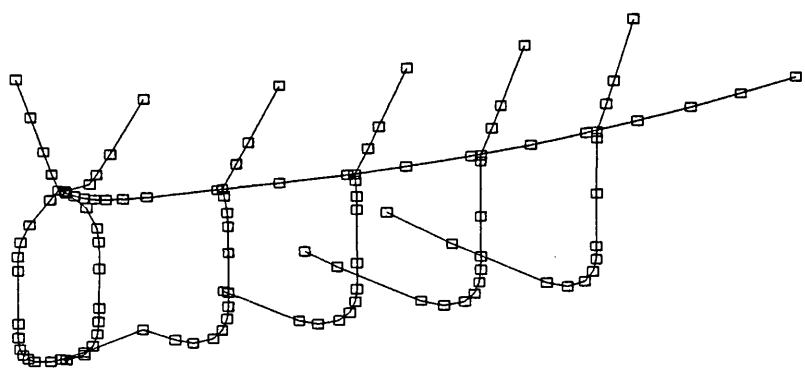

Figure 5.3 Input half-sections, knuckle and profile

In a first approach, a single surface path of order three was generated by lofting over the edited transverse sections and bow profile curves. As it can be seen from the

resulting surface (Fig. 5.4), not only was the knuckle line not well defined but also the shape of the surface near the bow profile was not "round" as expected.

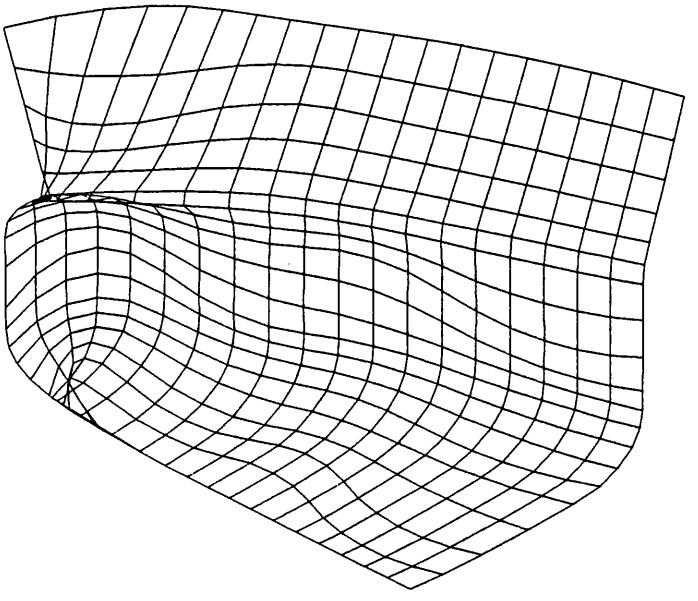

Figure 5.4 Bow represented by single surface patch

The generation of patches by lofting has a limitation which is the impossibility of imposing boundary conditions in the parametric direction opposite to the section curves. So, in an attempt to obtain a rounder shape in the bulb zone, the half section curves were mirrored in relation to the longitudinal symmetry plan, in order to obtain the full sections, portside and starboard (Fig. 5.5).

Another remark that can be made, is that the obtained surface is very complex, because it has a large number of rows of control points, which is a consequence of the compatibilisation process applied to the input curves. Therefore, it can be concluded that to keep the resulting surfaces simpler, the number of control points of each curve should be kept to a minimum and their position should also be kept on corresponding contour positions.

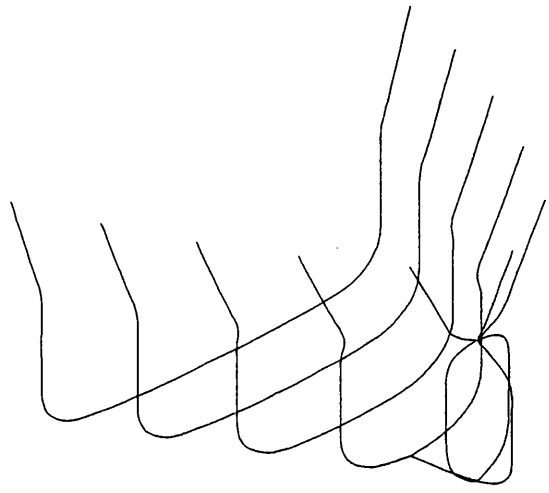

Figure 5.5 B-spline curves fitted to input data

Lofting the 10 transverse sections and the bow contour the resulting surface patch seems to model better the bulb area, but the longitudinal knuckle is still not well defined (Fig. 5.6).

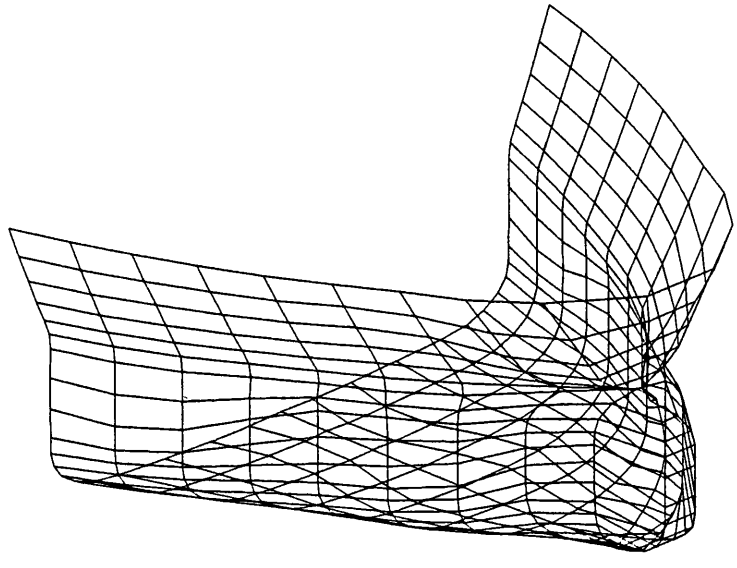

Figure 5.6 Bow represented with single surface

As an alternative, the knuckle line was used to split the transverse sections (ref. Section 4.6.2), lofting separately the lower and upper bow areas (Fig. 5.7).

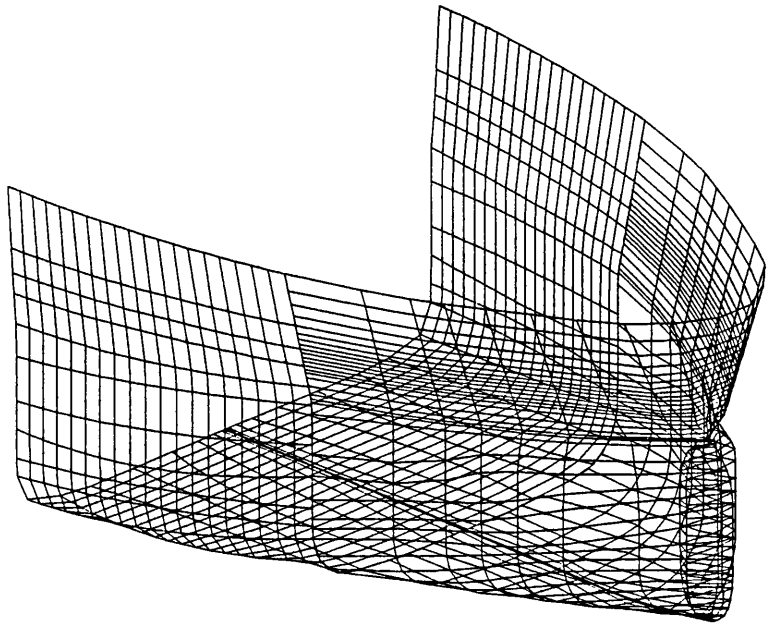

Figure 5.7 Bow represented by 4 surface patches

The resulting lower area forward, is still very rough, mainly due to the poorly defined shape of the bulb (Fig. 5.8).

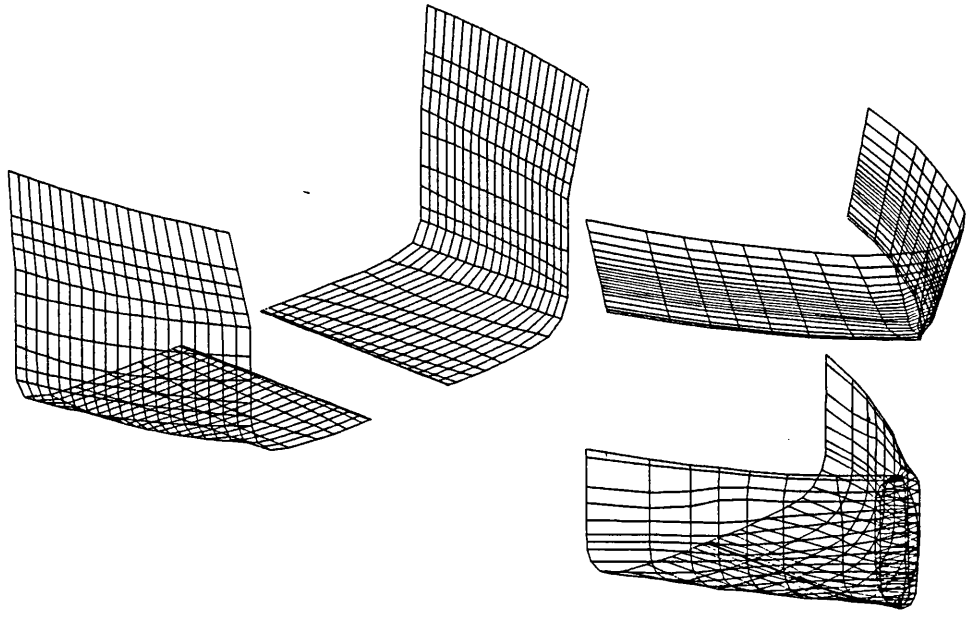

Figure 5.8 Exploded view of bow patches

5.2.2 Forebody

The transition zone between the bow and the midship zone, which has a regular change of section shape was modelled by a single surface patch lofted over 5 transverse sections (Fig. 5.9).

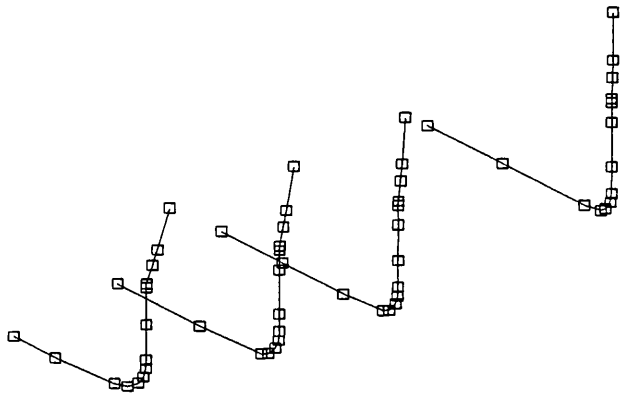

Figure 5.9 Lines defining the forebody

The resulting surface presents a good behaviour easily seen by the regular distribution of the isoparametric lines (Fig. 5.10).

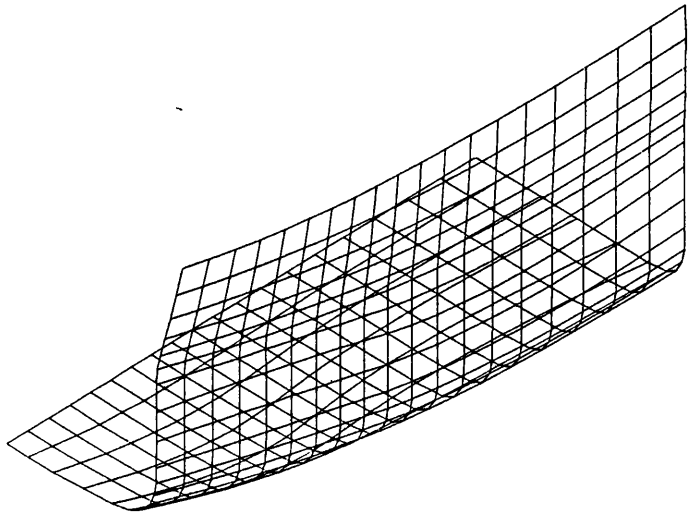

Figure 5.10 Forebody surface patch

5.2.3 Midships

The midship zone of the hull was defined by 6 transverse sections (Fig. 5.11).

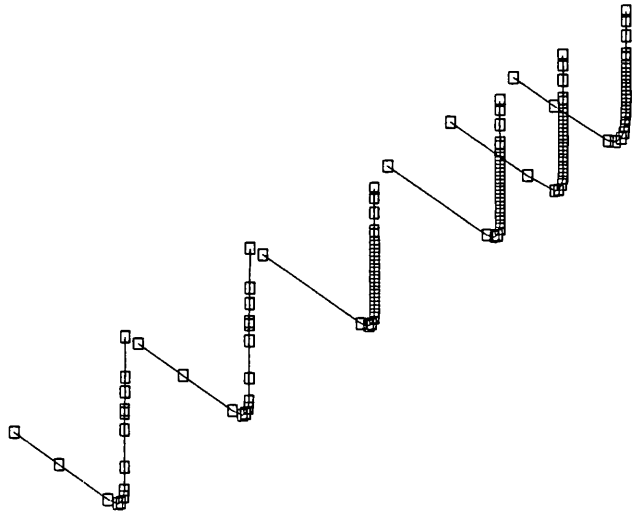

Figure 5.11 Lines defining the midship

Some of the oscillations obtained in the resulting surface patch (Fig. 5.12) are due to the very different number of input points in the after and forward sections. It is also visible that the distribution of the bottom isoparametric lines is irregular and the bottom is not flat all over as it should be. Checking the input lines, the cause can be traced to the insufficient number of points in the bottom, on two of the intermediate sections.

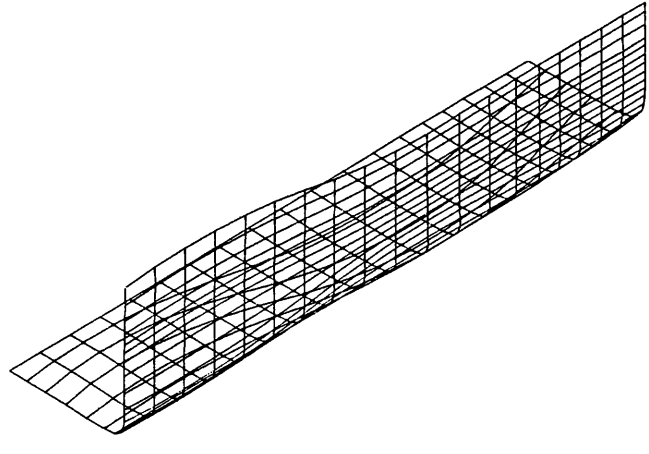

Figure 5.12 Midships surface patch

5.2.4 Aftbody

The aft body was defined by 6 transverse sections (Fig. 5.13). The curves fitted are shown if Fig. 5.14 and the resulting surface in Fig. 5.14.

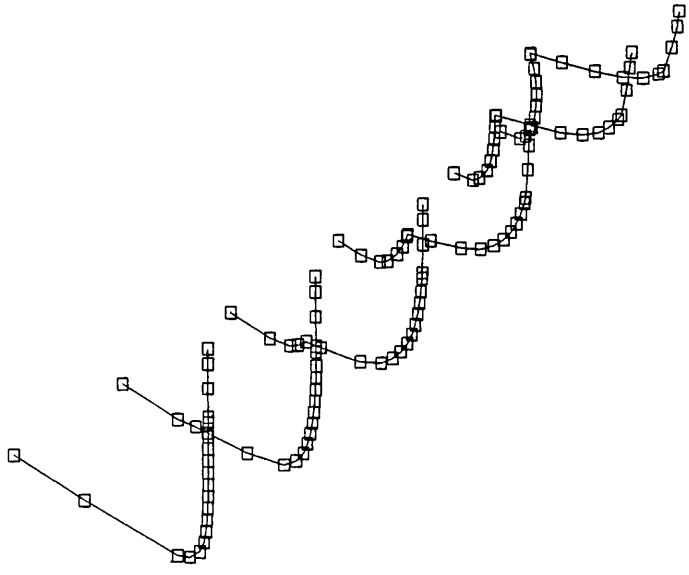

Figure 5.13 Lines defining the aftbody

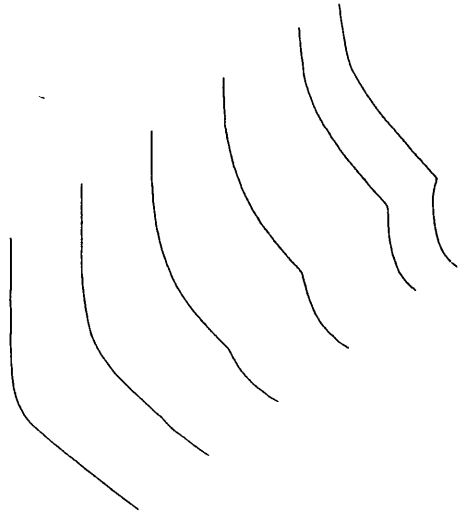

Figure 5.14 Curves defining the aftbody

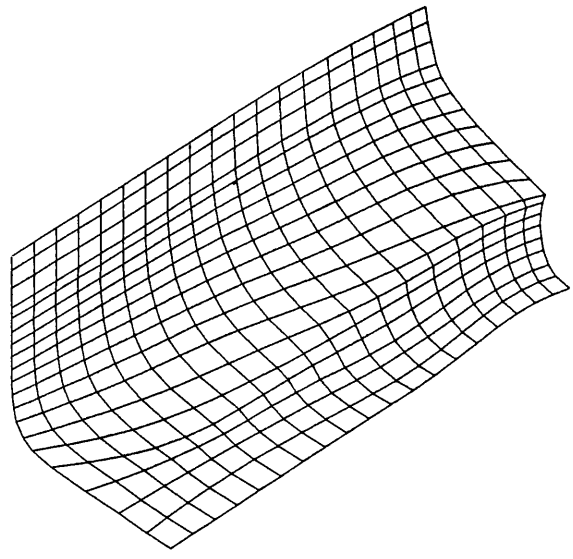

Figure 5.15 Aftbody surface patch

5.2.5 Stern

The stern of the hull has a more complex shape and the description of the knuckle lines was not available. The input data used was the stern contour and 5 transverse sections (Fig. 5.16).

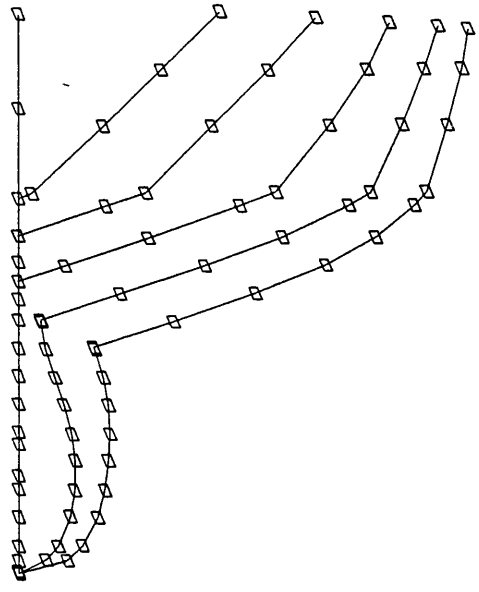

Figure 5.16 Input stern lines

As the stern knuckle lines are not clearly defined, each of the fitted curves was edited separately and two knuckles were defined so that the resulting curves matched reasonably the input lines. Then the curves were split by the knuckle points into 3 segments (Fig. 5.17).

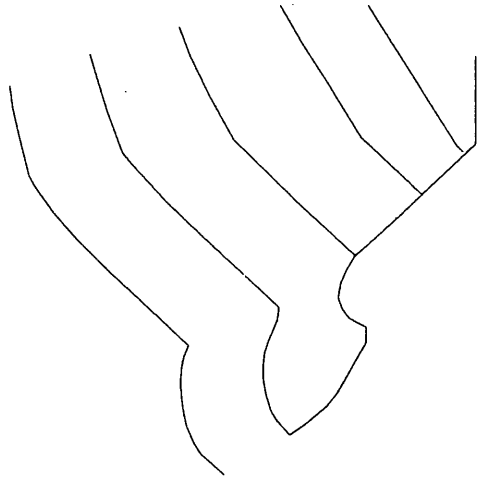

Figure 5.17 Stern curves

Lofting separately the curve segments, 3 surface patches were generated. The stern panel was defined by the stern contour and the aft transverse section (Fig 5.18).

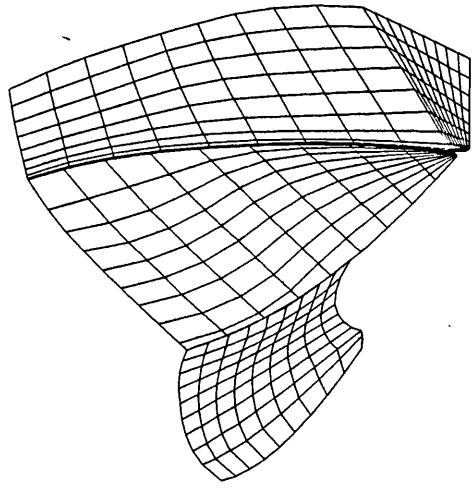

Figure 5.18 Stern surface patches

The complete stern was modelled by the 4 surface patches shown separately in Fig. 5.19.

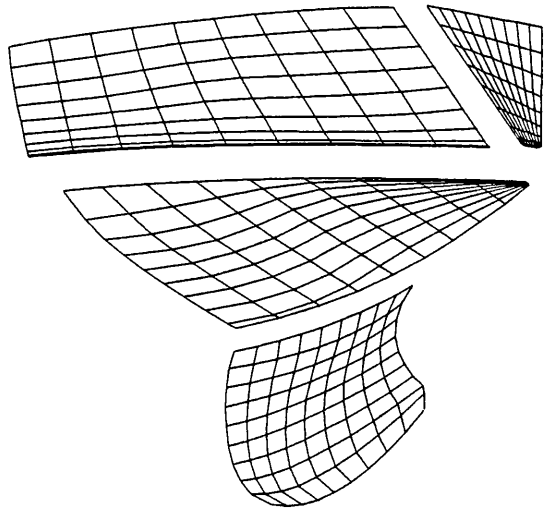

Figure 5.19 Exploded view of stern patches

To represent the complete ship the surface patches were joined and mirrored to obtain a full model (Fig. 5.20).

5.2.6 Conclusions

In general, the quality of the surfaces obtained is very poor and some reasons found are as follows:

- A procedure for generating the hull surface must not use too much information to start with. Surfaces generated from lofting too many curves are not closer to the desired shape, on the contrary, they start to present undesired shape characteristics. The same principle also applies to curves, which should not be generated from an excessive number of points.
- It is difficult with the tools developed so far, to obtain good results with an exclusively automatic procedure, i.e, avoiding the manual editing of the supporting curves.

It also seems a logical conclusion that an efficient surface fairing algorithm is a fundamental complement of a hull surface modeller.

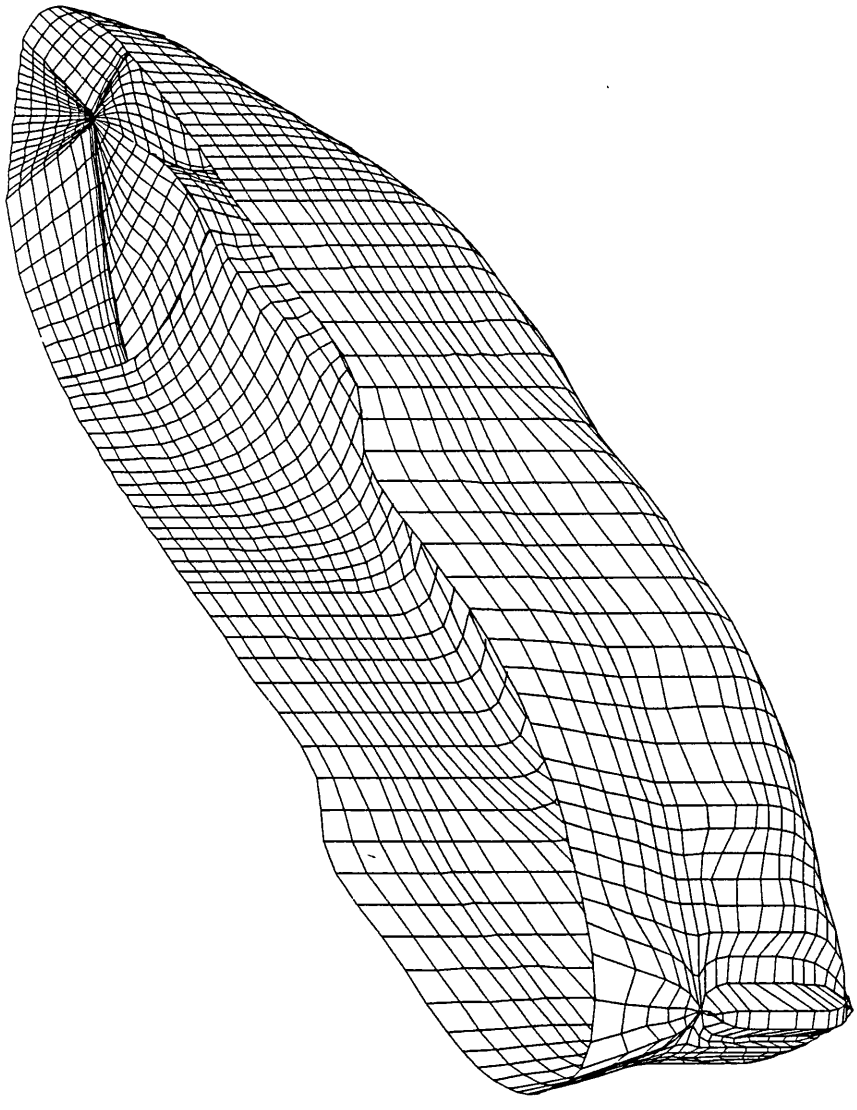

Figure 5.20 Model of complete ship

5.3 Modelling of the bow of a tanker of about 160,000 TDW

The first test was based on a preliminary body plan from which the lines used as input were selected, using almost all the points available from the grid of sections and waterlines. To check if the roughness of some areas of the final model were only due to the poor quality of the input, another test is was made, trying to model the bow of a 160,000 TDW tanker, provided with knuckle lines and a bulb, using the information from a fully developed body plan. The ship has the following main dimensions:

Length, overall	270.00 m
Length, between perpendiculars	260.00 m
Breadth, moulded	48.25 m
Depth, moulded	24.10 m
Draft, design	16.25 m

The bodyplan used, corresponding to a more advanced stage of the ship design, not only has more lines represented (transverse sections on every building frame) but also the shape quality of each line is better. However, and learning from the first example, on which the volume of information used did not correspond to an equivalent quality of the results, in this example, each curve was edited after the curve fitting to remove as many points as possible without losing the shape characteristics.

5.3.1 Boundary and knuckle curves

The boundary lines selected for input, were the longitudinal stem contour and three knuckle lines forward (Fig. 5.21).

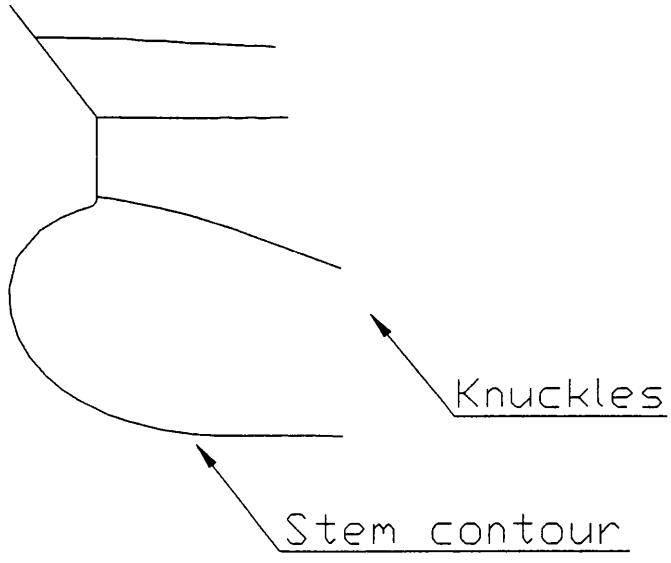

Figure 5.21 Boundary and knuckle curves, forward

5.3.2 Transverse sections

A set of 30 transverse sections were digitised in the forebody, including extra sections added with a smaller spacing in the bulb area, as shown on Fig. 5.22.

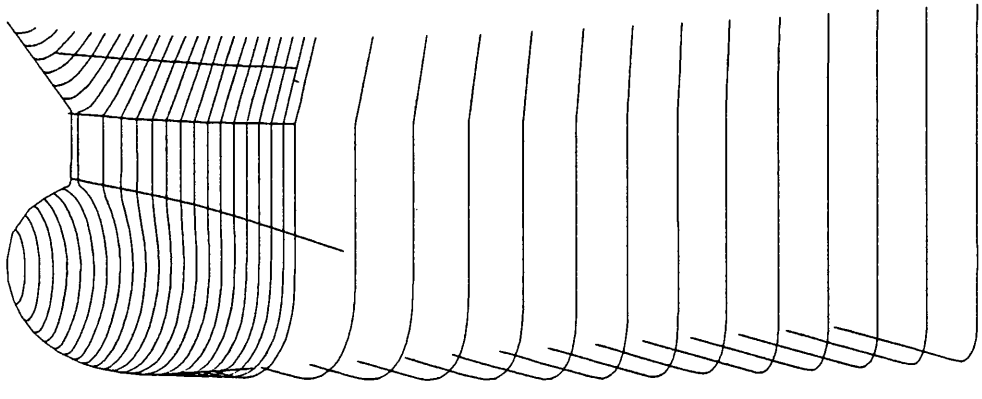

Figure 5.22 Digitised transverse sections (forward)

5.3.3 Surface patch generation

The bow zone, which has a complex shape due to the existence of the two knuckle lines and the bulb was divided into several patches.

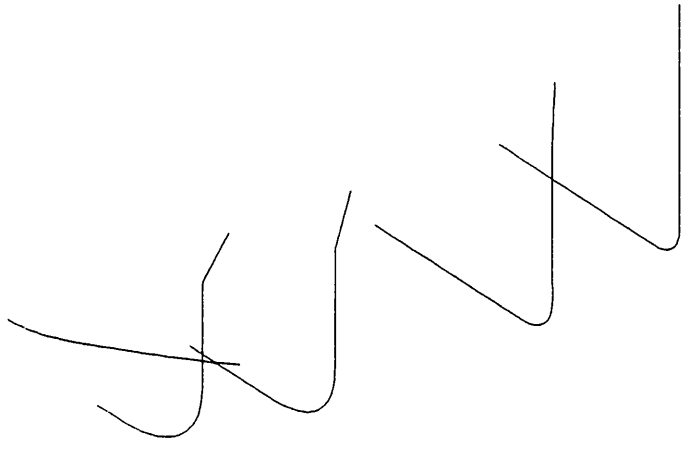

Figure 5.23 Initial curves used to model the forebody surface

In general, knuckle lines should not cross patch boundaries and so the first step was to create a new section coinciding with the beginning of the knuckle line. To obtain this section, a preliminary surface was generated by lofting to the first section forward of the knuckle line beginning (Fig. 5.24).

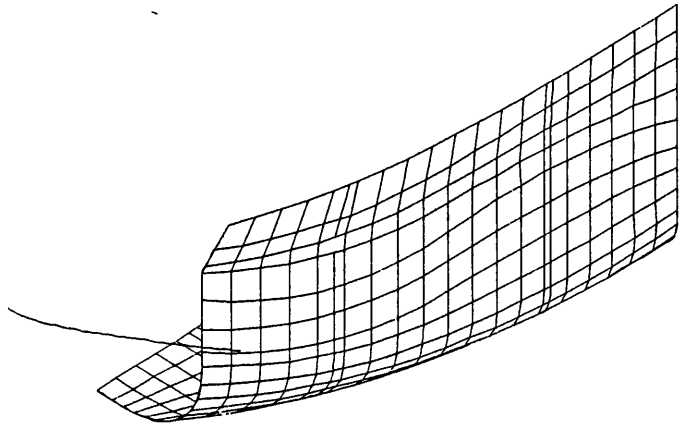

Figure 5.24 Auxiliary surface

With this auxiliary surface generated, a routine using the surface intersection method described in Section 4.11.1 was used to compute a new section coinciding with the beginning of the knuckle line (Fig.5.25). The curve fitted to this section was then used as the boundary between the aft and forward patches.

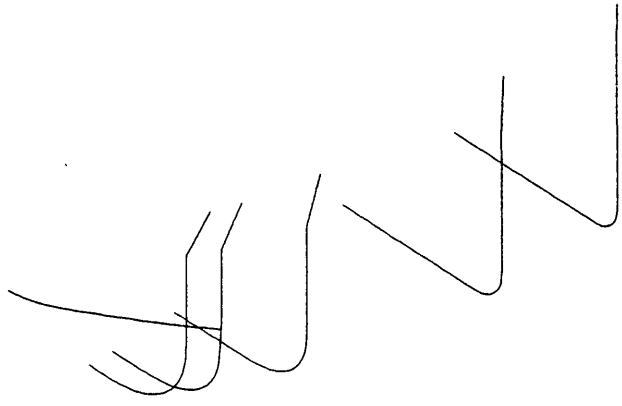

Figure 5.25 New section obtained by surface intersection

The bulb surface was lofted from a set of 4 section curves, the last of which was a dummy section curve, with the shape of a circular arc of 180° with zero radius, used to define the closing tangent point. An arc of 180° is represented by a curve of third order, with five control points as shown on the Fig. 5.26 and a knot vector *{0,0,0,0.5,0.5,1,1,1}* and the weights *{1,0.707,1,0.707,1}*. In this particular case of zero radius, all the five control points will coincide with the centre point of the arc.

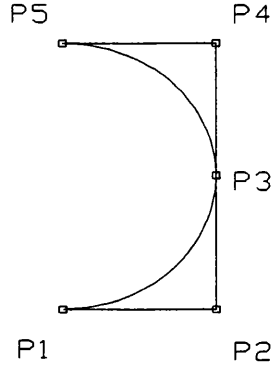

Figure 5.26 Arc of 180 deg. represented by NURBS curve

The resulting surface was then submitted to knot removal which eliminated 10 knots, i.e., 10 rows of the control points grid. Finally, the surface was faired by the least square method presented in Section 4.10.2. and the result, mirrored to show the full bulb, is shown in Fig. 5.27.

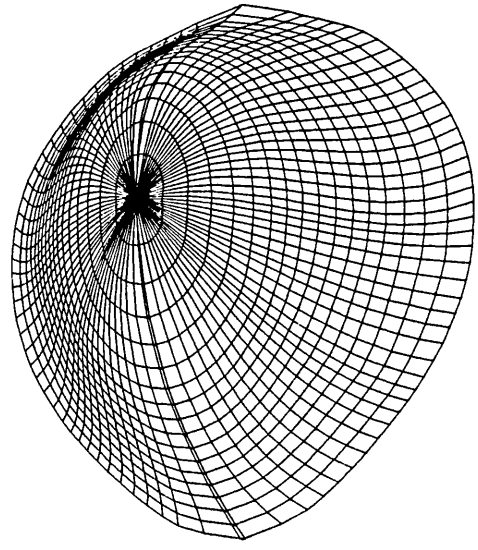

Figure 5.27 Bulb surface mirrored about centreline

The connection between the bulb and the adjacent upper patch presents a gap that the methods developed could not cope with (Fig. 5.28). A better approach would be to generate both surfaces beyond the connection line. The surfaces would then be intersected and the redundant areas removed. This would require the implementation of a surface/surface intersection method and also a data definition capable of describing trimmed surfaces (Refer to Chapter 6).

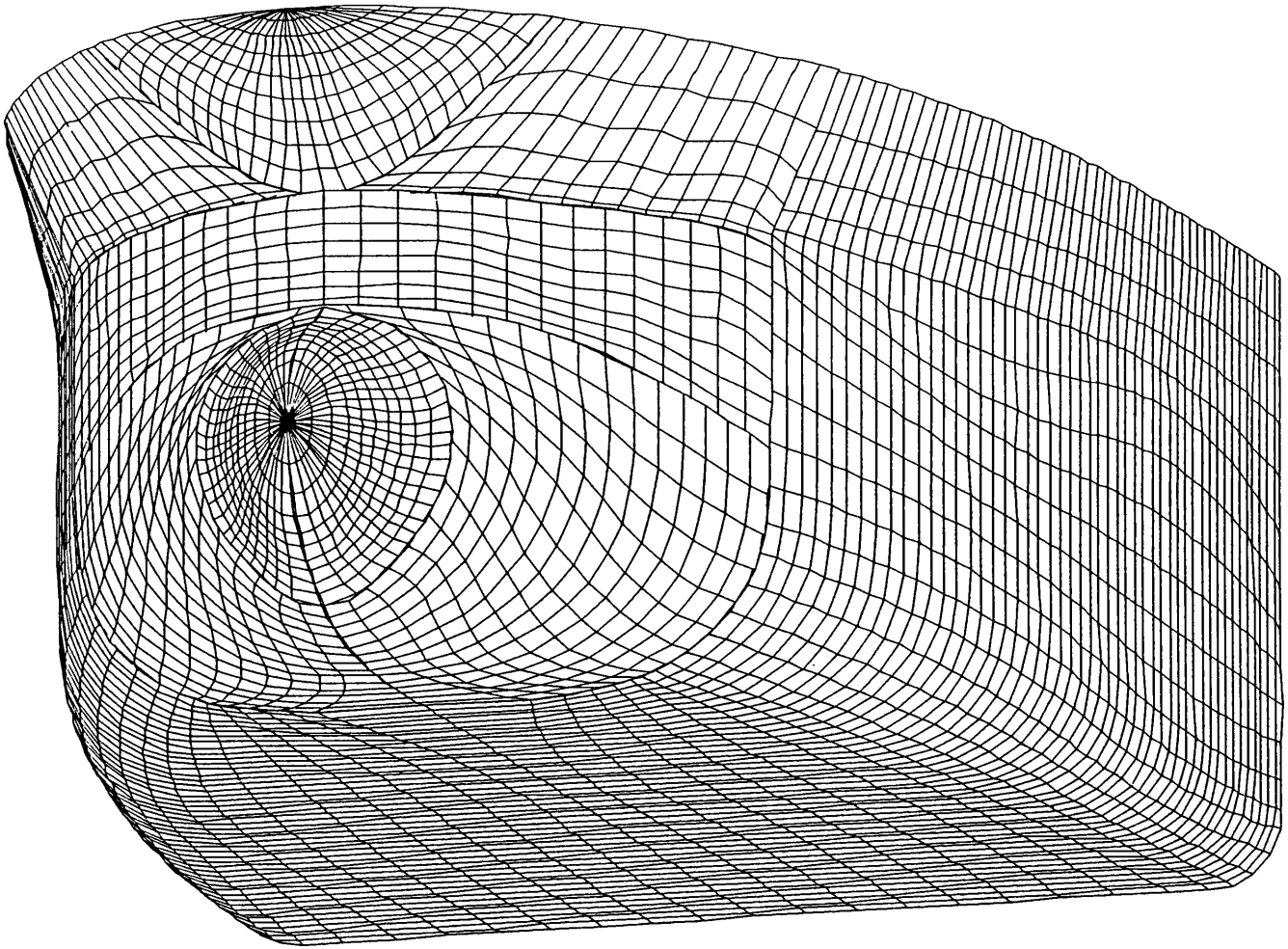

Figure 5.28 Bow modelled with 6 surface patches

5.3.4 Hawse pipe generation and intersection with the hull

For testing the cylinder generation and intersection with the hull, a hawse pipe of 500 mm radius at one end and 900 mm radius at the other diameter was generated and the result with the hidden lines removed is presented in Fig. 5.29.

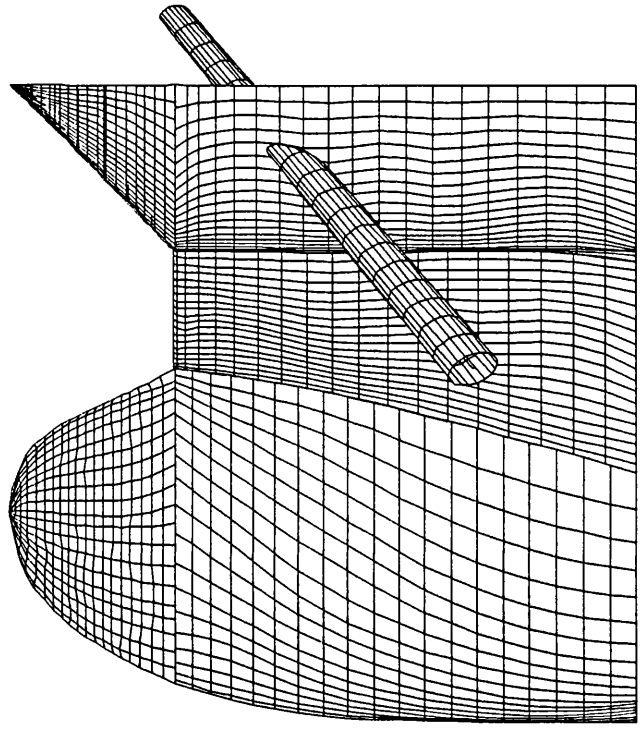

Figure 5.29 Intersection of cylinder with hull

5.3.5 Conclusions

The reduction of the input data and the manual editing and fairing of each supporting curve seemed to improve the quality of the obtained surfaces. Even so, the manual editing of the surface itself is inevitable most of the times, not only to introduce local characteristics which the generating process by itself cannot produce but also to improve the smoothness, whenever the surface analysis detects undesired shape characteristics. As the interactive editing of a surface, even with a reduced control mesh, is difficult to implement and to use, a semi-automatic post-processing fairing procedure would be a better solution.

5.4 Comparison with a commercial package

Due to the difficulties found in representing an existing hull with the developed tools, it was considered interesting to execute the same task with an existing commercial package, fully developed and tested, to evaluate how the available tools and the implemented algorithms perform in similar conditions.

During the development of the present work, the new release 5.3 of AutoShip, a commercial software package designed for the surface modelling of ships, became available.

AutoShip, runs on PC computers, and has been recently updated from a mixed cubic spline/B-spline curves model to a full NURBS surface ship hull representation. Since the algorithms used are not described in the package documentation, the comparison will be made in terms of the results of curve fitting and of the tools for surface generation and analysis.

5.4.1 Curves

Although more oriented to basic hull design, the package includes a module, AutoMatch, whose purpose is the automatic generation of curves and surfaces to match an existing hull described by an offset table.

The input to AutoMatch is composed by a set of transverse sections, sorted from fore to aft. Each section is defined by a set of points described by its 3D coordinates *(x,y,z)* and an attribute code (ch), equal to 1 if the point is a chine, or equal to 0, if not. The procedure automatically fits B-spline curves to each input section. The order and the number of control points generated for each curve is selected by the user and applied to all the curves. The order must be in the range 2-3 and the maximum number of control points generated is 9. This last limitation has been shown to be a problem when trying to use as an input lines digitised from a body-plan drawing.

Curves can be created by defining the coordinates of the control points. Regarding conics, the system allows the creation of circles and circular and elliptical arcs, but is limited to planes parallel to the projecting planes.

For editing the curves, a set of tools including refinement, splitting, joining, mirroring and reversing is available. Segments defined by the extreme points can be made straight. Knuckles can be generated/eliminated on selected points by automatically inserting/deleting repeated control points. The weight of individual control points can be changed manually, although in the documentation the user is not encouraged to use this feature.

The fairness of the curves generated can be evaluated by the inspection of the curvature, either in the porcupine form or by curvature plot display.

There are no automatic curve fairing or curve/curve intersections developed.

5.4.2 Surfaces

Surfaces can be generated in AutoShip by lofting or sweeping existing curves or from the offset of an existing surface. The maximum order allowed in both parametric directions is five. The type of sweeping implemented is different from the one described on Section 4.8.4, using one profile curve supported on both extremities by two trajectory curves, which provide an automatic scaling of the profile curve.

The package presents two different types of lofting. One, is not a true lofting because the curves used to generate the surface are not the ones selected but only a user defined number of points obtained by interpolation The second type of lofting is a real one, but with the limitation that the input curves must have the same number of control points, although accepting different orders.

The tools for surface analysis are the rendering and the representation of Gaussian and mean curvatures, by colour mapping on the surface, using a three colour palette.

There are no tools implemented for the automatic fairing of surfaces and for that reason the correction of the surfaces obtained is only possible by manual editing. The only editing operation available is to move points. As for the curves, these points can be either the control points or the corresponding nodes on the surface.

Contours can be generated automatically for constant values of x, y or z, but the curves generated are not available to be used further in the modelling work.

The package also offers routines for surface/surface intersection, although some implementation problems originate unpredictable results in some situations. The definitions of the generated curves and surfaces are not available to the user and therefore they cannot be used for post-processing by other applications.

5.4.3 Application examples

The same ship hull used in the application example of Section 5.2 was modelled with AutoShip and the results are presented.

In a first approach, and in order to allow a more direct comparison, the hull was divided in a similar way and the same surface generation technique, lofting, were used in both cases. As it can be seen (Figs. 5.29-5.31), the surfaces obtained directly from the curves are also very rough and they would require a lot of manual editing work to be considered acceptable. The problem with this approach is that the result from the lofting process applied to a set of curves, each defined by approximately 8 to 9 points, generates surfaces with control grids too dense. These grids imply surfaces too constrained, and even if they interpolate the hull shape at the sections, they present an unfair behaviour inbetween. As the system does not provide any surface fairing tools, the only alternative for improving the shape would be the manual editing, but the size of the control grid makes that task very difficult and time consuming. More practical alternatives could be the simplification of the grid within a defined tolerance using knot removal, which the system also does not present at the moment, or simply to avoid the generation of complex surfaces to start with.

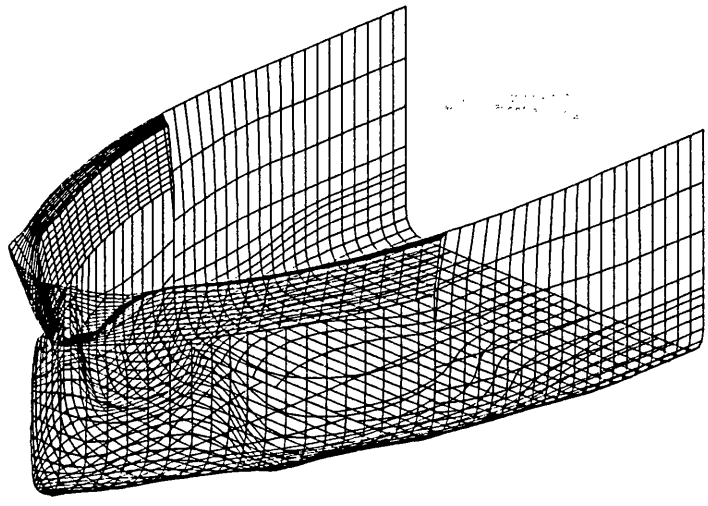

Figure 5.30 Forebody of tanker

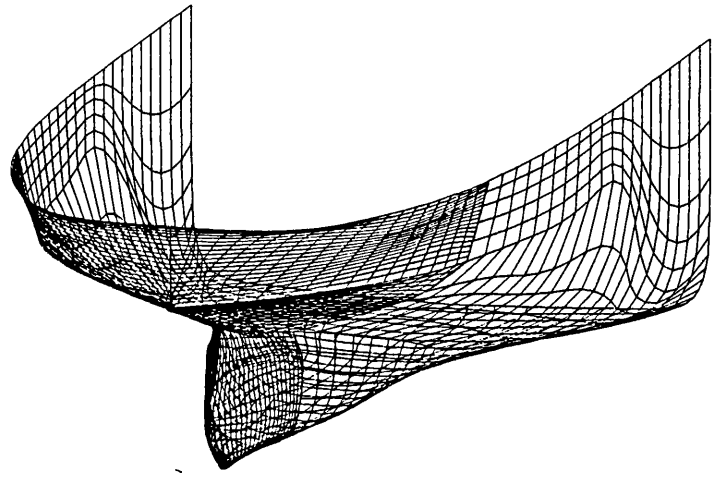

Figure 5.31 Aftbody of tanker

In order to explore this second alternative, a new attempt was made to model the hull. This time, and following the same guidelines used in the example of Section 4.3, the number of surfaces was reduced to a minimum - one for the forebody and two for the aftbody. Each of the surfaces was lofted from a minimum of curves, three or four, to avoid having too many rows on the control grid. As the resulting surfaces are necessarily very different from the intended, extra columns are generated and moved longitudinally to the exact location of known section curves. These column curves are then edited to match the shape of the corresponding section curves. The resulting surfaces (Fig. 5.33) are simpler and closer to the desired hull shape.

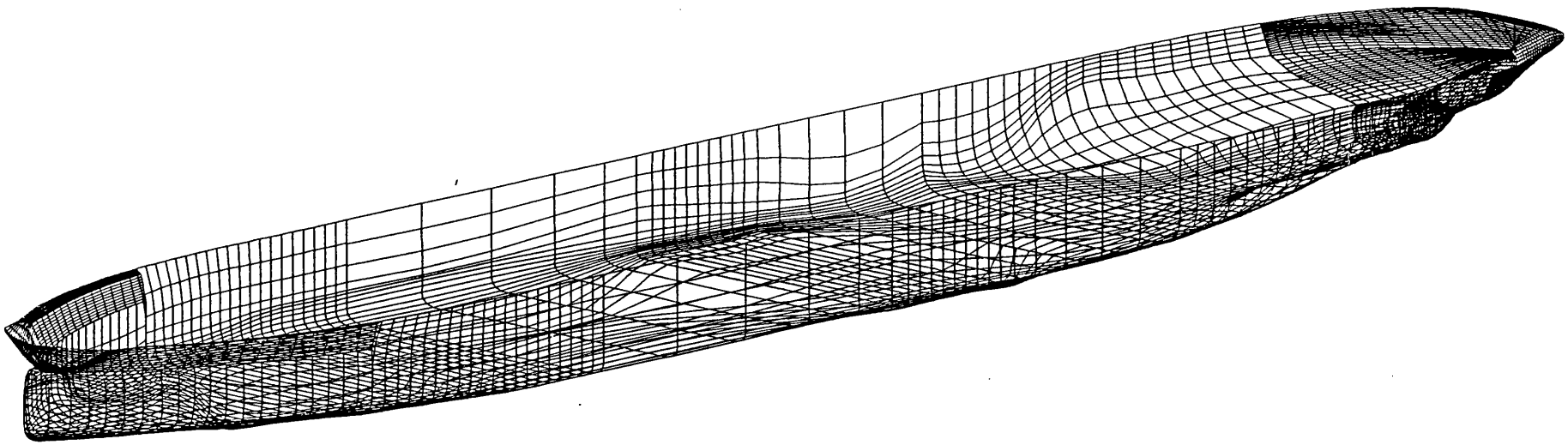

Figure 5.32 Full hull of tanker

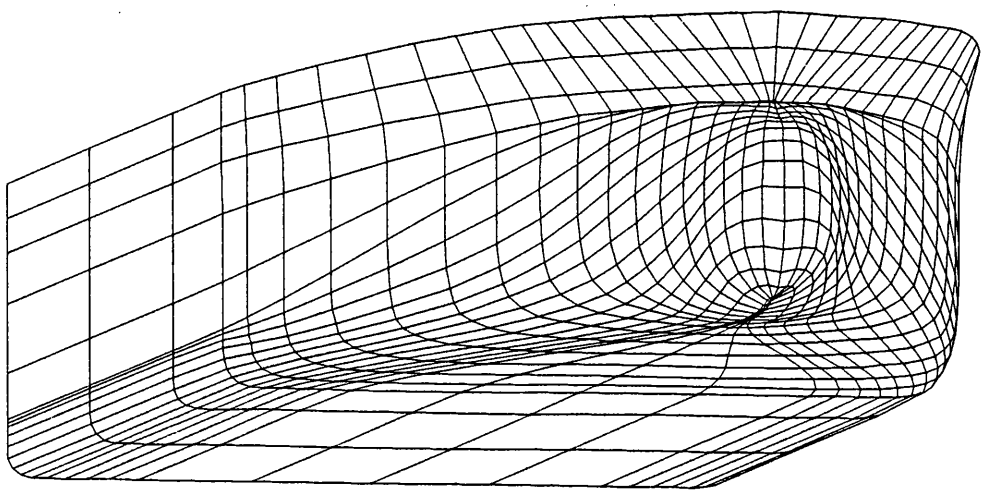

Figure 5.33 Forebody of tanker modelled with a single surface

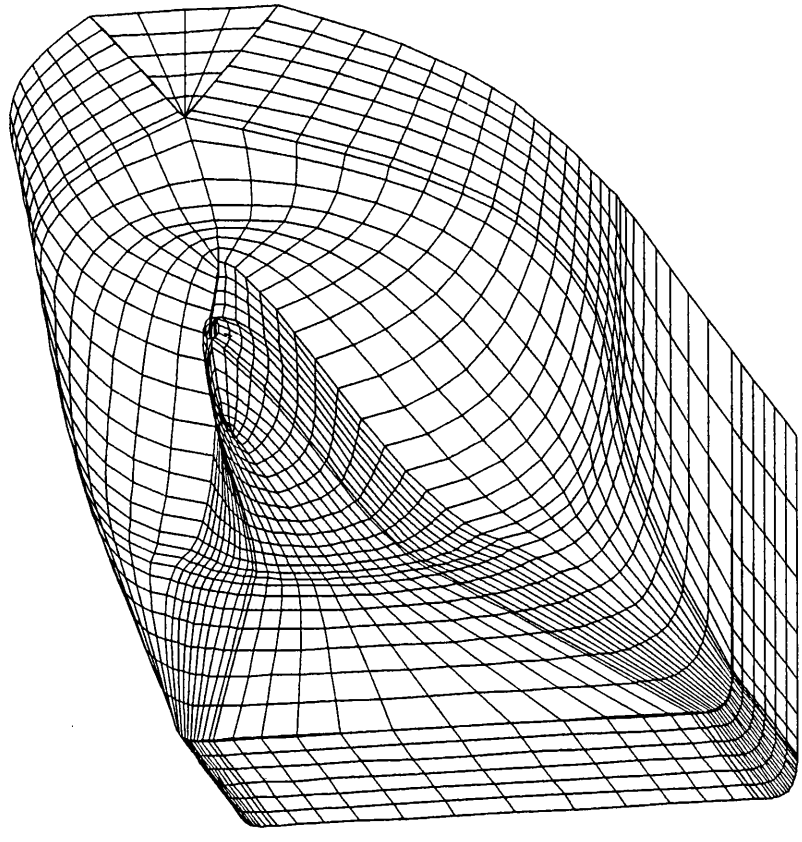

Figure 5.34 Aftbody and stern panel of tanker modelled with two surfaces

As concluded before, this approach produces simpler and fairer surfaces. As a single surface was used to model complex areas some care has to be taken to represent zones with different continuity characteristics. To generate planar bottom and side areas, two rows of the control grid are made to coincide with the corresponding tangency lines and then converted into knuckles.

To model the aftbody one surface patch was used to represent the stern panel and another to represent the rest of the hull. The procedure described above for the forebody was not so efficient here since the contour of the stem and the shape of the propeller boss are more difficult to represent with a simplified grid, as it can be seen in Fig. 5.34.

5.5 General remarks about modelling procedures

From the previous sections of this chapter some remarks can be made about the utilisation of software systems oriented for the representation of ship hulls using surfaces.

With the tools described in the present work, which are similar to the ones available in other surface modellers reviewed, it is not yet possible to have an automatic procedure to create directly surfaces from offset data, or to convert offset directly into curves and curves directly to surfaces.

Is not enough to have the mathematical tools available to perform good work. The procedure applied must be carefully selected to each case. A good approach will be to start with fewer curves, defined by the minimum number of points, to generate relatively simple surfaces. New sections can be obtained by intersection with vertical planes and edited to match known sections of the hull. Whenever obtained surfaces are not satisfactory, it is sometimes easier to delete them and return to the generating curves, instead of wasting to much time interactively editing a surface, which is a non trivial task.

6. Conclusions

6.1 Overall conclusions

The initial purpose of this work was to develop a procedure that could generate the surface hull surface from a set of transverse sections and boundary curves described by offsets in an almost automatic fashion. However, it soon became clear that the manual editing could not be avoided and so, instead of defining a precise sequence of operations, it was decided, by implementing several user interfaces to the basic algorithms, to create a wide set of curve and surface modelling tools. These tools allow the user to select in each case the methods and the sequences to use.

The work presented is a method of representation of the shape of ship hulls by a set of mathematical surface patches. For this purpose, the available formulations were studied and compared, taking into account the required characteristics of the surfaces.

From the parametric curves and surfaces reviewed, NURBS were identified as being the most suitable to represent ship hulls, because being a generalisation of Bézier and B-splines, they keep all their basic properties while adding extra degrees of freedom for shape control. Therefore, the NURBS formulation was selected as the basis for the system developed.

The method proposed is a two step approach. In the first step a wireframe model is created and in the second one, the surface patches are generated and faired.

The generation of surface patches requires different methods, depending on the constraints and geometric information available on each particular area of the hull. Taking this into account, it was decided to develop several different methods of generating surfaces, although keeping a unified definition, the Cartesian tensor

product, in order to simplify storage and further processing for curvature analysis, fairing, computation of intersections, and so on.

After experimenting with the various methods, a general procedure for approaching the representation of the hull shape was sketched as follows. First, NURBS curves are fitted to a set of transverse sections defined by points from an offset table or digitised on a lines plan drawing. Next, the same procedure is applied to generate curves for the stem and stern profiles and knuckles, if any. All curves are then edited, analysed and faired until a reasonable accurate wireframe model is obtained. The existing boundary lines which define in the hull regions of similar geometric characteristics, are used to split the internal wireframe lines. Finally, the resulting split sections are used as support for the generation of the surface patches.

For ships of simpler forms, capable of being represented by a single surface, an alternative approach proved to be efficient. A midship section was defined with a minimum of points, generally less than ten. Next a surface was generated by extrusion along the full length of the ship. The first and last columns of the control grid were then edited dynamically to approximate the stem and stern longitudinal contours. Finally the locations of the internal columns of the control grid were changed where necessary to coincide with known sections of the ship and then edited to approximate them. During this process, the surface may be refined by generating extra columns or rows of the control grid, if necessary.

Regarding the application of the several surface generation techniques used, in general ruled or extruded surfaces can be used in some extent of the parallel middle body, if any, lofted surfaces can be used in the fore and after bodies, blended surfaces can be used in some transitions areas between the bulb and the stem and on stern panels, and sweep surfaces can be used to generate decks, bulwarks, etc.

The methods of surface interrogation developed to analyse the quality of the patches were curvature analysis (Gaussian, mean and principal), isophotes for testing geometric continuity across patch boundaries and reflection lines, to evaluate the aesthetic quality. From the experience of the application examples, the most used tool

was the Gaussian curvature. However, the quantitative analysis of the curvature distribution displayed by a wide range of colours or iso-contours proved to be of limited usefulness to the user. So, in alternative, a qualitative display identifying the areas of positive, negative or null curvature was developed and used with more efficiency.

A preliminary fairing method was developed, by applying to rows and columns of the surface control grid the fairing algorithms developed to curves.

Some procedures to obtain information from the model were developed, such as the computation of the intersection of the hull with simple conic surfaces and with sets of planes parallel to the three orthogonal projection planes. An approximated method to compute the areas of the patches was also developed.

Finally, the procedure was applied to practical examples and it was concluded it is not yet possible to have a fully automatic procedure to create directly surfaces from offset data, or even to convert offsets directly into curves.

Although the system developed during this thesis can only be considered a prototype for demonstration of the algorithms and methods presented, the comparison with a commercial system showed that:

- The system contains already all the functionalities for the creation, edition and curvature analysis of curves and surfaces found in the commercial package.
- The system has tools for the simplification of the input data (filtering) and of the curves and surfaces (knot removal) not covered in the commercial package.
- The system performs already some simple curve and surface fairing capabilities not found in the commercial system
- The commercial system provides capabilities of surface/surface intersection and some derived capabilities such as the trim of surfaces not implemented in the prototype system.

It was also concluded that to have good results, the model should be kept as simple as possible, i.e, with a reduced volume of data. Curves should be defined by a minimum of points and surfaces should be generated from a minimum of curves in order to be fairer and easy to work with.

6.2 Specific conclusions

For the development of the wireframe model, it has been necessary to study the algorithms and procedures for the creation and editing of curves. From the comparative analysis of the algorithms for curve fitting it was concluded that sometimes, in particular when dealing with digitised input points, the efficiency of the curve fitting works against the designer since it attempts to reproduce all the shape characteristics, including some undesirable oscillations, that are not a characteristic of the shape, but only a product of the data input process. It was also noted that the excess of input points generally creates problems of numeric instability in the results of curve fitting and finally, that there are some shape characteristics such as knuckles, straight and conic segments, among others, that the fitting procedure cannot reproduce automatically.

To keep the model simple, measures were taken at the three levels of entities that constitute the hull model. For lines, a method was developed to filter data points to some extent, reducing errors and redundancy in input information. For curves and surfaces the simplification is first obtained by knot removal to a defined tolerance and next by interactively editing and fairing locally.

Filtering digitised input lines with the simple method presented was an efficient way to reduce the volume of input data to a minimum, removing redundant points. The user defined tolerance distance between the initial and the final line, should depend on the main dimensions of the ship and on the scale of the drawing being digitised.

The curve editing techniques developed allow the designer to exploit NURBS capabilities of representing conics like straight lines, circular and elliptical arcs of defined dimensions and they also provide tools to cut and join curve segments,

operations also required to prepare the curves to be used as support for surface generation.

The curve fitting and approximation methods are sensitive to the parameterisation selected by the user. From those parameterisations presented, the centripetal one showed better results. The developed curve fitting and approximation methods proved to be particularly sensitive to the distribution of the data points. In general, the methods based on knot averaging showed better results.

The accuracy of the curves automatically obtained from data points proved to be not so important for the global procedure as expected. Due to the implementation of an efficient interactive curve editing tool, provided with a large set of options (to move, delete, insert, align control points, to create conic curve segments with imposed geometric characteristics, change weights) all made in real time and with the possibility of simultaneously viewing the curvature displayed in the form of a porcupine.

The methods for surface generation considered were extrusion, ruling, lofting, sweeping and blending, that seemed to cover most of the situations found while modelling a ship hull surface. All the above methods could be reduced to particular applications of surface lofting. The implementation of surface lofting requires that all the supporting curves are made compatible, meaning that they must be raised to the same order and share a common knot vector, without changing their shape. The procedure to obtain curve compatibility relies entirely on two basic algorithms, degree raising and knot insertion, that can also be generalised to surfaces.

The basic surface algorithms were also implemented with a common user interface as a dynamic surface editor. However, due to the difficulty in interactively editing a surface in 3D space, this should be reduced only to minor local corrections. From the application examples it was concluded to be more efficient to delete a surface with unwanted shape characteristics, to edit the supporting curves and to generate the surface again.

In general, the procedure developed guarantees only C^0 continuity across the boundary lines between patches and so, they must be carefully chosen (boundary lines with known tangent values, knuckles, etc.). However, boundary conditions in the parametric direction of the support curves can be imposed by fitting curves with imposed end tangent values, for example, and these are naturally inherited by the surface. This procedure allows to obtain G^1 continuity across the patches boundaries.

In general NURBS proved to be a reliable basis for the representation of curves and surfaces used to model a ship's hull. The set of generation and editing techniques developed from the basic algorithms available proved to be a good foundation for a system dedicated to ship surface modelling. The extra degree of freedom provided by the weights, has potential capabilities which extend beyond the implemented representation of conics.

6.3 Future developments

The possible future developments of the present approach to ship surface modelling can be grouped as follows:

- further development of the mathematical algorithms and data structures
- development of features specific to ship shape in order to speed-up the most common modelling tasks

6.3.1 Mathematical algorithms

The quality of the curve fitting algorithms is very important not only for the creation of the wireframe model but also for the generation of the surface patches. The efficiency of the curve fitting algorithms depends of the type of parameterisation, and to be more effective, the parameterisation must have a close relationship with the geometry of the data polygon. The parameterisations used in this work are the *chord length* and the *centripetal*, that do not take into account, for instance, the changes of direction of the data polygon.

Parameterisation for interpolation and approximation is still a matter of research and some alternatives have been suggested. The *variable speed* parameterisation [91] tries to incorporate curvature information into the parameterisation. In [92] an *affine invariant angle knot spacing* parameterisation that takes into consideration the angles or corners implicit in the data is presented. The parameter is a function of both chord-length and the external angles between polygon segments (Fig. 6.1).

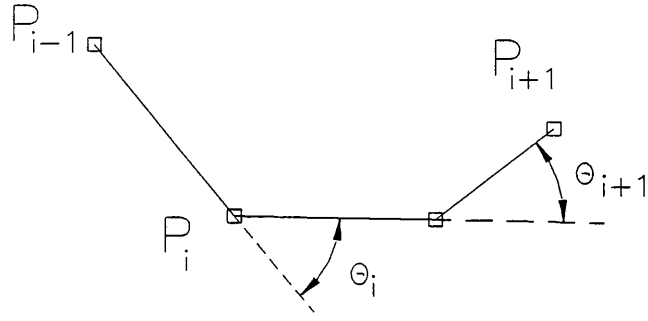

Figure 6.1 Affine invariant angle knot spacing

In [93] the parameter t_i is a function of the chord-length s_i and of the area δ_i of a triangle defined by the data points P_{i-1}, P_i and Q_i, the intersection of the straight lines defined by $\overline{P}_{i-2}\overline{P}_{i-1}$ and $\overline{P}_i\overline{P}_{i+1}$ (Fig. 6.2).

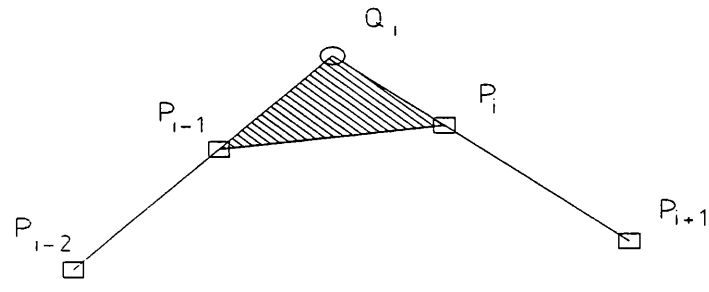

Figure 6.2 Triangle area for parameterisation

Although some of these parameterisations are applicable only to plane curves, they represent some of the directions that should be studied as an attempt to improve the quality of curve fitting.

As mentioned in Section 6.2 the accuracy of the curve fitting algorithms proved to be not as important as expected, but the development of new more efficient approximation algorithms would be welcome to produce simpler curves (less control points) reducing the manual editing time.

Several modelling tools depend on the correct determination of the curve/surface parameters corresponding to a given point *(x,y,z)*, i.e, to find u and v so that

$$(x, y, z) \rightarrow r(u)$$

or

$$(x, y, z) \rightarrow s(u, v)$$

The numerical method presented, based on the Newton-Raphson and the bisection algorithms should be replaced by a foolproof algorithm capable of providing accurate results even when the curve/surface have more complex shapes.

The basic editing methods (straightening, splitting, joining) presented in Section 4.5 for curves, should be generalised to surfaces. A new basic algorithm, degree reduction [94,95], should be implemented both for curves and surfaces.

As mentioned in Section 6.2, the present approach is only capable of producing C^0 continuity between surface patches. G^1 continuity, i.e., the existence of a common tangent direction along the boundary line between patches is a good target for practical purposes. Automatic surface editing capabilities can be developed to obtain G^1 continuity across contiguous patches, by inserting extra internal supporting curves on each patch and changing their position so that they become contained in a single tangent plane.

The simplified intersections of surfaces with planes parallel to the projection planes based on contouring techniques cannot deal with inclined planes or other generic

surfaces. The implementation of algorithms for intersection of arbitrary surfaces is one of the contributions required to allow several modelling operations like, for instance, the trimming of the deck surface by the hull surface.

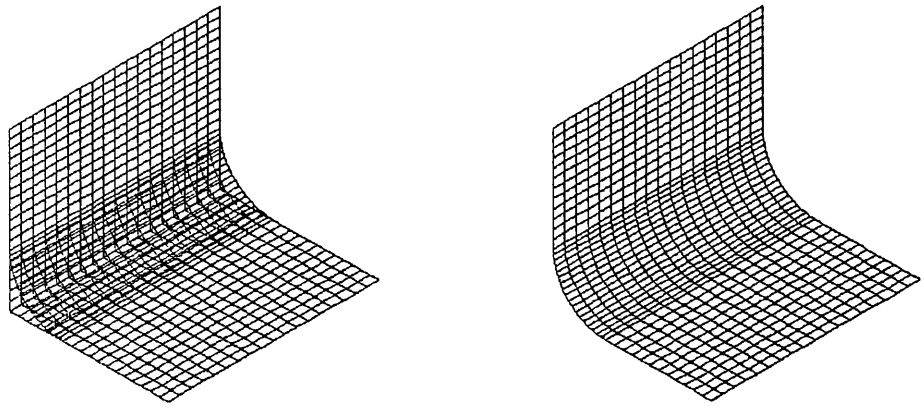

Figure 6.3 Surface filleting

An efficient algorithm for surface intersection allows also the development of other modelling tools like filleting (Fig. 6.3) and chamfering (Fig. 6.4) between two surfaces.

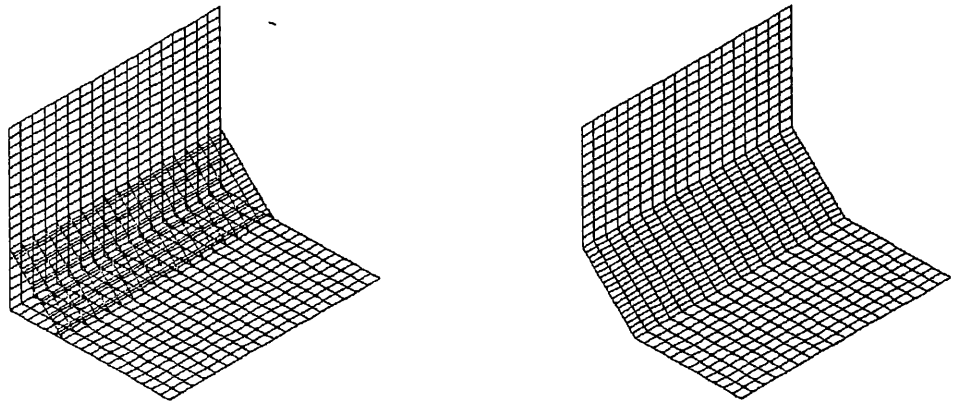

Figure 6.4 Surface chamfering

Both these techniques require first the intersection of the auxiliary cylindrical (in filleting) or plane (in chamfering) surfaces with each of the work surfaces and then the removal (trimming) of the areas in excess.

The generation of curves on a given surface is a topic that should be studied. Several operations such as surface intersections or projection of lines, generate series of points on a surface that normally must be treated afterwards as curves. If a general curve fitting procedure is used, the curve obtained is not necessarily entirely contained in the surface. A new type of curve fitting taking into account the surface parameter values of each data point is one possible direction of research.

The representation of surface patches trimmed or with openings requires an alteration of the data structure in order to accommodate the description of on or more closed curves on the surface, obtaining what is currently designated by *trimmed surfaces* (Fig. 6.5).

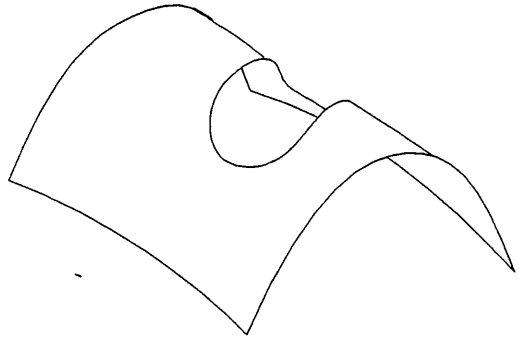

Figure 6.5 Trimmed surface

The developed preliminary methods of surface fairing working on rows and columns of the control grid, are not enough to obtain a surface of production quality. A global surface fairing method, by reducing user editing would improve speed and final quality. In terms of fairing, larger changes should be taken care using non-linear constrained optimisation procedures building on the work of Ferguson [83]. For fine tuning, particular interest should be given to methods that explore the potential of the extra degree of freedom provided by the weights, like in the recent work of Hohenberger and Reuding [96].

A procedure for the determination of the geodesic on surface patches will be quite useful as a basis for plate development algorithms.

6.3.2 Features specific to ship modelling

The present approach does not take into account the information on tangency points and directions provided by the flat-of-bottom and flat-of-side curves, when available, which could greatly improve the quality of the transverse section curves generated and therefore the corresponding surface patches.

The shape of some of the ship main curves defined during the basic design, like the midship section or the stem contour depend of the values of a set of design variables. Instead of trying to fit B-Spline curves to this lines, semi-automatic methods of generating the curve directly from known design shape parameters can be developed.

Based in the time consumed on the experiments of modelling specific surfaces like a deck with both camber and sheer, a special purpose method should be developed to reduce the time consumed for the generation of this particular type of surfaces where some dimensions and shapes can be parameterised.

References

1. **Comstock, J. (ed.)** *"Principles of Naval Architecture"*, SNAME, 1980.

2. **Lasky, Marc P. and Daidola, John C.** *"Design Experience with Hull FormDefinition During Pre-Detail Design"*, Proceedings of SCAHD'77, SNAME, 1977, pp. 31-59.

3. **Nowacki, H.** *"Computer Aided Design"*, Lecture Notes, University of Michigan, 1973.

4. **Buczkowski, Leslaw** *"Mathematical Construction, Approximation and Design of the Ship Body Form"*, Journal of Ship Research, Vol.13, No.3, September 1969.

5. **Kuiper, G.** *"Preliminary Design of Ship Lines by Mathematical Methods"*, Journal of Ship Research, Vol.14, No.1, 1969, pp. 52-66.

6. **Kuo, C. and Kyan, A.** *"Direct Generation of Fair Ship Hull Surface from Design Parameters"*, Proceedings of Computer Applications in the Automation of Shipyard Operation and Ship Design, 1974.

7. **Tokumaru, E.** *"New Lines System - Its Concept and Application"*, Proceedings of SCAHD'77, 1977, pp. 185-194.

8. **Fuller, Arthur L. and Aughey, Michael E.** *"Computer-Aided Ship Hull Definition at the Naval Ship Engineering Center"*, Proceedings of SCAHD'77, SNAME, 1977, pp. 131-148.

9. **Lewis, F. M.** *"The Inertia of Water Surrounding a Vibrating Ship"*, Transactions of SNAME, 1929.

10. **Reed, Arthur M. and Nowacki, Horst** *"Interactive Creation of Fair Ship Lines"*, Journal of Ship Research, Vol.18, No.2, June 1974, pp. 96-112.

11. **Hoffman, Dan and Zielenski, Thomas** *"The Use of Conformall Mapping Techniques for Hull Surface Definition"*, Proceedings of SCAHD'77, SNAME, 1977, pp. 159-174.

12. **Keane, A. J.** *"A Computer Based Method for Hull Form Cocept Design: Applications to Stability Analysis"*, RINA, 1987, pp. 61-75.

13. **Hattori, Yukihide and Matida, Yuokio** *"Some Problems in Practical Improvement of Mathematical Fairing"*, Proceedings of SCAHD'77, SNAME, 1977, pp. 61-70.

14. **Tsujita, T. and Eida, Y.** *"Mathematic Representation of Ship Hull Form by Using of Sine Curve and Ellipse"*, Proceedings of PRADS, 1995, pp. 2.1369-2.1380.

15. **Rabien, U.** *"Ship Surface Design by Transformating Given Mesh Representations"*, Proceedings of Computer Applications in the Automation of Shipyard Operation and Ship Design, 1979, pp. 85-93.

16. **Collatz, G. and Seiffert, E.** *"Interactivite fairing of Ship Lines - A Procedure Developed for the Model Manufature at the Hamburg Model Basin"*, Proceedings of SCAHD'77, SNAME, 1977, pp. 175-184.

17. **Kouh, J.S. and Chau, S. W** *"Computer-aided Geometric Design and Panel Generation for Hull Forms Based on Rational Cubic Bézier Curves"*, Computer Aided Geometric Design, N0.10, 1993, pp.537-549.

18. **Rogers, D. F.** *"B-spline Curves and Surfaces for Ship Hull Definition"*, Proceedings of SCAHD'77, 1977, pp. 79-96.

19. **Nowacki, H. and Creutz, G.** *"Ship Lines Creation by Computer - Objectives, Methods and Results"*, Proceedings of SCAHD'77, SNAME, 1977.

20. **"A Description of the BMT Hull Design Software"**, British Maritime Technology, 1985

21. **Yuille, I.M.** *"Interactive Program for the Design of Ship Hull Forms"*, Proceedings of ICCAS III, 1979.

22. **Munchmeyer, F.C., Schubert, C. and Nowacki, H.** *"Interactive Design of Fair Hull Surfaces Using B-splines"*, Proceedings of ICCAS III, 1979.

23. **Stroobant, G. and Mars, D.** *"Ship Hull Form Fairing"*, Proceedings of ICCAS IV, 1982, pp. 177-181.

24. **Reese, Dirk** *"Fairing of Ship Lines and Ship Surfaces"*, Proceedings of ICCAS V, 1985.

25. *"STEERBEAR Specification"*, Kockums Computer Systems, Malmo, 1972.

26. **Lauritsen, Ole N.** *"Practical Application of Single Curved Hull Definition. Background, Application, Software and Experience"*, Proceedings of ICCAS V, 1985.

27. **Bézier, Pierre** *"Mathematical and Practical Possibilities of UNISURF"*, Computer Aided Geometric Design, Barnhil, R. and Riesenfeld, R. (eds.), Academic Press, 1974.

28. **Chaojun, Zhou** *"The Use of Bézier Surface in the Design of a Ship Hull Surface"*, Proceedings of ICCAS V, 1985.

29. **Pommelet, Marc** *"Advance Use of Bézier Surfaces for Computer Aided Hull Definition"*, Proceedings of ICCAS V, 1985.

30. **Fog, Nils G.** , *"Creative Definition and Fairing of Ship Hulls using a B-Spline Surface"*, Computer Aided Design, Vol.16, No.4, July 1984.

31. **Beyer, Klaus-Peter** *"Direct Curve and Surface Manipulation for Hull Form Design"*, Proceedings of Ship Meeting/STAR Symposium, SNAME, 1988, pp. 247-256.

32. **Catley, D., Davidson, G., Okan, M., Whittle, C. and Thornton, P.** *"A System for General Surface Definition and Manipulation"*, Fundamental Algorithms for Computer Graphics, Earnshow, R. (Editor), Springer-Verlag, 1985.

33. **Jensen, J. J. and Baatrup, J.** *"Transformation of Ship Body Plans into a B-Spline Surface"*, The Technical University of Denmark, Report No.373, August 1988.

34. **Standersky, Nelson Bianco** *"The Generation of Ship Hulls with Given Design Parameters Using Tensor Product Surfaces"*, Proceedings of Theory and Practice of Geometric Modelling, University of Tubingen, 1988.

35. **Bardis, L. and Vafiadou, M.** *"Ship Hull Representation with B-spline Surface Patches"*, Computer Aided Design, Vol.24, No.4, 1992, pp. 217-222.

36. **Kouh, J.S. and Söding, H.** *"Rational Cubic Splines for Ship Hull Representation"*, Proceedings of ICCAS V, 1985.

37. **Clemens, J. C.** *"Ship Lines Using Convexity-Preserving Rational Cubic Interpolation"*, Journal of Ship Research, Vol.35, No.1, March 1991, pp. 28-31.

38. **Coons, S. A.** *"Surfaces for Computer Aided Design of Space Forms"*, Project MAC-TR41 Technical Report, MIT, 1967.

39. **Rogers, D. F. and Adams, J. A.** *"Mathematical Elements for Computer Graphics"*, MacGraw-Hill, 1990.

40. **Farin, G.** *"Algorithms for Rational Bézier Curves"*, Computer Aided Design, Vol.15, No.2, March 1983, pp. 73-77.

41. **Ferguson, David R.** *"Multivariate Curve Interpolation"*, 1964.

42. **Barnhil, R.** *"A new Twist in CAGD", Computer Graphics and Image Processing"*, 1978.

43. **Coons, S. A.** *"Surface Patches and B-spline Curves"*, Computer Aided Geometric Design, Barnhil, R.E. and Riesenfeld, R.F. (eds), Academic Press, 1974, pp. 95-126.

44. **Gordon, W.** *"Spline-based Surface Interpolation through Curve Networks"*, Journal of Maths. And Mechanics, Vol.18, No.10, 1969, pp. 931-952.

45. **Cox, M. G.** *"The Numerical Evaluation of B-Splines"*, Journal of Inst. Maths Applics., 1972, pp. 134-149.

46. **de Boor, Carl** "*On Calculating with B-splines"*, Journal of Approximation Theory, Vol.6, 1972, pp. 50-62.

47. **Barsky, Brian** *"Computer Graphics and Geometric Modeling Using Beta-splines"*, Springer-Verlag, 1988.

48. **Versprille, K.** *"Computer Aided Design Applications of the Rational B-spline Approximation Form"*, PhD. thesys, Syracuse University, 1975.

49. **Piegl, L.** *"A Menagerie of Rational B-spline Circles"*, IEEE Computer Graphics & Applications, September 1989, pp. 48-56.

50. **Piegl, L.** *"Modifying the Shape of Rational B-splines - Part 1"*, Computer Aided Design, Vol.21, No.8, October 1989.

51. **Lee, E.T.Y.** *"Choosing Nodes in Parametric Curve Interpolation"*, Computer Aided Design, Vol.21, July/August 1989, pp. 363-370.

52. **Gordon, W.J. and Riesenfeld, R.F.** *"B-spline Curves and Surfaces"*, Computer Aided Geometric Design, Barnhil, R.E. and Riesenfeld, R.F. (Eds), Academic Press, 1974, pp. 95-126.

53. **Tiller, W.** *"Geometric Modelling Using Non-Uniform Rational B-splines: Mathematical Techniques"*, SIGGRAPH'86, 1986.

54. **Rogers, D. F. and Satterfield, S. G.** *"Dynamic B-spline Surfaces"*, Computer Applications in the Automation of Shipyard Operation and Ship Design IV, 1982, pp. 189-196.

55. **Piegl, L.** *"Rational B-spline Curves and Surfaces for CAD and Graphics"*, State of the Art in Computer Graphics, in Rogers, D.F. and Earnshaw, R.A. (Eds), Springer-Verlag, 1991, pp. 226-269.

56. **Akima, H.** *"A New Method of Interpolation and Smooth Curve Fitting Based on Local Procedures"*, Journal of ACM, Vol.17, No4, 1970, pp. 589-602.

57. **Piegl, L.** *"Interactive Data Interpolation by Rational Bézier Curves"*, IEEE Computer Graphics & Applications, April 1987, pp. 45-58.

58. **Rogers, D. F. and Fog, N. G.** *"Constrained B-Spline Curve and Surface Fitting"*, Computer Aided Design, Vol.21, No.10, December 1989, pp. 641-648.

59. **Piegl, L.** *"Modifying the Shape of Rational B-spline Curves. Part 1: Curves"*, Computer Aided Design, Vol.21, No.8, October 1989, pp- 509-518.

60. **Cohen, E., Lyche, T. and Schumaker, L.** *"Algorithms for Degree-Raising of Splines"*, ACM Transactions on Graphics, Vol.4, No.3, July 1985, pp. 171-181.

61. **Chaikin, G.** *"An Algorithm for High-speed Curve Generation"*, Computer Graphics and Image Processing, Vol.3, 1974, pp. 346-349.

62. **Cohen, E. Lyche, T. and Riesenfeld, R.** *"Discrete B-splinee and Subdivision Techniques in Compuyter-Aided Geometric Design and Computer Graphics"*, Computer Graphics and Image Processing, Vol.14, No.2, October 1980, pp. 87-111.

63. **Bohem W.** *"Inserting New Knots into B-spline Curves"*, Computer Aided Design", Vol.12, No.4, July 1980, pp. 199-201.

64. **Tiller, W.** *"Knot Removal Algorithms for NURBS Curves and Surfaces"*, Computer Aided Design, Vol.24, No.8, 1992, pp. 445-453.

65. **Nowacki, H., Bloor, M. and Oleksiewicz, B. (Editors)** *"Computational Geometry for Ships"*, World Scientific Publishing, 1995.

66. **Ball, A. A.** *"CONSURF. Part one: introduction of the conic lofting tile"*, Computer Aided Design, Vol.6, No.4, October 1974, pp. 243-249.

67. **Ball, A. A.** *"CONSURF. Part two: description of the algorithms"*, Computer Aided Design, Vol.7, No.4, October 1975, pp. 237-242.

68. **Ball, A. A.** *"CONSURF. Part three: how the program is used"*, Computer Aided Design, Vol.9, No.1, January 1977, pp. 9-12.

69. **Woodward, C. D.** "Skinning Techniques for Interactive B-spline Surface Interpolation", Computer Aided Design, Vol.20, No.8, 1988, pp. 441-451.

70. **Cohen, E., Lyche, T. and Schumaker, L.** *"Algorithms for Degree-Raising of Splines"*, ACM Transactions on Graphics, Vol.4, No.3, July 1985, pp. 171-181.

71. **Bloomenthal, J.** *"Modeling the Mighty Maple"*, Proceedings of SIGGRAPH 1985, Computer Graphics 19, pp. 305-311.

72. **Lin, Fenqiang and Hewit, W.T.** *"Expressing Coons-Gordon Surfaces as NURBs"*, Computer Aided Design, Vol.26, February 1994, pp. 145-155.

73. **Farin, G.** *"Curves and Surfaces for Computer Aided Geometric Design - A Practical Guide"*, Academic Press, 1988.

74. **Beck, J. M., Farouki, R. T. and Hinds, J. K.** *"Surface Analysis Methods"*, IEEE Computer Graphics and Applications, December 1986, pp. 18-36.

75. **Wolter, Franz-Erich and Tuohy, Séamus T.** *"Curvature Computations for Degenerate Surface Patches"*, Computer Aided Geometric Design, No.9, 1992, pp. 241-270.

76. **Heap, B. R.** *"Algorithms for the Production of Contour Maps Over an Irregular Triangular Mesh"*, National Physical Laboratory, February 1972.

77. **Atkins, D.A., Berger, S.A., Webster, W.C. and Tapia, R.** *"Mathematical Ship Lofting"*, Journal of Ship Research, Vol.10, No.4, December 1966.

78. **Kjellander, Johan A. P.** *"Smoothing of Cubic Parametric Splines"*, Computer Aided Design, Vol.15, No.3, May 1983, pp. 175-179.

79. **Kjellander, Johan A. P.** *"Smoothing ofBicubic Parametric Surfaces"*, Computer Aided Design, Vol.15, No.3, September 1983, pp. 288-293.

80. **Farin, G., Rein, G, Sapidis, N. and Worsey, A. J.** *"Fairing Cubic B-spline Curves"*, Computer Aided Geometric Design, 4, 1987, pp. 91-103.

81. **Sapidis, N. and Farin, G.** *"Automatic Fairing Algorithm for B-spline Curves"*, Computer Aided Design, Vol.22, No.2, March 1990.

82. **Huanzong, R., Gang, C. and Weirong, Z.** *"Nonuniform B-spline Mesh Fairing Method"*, Proceedings of ICCAS VI, 1992, pp.261-272

83. **Ferguson, David R.** *"Construction of Curves and Surfaces Using Numerical Optimization Techniques"*, Computer Aided Design, Vol.18, No.1, January/February 1986, pp. 15-21.

84. **Andersson, E., Andersson, R., Boman, M. Elmroth, T., Dahlberg, B. and Johansson, B.** *"Automatic Construction of Surfaces with Prescribed Shape"*, Computer Aided Design, Vol.20, No.6, July/August 1988, pp. 317-324

85. **Strang, Gilbert** *"Introduction to Applied Mathematics"*, Wellesley-Cambridge Press, 1986.

86. **Lott, N. J. and Pullin, D. I.** *"Method for Fairing B-Spline Surfaces"*, Computer Aided Design, Vol.20, No.10, December 1988, pp. 597-604.

87. **Liu, J., Koyama, T. and Yamato, H.** *"Constrained Smoothing B-spline Curve Fitting for Ship Hull Generation and Fairing"*, Computer Applications in the Automation of Shipyard Operation and Ship Design VI, 1992.

88. **Satterfield, Steven G. and Rogers, David** *"A Procedure for Generating Contour Lines from a B-spline Surface"*, Proceedings of Computer Graphics Tokyo 84, 1984.

89. **AutoCad 12**, *"AutoCad Reference Manual"*, Autodesk, 1992.

90. **AutoCad 12**, *"AutoCad Development System Programmer's Reference Manual"*, Autodesk, 1992.

91. **Cohen, E. and O'Donell, C.L.** *"A Data dependent Parametrization for Spline Approximation"*, Mathematical Methods in Computer Aided Design, Lyche, T. and Schumaker, L. (ed.), Academic Press, 1989, pp. 155-166.

92. **Foley, T. A. and Nielson, G. M.** *"Knot Selection for Parametric Spline Interpolation"*, Mathematical Methods in Computer Aided Geometric Design, Academic Press, 1989, pp. 261-271.

93. **Filipe, A.** *"A New Approach for Curve Parameterizations Based on Areas, Useful in Computer Aided Design"*, Proceedings of Compugraphics'91, September 1991, pp. 163-168.

94. **Eck, Matthias** *"Degree Reduction of Bézier Curves"*, Computer Aided Geometric Design, Vol.10, 1993, pp. 237-251.

95 **Piegl, L. and Tiller, W.** *"Algorithm for Degree Reduction of B-spline Curves"*, Computer Aided Design, Vol.27, No.2, February 1995, pp. 101-110.

96. **Hohenberger, W. and Reuding, T.** *"Smoothing Rational B-spline Curves and Surfaces Using the Weights in an Optimization Procedure"*, Computer Aided Geometric Design, Vol.12, No.8, December 1995, pp. 837-848.