Computer Support for Design, Engineering and Prototyping of the Shape of Ship Hulls

Computer Support for Design, Engineering and Prototyping of the Shape of Ship Hulls

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof. ir. K.F. Wakker
in het openbaar te verdedigen ten overstaan van een commissie,
door het College voor Promoties aangewezen,

op donderdag 2 december 1999 te 16.00 uur

door

Herbert Jan KOELMAN

scheepsbouwkundig ingenieur
geboren te Amsterdam

Dit proefschrift is goedgekeurd door de promotoren:
Prof. ir. A. Aalbers
Prof. Dr. I. Horváth

Samenstelling van de promotiecommissie:

Rector magnificus	Voorzitter
Prof. ir. A. Aalbers	Technische Universiteit Delft, promotor
Prof. Dr. I. Horváth	Technische Universiteit Delft, promotor
Prof. dr. D. Dutta	University of Michigan, Verenigde Staten
Prof. dr. ir. F.W. Jansen	Technische Universiteit Delft
Prof. dr. ir. G. Kuiper	Technische Universiteit Delft
Prof. Dr.-Ing. Dr. h.c. H. Nowacki	Technische Universität Berlin, Duitsland
Prof. Dr. habil T. Tóth	University of Miskolc, Hongarije

Published and distributed by:

Scheepsbouwkundig Advies en Reken Centrum (SARC) BV
Eikenlaan 3
1406 PK Bussum
The Netherlands
Telephone: +31 35 6915024
Telefax: +31 35 6918303
E-mail: sarc@sarc.nl

ISBN 90-901-2888-3

Graphic design: A.R. Vredenduin
Printed in the Netherlands

Contents

Introduction

Scope of this thesis

A determining factor in the appearance and performance of a ship is the hull form. It exerts its influence on many properties, such as resistance, intact and damage stability, behaviour in seaway, manoeuvrability, deadweight, tank capacities, longitudinal strength, production costs and the aesthetic appearance of the vessel. This notion is not new, the importance of the ship hull has already been stipulated in [Weinblum, 1953] rather concisely: 'Die Entwicklung günstiger Schiffsformen ist die wichtigste Aufgabe der Schiffsbauwissenschaft'.

Neither from recent times is the desire to use the computer to support the design, engineering and manufacturing of the ship hull. This is quite understandable because, after all, the manual drawing of a lines plan, manual lofting and the manual construction of shell plate developments are time consuming and often cumbersome processes.

One would expect that, in the course of time, computer systems should have evolved to become efficient and versatile instruments. To investigate this assumption, in this thesis we will sketch requirements for a ship hull Computer-Aided Design (CAD) or Computer-Aided Engineering (CAE) system. When contemporary computer systems are tested against these requirements, it appears that they do not meet essential elements, and that academic research is not always directed towards fundamental improvements. On the contrary, the Computer-Aided Ship Design community adopted a paradigm which is unsuitable for efficient and easy manipulation of a vessel's hull form.

So the core of this thesis is dedicated to the design and development of a new computer system, with as few limitations as possible, which supports all major ship hull design activities, and all relevant processing of the ship hull shape in a straightforward way.

Besides this main research topic, there is a number of secondary questions which will be addressed, such as the meaning of the concept 'user-friendliness', the quality of a user interface, the benchmarking of ship hull design systems, the necessity of GC^2 surface continuity for ship hulls, and the proper way to react on comments and desires of users of a ship hull design system.

Overview of this thesis

In order to be able to formulate objectives and requirements for a computer system, in the first chapter we analyse the hull design process and commonly used hull design methods.

To explore the state-of-the-art in the CAD field, the second chapter contains an overview of CAD methods which can be used for our subject. The third chapter will focus on those

CAD methods applied to ship hull modelling, and it will be investigated whether present CAD ship hull modelling methods meet the requirements of Chapter One, first in a normative comparison, and finally illustrated by practical examples of hull form designs. At the end of the third chapter the research in this field will be discussed.

After the conclusion about contemporary systems, in the fourth chapter the conceptual design of a novel ship hull CAD/CAE system will be presented, with its implementation in Chapter Five.

Chapter Six presents the practical experiences of professional designers using the new system, and some hull designs. Their comments are included, and a benchmark of the new system is presented.

Finally, in the seventh chapter conclusions are drawn, and subjects for further research and development will be identified.

1 Requirements for computer support of ship hull design

Our goal is to develop a computer system which supports all major ship hull design activities, as well as all relevant supporting processes applied to the development of a ship's hull shape.

The first action is to identify those relevant design activities and supporting processes, and use them to formulate objectives and requirements for the computer system. This will be elaborated in this first chapter, which starts with an exploration of the methodology of ship hull design.

1.1 General methodology of ship hull design

1.1.1 Ship hull design process

To visualize the ship hull design process, the ship design spiral is often proposed, which is quite akin to the helix of mechanical engineering of [Hubka, 1982]. Hubka's four design phases (Concept – Preliminary Layout – Dimensional Layout – Detail and assembly drawings) have their equivalent in naval architecture:

- Conceptual model, and possibly numerical conceptual model;
- Preliminary model;
- Final design model;
- Detailed and faired hull form model.

However, we must realise that these design phases are abstractions only, in practice there is an overlap between the phases which is not expressed in the spiral or the helix. Another flaw of the spiral or helix is that they imply within each design phase an equal and fixed sequence of analysis and other actions, which in practice never occurs.

As an alternative, to represent the design process we adhere to a model which expresses the division between design and analysis. This model, which is sketched in Figure 1.1, shows on the left side a number of *design choices* and *design activities*, and on the right a *box of analytical tools* to analyse aspects of the design. The results of the analysis are used in an *evaluation phase*, where modifications to the design or even the design criteria are applied.

Although the design phases of the helix are abstractions, in order to acquire a framework of data and methods, we will analyse what kind of data are used in each design phase.

In the *conceptual design phase* two kinds of data can play a role. Firstly, we have shape knowledge, which at this stage mainly consists of mental images, or rough sketches, of important layout items. Examples of shape data are deck contours and plan contours. Secondly, we have non-shape data, which are based on relationships between parameters. Out of the

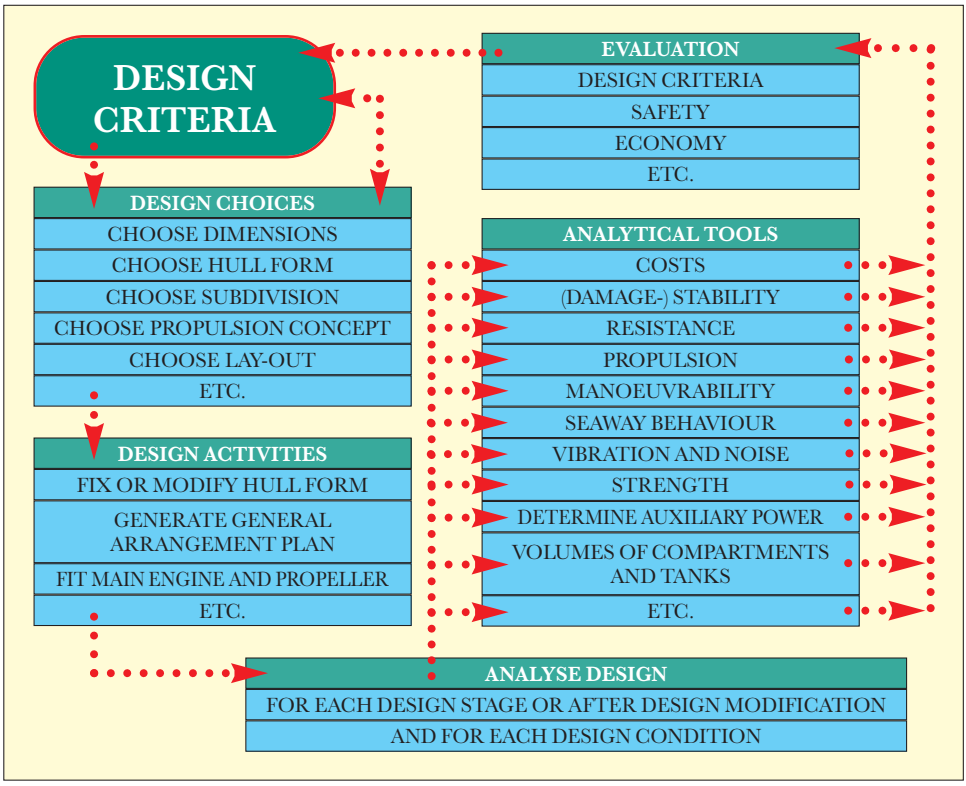

Figure 1.1 *Model of ship hull design process.*

many types of relations, for hull design the most relevant ones are physical, definitional and empirical relations.

Relations are physical if they relate physical events, and definitional if they define a concept, e.g. *deadweight* = *displacement* minus *lightweight*[1].

Empirical relations are based on past experiences in comparable cases, or on research on relations between parameters of systematically transformed series of hull forms. An example is the resistance estimation based on main dimensions and hull form coefficients according to [Holtrop, 1983].

All three kinds of relations can be utilized in an ad hoc process, or with a processing system such as a Concept Exploration Model (CEM) or an Expert Parametric Model (EPM), for example that of [van Hees, 1997].

In the *preliminary design phase* the body of the vessel gets shape, often in a rather rough form. It might be that in the conceptual phase insufficient empirical relations are available. For example, the hull form to be designed may fall outside the domain of available empirical methods. In that case, the preliminary model can be utilized by analytical calculations (such as damage stability calculations, or potential flow calculations) to derive numerical

[1] At the back of this booklet a short glossary of specific terms is included.

qualities from the hull form. The figures obtained this way can be used with CEM's or with an EPM's, as described in the previous paragraph.

In practice this preliminary design phase is often reiterated, because the results of the analytical calculations show that the hull form does not have the desired properties. In this case the designer has to modify the hull form, and re-enter the preliminary design phase.

In the *final design phase* the 'to be built' shape of the hull form is determined, while all design aspects are taken into account. The result of this design phase is materialized into a scale lines drawing, 3-D views or a scale model.

The final shape of hull form is determined in the *final phase*, where the hull form is faired for production, and equipped with production-level details, such as exact radii of roundings.

Finally, after the four design phases there is an additional phase, the *engineering phase*, where the designed hull form is utilized, in a preparation to the production process. Examples of engineering activities are:

- Determination of construction details, and creation of construction drawings;
- Determination of piping arrangement;
- Making an arrangement of shell plates over the hull and generation of shell plate expansions;
- Generation of NC or CAM data.

So we see that the vessel gets its first shape somewhere between the conceptual and the preliminary design phase. The development of this shape is the subject of the next section.

1.1.2 Shape generation

In practice the shape of hulls is gradually improved in years of development, testing and experience. The quality of the analytical tools of Figure 1.1 is not sufficient to make a complete quantitative analysis of all affected aspects. It is the ship designer who has general notions about relationships between shape characteristics and effects, and who uses these notions to construct some mental image of the ship's hull form.
Such relationships can be rather unprecise, such as:

- If seaway behaviour is important: do not use too much flare;
- For more tank capacity: give ordinates slightly more U-shape;
- In case of danger of vibration: create better inflow of water into the propeller disc;
- For an additional container: create extra space by an additional knuckle;
- For better initial stability: make the aft body more pram-type.

Based on an (implicit or explicit) mental image of shape, the shape of the hull form is in general generated with one of the following methods:

- Hull form transformation;
- Systematically varied standard series;
- Using mathematical formulae;
- Fuzzy modelling;
- Arbitrary free form design.

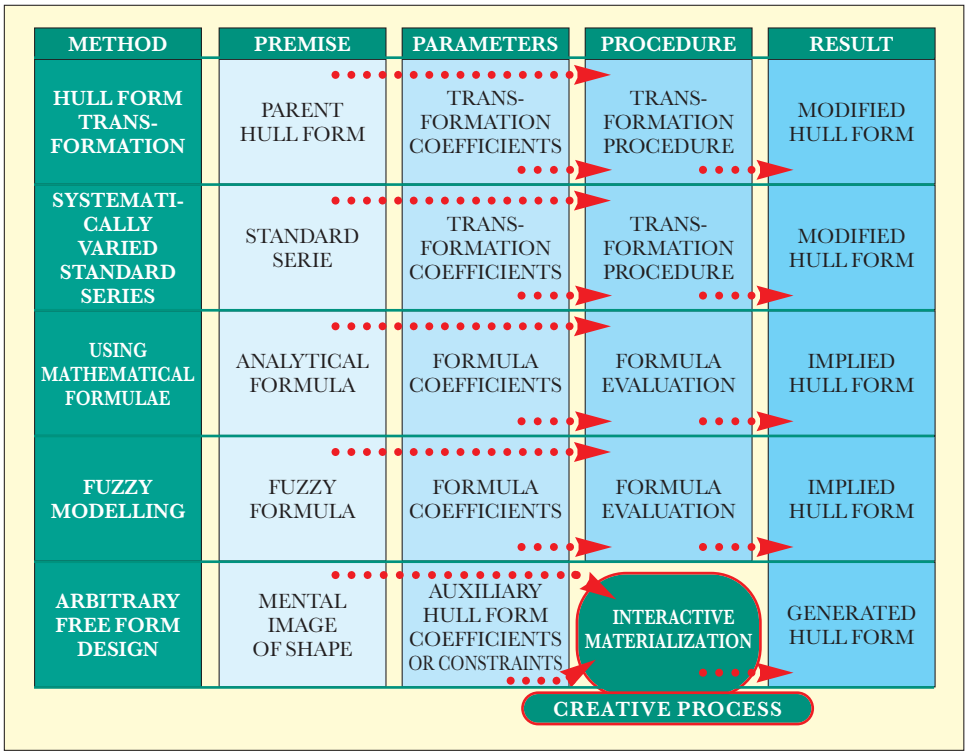

Figure 1.2 *Hull form generation methods.*

The first four methods are essentially methods of *hull form modification*, because each new hull form will always inherit the characteristics implied by the parent form or the formulae. The last method, arbitrary free form design, is for *design generation*. A designer may use this method to create truly new hull forms. The different premises, parameters, generation procedures and type of results of these five methods are summarized in Figure 1.2.

The basis for *hull form transformation* is a library of parent forms. From that library a form is selected which resembles the (mental image of the required) shape of the hull form to be designed, and with mathematical transformation or distortion the new hull form (the daughter form) is derived. Transformations can be local or global, but local transformations are little used[2].

Global transformation simply works on the basis of hull form coefficients and is therefore easier to use. Examples of global transformations can be found in [Alef and Collatz 1976], [Kovachev and Yovev 1983], [Lackenby, 1950] and [Rabien, 1979], and an overview of transformation methods is listed in [Schneekluth and Bertram, 1998].

[2] The author assumes that the reason is that for local transformations the ship designer must choose the extent and nature of transformation functions while the impact of those choices on the resulting hull form is not evident.

Model basins and university laboratories have published diagrams or formulae for *standard series* of hull forms. Some of the most famous ones stem from decades ago, such as 'Taylor', 'Series 60' and 'Guldhammer' ([Taylor, 1933], [Gertler, 1954], [Todd, 1963] and [Guldhammer, 1969]), while others are more recent ([Keuning, Gerritsma, Terwisga, 1993] and [van Oossanen and Pieffers, 1985]).

Mathematical formulae or calculation schemes for the initial hull form generation have been developed by a number of authors, see [Kuo, 1971] for an overview. The basic approach of the mathematical method is that formulae are derived which generate a hull form. The formulae can be analytical (as in [de Groot, 1977]), they may be polynomials (with [Weinblum, 1953] as example) or involve relationships between hull shape, hull form coefficients and sectional area characteristics, which are used by, for instance, [Harries, 1998], [Jorde, 1997], [Koelman, 1978] and [Kuiper, 1970].

Very recent *fuzzy modelling* techniques have been applied to initial hull form design (see [Kim et al, 1996]). With fuzzy function approximation methods (see [Kosko, 1997] for an overview) function coefficients for fuzzy functions are derived. With a few design parameters these functions can be applied to generate a SAC and a hull form. This method, which is still in its infancy, can be regarded as a potential replacement for the mathematical formulae method and for the standard series.

The methods discussed can be qualified as 'parametric' or 'procedural'. There is a standardized procedure which, given the initial parameters or choices, inevitably leads to a predetermined result. The advantages of such a procedure are its speed and simplicity for the designer. The great disadvantages are the inflexibility and the lack of shape control. Also, taking into account the lack of variation in hull form types, many designers favour an arbitrary free form method for the ab initio design, or at least free form manipulation after the initial design has been produced by a procedural method. This free form issue will be discussed in the next section.

1.1.3 Arbitrary free form design

Arbitrary hull form design is defined as the process to convert the designer's mental images into a virtual or physical representation. For example a representation in clay, on paper or in a computer memory.

An unresolved question with respect to free form design is whether a human 'thinks' in terms of 3-D objects, or in terms of 2-D views (such as contour lines or intersections). We have concluded that we cannot resolve this question in a scientific way[3], so we explore some practical ways of looking, which support one view or the other.

[3] This question is similar to the Great Questions of philosophy, such as "do we think in language" or "do we know the real world?", which questions are not subject to the methods of the positive sciences. As an illustration of the applied methodology in this field we quote [Chomsky 1975]: "*We do, I am sure, think without words too – at least so introspection seems to show*". We conclude that this level of reasoning cannot be falsified in a Popperian sense, and that the scientific answer to our question lies far beyond the scope of this thesis.

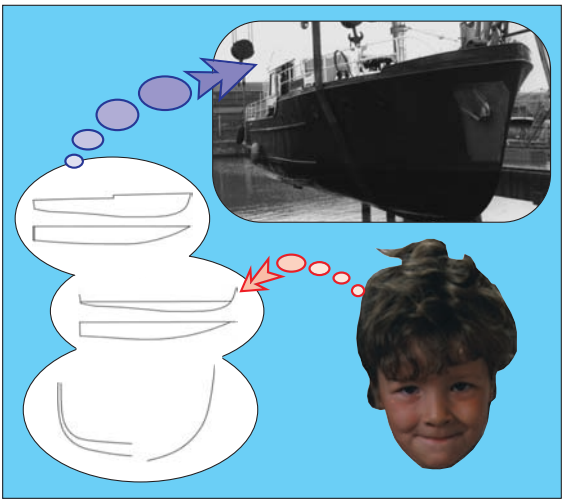

Figure 1.3 *Mental representation of 3-D object, seen as collection of 2-D views.*

Let us assume that, traditionally, a ship designer mainly thinks in terms of 2-D views and uses these 2-D views to construct a 3-D mental representation, as illustrated in Figure 1.3.
This assumption seems plausible when one observes ship designers talk or work and notice that many of the subjects they use relate to 2-D entities. Examples of ship hull designers thinking in 2-D are:

- Sketches are made of all kinds of 2-D views;
- The midship *section* has a circular bilge, with some radius;
- The cross *sections* in the foreship must be U-shaped;
- The area of the cross *sections* must match the SAC;
- The design *waterline* may not become concave in the forebody;
- The use of coefficients of the *waterline*;
- At *frame* X the width at height H must be at least B;
- The *Buttocks* must have fluent shoulders;
- The vessel has shear (where shear is a *2-D projection*) of Y%;
- The vessel has an elliptical stern part (a *2-D projection*);
- The vessel has a goose-neck bulbous bow (as can be seen in *plan view*);
- The fairness of the diagonal (which is a longitudinal *section*) is a measure for overall fairness.

(All italicized terms in these examples are 2-D by nature.)

On the other hand, we can also advocate that a human has a mental representation of the 3-D object, and derives from this representation 2-D views, which are only used to communicate ideas about the object to the outside world, as illustrated in Figure 1.4.

Figure 1.4 *Mental representation of 3-D object.*

This assumption also seems plausible, considering these examples:

- Working with 3-D entities, such as with a cylindrical shape which has a certain radius, regardless of the orientation of the centerline of the cylinder;
- Some designers really 'think' spatially, which is demonstrated by 3-D sketches they produce of even the most preliminary shape concept, and by gesticulation when they try to communicate their ideas of the hull form to other persons;
- It would be interesting to explore human nature in this field through an investigation of the behaviour of those uninfluenced by tradition and education: children. A systematic investigation falls beyond the scope of our research, but we have one anecdotical example: when the author's seven-year-old son was shown a vessel's lines plan for the first time in his life, the plan view was recognized as 'a boat, seen from underwater' by the boy. Apparently his orientation is more 3-D then 2-D, at that age.

Our last considerations about this subject are:

- It might be that a 2-D or 3-D approach also depends on the design phase. The ship hull model in the final design phase is often rather detailed, with all kinds of specific shapes or shape constraints, which gives rise to a more 2-D-like approach. On the other hand the ship hull model in the preliminary design can be rather vague, with a more 3-D-like approach;
- One could postulate that a 2-D orientation is not 'natural' but 'nurtural'; that it might stem only from education and convention. No matter if this statement is true, a 2-D orientation must seriously be taken into account for it might take generations before it would eventually vanish;
- In [Ferguson, 1992] 2-D and 3-D approaches are ranked equally. In that essay three tools of visualization are identified; the pictorial perspective, the orthographic projection (which is an engineering drawing) and the tactile model.

1.1.4 Conclusions about design methodology

We end this sub-chapter with four conclusions:

- It is questionable whether the design helix is a proper representation of the ship design process, instead we propose the design and analysis model of Figure 1.1. Additionally, as abstractions, the four design phases of the design spiral can be recognized, each with its own data set: a conceptual model, preliminary model, final design model and detailed model;
- The most commonly used ship hull design method is 'arbitrary free form design'. The question whether a designer thinks in 2-D or in 3-D entities cannot be resolved, both approaches may be used;
- Design methods are intermixed, and there is no prevailing sequence of activities. Even within the arbitrary free form design method, choices of parameters or geometrical entities to use may vary from project to project, from company to company, and from designer to designer;
- With an eye on the previous three conclusions, a solemn description of the prevailing ship design methodology can be 'The freedom to execute any activity in any sequence'. A distant observer could describe it boldly with 'chaos'.

1.2 Objectives and requirements for a CAD/CAE system

1.2.1 Objectives of CAD and CAE applied to hull forms

The ultimate objectives of every tool used for economic human activity are:
- To obtain a greater effectiveness and a better quality;
- To obtain a greater efficiency.

Concentrating on CAD and CAE in the field of ship hull design and engineering, these primary objectives can be split into secondary ones:

A *greater effectiveness* implies that more topics can be dealt with than without the use of CAD/CAE, which is a goal in itself, but which also leads to better quality because in the design stage more knowledge about the ship will be available. We can think of:
- More design iterations, to come to a more optimal final design;
- Integration of analytical calculations, such as (damage-) stability, or (first principle) flow or structural strength calculations;
- 3-D visualization, or automatic manufacturing of a tactile scale model, to give all persons involved a better image of the vessel;
- Higher precision of the hull form definition.

A *greater efficiency* means that less time, material and labour are necessary to obtain the desired results. Greater efficiency leads to:
- A shorter time to reach a certain design stage;
- Fast analytical calculations (from the box of Figure 1.1) possible;
- Integration between CAD and CAE;
- Fast geometrical manipulations (e.g. projection of shell plate boundaries onto the hull surface).
- More freedom in the sequence of design activities (e.g. calculate stability based on a preliminary CAD model, while without CAD insufficient information would be available to perform these calculations at this design stage);
- Increased job satisfaction.

Unfortunately the use of CAD/CAE is not always beneficial, there are examples of inefficient or ineffective use of CAD or CAE, such as:
- The use of improper CAD/CAE systems, which force the designer into a corner;
- An exaggerated attention towards presentation or layout issues, which draws the user's attention away from the core of the work;
- A tendency to 'over-calculate': to make too many calculations, because of the single reason it is so simple to calculate with the computer;
- A tendency to use always the latest CAD/CAE products or operating systems, which may be unstable and error prone.

1.2.2 System requirements

If we combine the listed secondary objectives with our conclusions of the ship hull design methodology of Section 1.1.4, we get the following set of system requirements:

- It must be possible to manipulate two-dimensional views or projections, while the system always maintains a coherent 3-D model (this requirement is called 'Draw 2-D & Model 3-D');
- It must also be possible to manipulate directly in three dimensions (called 'Work 3-D');
- The system must allow for as much freedom as possible to execute any activity in any sequence;
- A system must be applicable for all design phases;
- The level of precision must be controllable by the system user;
- There must be integrated data and functions for CAD and CAE;
- Integration or data exchange with analysis software must be possible;
- The system must be stable and predictable;
- The model used by the system must be processable, so that a variety of derived information can be generated.

The first three requirements, the ability to work in both *two* and *three dimensions* and the requirement for *freedom*, stem directly from the conclusions for the ship hull design methodology, as listed in the previous sub-chapter. Of course all CAD systems have a particular functionality which is offered to the user, and as a matter of principle the confines of this functionality cannot be crossed. To underwrite this, in the sequel we shall call this requirement *enhanced freedom*. It implies that, as far as possible, the designer must be able to work any way he likes.

Unfortunately we cannot make an exhaustive list of possibilities implied by this requirement; freedom is unlimited. However, we can give some hints, for example it must be possible to:

- Use no prescribed working sequence. Allow any action at any time;
- Work with (planar or 3-D) curves or with surfaces;
- Manipulate control points (points of attraction of curves or surfaces) or data points (coordinates of the hull form itself);
- Work with hull form coefficients, or without;
- Use hull form transformation, at any moment;
- Import hull forms from other sources, for example from parametric or procedural methods as discussed in Section 1.1.2;
- Export hull forms to other systems. For example, for the analysis of flow around the hull, or for the calculation of stability or strength.

The *applicability* requirement implies that the method must be so versatile that it can be used in all four design phases.

The *precision* requirement follows from the applicability requirement, and it means that it must be possible to work to a degree of precision chosen by the designer. Of course, in practice, in a more mature design phase a higher degree of precision (and less vagueness) will be used.

The *integration* requirement is also derived from the applicability requirement, and it implies that for the system there is no difference between modelling for design and for engineering or manufacturing. All system functions must be available for CAD as well as CAE, and the hull form model must be shared.

Stability and *predictability* requirements must be included, because in order to be efficient, a system must not fail, nor must it surprise a trained user.

The *processability* requirement means that the system must allow easy post-processing, such as for generation of drawings, CAE data, rapid prototyping and mesh generation for potential-flow analysis.

1.2.3 User interface requirements and system goals

In addition to the system requirements formulated in the previous section, in this one we will focus on user interface requirements. Because it is our intention to create a *practically useful* software system, which can be used by any skilled naval architect, it is quite obvious that the implementation must use a visual user interface which is intuitive, versatile, consistent and also offers much freedom for the software user.
However, these requirements do not only apply on the visual interface (*how* the information is presented), but also on the interface content (*what* is presented to the user). The *intuitivity* and *versatility* requirements applied on the interface content imply for example that:

- From a certain level of abstraction, issues of mathematics must be hidden for the user. Because our attention is focussed on a ship designer with a technical background, geometrical interpretations of shape (such as tangents, curvature continuity or the classification of simple curves) can be presented to a system user. However, the user must be shielded from mathematical aspects of e.g. topology, geometric continuity or geometry representation;
- Any fairing or smoothing method incorporated into the new computer system must:
  - Enable global as well as local smoothing;
  - Use a smoothing criterion with a geometrical sense.

To summarize, the system and its implementation should lead to a practical system, which is suitable for *any activity* applied on the hull form, such as:

- Ab initio design;
- Design modifications during preliminary design and final design;
- Fairing with an arbitrary accuracy, including accuracy sufficient for production;
- Generation of engineering data or CAE data;
- Generation of drawings and tactile scale models;
- Import or digitize data of hull form, or parts of it;
- Perform numerical analysis, or farm out analysis to external software.

2. CAD Fundamentals

In this chapter we present a classification and a discussion of modelling fundamentals and of representations of curves and surfaces. Subsequently, we will investigate several existing methods for geometric modelling and for surface representations, which could be applied for a computer system for ship hull design and engineering. Because the issue of 'fairness' plays an important role in hull design, curve fairing algorithms will be discussed in detail. Finally, we deal with the place and role of rapid prototyping in the development of ship hulls.

The goal of this chapter is to give an overview of existing CAD methods, which can be used in a hull form design system. It is not our intention to draw final conclusions about the applicability of specific methods in this chapter. This will be done in Chapter Four.

2.1 Modelling and representations of geometries

Each system for modelling rigid solids uses one or more *geometric models* and one or more *geometry representations*. A **geometric model** is a theoretically supported information structure to describe the metric properties of objects, while a **geometry representation** describes the shape of curves and surfaces by means of equations and coefficients.

Alternative geometric modelling methods and geometry representations will be discussed in the next two sections.

2.1.1 Taxonomy of geometric modelling methods

In published literature, it is surprising that there is no consensus about terminology and classification of geometric models. [Baumgart, 1974], [Mäntylä, 1988], [Mortenson, 1985], [Muuss and Butler, 1991], [Piegl, 1993] and [Zeid, 1991] all use their own definitions and classifications.

Matters are still more confused by the fact that technical publications, e.g. [Michelsen, 1994], STEP [Owen, 1997] and [Koelman, 1997b], tend to make only a division between *geometry* and *topology*, while papers which revert to Requicha's work ([Requicha, 1980]) distinguish between *complete models* and *non-complete models*.

In order to be able to classify modelling approaches, we have to define some concepts:

- A **solid** is a rigid, finite, continuous and continuously bounded subset of R^3;
- A **topology** of shape S expresses the non-metric continuity properties of subsets of S (this is an interpretation of the formal definition of [Gomes and Middleditch, 1997]);
- A **2-manifold** is an open topological space where every point has a neighbourhood which is topologically equivalent to an open disk of E^2 (According to [Mäntylä, 1988]);
- A **manifold solid** is a solid, with a boundary which is topologically equivalent to a

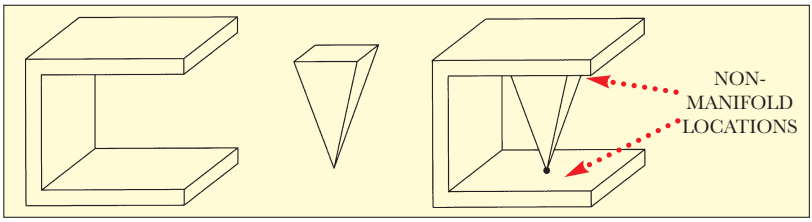

Figure 2.1 *Manifold and non-manifold solids.*

2-manifold. A solid which is not manifold, is called a non-manifold. Figure 2.1 gives examples of two manifold solids at the left, and a non-manifold solid at the right;

- A **geometrical model** is **complete**, if it represents one and only one object. The completeness of the geometrical model in this sense is related to the information content, rather than to its uniqueness. Consequently a complete geometric model fully characterizes an object in terms of:
  - Identification (such as names and identifiers);
  - Geometry (the shapes, and dimensions of objects);
  - Topology (information about the entities, and explicit information about connections between those entities);
  - Location (the position of objects);
  - Attributes (such as material, and colour).

  A model which is not complete is called *incomplete*.

The taxonomy of modelling methods according to [Horváth and Juhasz, 1997], which is used in this thesis, is presented in Figure 2.2.

2.1.2 Alternative geometry representations for curves and surfaces

Numerous geometry representations have been developed in the course of time, and it is not the intention to describe them all in this section. Only those with relevance to free form ship hull design will be discussed. Detailed discussions of a wide range of representations of geometry can be found in [Faux and Pratt, 1979], [Mortenson, 1985], [Farin, 1990], [Zeid, 1991], [Hoschek and Lasser, 1992] and [Piegl, 1993].

Figure 2.3 presents a taxonomy of geometry representations for curves. The primary distinction between different representations concerns the use of the coordinate system:

- Explicit: $y = f_1(x)$, $z = f_2(x)$;
- Implicit: $f_1(x,y) = 0$, $f_2(x,z) = 0$;
- Parametric: $x = f_1(t)$, $y = f_2(t)$ and $z = f_3(t)$, where t is the curve parameter.

In the figure we see some representations which may be used for special purposes (for example for roundings of waterlines), such as clothoids and conics. We also recognize the polynomial, which is in parametric form defined as

$$\mathbf{p}(t) = \sum_{i=0}^{n} \mathbf{a}_i \, . \, t^i ,$$

with n the degree, $\mathbf{a}$ the polynomial coefficient vector, and t the parameter,

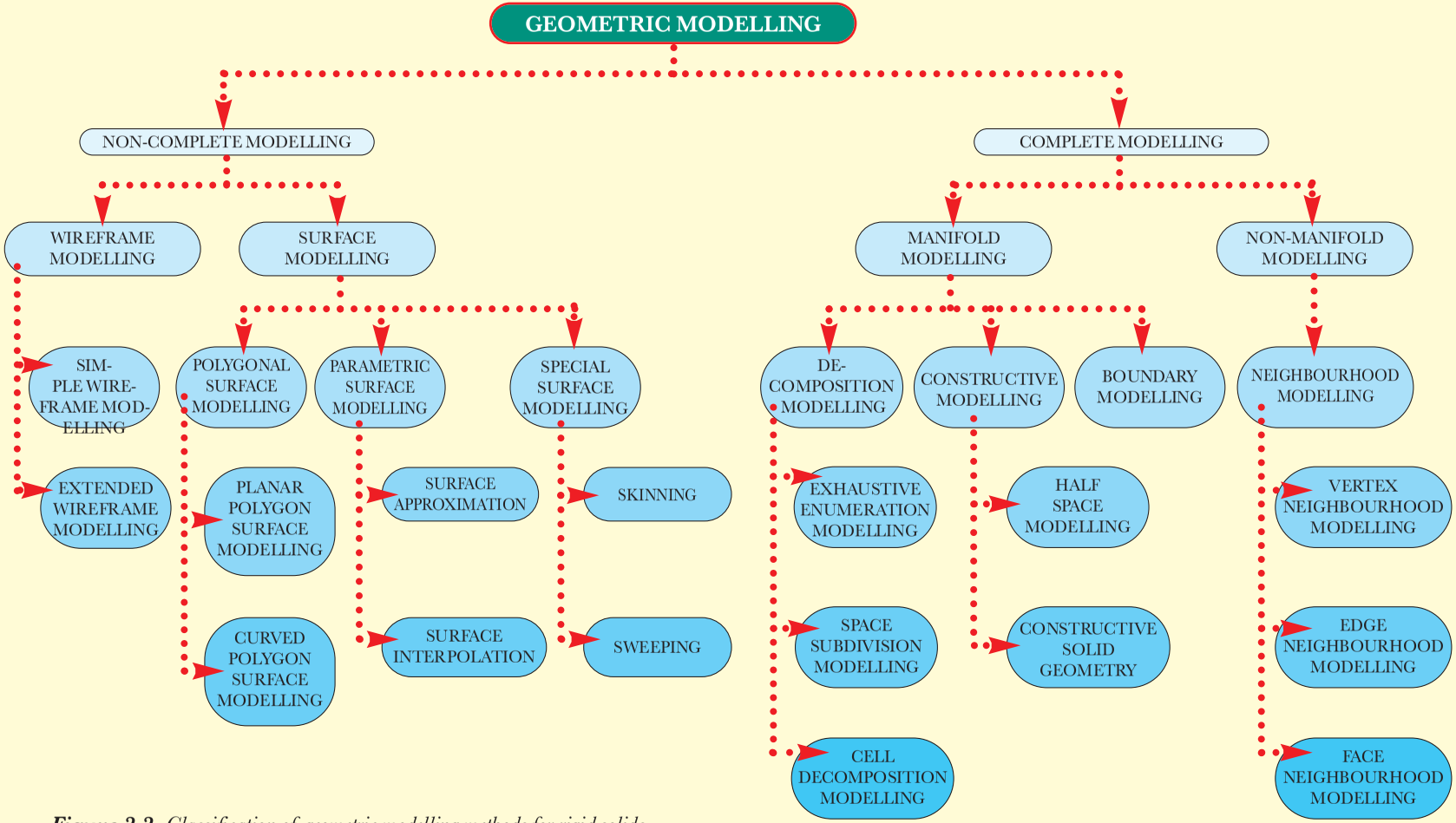

Figure 2.2 *Classification of geometric modelling methods for rigid solids.*

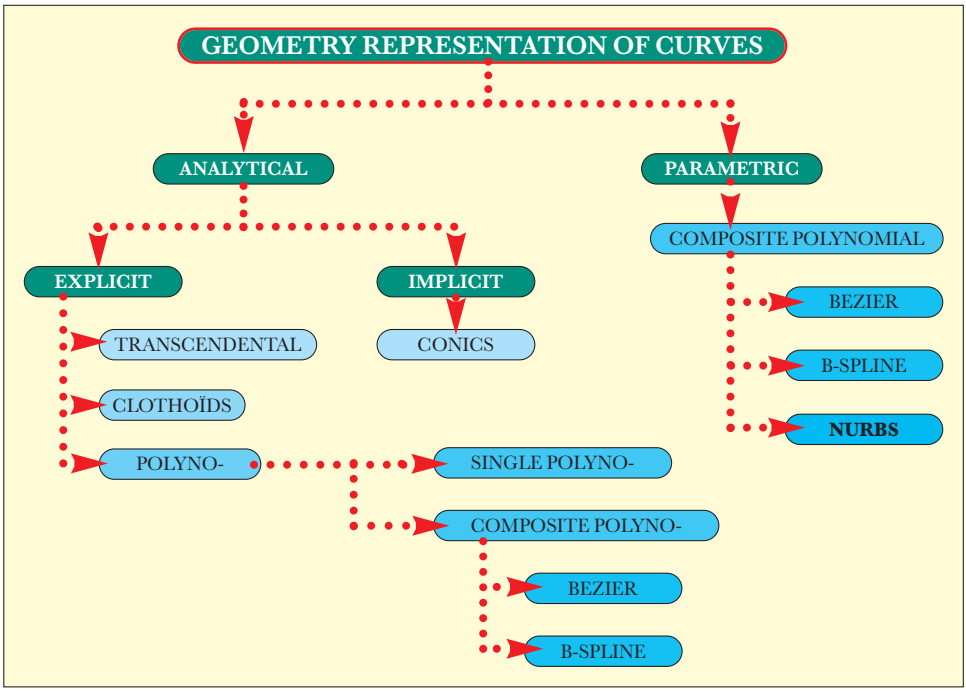

Figure 2.3 *Geometry representations for curves.*

the B-spline, with a parametrical definition of

$$\mathbf{p}(t) = \sum_{i=1}^{L} \mathbf{P}_i N_i^n(t),$$

with L the number of control points, N the B-spline basis function, **P** the vector control points, and t the parameter,

and the NURBS, with a parametric definition of

$$\mathbf{p}(t) = \frac{\sum_{i=1}^{L} \mathbf{P}_i w_i N_i^n(t)}{\sum_{i=1}^{L} w_i N_i^n(t)}, \qquad (2.1)$$

with w_i the additional weight factors.

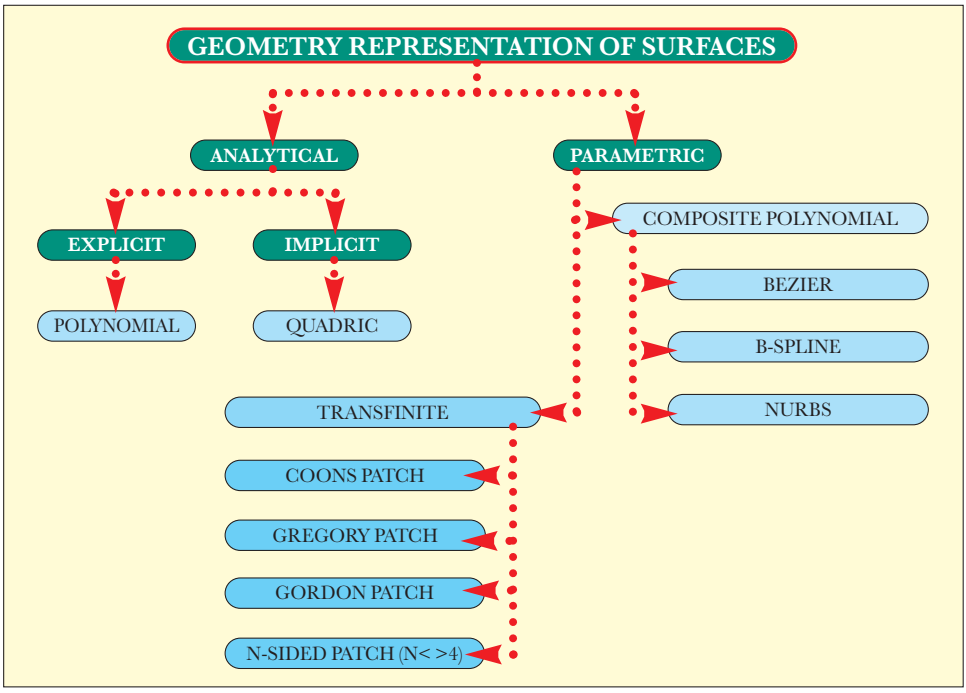

Figure 2.4 *Geometry representations for surfaces.*

Figure 2.4 contains a classification of geometry representation for surfaces, where also three systems are used:

- Explicit: $z = f(x,y)$;
- Implicit: $f(x,y,z) = 0$;
- Parametric: $x = f_1(u,v)$, $y = f_2(u,v)$ and $z = f_3(u,v)$, where u and v are the surface parameters.

We see the class of transfinite patches, which can be used to represent a surface, where the shape of the surface is derived from the shape of curves inside or bounding the surface.

We also see the polynomial surface

$$\mathbf{r}(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} \mathbf{a}_{i,j}\, u^i\, v^i \,,$$

with **a** the coefficient vector, and u and v the surface parameters,

the parametric B-spline surface

$$\mathbf{s}(u,v) = \sum_{j=1}^{K} \sum_{i=1}^{L} \mathbf{P}_{i,j}\, \mathrm{N}_i^{\mathrm{n}}(u)\, \mathrm{M}_j^{\mathrm{n}}(v) \,,$$

where N and M are the B-spline basis functions, P the control points and u and v the parameters,

and the parametric NURBS surface

$$\mathbf{s}(u,v) = \frac{\sum_{j=1}^{K}\sum_{i=1}^{L} \mathbf{P}_{i,j}\, w_{i,j}\, N_i^n(u)\, M_j^n(v)}{\sum_{j=1}^{K}\sum_{i=1}^{L} w_{i,j}\, N_i^n(u)\, M_j^n(v)}.$$

It is widely recognized that parametric B-spline and NURBS do offer a number of attractive properties for the design of curves and surfaces. They are:

- Local control; moving one control point only results in local change of the curve or surface;
- Affine invariance; an affine transformation (a combination of rotation, translation, shear or scaling) of the control points, is also applied to the curve or surface;
- Linear precision; the ability to create a straight line;
- Convex hull property; each point of the curve or surface lies in the convex hull of the control points;
- Variation diminishing; a curve is not intersected by any straight line more often than the polygon of control points itself.

B-spline and NURBS surfaces can be manipulated by means of direct manipulation of the 3-D control points (and the corresponding weight in case of NURBS), or by means of data interpolation, where the surface is reconstructed through a set of predefined data points.

A problem with surface reconstruction, however, is the assignment of parameter values to the data points (see [Ma and Kruth, 1995], [Sarkar and Menq, 1991] and [Alfeld, 1989] for more details).

2.2 Techniques for complete geometric modelling

2.2.1 General overview of modelling methods

According to the classification of Figure 2.2, geometrically complete modelling is subdivided into manifold and non-manifold modelling. We will first discuss manifold modelling.

In Figure 2.1 we see that three different methods of manifold modelling are available: *decomposition modelling*, *constructive modelling* and *boundary* modelling:

- A **decomposition model** represents a solid as a collection of simple objects, which share the common boundary;
- A **constructive model** represents a solid as boolean combinations of primitive solids;
- A **boundary model** represents the topology of a solid by connected faces, which are bounded by loops of edges, while the edges are bounded by vertices. It represents the geometry by curves and surfaces.

An example of a decomposition model is given in Figure 2.5, and of a constructive model in Figure 2.6. Figure 2.7 shows the elements of a boundary model: the shell (a), the faces (b) and the vertices and edges (c).
Concentrating on application for a ship hull, we consider a decomposition model to be unsuitable, because it cannot exactly represent the curved hull surface. Even so, a constructive model does seem less appropriate because a ship hull is, in general, not a combination of primitives.

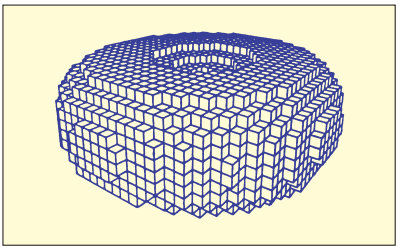

Figure 2.5 *Decomposition model (reprint from [Mäntylä, 1988]).*

The approach of a boundary model, to represent entities on the boundary, is the best one for ship hull surface modelling, albeit that for ship hull modelling we are interested in curved faces. Therefore, in the subsequent section we will concentrate on the boundary modelling subject.

2.2.2 Boundary modelling

In its early manifestations of the boundary model, the object is represented by the relationships between adjacent non-curved faces, non-curved edges and vertices. Such a model is called a polyhedral Boundary REPresentation, or polyhedral BREP. In more recent implementations the BREP is extended to allow for curved faces and edges, and is thus not polyhedral anymore.

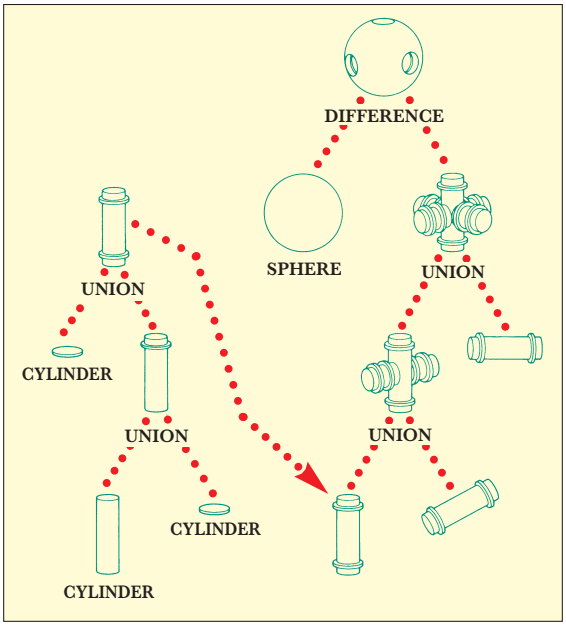

Figure 2.6 *Constructive model (reprint from [Mortenson, 1985]. Copyright © 1985 John Wiley & Sons).*

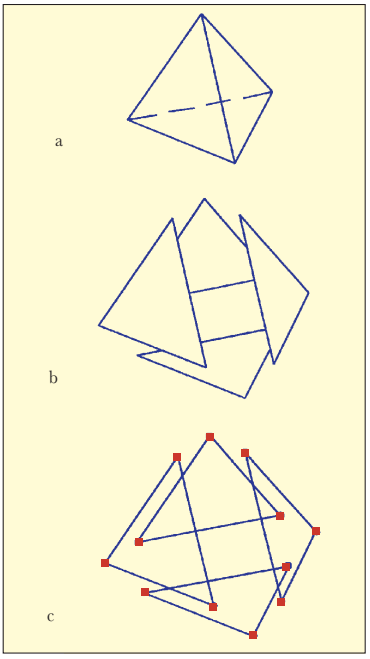

Figure 2.7 *Boundary model.*

In this section we limit ourselves to the fundamentals of the polyhedral BREP, the inclusion of curved elements will be discussed in the next sub-chapter.

With the three types of elements of a BREP, nine adjacency relationships are possible. In [Weiler, 1985] it was shown, however, that only three relationships are sufficient:
- Vertex – edge;
- Edge – edge;
- Face – edge.

The most commonly used structures are the *winged-edge*, which is based on the edge – edge relationship, and the face – edge based *halfedge*. However, with the winged-edge, an edge can be traversed in two directions, so every time an edge is accessed, it must be determined which edge side was intended. So we will discuss the halfedge structure, which was also used in [Mäntylä, 1988], in more detail.

The basis of the halfedge structure is a set of pointers which belongs to one half of an edge, and which points to adjacent topological entities, such as to the adjacent vertex, face or other halfedges. See Figure 2.8 for a schematic representation of the use of halfedges HE1 and HE2 of the edge between vertices V1 and V2.

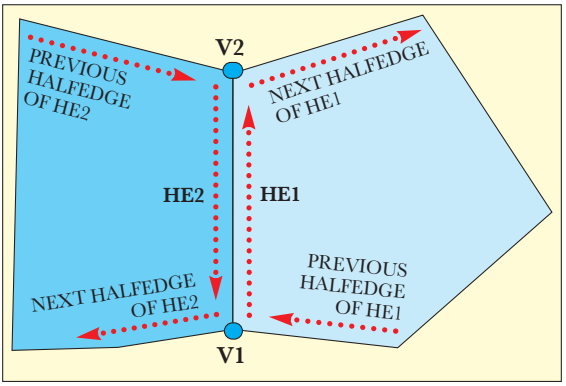

Figure 2.8 *Halfedge data scheme.*

With this foundation the solid is modelled as lists of elementary topological elements, which all point to elements in their neighbourhood. The used topological elements are:
- The **shell**; the boundary of a solid;
- The **face**; a finite and non self-intersecting part of a shell. The boundary of a face consists of edges, which can be organized in loops of edges. A face can be bounded by more than one loop. In that case one loop is designated to be the outer boundary, and the others represent 'gaps' in the face;
- The **edge**; a non self-intersecting topological entity which corresponds to a metric curve, and is bounded by two vertices;
- The **loop**; a closed, non self-intersecting boundary of a face, which consists of an ordered sequence of edges;
- The **halfedge**; a logical entity to indicate the two possible orientations of a physical edge;
- The **vertex**; the topological entity which corresponds to a metric point.

The relationships between these elements are sketched in Figure 2.9

It would be perfectly possible to implement a solid modeller with this data structure only, but maintaining all these relationships would make implementation of the system quite laborious. Fortunately the theory of topology gives extra support with a simple relationship between the topological entities. When V is the number of vertices, E the number of edges,

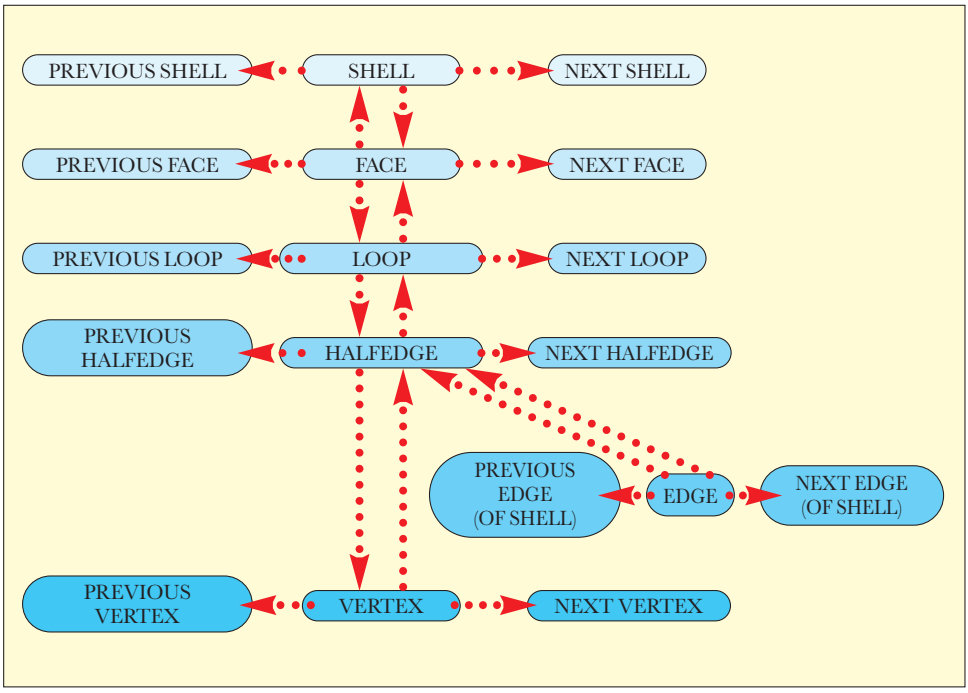

Figure 2.9 *Halfedge data structure.*

F the number of faces and S the number of shells, then for every manifold solid the Euler-Poincaré formula, is V – E + F – 2S = 0. For more complex solids an extended equation is available, which can be given after one extra topological entity is defined:

- The **hole** is a geometric entity which makes a manifold solid multiple connected. Intuitively this is a hole *through* a closed solid. The number of holes is also known as the *genus* of a solid.

For a solid containing multiple loops (L) and holes (H), the relation reads V + F – E – L = 2(S – H).

The Euler-Poincaré formula can be used to define so-called Euler operators, which are to maintain the validity of the solid. If, for example, an edge is added to any solid object, a vertex or a face must also be added to that object, in order to maintain V + F – E – L – 2(S – H) at zero.

For basic manipulations of the solid, ten useful Euler operators can be defined. Constructive operators are:

- MEV Make edge and vertex;
- MEF Make edge and face;
- MVFS Make vertex, face and shell;
- KEML Kill edge, make loop;
- KFMLH Kill face, make loop and hole.

and the destructive operators are:
- KEV Kill edge and vertex;
- KEF Kill edge and face;
- KVFS Kill vertex, face and shell;
- MEKL Make edge, kill loop;
- MFKLH Make face, kill loop and hole.

In a software implementation of boundary modelling it is good practice, according to the ideas of [Baumgart, 1974], to make software functions for each Euler operator, and to process all effects on the relationships between the topological elements strictly within those Euler functions. In this way the Euler functions serve as an abstraction layer, and hide the nasties of the manipulation of relationships for the higher level application layer.

In order to provide a comprehensive survey, we also mention the most important properties of non-manifold modelling. Methods for non-manifold modelling may be viewed as a replacement of manifold modelling, and implementations are available (see for instance [Yamaguchi and Kimura, 1995] or the radial-edge structure of [Weiler, 1986a]), but the added complexity compared with manifold modelling is enormous. Whereas for the manifold boundary representation the 10 Euler operators are sufficient, according to [Weiler, 1986b], this scheme requires about 50 Euler-like operators.

2.3 Representation of surface patches

In the previous sub-chapter the polyhedral BREP was discussed. Because a ship hull surface is typically curved, we have to look into methods for representing a curved surface. Because our analysis of Chapter One indicates that it must be possible to design a ship hull with curves only, in this sub-chapter we will investigate techniques which generate a curved surface on the basis of the shape of curves lying on, or at the boundary of, the surface.

First, however, we have to define some notions which will be used in this sub-chapter, and in the remainder of this thesis:
- The **patch** is a continuous surface bounded by non self-intersecting curves, with no curves in its interior;
- A **patch complex** is an arrangement of patches which are connected to each other on their boundary, with explicit or implicit continuity conditions;
- A **regular patch complex** is a patch complex which:
  - Is represented by parametric surface **S**(u,v), with N continuous curves of constant u parameter and M continuous curves of constant v parameter;
  - Consists of NM four-sided patches.

  Usually the *u* and *v* curves are orthogonal in parameter space.

2.3.1 Single four-sided patch

As presented in [Coons, 1974], the basic idea is to interpolate a surface on four arbitrary bounding curves. This *transfinite interpolation* consists of four steps: first (a) interpolate the ruled surface between two opposite curves, then (b) interpolate the ruled surface between the other two curves, add the two surfaces and (c) subtract the surplus, which happens to be the bilinear interpolant to the four corners. This is illustrated in Figure 2.10, which shows the first ruled surface, the second ruled surface, the bilinear interpolant and, finally, the resulting interpolating surface.

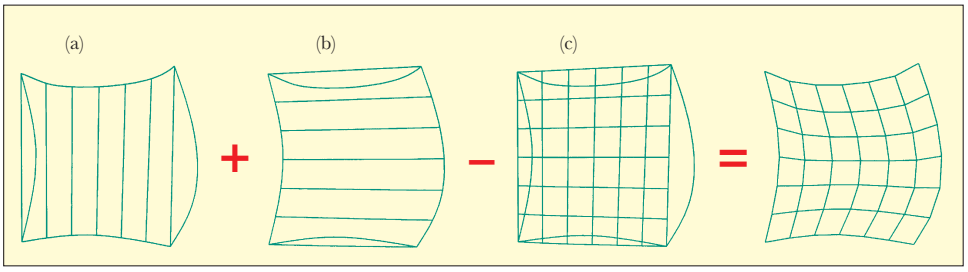

Figure 2.10 *Boolean sum of surfaces.*

To maintain GC^1 continuity, first derivative information must also be included. Suppose we have four bounding curves and for each curve we know the tangent ribbon (which is the collection of tangent vectors along the curve).
It is illustrated in Figure 2.11, where the subscripts $_u$ and $_v$ denote differentiation in terms of u and v respectively. So for example $\mathrm{F}_v(u,0)$ is the tangent vector in v direction at $\mathrm{F}(u,0)$, which is

$$\frac{\partial \mathrm{F}(u,0)}{\partial v}$$

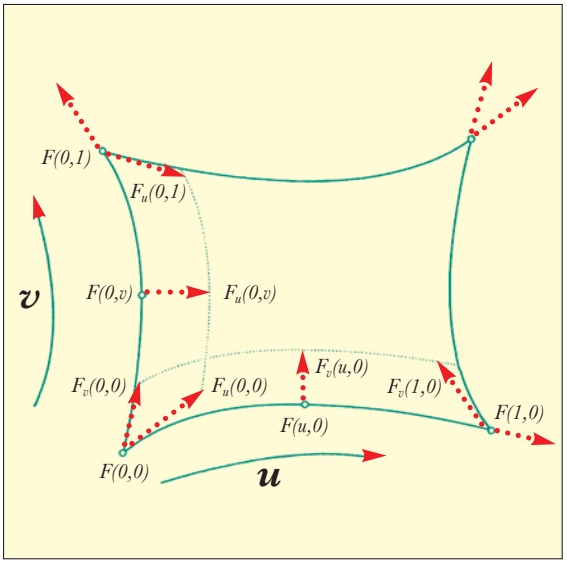

Figure 2.11 *Position and tangent vectors.*

We first define the four cubic Hermite interpolation functions (expressed in terms of the parameter u):

$\mathrm{H}_0(u) = 1 - 3u^2 + 2u^3$,
$\mathrm{H}_1(u) = 3u^2 - 2u^3$,
$\mathrm{H}_2(u) = u - 2u^2 + u^3$,
$\mathrm{H}_3(u) = -u^2 + u^3$.

Then the Coons patch $\mathbf{F}(u,v) = \mathbf{F}_1(u,v) + \mathbf{F}_2(u,v) - \mathbf{F}_{12}(u,v)$,

with the two ruled surfaces:

$$\mathbf{F}_1(u,v) = \mathrm{H}_0(u).\mathbf{F}(0,v) + \mathrm{H}_1(u).\mathbf{F}(1,v) + \mathrm{H}_2(u).\mathbf{F}_u(0,v) + \mathrm{H}_3(u).\mathbf{F}_u(1,v)\,,$$
$$\mathbf{F}_2(u,v) = \mathrm{H}_0(v).\mathbf{F}(u,0) + \mathrm{H}_1(v).\mathbf{F}(u,1) + \mathrm{H}_2(v).\mathbf{F}_v(u,0) + \mathrm{H}_3(v).\mathbf{F}_v(u,1)\,,$$

and the bicubic tensor product interpolant

$$\mathbf{F}_{12} = \mathbf{h}(u)\ \mathbf{M}\ \mathbf{h}^{\mathrm{T}}(v)\,,$$

where

$$\mathbf{h}(u) = [\mathrm{H}_0(u)\,,\ \mathrm{H}_1(u)\,,\ \mathrm{H}_2(u)\,,\ \mathrm{H}_3(u)]$$

and

$$\mathrm{M} = \begin{bmatrix} \mathbf{F}(0,0) & \mathbf{F}(0,1) & \mathbf{F}_v(0,0) & \mathbf{F}_v(0,1) \\ \mathbf{F}(1,0) & \mathbf{F}(1,1) & \mathbf{F}_v(1,0) & \mathbf{F}_v(1,1) \\ \mathbf{F}_u(0,0) & \mathbf{F}_u(0,1) & \mathbf{F}_{uv}(0,0) & \mathbf{F}_{uv}(0,1) \\ \mathbf{F}_u(1,0) & \mathbf{F}_u(1,1) & \mathbf{F}_{uv}(1,0) & \mathbf{F}_{uv}(1,1) \end{bmatrix},$$

with the twist vector

$$\mathbf{F}_{uv}(u,v) = \frac{\partial \mathbf{F}_v(u,v)}{\partial u}\,.$$

Theoretically

$$\frac{\partial \mathbf{F}_u(u,v)}{\partial v} = \frac{\partial \mathbf{F}_v(u,v)}{\partial u}\,,$$

but unfortunately, in practical implementations, derivative information may be based on estimation or interpolation. In this case twist incompatibility is encountered, it means that

$$\frac{\partial \mathbf{F}_u(u,v)}{\partial v} \neq \frac{\partial \mathbf{F}_v(u,v)}{\partial u}.$$

Twist incompatibility may cause undulations in the surface representation, and can also cause numerical instability.

To combat these effects, several compatibility corrected schemes have been developed, from which Gregory's ([Gregory, 1982]) is the most commonly used, where the $\mathbf{F}_{uv}$'s at the corners are replaced by a rational combination of

$$\frac{\partial \mathbf{F}_u(u,v)}{\partial v} \quad \text{and} \quad \frac{\partial \mathbf{F}_v(u,v)}{\partial u}.$$

To obtain GC^2 continuity an extended representation can be constructed, see [Hagen and Schulze, 1987]. To reflect the increased continuity information, additional coefficients, containing second-order partial derivatives, are of course required. The interpolant to the corners is biquintic tensor product for this case.

2.3.2 Regular patch complex

Just as the Coons/Gregory patch interpolates a surface between four boundary curves, the Gordon surface interpolates a surface through a regular network of curves (sketched in Figure 2.12). Similar to the Coons/Gregory patch, according to [Gordon, 1969] the surface through the network is the boolean sum of three interpolants.

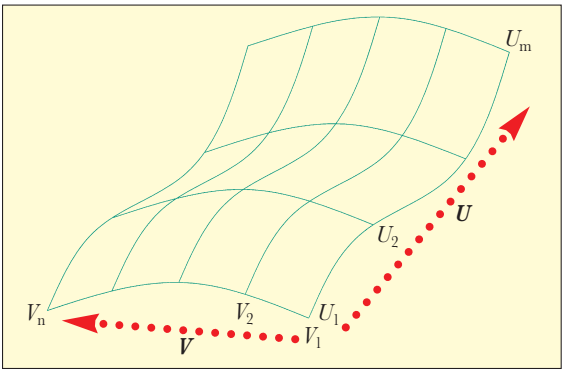

Figure 2.12 *Regular patch complex.*

The mathematical formulation of the Gordon surface first requires the definition of scalar cardinal functions $C(x_i)$, where $C(x_i) = 1$ for one specific value of i, and zero for all other integer values of i. We define two sets of cardinal functions:

- n cardinal functions $C_j(v)$, $j = 1..n$. For the n values on the domain of v (v_k, $k = 1..n$) the cardinal functions are defined as $C_j(v_k) = \delta_{jk}$;
- m cardinal functions $C_i(u)$, $i = 1..m$. For the m values on the domain of u (u_l, $l = 1..m$) the cardinal functions are defined as $C_i(v_l) = \delta_{il}$.

For both cases δ is the Kronecker delta.
For C any appropriate function satisfying the cardinality conditions can be chosen.

Let's have a network of n curves which run in u-direction (the curves $F(u,v_j)$, $j = 1..n$) and m curves which run in v-direction (the curves $F(u_i,v)$, $i = 1..m$). The first interpolant, parametrically orthogonal to the v-curves, is

$$\mathbf{F}_1(u,v) = \sum_{i=1}^{m} \mathbf{F}(u_i,v) \,.\, C_i(u) \,,$$

and the second interpolant, orthogonal to the u-curves, is

$$\mathbf{F}_2(u,v) = \sum_{j=1}^{n} \mathbf{F}(u,v_j) \,.\, C_j(v) \,.$$

The tensor product surface $\mathbf{F}_{12}$ that interpolates all n.m network points is:

$$\mathbf{F}_{12}(u,v) = \sum_{i=1}^{m} \sum_{j=1}^{n} \mathbf{F}(u_i,v_j) \,.\, C_i(u) \,.\, C_j(v) \,.$$

The Gordon surface is $\mathbf{F}(u,v) = \mathbf{F}_1 + \mathbf{F}_2 - \mathbf{F}_{12}$.

2.3.3 Methods for representation of N-sided patches

In section 2.3.1 the representation of a four-sided patch was discussed, in this section we will concentrate on methods for patches with a number of sides $N \neq 4$. The modelling of a (single) N-sided patch is the subject of quite a large number of publications (see [Peters, 1990a] for an overview). The methods discussed in those publications can be grouped into three categories:

- Refinement of a control network by subdivisioning;
- Local parametrizations followed by a boolean sum, or a convex combination;
- Hierarchical decomposition.

2.3.3.1 *Refinement by subdivisioning*

It has been proved, among others by de Boor in [Piegl, 1993], that when an extra knot is added to a B-spline control polygon, without changing the shape of the curve (a process called knot line refinement), the new net of control points lies between the old polygon and the curve. When many more vertices are added, the polygon converges to the curve itself. Quite similar, for N-sided surface patches a net of control points can be recursively subdivided until a sufficiently accurate representation of the surface is obtained. This process, as discussed in [Nasri, 1987], [Nasri, 1991], [Peters, 1990b], [Peters, 1994] and [Warren, 1992], is illustrated in Figure 2.13.

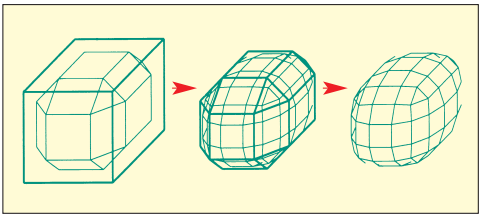

Figure 2.13 *Subdivision method (reprint from [Peters, 1994]).*

2.3.3.2 *Boolean sums (or convex combinations)*

In [Charrot and Gregory, 1984], [Gregory, 1982], [Gregory, 1984], [Gregory, 1989], [Gregory and Hahn, 1989], [Gregory et al, 1993], [Kato, 1991], [Kuriyama, 1994] and [Varady, 1991] variations are described of a method where an N-sided single patch is represented by a combination of N corner patches. For a pentagonal patch this is illustrated in Figure 2.14.

Here we will discuss the method according to [Gregory, 1982]. In Figure 2.15 the coordinate map of an N-sided patch is sketched. In parameter space this patch is convex by definition, however, in model space the patch may be concave.

For a parameter point $\mathbf{X}$ inside the boundary of the N-sided patch a barycentric parametrization is used. j is the index of each corner ($j = 1..N$), as in Figure 2.15, and d_j is the perpendicular distance of $\mathbf{X}$ to the side E_j. For each corner we have local parameters u_j and v_j, with

$$u_j = \frac{d_{j-1}}{(d_{j-1} + d_{j+1})}$$

and

$$v_j = \frac{d_j}{(d_{j-2} + d_j)}.$$

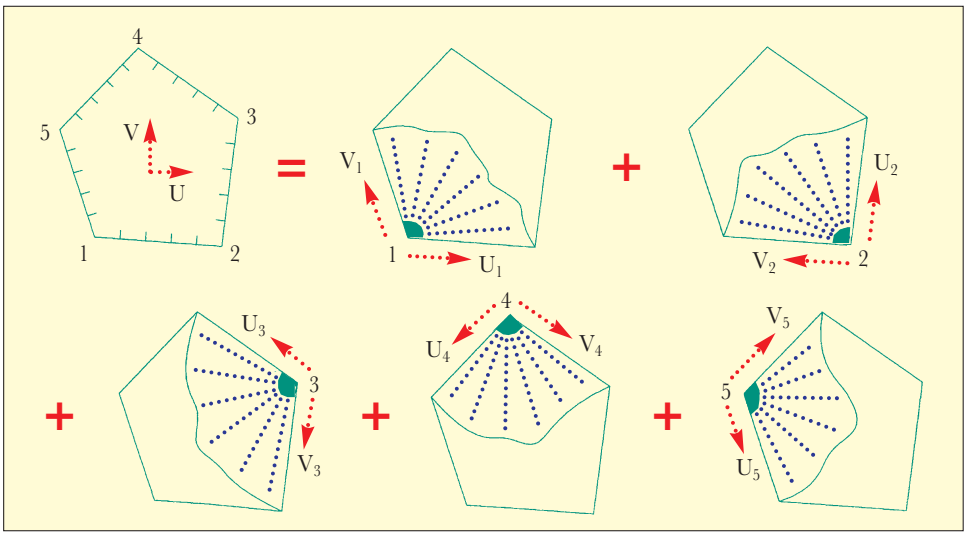

Figure 2.14 *A 5-sided patch as a combination of five corner patches.*

Then for each of the N sides we have a positional function $\mathbf{f}_j(u)$, $j = 1..N$, and the cross-boundary vector function $\mathbf{t}_j(u)$. For the j^{th} corner we have two linear interpolants, defined in the local parameters u_j and v_j:

$$\mathbf{T}_1(u_j,v_j) = \mathbf{f}_{j-1}(v_j) + u_j\,\mathbf{t}_{j-1}(v_j)$$
$$\mathbf{T}_2(u_j,v_j) = \mathbf{f}_j(u_j) + v_j\,\mathbf{t}_j(u_j)$$

and the tensor product interpolant:

$$\mathbf{T}_{12}(u_j,v_j) = [1 \;\; u_j] \begin{bmatrix} \mathbf{f}_j(0) & \mathbf{t}_j(0) \\ \mathbf{t}_{j-1}(0) & \dfrac{\partial \mathbf{t}_j(0)}{\partial u} \end{bmatrix} \begin{bmatrix} 1 \\ v_j \end{bmatrix}, \tag{2.2}$$

where

$$\frac{\partial \mathbf{t}_j(0)}{\partial u}$$

is the twist vector. If twist vector incompatibility occurs, this term can be replaced by a rational combination of

$$\frac{\partial \mathbf{t}_j(0)}{\partial u} \quad \text{and} \quad \frac{\partial \mathbf{t}_{j-1}(0)}{\partial v}.$$

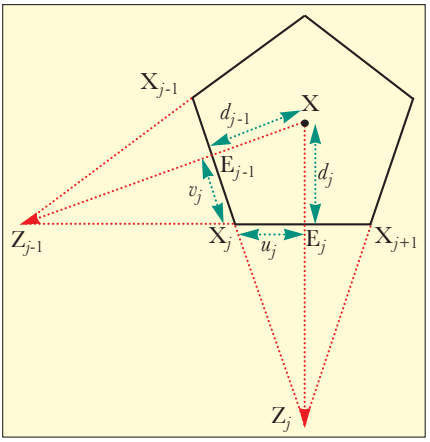

Figure 2.15 *Coordinate map (in parameter space) of N-sided patch.*

With these, the boolean sum interpolant for the patch in each corner is

$$\mathbf{p}_j(\mathbf{X}) = \mathbf{T}_1(\mathbf{X}) + \mathbf{T}_2(\mathbf{X}) - \mathbf{T}_{12}(\mathbf{X}) . \tag{2.3}$$

The final interpolant is a weighted combination of the N corner patches:

$$\mathbf{p}(\mathbf{X}) = \sum_{j=1}^{N} w_j(\mathbf{X})\, \mathbf{p}_j(\mathbf{X}) ,$$

where $w_j(\mathbf{X})$ is a weight factor (and Π denotes repeated multiplication):

$$w_j(\mathbf{X}) = \frac{\prod_{i=1,\, i\neq j-1,\, i\neq j}^{N} d_i^2}{\sum_{k=1}^{N} \prod_{i=1,\, i\neq k-1,\, i\neq k}^{N} d_i^2} .$$

2.3.3.3 *Hierarchical decomposition*

As described in [Chiyokura et al, 1991], [Gregory et al, 1989], [Hahn, 1989a] and [Hahn, 1989b] a possibility to represent an N-sided patch by subdivision into N connected 4-sided patches, as illustrated in Figure 2.16 for a hexagon. As the first step in the construction of the patch the location of and the tangent plane at the central vertex **S** must be specified, either by the user or by an estimating algorithm. In the second step the 4-sided patches are constructed, with the boundary condition that the neighbouring patches connect with GC[1] surface continuity.

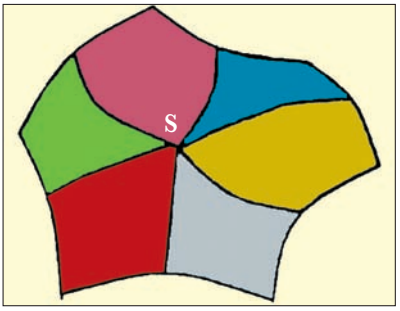

Figure 2.16 *Dividing a hexagon into six 4-sided patches.*

2.4 Curve fairing[1]

2.4.1. Interpretation of fairness

The question of the fairness of a curve is essentially subjective. Let us take, for instance, the curves of Figure 2.17, which both go through the same three points. However, we may think the green curve is considered more fair than the blue one; this judgement is based

[1] In this thesis the words *fair* and *smooth* will be used interchangeably. According to our understanding they refer to the same property, although they express different points of view.

on personal expectations. Several authors have tried to formulate more objective criteria, such as Su and Liu [Su and Liu, 1989] who consider a curve *fair* if three conditions are fulfilled:

- The curve is of GC^2 continuity;
- There are no unwanted inflection points;
- The curvature varies in an even manner.

Nevertheless, even these definitions describe no sound mathematical properties. In order to avoid confusion for a user of the software, we need an objective measurement of fairness.

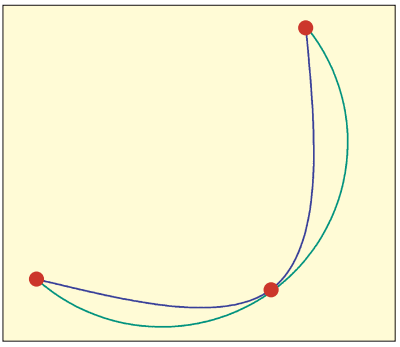

Figure 2.17 *Two different curves through the same three points.*

There are multiple mathematical definitions around, which can be grouped into two categories. The first category specifies the global fairness, and is based on a mathematical-mechanical model of the complete curve. The second category relies on a more intuitive judgement of local properties.

In [Giunnis and Wahl, 1998] and [Roulier and Rando, 1994] many definitions of fairness are listed. Some typical definitions of global fairness are:

- **Definition #1 of global fairness**. The fairest curve **r** is the one which minimizes the strain energy E, which is given by

$$E = \int (\kappa(s))^2 \, ds,$$

where s is the arc length of the curve and κ is the curvature:

$$\kappa = \frac{|\mathbf{r}' \times \mathbf{r}''|}{|\mathbf{r}'|^3};$$

- **Definition #2 of global fairness**. The fairest curve **r** is the one which minimizes

$$\int (\mathbf{r}''(t))^2 \, dt,$$

where **r** is the vector function, and t the curve parameter. This is called the 'minimum property';

- **Definition #3 of global fairness**. A curve is fair if its curvature plot is continuous and consists of only a few monotonic pieces [see Farin, 1990].

Measurements for local fairness are:

- **κ-discontinuity**. The local fairness at the knot u_j of a cubic B-spline is represented by

$$\kappa\text{-discontinuity} = \left| \frac{d\kappa}{ds}(u_j+) - \frac{d\kappa}{ds}(u_j-) \right| \text{ (See [Pigounakis et al, 1996]).}$$

Here u_j+ means $u_j + \Delta u$ and u_j- is $u_j - \Delta u$;

- **f '''-discontinuity**. The local fairness of a curve **f** is represented by the square of the third-derivative discontinuity at u_j:
  f '''-discontinuity = $[f'''(u_j+) - f'''(u_j-)]^2$. See [Farin et al, 1987]. The f '''- discontinuity is also called the third-derivative jump.

With these definitions of fairness, local or global fairing algorithms can be developed, which are the subject of discussion in the next two sections.

2.4.2 Local fairing algorithms

[Kjellander, 1983] describes a very simple fairing principle, which consists of five steps. Given N points in 3-D Euclidian space:

1. Generate a cubic parametric curve **C** through the points;
2. Study the curve **C**, and decide which point ($\mathbf{p}_i$) should be moved in order to improve fairness;
3. Construct a local cubic curve $\mathbf{C}^*$ in the surroundings of $\mathbf{p}_i$, solely based on the neighbouring points $\mathbf{p}_{i-1}$ and $\mathbf{p}_{i+1}$, and the tangents $\mathbf{p'}_{i-1}$ and $\mathbf{p'}_{i+1}$;
4. Move the offending point $\mathbf{p}_i$ to its new location $\mathbf{p}_i^*$ on the local cubic curve $\mathbf{C}^*$;
5. If the curve is not fair enough, according to a fairness definition which can be freely chosen, go back to the first step.

Although the modifications are of local nature, the scheme may be called semi-global, because the first step involves a global construction of the curve, through all data points.

Based on this idea [Farin et al, 1987] proposed a truly local fairing scheme.
Given a B-spline with a knot sequence $t = \{t_0, t_1 ... t_N\}$, polygon points $\mathbf{d}_{i,0}$ and a maximum error σ, the developed knot removal / knot reinsertion algorithm comprises four steps:

1. Identify the offending knot t_j, which is the knot with the highest square of the third-derivative jump;
2. Remove this knot t_j, and compute a new polygon, approximating the original one;
3. Using this polygon, reinsert t_j so that the new curve is again defined over the original knot sequence;
4. Calculate the sum of differences between the original polygon points $\mathbf{d}_{i,0}$ and the new polygon points $\mathbf{d}_{i,1}$. If this sum is smaller than a permitted error σ the algorithm has come to an end, otherwise go to Step One.

2.4.3 Global fairing algorithms

The process of fairing of a sequence of data points can be regarded as a compromise between two criteria:

- To comply with one of the global fairness indicators from Section 2.4.1;
- An acceptable difference between the original and the faired data points.

Given a set of N parameters $\{t_1 < t_2 < t_3 ... < t_N\}$, a corresponding set of N data points $\mathbf{q}_i$, a corresponding set of N positive weights w_i, and a B-spline function $\mathbf{f}(t)$, the *complete smoothing algorithm* by [Pigounakis et al, 1996] uses two scalar functions. $E(\mathbf{f})$ which indicates the differences between the original and the faired data points (as well as the differences of end-derivatives, which are omitted here), and $\mathcal{J}(\mathbf{f})$, which indicates the fairness of the curve:

$$E(\mathbf{f}) = \sum_{i=1}^{N} w_i \left| \mathbf{f}(t_i) - \mathbf{q}_i \right|^2$$

$$\mathcal{J}(\mathbf{f}) = \int (\mathbf{f}''(t))^2 \, dt .$$

The goal is to minimize the function $\mathbf{J}(f) + W\mathbf{E}(f)$, where W is a global weight factor, governing the balance between fairness and the desire to stick close to the original data points.

Another solution is proposed in [Dierckx, 1993]. Here $E(\mathbf{f})$, the sum of the squared deviations between the faired points and the original data, is not a part of the function to be minimized. It is constrained to be smaller than a predefined maximum error S. The fairness criterion is based on the f ‴-discontinuity.

$$E(\mathbf{f}) = \sum_{i=1}^{M} w_i \left| \mathbf{f}(t_i) - \mathbf{q}_i \right|^2 < S , \tag{2.4}$$

where M is the number of data points, and

$$\mathcal{J}(\mathbf{f}) = \sum_{j=2}^{g-1} (\mathbf{f}'''(t_j+) - \mathbf{f}'''(t_j-))^2 , \tag{2.5}$$

where g is the number of knots.

The solution of this set of equations is an optimal balance between $E(\mathbf{f})$ and $\mathcal{J}(\mathbf{f})$, resulting in $E(\mathbf{f}) = S$. Because the fairing parameter S has a clear geometrical meaning (it is the square of the mean deviation between faired and original points), the Dierckx approach is the most intuitive one.

When considering the discussed global fairing algorithms we must realise that, although they act globally, for each data point local control can be achieved through the application of the individual weight factors w_i.

2.4.4 Human intervention at fairing

The author's experience is that in the field of naval architectural design, automatic fairing algorithms are not always applicable. A ship hull may contain many shape constraints, which arise from considerations of layout, aesthetics, predefined dimensions or hydrodynamics. Examples are a maximum dimension, or the intended second order discontinuity at the transition of flat bottom and circular bilge.

These constraints may be known beforehand, but it can easily happen that they only be-

come apparent during the fairing process. Due to these effects, completely automatic fairing, i.e. as a black box process, will not always be applicable. In the majority of the cases manual intervention is needed, and in fact the weight factors w_i and the maximum error S of Equation (2.4) may serve as a handle for the ship designer.

Finally we note that, whichever fairing criterion or fairing algorithm is used, visual inspection of the curvature plot remains the best tool for a human to judge fairness, see for example Figures 2.18 and 2.19 where the curvature is plotted (in green) perpendicular to the curve.

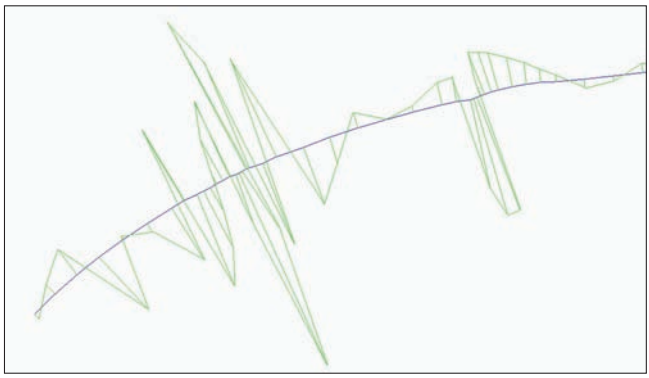

Figure 2.18 *Curvature discontinuities of a less fair curve.*

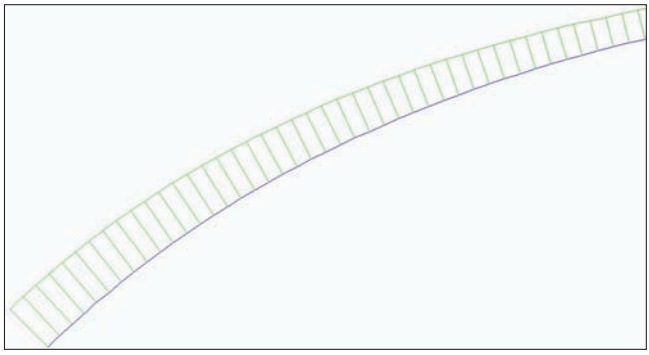

Figure 2.19 *Continuous curvature of a fair curve.*

When the signed curvature is projected in the osculating plane[2], the curvature distribution can be observed from all viewpoints. For instance in Figure 2.20 this so-called porcupine visualisation of curvature is viewed from four directions.

[2] At point **x** of a curve **c** in 3-D space, the plane spanned by **x** and the vectors **c'** and **c''** is called the osculating plane at **x**. It is the plane which coincides with the infinitesimal part of the curve around **x**.

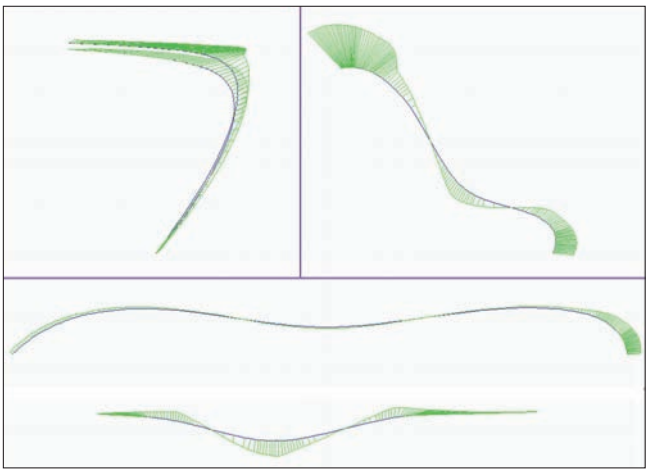

Figure 2.20 *Curvature projected in osculating plane.*

2.5 Physical models for the support of ship hull design

In many areas of industrial design, the design process is supported by the use of tangible models. Because we are also interested in using physical models to support the ship hull design process, in this sub-chapter we discuss methodological aspects and technologies of model manufacturing. In the last section we will focus on the role of model making in ship design.

2.5.1 Physical (materialized) modelling methodology

Numerical controlled (NC) milling was invented 45 years ago, and since this time it has been used for the manufacture of physical scale models or actual size models. Traditionally, the milling machines were large and expensive, while the operation was quite specialistic. NC machines were typically used in the automotive and the shipbuilding industry.

Initially, all physical modelling was called Rapid Prototyping (RP), and in fact it involved NC milling of wooden, plastic or foam models. With the emergence of new (additive) prototyping technologies in the 1980's, models could be created of hard material, so the creation of functional models or tools became possible. These applications are called *Rapid Manufacturing* (RM) or *Rapid Tooling* (RT). The reader is referred to [Kasteren, 1997] for an introduction in this field.

The decreasing costs and size of prototyping machines, combined with easier operation, have moved the machine towards the designer, where it can be used to materialize a design concept rather quickly. This is called *Physical Concept Modelling* (PCM).

This process of this divergence of RP is sketched in Figure 2.21 (from [Horváth, 1999]). In the three branches, the physical model serves different purposes:

- The purpose of RM/RT is direct fabrication of objects or moulds;
- PCM is used for synthesis in the cycle of design and development;

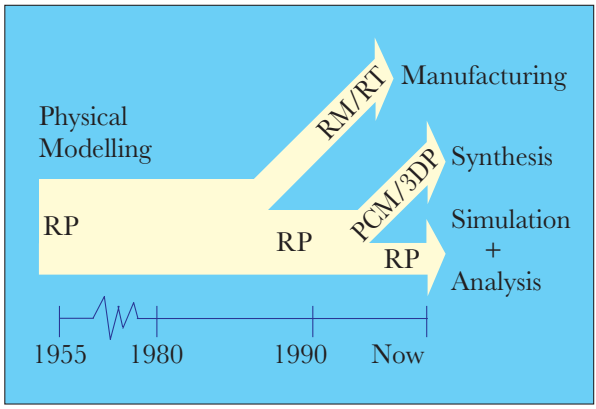

Figure 2.21 *Divergence of rapid prototyping.*

- The remaining RP branch is now used to support the final evaluation of the model (such as simulation and analysis).

As noted in [Lennings, 1998] the nature of PCM induces that a modelling device must fulfil several criteria. It must be:

- Cheap;
- Office friendly (non-toxic and low noise);
- Small enough to fit on a desktop;
- Easy to operate;
- Fast.

2.5.2 General principles of conventional rapid prototyping

In the early days of CAD the computer controlled manufacturing techniques were limited to numerical controlled milling, but in the recent years an explosion has happened in the number of techniques which, according to [Horváth, Vergeest and Juhsz, 1998], are classified in Figure 2.22.

Incremental processes form an object by adding material to the object. Zero-dimensional deposition processes work on a drop-by-drop basis, while the one-dimensional and two-dimensional processes build up a layer by adding lines or spots to the object. So these techniques belong to the class of layered manufacturing.

Decremental processes form an object by removing material. Although it is not strictly necessary, these processes can also work on a layered base.

The recently developed *hybrid process*, Thick-Layered Object Manufacturing (TLOM), is also layer-based. With TLOM the layers have curved front surfaces, so they can be of much larger thickness.

An important aspect of a particular technique is the maximum manufacturable dimension. For the commercially available machines working with incremental processes this happens to be relatively small. According to [Bailey, 1996], there exists a Laminated Object Manufacturing apparatus which is able to make a model of maximum dimensions of $800 \times 550 \times$

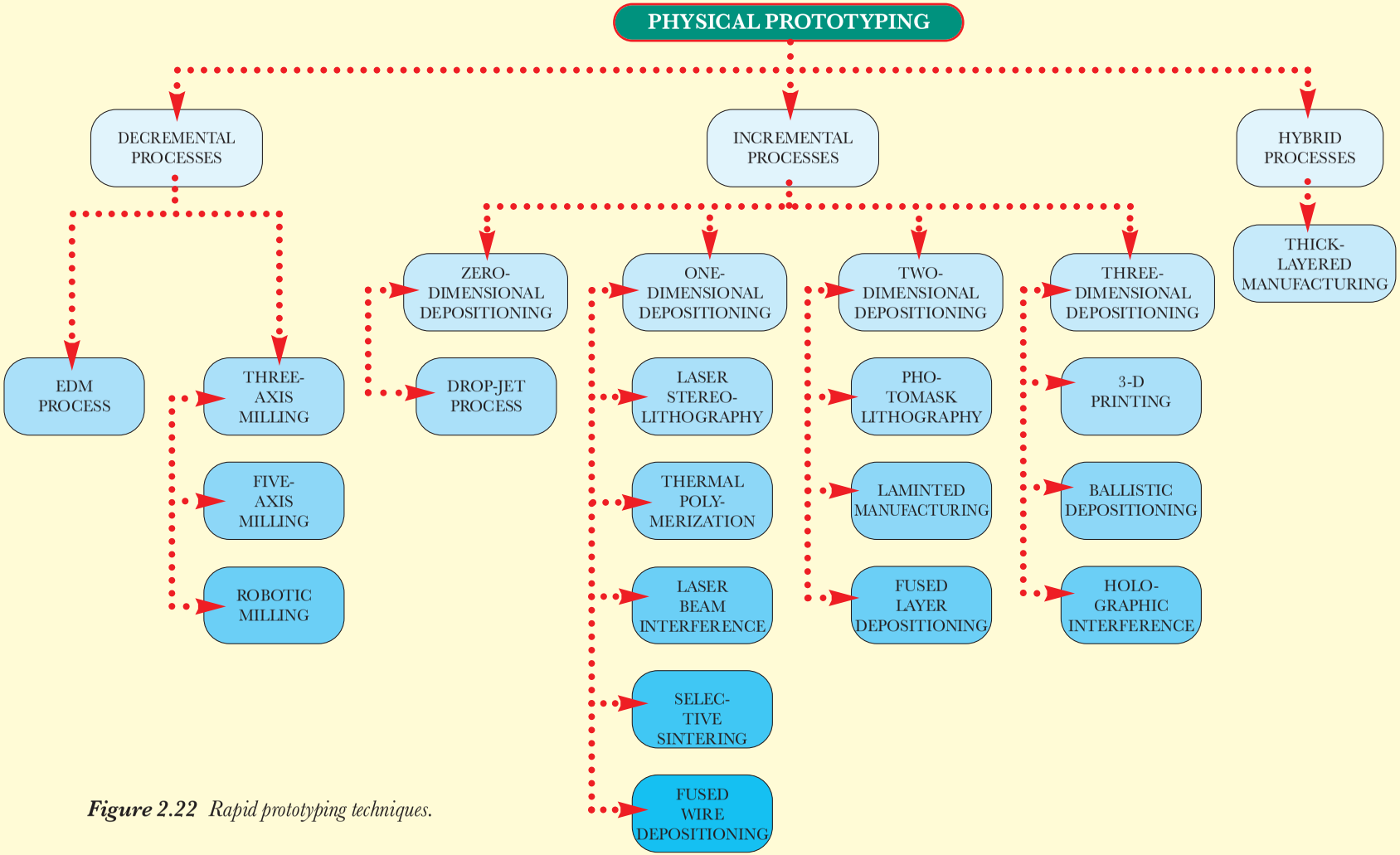

Figure 2.22 *Rapid prototyping techniques.*

500 mm, but most of the other devices are limited to model dimensions of about 500 × 500 × 500 mm.

Milling machines are available in a wide range of dimensions, from 300 × 150 × 230 mm to 22.40 × 4.50 × 4.50 m (see [Dong, 1998]).

At present several TLOM devices are subject to academic research. This technique allows large prototype dimensions. The approach of TLOM is that layers of foam are individually cut by a heated flexible blade, and then stacked together. Because the shape of the blade is continuously adapted, the approximation of the surface is of a higher order. For a detailed description of the process the reader is referred to [Broek et al, 1998].

2.5.3 Processing of CAD models for physical prototyping

The manufacturing process common to all fabrication techniques comprises three steps:

1. Decomposition of the CAD model;
2. Slicing;
3. Possibly stacking, and assembling the prototype.

Decomposition may be necessary to meet technological or dimensional constraints of the applied fabrication apparatus. Because the decomposition strategy is very much dependent on the particular fabrication technique, it will not be discussed here. The same argument goes for *stacking* and *assembly*. Given a required accuracy, the thickness of the slices is determined by the applied manufacturing technology. Figure 2.23 shows a classification of approximation methods.

The simplest is the zero order *stepped* approximation which is, for instance, typically used in three-axis milling.

First order, and more accurate, is the *ruled* or *sloped* approximation. Even more accurate can be the *circular* approximation, with constant radius, and the *higher order* approximation, where polynomials are used to approximate the shape of an object.

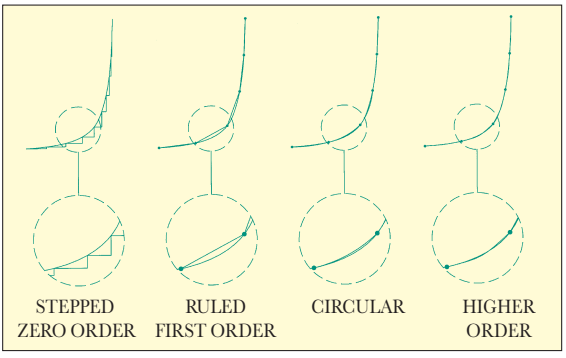

Figure 2.23 *Shape approximation for layered manufacturing.*

The determination of the actual layer thickness with a *stepped* approximation was subject of research in [Kulkarni and Dutta, 1996], [Dolenc and Mäkelä, 1994] and [Jager et al, 1996].

The minimum layer thickness for the *higher order* approximation is an iterative process, which is described into great detail in [Horváth, Vergeest, Broek and Smit, 1998]. One of the steps in the iteration process is the approximation of the geometry by the carving blade. A numerical solution for this problem has been developed in [Horváth, Vergeest and Juhász, 1998].

2.5.4 The role of physical modelling in ship design

Two types of physical modelling of Section 2.5.1 can support the ship design process in several ways.

PCM can be used for design synthesis. The ship design process, as sketched in Figure 1.1, shows an 'evaluation' and 'design choices' phase. In both phases a model (either complete or partial) can be an aid for the designer, for example, to evaluate the appearance of the shape, or as an aid in choosing layout.

Another application of PCM, for presentation to shipowners or other principals, is the creation of a small size scale model (abt. 500-1000 mm). If necessary, it can be finished and possibly decorated for display purposes.

The nature of ship design is that synthesis and evaluation involve many evaluation tools, which may involve a considerable amount of time. This implies that the synthesis of shape will not appear very frequently, so that the requirements of a PCM device, as formulated in Section 2.5.1, can be relaxed. Of course the device must be cheap and easy to operate, but in practice the aspects of size and speed are of less importance than they might be for other areas of industrial design.

The simulation and analysis aspects of *RP* are also relevant if we think of the fabrication of scale models for tests in towing tanks. An essential aspect in this respect is the size of the model, because test models need to be of adequate size. On the other hand, the speed of fabrication is less important, because only one or a small number of prototypes are actually tested.

Only the *RM/RT* issue lies out of the scope of the ship *design* subject. However, it is not unthinkable that in the future details of a ship hull on scale 1/1 will be produced on an appropriate apparatus.

3 Modelling ship hulls by computer

Where the previous chapter concerned *general* methods of computer-aided design, in this chapter we will concentrate on the *ship hull design* topic. First we will present an overview of CAD methods which are used today. These methods will further be evaluated in a normative sense in the second sub-chapter, which is followed by a practical discussion of actual ship designs.

In the fourth sub-chapter we will discuss some recent scientific papers, in order to see which direction academic research is heading, and finally conclusions will be drawn.

Before going into details, it must be clear that the aim of this chapter is not to construct a framework for a new hull design system. The purpose is to show the capabilities of the existing computer systems.

3.1 Overview of present modelling methods for ship hulls

In this sub-chapter, we investigate how the geometric models listed in Figure 2.1 are employed in existing hull form modelling systems. It is also our objective to explore which combinations of geometry representations and geometric models are used by the systems concerned.

3.1.1 Parametric methods

With parametric methods, a ship hull can be generated or modified with just a few parameters. In this respect, the word *parameter* refers to very specific naval architectural entities such as Cb, Cp, LCB or the parameters of a specific hull surface equation.

Prerequisite for this method is a fixed *procedure* for shape generation or modification, for which reason these methods are also called procedural methods.

A parametric method is neither a model, nor a representation, so it is not listed in the taxonomy of Figure 2.1. It is mentioned here because this method is often used by private, commercial or academic systems for hull form transformation, standard series or mathematical formulae for hull form generation. The procedures contain fixed actions which must always be applied to some geometric model. They are no substitute for it.

3.1.2 Simple wireframe modelling method

Several systems exist which apply simple wireframe models. For the representation of the geometry of the curves in some former systems non-polynomial representations have been applied. For instance, the FORAN[1] system used transcendental functions, and a (former)

[1] In this chapter only the brand names of commercial systems are mentioned. Appendix B lists the manufacturing companies.

Norwegian system called Autokon used circular arcs or clothoids. This category of systems has either vanished, or has been updated to the B-spline/NURBS representation. So the recent systems use polynomial or spline-based representation of curves.

Actually, the majority of systems which use simple wireframe models are not specifically used for hull form design, but more for analysis and engineering. For example, the systems Seasafe and PIAS, and the one described in [Michelsen et al, 1993], use simple wireframes to represent the hull form with an accuracy and completeness which is sufficient to make calculations of hydrostatics, (damage-) stability, tank soundings and longitudinal strength. Another example is the NUPAS system which uses this method for engineering, for construction drawings and for CAM.

To the author's knowledge, the only two *design* systems which use a simple wireframe are Mastership and Shipshape.

The reason that simple wireframe modelling is scarcely used for design is that it is geometrically non-complete. Curves may run in 3-D space, and there is no explicit information regarding relationships between curves and associated surfaces between the curves. Generation of surfaces between the curves, or generation of additional curves, cannot be applied on such an ambiguous model.

3.1.3. Extended wireframe modelling method

To the author's knowledge there are no commercially available systems that use this approach, at least not in the naval architectural field. Recent research led to a ship hull design system which uses this method. It will be discussed in Section 3.4.5.

3.1.4 Curved polygon based surface modelling method

There have been times that this approach received much attention from researchers. Consider the following papers:

- In [Earnshaw and Yuille, 1971] and [Yuille, 1979], a system of connected Coons patches is described;
- In [Munchmeyer et al, 1979] Coons patches are used, in combination with a simple wireframe;
- In [Reuding, 1989] a system for the design of yachts is presented, which works with Bézier surface patches, in combination with a simple wireframe.

Furthermore, from its documentation it is noted that the NAPA system also uses surface modelling (with Coons patches), but just like the systems discussed in the previous section, this system can be considered to be a system which is more suitable for analysis, than for design.

3.1.5 Parametric surface modelling method

In the last 15 years numerous scientific papers have been published on this technique, applied to ship hull modelling. The variety of publishing institutes indicates the wide interest in this method. They are:

- Chanic, Brussels and the Institut de recherches de la construction naval, Paris ([Stroobant and Mars, 1982]);
- U.S. Naval Academy, Annapolis ([Roger and Satterfield, 1982]);

- TU Denmark ([Fog, 1984]);
- University of Newcastle upon Tyne ([Hills and Welsh, 1986]);
- TU Berlin ([Standerski, 1989]);
- TU of Athens ([Bardis and Vafiadou, 1992]);
- Massachusetts Institute of Technology ([Abrams et al, 1995]);
- TU of Lisbon ([Ventura and Soares, 1998]);
- Indian Institute of Technology ([Anantha Subramanian and Suchithran, 1999]).

Actually the difference between the papers is surprisingly small. They all describe the advantages of the Bézier, NURBS or B-spline surface representations and they all discuss their specific implementations.

The favourable properties of the B-spline/NURBS surfaces have been discussed in Section 2.1.2 in a formal way. Formulated in a more practical way, by manipulating the control points, a great variety of sculptured objects can be produced, which can also be processed and visualized very efficiently.

When the author developed a B-spline surface based hull form generation system ([Koelman, 1985]) it was the idea that it should be just as easy to design a ship hull as to design a teapot. Indeed it proved to be rather efficient for a simple hull surface (the yacht of Figure 3.1 was produced within two hours).

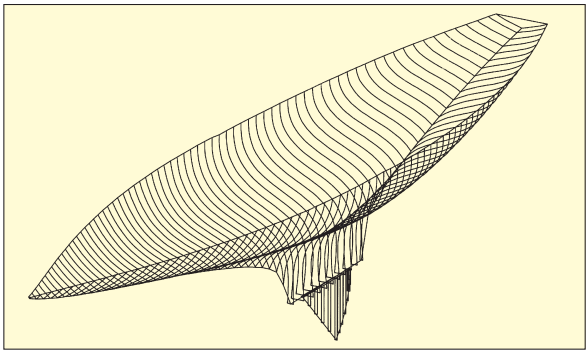

Figure 3.1 *Hull form generated with B-sline surface based software (1988).*

Due to its strength in representing and visualizing a closed hull surface, nearly all existing commercial hull form design systems have adopted the B-spline or NURBS surface representation. These systems[2] are:

- Autoship;
- Defcar;
- Fastship;
- FORAN ([Wake, 1997]);
- L/GRAND ([Laansma, 1993]);
- Macsurf / Maxsurf;
- Multisurf ;
- PIAS/Hull form generation ([PIAS, 1985]).

Because all these systems adopt the same method, the functional differences between these systems are relatively small. One difference concerns the number of surfaces used. Some systems use a single surface, while other systems use multiple surfaces, which can be connected. Besides, B-spline or NURBS representations are also used in software which is intended for engineering, such as the systems TRIBON and Nauship.

[2] This list cannot be exhaustive, but according to the author it contains all widespread systems.

These contemporary specific-purpose commercial systems cannot be called complete, because they lack an explicit representation of the topology (they are surface models).

3.1.6 Special surface modelling method

A method which is relevant for ship hull modelling is cross-sectional design. With this method a surface is generated through an ordered set of space curves (called *skinning*), or through space curves which are distributed along another space curve (called *sweeping*). The general application of these methods is discussed in [Woodward, 1987], [Woodward, 1988], [Woodward, 1990] and [Piegl and Tiller, 1996]. In [Kouh and Chau, 1990] this method is applied to generate a ship hull surface, and in [Harries, 1998] the method is further extended to *fair* skinning of sections and contour curves (and applied within a system for parametric hull form design).

In view of hull form modelling, a favourable aspect of skinning is the possibility to generate a surface, on the basis of design curves. However, a limitation is that these curves must be ordered in a regular way.

3.1.7 Complete geometric modelling

To the author's knowledge, no computer system, commercially available for naval architectural use, is based on a complete geometric model (as defined in Section 2.1.1).

Also, academic research in this field is scarce, a topic which will be addressed in Sub-Chapter 3.4.

	GEOMETRIC MODEL					
	INCOMPLETE MODEL					COMPLETE MODEL
GEOMETRY REPRESENTATION		SIMPLE WIREFRAME MODEL	EXTENDED WIREFRAME MODEL	POLYGONAL SURFACE MODEL	PARAMETRIC SURFACE MODEL	
	NON-POLYNOMIAL	OBSOLETE				
	SINGLE POLYNOMIAL	FOR ANALYSIS ONLY				
	B-SPLINE	FOR ANALYSIS ONLY	TO BE DISCUSSED IN SECTION 3.4.5		DE FACTO STANDARD	
	NURBS				DE FACTO STANDARD	
	TRANSFINITE SURFACE		TO BE DISCUSSED IN SECTION 3.4.5	FOR ANALYSIS ONLY		

Figure 3.2 *Models and representations used in existing hull form systems.*

3.1.8 Summary of the application of modelling methods

Based on the overviews of the previous sections, Figure 3.2 gives a summary of models and representations, such as discussed in literature and used in contemporary hull form modelling systems.

The majority of scientific publications and contemporary computer systems are stuck with the one combination of model and representation: the parametric surface model, combined with a B-spline or NURBS surface representation. This is the de facto standard, the leading paradigm.

3.2 Fundamental investigation of problems with the parametric surface modelling method

It was proposed in Sub-Chapter 1.1 that in some design stage virtually everyone will use the arbitrary free form design method. We also have seen in the previous sub-chapter that the vast majority of today's hull form applications work with a parametric surface model, and uses B-splines or NURBS for geometry representation.

For these reasons, in this sub-chapter we will discuss the merits of the parametric surface modelling method applied to free form ship hull design.

We have already demonstrated the fitness of the B-spline or NURBS surface to represent a simple ship hull, but it appeared that with this representation a more complex hull form is much harder to create. It is far easier to create a teapot than an average ship hull.

The difficulties when using the B-spline/NURBS surface method to create good, complete and detailed hull forms in a straightforward way stem from three sources:

- Discontinuous nature of the ship hull;
- Rigidity of the network;
- No interpolation possibility.

These three aspects will be the subjects of the next sections.

3.2.1 Discontinuity aspects of a ship hull

B-splines/NURBS have been developed for a GC^X continuous surface, and in practice X is limited to 2, so these surfaces possess continuity of tangency and curvature. When we look at a typical hull form, however, we see that there are many discontinuities, either first order or second order. For example, over knuckles (1^{st} order), at the extremes of the mid-ship section bilge or at a waterline rounding where the curvature is discontinuous (2^{nd} order).

There are even regions where the curvature is not strictly discontinuous but where, due to the sudden change in scale of curvature, there is a kind of pseudo-discontinuity (for example at a bulbous bow). In [Horváth and Vergeest, 1998] these are called 'second order singularities'.

As an illustration, in Figure 3.3 two waterlines in the stem region curvatures are plotted in green. We see that the upper waterline shows a very slight change in curvature, while

the waterline over the bulb shows a pseudo-discontinuity, because the curvature increases steeply over a small interval.

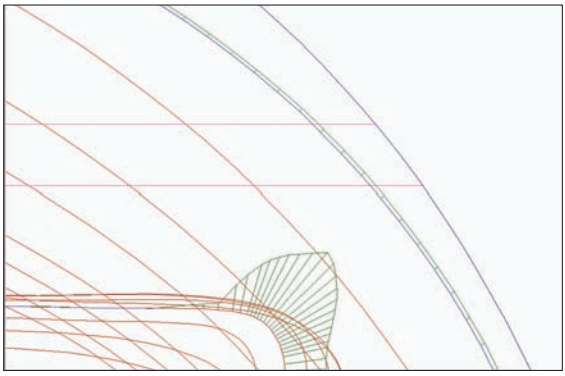

Figure 3.3 *Pseudo-discontinuities.*

This characteristic of the hull form makes it hard to fit it into one single B-spline/NURBS surface and, indeed, this is the reason that an average teapot can be more easily designed with a B-spline/NURBS surface than the majority of hull forms. The teapot contains no (or fewer) discontinuities.

Even the exact modelling of a simple midship section consisting of 'Flat of bottom', bilge and 'flat of side' (FOB - bilge - FOS, see Figure 3.4) is not possible with a single B-spline surface, because it tends to smooth away the discontinuities at both sides of the bilge, regardless of the order and the number of knots of the spline. An exactly cylindrical bilge part cannot be produced with a B-spline (in [Harries and Abt, 1998] the accuracy of an approximation is discussed).

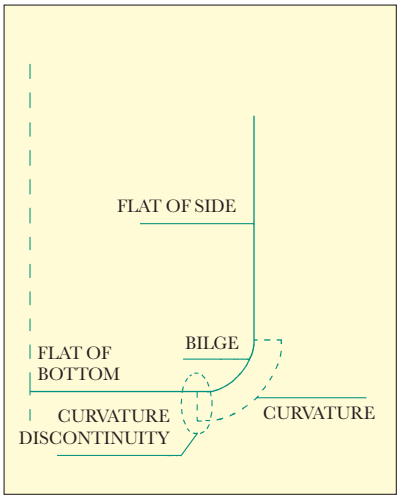

Figure 3.4 *A typical midship section.*

A NURBS can produce an exact cylindrical part (see [Piegl and Tiller, 1987]) but that involves exact (mathematically predefined) values of parametrization, weight factors and control points, and it cannot be expected that a ship hull designer manipulates these mathematical entities.

3.2.2 Rigidity of the network

The formulae for the B-spline and NURBS imply a regular network, a network which is orthogonal in parameter space. This regular network has the following disadvantages:

- The defining curves of the regular network are *in general* not parallel to the main orthogonal planes of the vessel. So the ship designer must be prepared to work with more or less arbitrary 3-D curves over the surface, and cannot work with curves of his own choice. In Figure 3.5

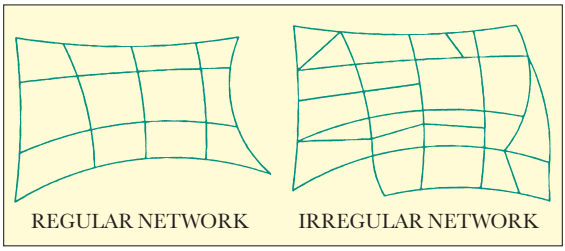

Figure 3.5 *Network regularity.*

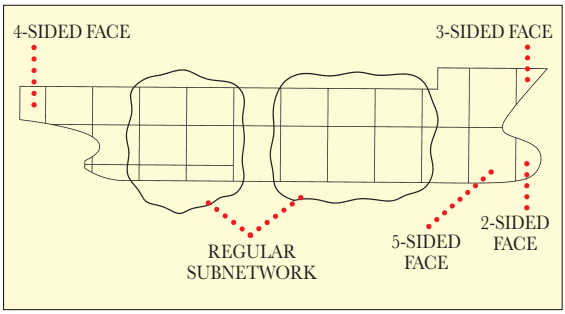

Figure 3.6 *Network of curves, including partial waterline.*

regularity and irregularity are illustrated, and the example of Figure 3.6 shows that even for simple hull layouts the network can be highly irregular;

- The regular network cannot include partial network curves. Partial network curves are desirable for, for example, partial waterlines, additional local shape details, additional local shape control and integrated stem roundings;

- Additional network curves may be necessary for a good definition of some details, but if they run over the complete surface (to maintain regularity), they may cause undulations in the areas where they are superfluous. This effect is caused by the simple fact that in those excess regions too many network curves determine the shape of the surface. It is a well-known experience that the fairest surface is obtained with a minimum number of network curves;

- Because the course of the network curves cannot be chosen by the ship designer, curves which possess important shape characteristics must be created by projection or intersection. As a matter of principle, the results of these actions are unpredictable;

- The location and nature of the network curves must be defined at the start of the design. When it appears later on that a different setup is more appropriate, a modification towards the desired setup is difficult. It is often quicker and easier to forget the work already done, and to restart the complete design.

To relieve the problems discussed in this and in the previous section, one could suggest the use of multiple B-spline/NURBS surfaces instead of a single surface, but with that 'solution' three other problems are introduced:
- A system of surfaces is constructed which, without further supporting topology, will not be geometrically complete;
- The arrangement of surfaces is the responsibility of the ship designer;
- Continuity and discontinuity conditions between the surfaces have to be specified explicitly by the ship designer.

3.2.3 Interpolation possibility

As discussed in Section 2.1.2 the 'reverse problem' (this is the interpolation issue) is difficult because of the question of parameter assignment. Techniques as described in [Birmingham and Smith, 1998] may solve this problem in the future, but at this moment there exists no practical and general solution. That implies that the designer always has to work

with control points just lying somewhere in 3-D space even if, for some points or sections, exact locations are available.

Another very common and important application of surface interpolation is simply importing digitized curves or manually typed tables of offsets of existing hull forms. With the B-spline/NURBS surface method, sometimes a satisfactory interpolation fails, because a suitable parametrization cannot be found.

3.2.4 Comparison with the 'requirements for a CAD/CAE system'

So far we have discussed aspects of existing computer methods in a descriptive sense. In this paragraph we will compare the abilities of the B-spline/NURBS surface method with the requirements as formulated in Section 1.2.2. The requirements which are *not* completely met are:

- The *draw 2-D & model 3-D* requirement, because manipulation of an arbitrary section (waterline, ordinate or buttock) is not possible;
- The *enhanced freedom* requirement. This is shown if we test the properties of the B-spline-NURBS surface method against the examples listed in Section 1.2.2:
  - Indeed the method prescribes no working sequence;
  - The method is restricted to work with surfaces (no curves);
  - It is restricted to manipulation of control points (not of points on the hull surface);
  - Hull form coefficients are allowed to be used;
  - Hull form transformation can be used;
  - In general, hull forms cannot be imported (just because they do not fit into the regular network);
  - There is no restriction on export capabilities.
- The *applicability* requirement, because of the restrictions mentioned, it will be very hard to create a very accurate, detailed and faired hull form model (in the final phase);
- Consequently, neither the related *precision* requirement;
- The *predictability* requirement (see the fourth item of Section 3.2.2).

While the following requirements *are met* with the B-spline/NURBS surface method:

- The *work 3-D* requirement;
- The *integrated data and functions for CAD and CAE* requirement;
- The requirements for *integration or data exchange with analytical software*;
- The *stability* requirements (which is a matter of implementation);
- And, finally, the *processability* requirement.

3.3 Practical experiences with the parametric surface modelling method

In this sub-chapter we will discuss some experiences gained with the application of the parametric surface models. In most cases commercial brochures show delightful colour plates of wonderful hull forms, but unfortunately the less favourable aspects are under-exposed. However, to obtain an honest view of the capabilities of the parametric surface model, some real life hull forms will be presented here, accompanied by a brief discussion of the details of the design process. The first three examples are vessels designed by design

offices or shipyards, and all three have been built this decade. These vessels have been selected because they represent a certain variety of hull forms. Finally, a few examples from literature are presented, which show some interesting details.

Before the actual discussion starts, we must emphasize that the less favourable aspects of the parametric surface model do not imply that it is impossible to design a proper hull form with this model. The whole issue of hull form modelling is more a matter of efficiency than of capabilities in a narrower sense. To exaggerate: we bet that, given sufficient time, even with the Windows 'Paint' application, a ship hull form can be designed. However, it would be neither efficient nor pleasant.

3.3.1 Schooner yacht

As presented in [Koelman, 1997a], Figure 6.1 shows a lines plan of a 25 m schooner yacht. A sailing yacht must be appealing, so the visual lines (characteristic lines on the *real* vessel) must be of perfect quality, and must match the artistic ideas of the designer. The design of this vessel was governed by four design criteria:

1. The nature of the stem, and the knuckled stern contour, in combination with the overhanging transom. Essentially, the transom consists of a heart-shaped segment of an angled cylinder. In one view the transom must be cylindrical, while in another view the shape must be completely free;
2. The line of the bulwark, with its characteristic bowsprit opening;
3. The connection line between sheer strake and shell, which is visible on the real vessel;
4. Displacement.

In the preliminary design stage the designer tried to define the vessel with one of the commercially available B-spline/NURBS surface modelling systems. During the design he experienced difficulties with the following aspects:

1. To create the combination of the overhanging stern part and the straight vertical stern. Either the network curves were chosen to run parallel (in parameter space) to the stern, or the network curves were extended in the 'negative' part of the centerline. In the first case, no smooth buttocks could be obtained. In the second case, no control over the stern was possible and the stern actually showed undulations.
   The choice of the network curves was a matter of trial and error and it took multiple restarts to find the most proper option;
2. The discontinuities of the bulwark near the stem could hardly be created;
3. The transom curve should smooth into the bulwark curve, which is most clearly shown in the top view of the lines plan. This proved to be practically impossible, because of the limited number of control points or control curves in this specific region.

3.3.2 Mooring launch

Figure 3.7 shows a lines plan of a $9.18 \times 3.30 \times 1.66$ m mooring launch, constructed in 1993. The boat is used to assist ship handling in port, and shows a traditional round bilge hull form. The hull form was designed in a hand-drawn lines plan by the yard, and the 1990-version of the PIAS hull form generation software was used to fair the body and to generate the lines plan.

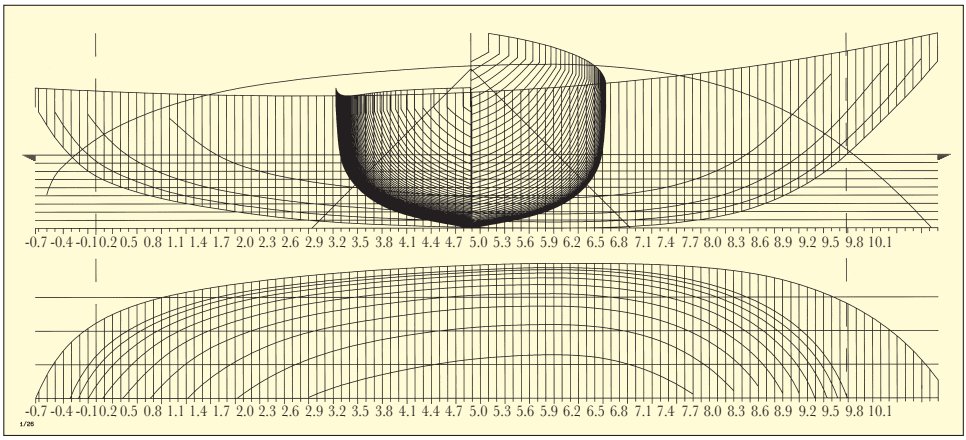

Figure 3.7 *Mooring launch (Courtesy of Engelaer Scheepsbouw, Beneden-Leeuwen).*

The yard's design office had very strict requirements on the shape of the ordinates and waterlines. Hence, to have a good control over the shape, it was decided to anchor the transverse network curves in the ordinate planes as much as possible. The consequence of this choice is that the hull surface must extend beyond centerplane. This last aspect is also favourable because an alternative would be to let the network curves run more or less 'parallel' to stem and stern, which would lead to an unnecessary and even unpleasant concentration of network curves in the tiny ordinates in the extreme fore and aft parts.

Figures 3.8 and 3.9 show all isoparametric curves (that are the curves on the surface corresponding with network curves) of aftship and foreship. The part of the hull surface extending beyond centerplane is clearly visible. This area will be trimmed in a later stage.

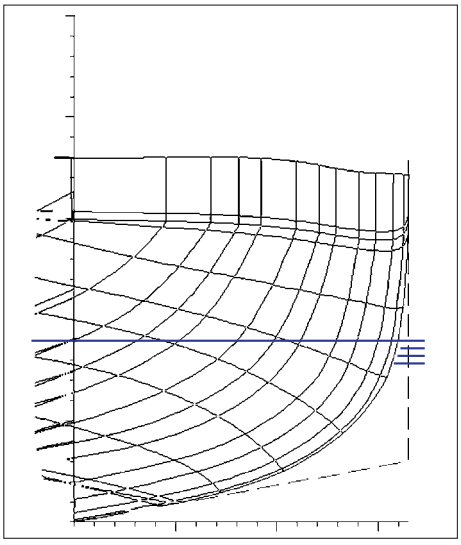

Figure 3.8 *Aft ship of mooring launch (Courtesy of Engelaer Scheepsbouw, Beneden-Leeuwen).*

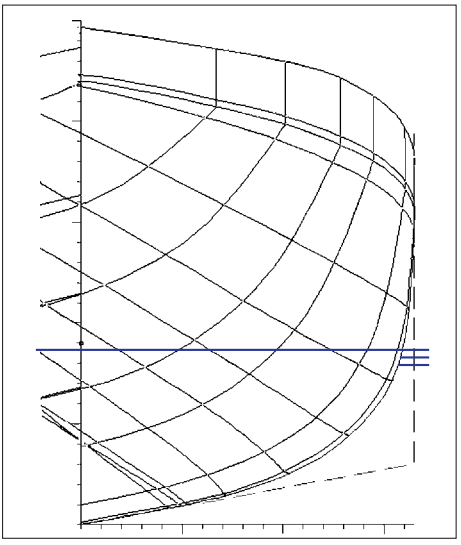

Figure 3.9 *Fore ship of mooring launch (Courtesy of Engelaer Scheepsbouw, Beneden-Leeuwen).*

Although at first sight the hull form seems simple, in the fairing phase the following aspects proved to be rather time consuming:

- Going in forward direction, the stem contour at CL should become steadily flatter, but should never be completely straight. The stem contour is not directly controllable, because it is the result of a trimming operation, and control over the contour was only possible by manipulating the network in the vicinity of the contour. Such a mechanism is of course inconvenient, while the result is unpredictable. Even the intersection between FPP and CWL does not have a one-to-one relation with a single point of the network, so each design modification which caused an alteration in this area, effectively altered waterline length;
- The yard desired the aft part of CWL to be rather full. As can be seen in Figure 3.8 no network curves coincide with, or are parallel to, CWL. So again it was rather difficult to combine the specific waterline shape with the desired shape of the stern and with the required waterline length, due to the indirect control.

The utilized software was strictly screen-oriented and did not allow for generating multiple views. With a windows-based version the described problems could, in a practical sense, have been diminished to some extent. The effect of manipulations of the network on the shape of waterlines, ordinates and contours could have been directly visible in other windows (provided that the trimming operations would take place in real time). However, even with a more powerful user-interface the indirect control of important design aspects (and the fact that an attempt to modify the waterline *shape* has an impact on the waterline *length*) is not very effective.

3.3.3 Cargo vessel

For our third example we have interviewed Mr. A.H. v.d. Horst, principal designer at Ferus Smit Shipyard. He was so kind to discuss the design process of the cargo vessel shown in Figure 3.10, which he designed in the early nineties with the same (B-spline surface based) hull form generation software as used in the previous section. The aim was to design a coaster with the following characteristics:

- Low air draft;
- L_{PP} just below 85 m;
- L_{OA} as short as possible;
- Bulbous bow not exceeding extreme length over deck;
- Propeller diameter 2.00 m;
- Depth 6.90 m and breadth 11.30 m.

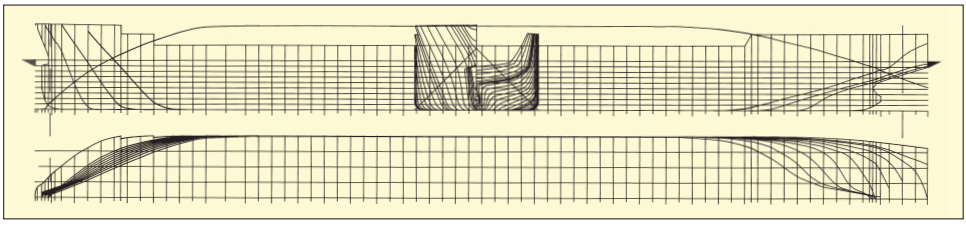

Figure 3.10 *Cargo vessel in preliminary design stage, (1990, Courtesy of Shipyard Ferus Smit, Westerbroek).*

The designer emphasizes that the hull form transformation method was not considered in this case, because according to his experiences, this method allows for small transformations only. For instance, the contour outline in the propeller area may easily be distorted.

The first design action is to define Cb, Cm and desired LCB. On basis of these parameters the SAC is generated, with the aid of a standard series. Furthermore, the deckedge and stem and stern contours are drawn.

It is the designer's experience that a minimum number of network curves should be used in order to obtain good control. However, the disadvantage of the obligatory regular network is encountered here, because the aft body, with its stern bulb, needs more shape control than the fore body. Therefore, two separate surfaces have been used. And in order not to create continuity problems, the surfaces meet at midship. For the aft body 16 longitudinal network curves have been used and for the fore body 10.

In both the fore and the aft body five ordinates are drawn at appropriate locations, so there is already a tendency towards the desired hull shape. The available SAC can be used as an aid to give the ordinates an area which leads to the desired fullness of the hull form. It appeared that the transom part needed additional control, so an extra network curve (an ordinate) was added in this part. To make network regularity bearable, the surface in the propeller boss area did continue on the 'negative' side of centerplane, it was trimmed later on.

Concerning this project, the designer draws the following conclusions:

- The fairing of the hull form is a very delicate question, because a network curve running in a longitudinal direction is not bound to any plane (contrary to the network curves in transverse direction, which were chosen to lie in ordinate planes);
- The upper part of the bulb intersects the hull (on a knuckle curve), and the lower part is faired into the hull. The surface model used does not allow for a partial knuckle curve, so the knuckles have been added manually at appropriate locations;
- The computer supported design process resulted in a hull form which was accurate and fair enough for *analysis*, but not for *production* (not even for scale model production);
- To include additional geometric details, such as the exact circularity of some frames near the stern tube or the exact straightness of a part of the stem and the stern contour, manual postprocessing was necessary. A lot of additional details of the lines plan of Figure 3.10 are produced by this postprocessing, and not from the B-spline surface representation.

The time involved was estimated at:

- Three days for the initial design resulting in an accuracy sufficient for analytical calculations;
- Two additional days for fairing, resulting in a hull form which is visually sufficiently fair;
- About 12 additional days for inclusion of all details and (manual) fairing to the desired level.

3.3.4 Examples from literature

In this sub-chapter we have so far described some experiences in the *process* of hull design, which illustrate difficulties which can be encountered when using the parametric surface modelling method. In this section we will look at some *design results* in literature which also indicate some problems.

The first example concerns a hull form which was designed with the system for preliminary design, as described in [Hills and Welsh, 1986]. Of course the paper is already more than a decade old, so the software used will not be representative for modern software, but it does clearly show the mechanisms of the utilized B-spline surface approach, which are universal.

In Figure 3.11 the body plan is shown, which is quite agreeable for preliminary design. However, less agreeable is the fact that the accompanying three-dimensional view of Figure 3.12 does show that the complete bow and stern regions have been omitted. We assume this simplification was introduced because the regular network of the B-spline surface does not allow direct control over the shape of the contour, as well as over the shape of the ordinates in the contour regions.

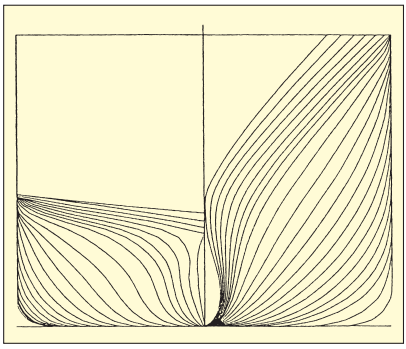

Figure 3.11 *Body plan (reprint from [Hills and Wells, 1986]).*

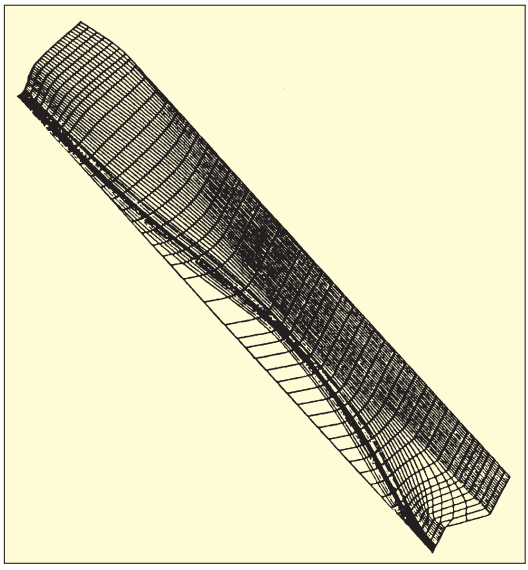

Figure 3.12 *Three-dimensional view of the vessel of figure 3.11 (reprint from [Hills and Wells, 1986]).*

Figure 3.13 shows a reprint of a brochure of a parametric surface modelling software package. Note that all vessels in this figure do have longitudinals (waterlines or buttocks), except for the SWATH vessel, where only ordinates are drawn. Apparently, for the vessels with the longitudinals a surface model was used, but the SWATH was only defined by editing or digitizing simple curves: the SWATH simply did not fit into the regular network! For it can easily be seen that a regular network can, for example, be 'folded' around the surface of the catamaran. However, at the fore part of the SWATH the gap between the struts and the submerged hull makes any network, which also must include ordinates, irregular.

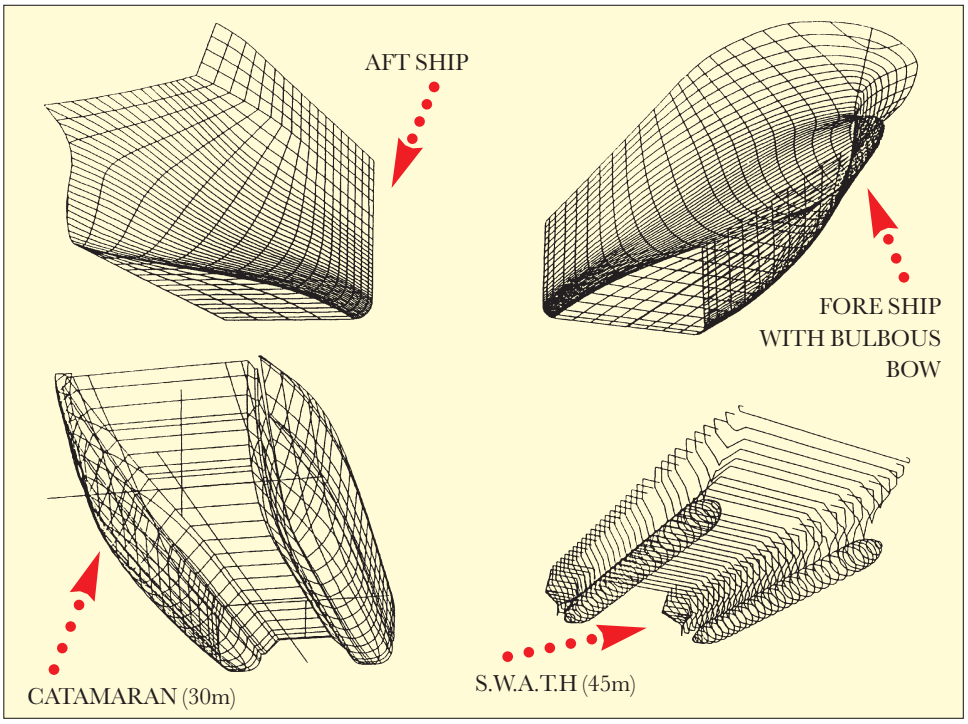

Figure 3.13 *Examples in a commercial brochure (reprint from [Coastdesign, 1992]).*

Also remarkable in this picture is the stern shape of the top left vessel. It shows a smooth stern contour without arrangements for the stern tube, exactly straight parts or other practical details. Therefore this stern shape is not according to standards for final design.

3.4 Survey of relevant recent research, applicable to ship hull design

We have seen in the previous sub-chapter that the majority of attention is directed to the method of a (non-complete) parametric surface model, with B-splines or NURBS for surface geometry representation. To investigate whether alternative research is dedicated to improvements within this paradigm, or towards a paradigm shift, in this sub-chapter we will discuss seven papers. The criteria for selecting these specific publications are:

- Papers must be quite recent, and reflect the state-of-the-art;
- The subject must be the design of a ship hull, or the discussed method must be very appropriate for ship hull modelling;
- In order to give a broad overview, there must be some variety between the subjects or solutions of the papers.

In the first publication the subject of surface fairing is addressed, and research to an accurate surface representation is discussed. The second paper is also involved with the subject of surface fairing, as well as fair mesh interpolation.

The third and fourth papers stick to the B-spline/NURBS surface paradigm, and try to explore an alternative way of shape generation. The last three publications show a gradual growth towards geometric completeness.

Because in this stage we are more interested in the direction of research than in the exact nature of the solutions, the discussion will be of a general nature, without any comments. In the last section we will take the liberty to express our opinion on the papers presented.

3.4.1 Accuracy of surface representation

In [Tuohy et al, 1996] it is investigated which representation approximates a test surface with the highest accuracy. The test surface is a double sinusoid, which was approximated by a Coons patch, a Gordon surface and two specific transfinite surfaces (which are variants of the Gordon surface), called UNO-1 and UNO-2 (where UNO abbreviates University of New Orleans). It was concluded that the Gordon surface and UNO-2 give the highest accuracy.

3.4.2 Surface patches and surface fairing

The subject of surface fairing is addressed in [Nowacki et al, 1997]. Given a regular or an irregular network of curves, their surface fairing procedure consists of three steps:

- Net fairing, based on the minimization of the net energy, by the method of Hosaka (as discussed in [Su Bu-qing and Liu Ding-Yuan, 1989]);
- Construction of fair GC^1 or GC^2 patches between the network curves;
- Automatic fairing of the complete hull surface (for a regular mesh only).

3.4.3 Constrained shape reconstruction

Given a set of points on sections of a ship hull, with the least squares method a B-spline surface can be reconstructed over the hull surface.

As discussed in Section 2.1.2 a problem to overcome is choosing appropriate parameters. Another difficulty, which is rather ship-specific, might be that second order discontinuities may cause undesired surface characteristics, such as superfluous points of inflection, or the extension of part of the surface below baseline or beyond half breadth. In [Rogers and Fog, 1989] this problem was recognized and attacked with iterative parameter assignment and surface fitting constrained to specific allowed locations.

3.4.4 Automatic hull form generation, based on a genetic algorithm

In [Birmingham and Smith, 1998] a method for the initial generation of the ship's lines is presented. The researched idea is to keep parametric surface modelling, to maintain the B-spline surface as geometric representation, and to use advanced search techniques to generate the desired hull form. The input for the method is described as 'design intent' and may consist of numerical data and of shape data. In an example (reprinted in Figure 3.14) to illustrate the use of numerical data, hull form coefficients are utilized. In a second example, to illustrate the use of shape data, a hull form is generated on the basis of a table of offsets.

A ship hull, represented by a number of B-spline surfaces, is modified iteratively, and for each iteration a fitness function is determined on the basis of:

- How near the desired numerical data are approximated;
- How near the target curves or the offsets are approximated;
- A degree of fairness.

A genetic algorithm optimization technique is used to find the surface with the highest fitness. That surface fits all input best.

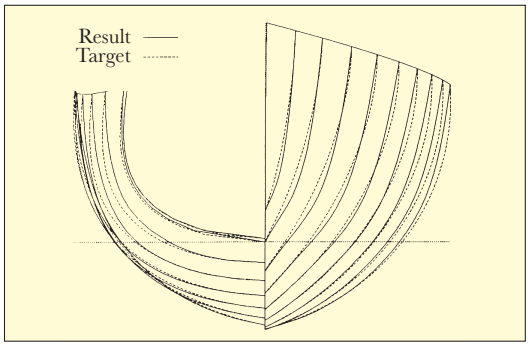

Figure 3.14 *Hull form generated on basis of 6 parameters (reprint from [Birmingham and Smith, 1998]).*

[Peacock et al, 1997] use alternative optimization techniques to reach a similar goal. However, their geometric model is wireframe, instead of parametric surface.

3.4.5 Extended Wireframe Modelling for ships

The first publication where the disadvantages of the use of a parametric surface model for ship hull design are discussed is [Michelsen, 1994]. In this Ph.D. thesis for the first time concepts of topology and transfinite surface interpolations of N-sided patches are applied to ship hull modelling. Michelsen's method has the following characteristics:

- Uses an extended wireframe as geometric model;
- Topology and geometry are initially separated. In a preprocessing step, topology is automatically derived from geometry, but this process may fail, in which case manual support is necessary and possible;
- Uses for curve representation GC^2 continuous cubic Hermite spline curves;
- Uses the concept of polycurves, which are ordered sequences of curves;
- Employs transfinite surfaces (Coons and N-sided patches) for surface representation.

3.4.6 Complete Modelling with surface patches

In [Jensen et al, 1991] a geometrically complete modelling alternative for the B-spline/NURBS surface representation is presented. The system is based on boundary modelling, and on transfinite geometry representation of surfaces by compatibility corrected Gordon patches. The paper also describes a method to derive cross boundary derivatives from a network of curves.

3.4.7 Complete Modelling with sketched design curves and surface patches

In [van Dijk, 1994] the ideas of [Jensen et al, 1991] are further elaborated. Van Dijk introduces an alternative data structure to accommodate sketched curves, and proposes a patch (the van Dijk patch) as an alternative for the Coons patch. This system was intended for fast design of industrial objects.

3.4.8 Comment on the surveyed research

In the paper of Section 3.4.1 the basic research question, to find the best approximation of a given surface, is not very relevant for hull *representation*, because a representation with a lower accuracy can also be appropriate, albeit in a more refined form. For interactive hull *design* the accuracy issue is irrelevant. More interesting is the issue of flexibility and capabilities of surface representations, from the viewpoint of practical ship design.

The second paper addresses the surface fairing subject in an advanced manner and the fact that the curve network is not supposed to be regular is certainly an advantage. Unfortunately, the geometric model used is not complete, because explicit handling of topology is missing, but we assume that this method could be implemented upon some complete geometric model. However, the method can be critized for its choice to fair the surface automatically. As outlined in Section 2.4.4, in many cases automatic fairing of ship hulls will not be applicable.

The constrained shape reconstruction of the third paper is more attractive than an ordinary interpolation, but a problem lies in the heuristic character of the constraints. For it is possible to formulate constraints at some well defined locations, but in large areas of the ship the constraints are not constant, but are themselves dependent on the location. For example, the vertical side at midship may be constrained to be exactly the half breadth of the vessel, in order to prevent oscillation in that region. However, a similar constraint for the narrowing fore and aftship cannot be formulated, so there is no mechanism to prevent oscillation in those regions. Besides, only positional constraints are included, but e.g. tangential constraints or volume properties may be just as important. By the way, this last issue is addressed in [Nowacki and Xinmin Lü, 1994], for Hermite polynomial curves.

The presented results of the automatic hull form generation of the fourth paper are impressive, but we still wonder what the results of this method will be when applied to more complex hull forms, such as hulls with discontinuities, bulbous bow, stern tube or other constrained geometric details. For it was around 1989 that the author has also experimented with a comparable method which was clearly less advanced, because there was no fairness criterion, and no genetic algorithm was used but only traditional optimization techniques. These experiments produced results for a simple surface (see Figure 3.15 for an example, where the surface must match given points in space). However, for higher accuracies, or a more complex hull form, satisfactory results could not be obtained.

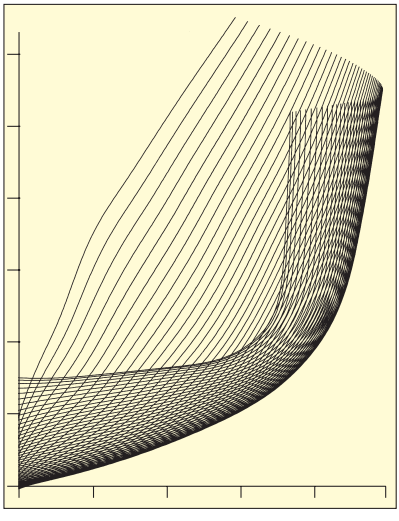

Figure 3.15 *Experimentally generated B-spline surface, with automatic parameter assignment (1989).*

Besides, both for this paper and for the previous one, a parametric surface model is used, which has a number of disadvantages which have already been discussed in the previous sub-chapters.

The thesis discussed in Section 3.4.5 does incorporate topology, and has actually been applied with success on a ship hull surface. The author of that thesis also indicates some aspects of the method which can be improved:

- Non-complete modelling is used, therefore there is no guaranteed consistent model of the rigid solid;
- Topological relations are to be maintained manually, or semiautomatically;
- No fairing possibilities;
- Only cubic splines are used for geometry representation of curves, there is no manipulation of control points possible.

The last two discussed papers use a complete geometric model, not specifically applied to a ship hull, which could serve very well for hull modelling purposes. The methods used in these papers deserve to be applied in shipbuilding (and the other way round).

3.5 Conclusion on the applicability of CAD modelling methods for ship design

In this chapter contemporary CAD methods applicable to ship hull representation have been discussed. Because the vast majority of academic or commercial systems for ship hull form design work with parametric surface modelling, this method received the greatest attention of this chapter. It was shown in Section 3.2.4 that this method does not meet our requirements for a good CAD/CAE system, and the problems have been illustrated in Sub-Chapter 3.3 with practical examples of vessels designed with a B-spline or NURBS surface representation.

It can be expected that geometrically complete models or extended wireframe models will conform much better to our requirements, and indeed the survey of research shows interest in this direction.

So we conclude that, in order to meet our requirements, a new system has to be developed, which cannot be based on the CAD methods typically used nowadays in naval architecture.

The papers cited in Sections 3.4.6 and 3.4.7 present contours of a method which can be quite applicable in a new system for ship hull design. The development of that system is the subject of the next chapter.

4 Development of a shape design system for ship hulls

Although this is one of the shorter chapters of this thesis it might be considered as the pivot, because the analysis of design methodology and CAD methods, as presented in the previous chapters, is used to develop the concept of a new shape design system. Subsequently that concept is used in the development of a data model and a functional specification.

4.1 Conceptualization of the system

In this sub-chapter we will indicate which elements are necessary for a new modelling method. We start, however, with two definitions:

- A **curve** is a point-bounded collection of points in 3-D Euclidian space, represented by a G^N continuous (which means with geometrically continuous derivatives up to degree N) function;
- A **polycurve** is a sequence of curves. So a curve is a segment of a polycurve.

To indicate the necessary elements we will traverse the list of requirements presented in Section 1.2.2:

The *draw 2-D & model 3-D* requirement implies support to work with 3-D curves, and a mechanism to embed two-dimensional curves into a three-dimensional framework of curves and surfaces. In other words, topology must be included and, according to the parlance of Section 2.1.1, we need a complete geometric model to fulfil this requirement.

The *work 3-D* requirement implies support for 3-D curves, 3-D surfaces and special surfaces, such as developable surfaces or quadric surfaces, including simple cylindrical surfaces.

The *enhanced freedom* requirement calls for support of a polycurve, control point manipulation, generating intersection curves and a variety of capabilities for import and export. The polycurve is included because a naval architect is used to work with continuous 'lines' that run over the complete surface (such as a waterline or an ordinate), which can be split into separate parts.

The *applicability* requirement implies that tools for all design phases have to be included. For preliminary design, amongst others, we need a hull form transformation function, support for the SAC and tools to manipulate the SAC. For the preliminary and the final design phase, global hull form manipulation tools are required, as well as the capability to generate sections from the hull form model. For the detailed design phase we need fairing

tools and the capability of curve projection, to project butts and seams onto the hull surface. The latter in order to be able to generate shell plate developments in the detailed design stage.
The *precision* requirement asks for fairing methods with variable levels of accuracy, and with the possibility to perform global as well as local fairing. Connected with the fairing issue is a standard library of simple geometrical entities and the ability to specify prescribed tangents or curvature at the ends of each curve (the boundary conditions). Support for the handling of discontinuities or singularities is also implied by this requirement.

The *stability* and *predictability* requirements imply the use of a complete geometric model, in order to ensure reliability and to avoid ambiguity.

The *integration* requirement implies several import and export functions.

And finally, in order to meet the *processability* requirement, functionality in the field of generation of lines plans and rapid prototypes must be available, as well as a substantial export capability.

Figure 4.1 shows the design for our new system. Around the complete geometric model, a kernel containing methods for the representation of topology and geometry, many functions are arranged. The primary functions are essential for the system. The secondary functions are necessary to fulfil the requirements of Section 1.2.2, but they are not crucial. The system could work without the secondary functions, albeit in a less advanced way.

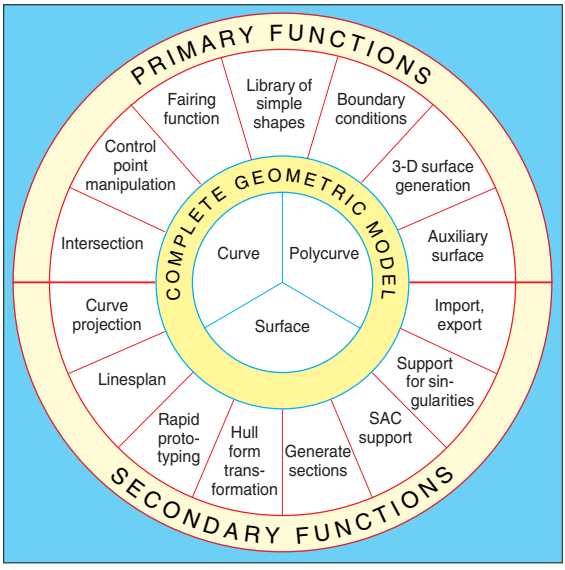

Figure 4.1 *Conceptual system design.*

The included primary functions perform the following tasks:

- With the *intersection* function the hull can be intersected with a plane, and the intersection curve(s) are generated, either based on the crossed network lines, or based on the hull surface;
- With *control point manipulation*, control over the shape of a curve can be obtained;
- The *fairing* function is used for fairing and smoothing;
- The *library of simple shapes* contains elementary curves, such as parabolas or arcs, to be used for roundings;
- With the *boundary condition* function the constraints of tangency or curvature can be imposed;
- With the *3-D surface generation* a surface is created;
- The *special surface* function is used to create developable or elementary surfaces.

The secondary functions are:
- *Curve projection*, to project an arbitrary curve onto the hull surface;
- *Lines plan*, to compose and plot a conventional body plan, for traditional visualisation of the hull;
- *Rapid prototyping*, in order to manufacture a physical scale model to serve two purposes:
  - Unambiguous visualisation of the hull, also suitable for people not acquainted with the traditional drawing in the body plan;
  - Tests in model basins.
- *Hull form transformation*, for modification of a particular design;
- *Generate sections*, to generate cross sections for construction frames (this function is similar to the intersection function);
- *SAC support*; an aid to create a hull with the desired hull form coefficients;
- Support for *singularities* in or between curves or surfaces;
- *Import* and *export* functions, for data exchange with other software.

4.2 Hybrid data model and functional specification for ship hull modelling

4.2.1 Concept of the data model

Based on the conceptual system design of the previous pages, now is the time to fill in the scheme of Figure 4.1 with appropriate CAD methods discussed in Chapter 2. Our new system will be based on multiple data types or methods, therefore we baptize it **H**ybrid model for ship hull **REP**resentation and abbreviate it as **HREP**.

In the previous sub-chapter we concluded that a *complete geometric model* is imperative, and, according to the classification of Figure 2.2, such a model can be manifold or non-manifold. The choice between manifold and non-manifold modelling is led by the following considerations:
- Non-manifold modelling requires a much more extensive and complicated system design;
- Non-manifold modelling increases complexity for the system's user, the ship designer, because all the connectivity possibilities of the non-manifold model must be presented and discussed;
- Our modelling object, a ship moulded hull can be mapped to two-manifolds;
- Non-manifold modelling offers fascinating additional possibilities, such as modelling of the internal subdivision of the hull (for tanks and compartments), and construction details, such as a steel plate connected to the hull surface on an edge only.

We decided that for our current purposes manifold modelling is sufficient and that the additional possibilities of non-manifold modelling are not needed. Because non-manifold modelling increases complexity, we will not use it in HREP.

We already concluded in Sub-Chapter 2.2 that the most suitable complete geometric model to represent a ship hull is the **B**oundary **REP**resentation. However, with the polyhedral halfedge-based BREP described in Section 2.2.2 two important elements are

missing, so we extend the BREP with:
- Continuous curves and polycurves, because the edges of an ordinary BREP are only ordered around the faces, not over the complete hull surface;
- Curved geometry, because the edges of the BREP are straight, and the surfaces are polyhedral, while for a ship hull curved surfaces and curves are required.

To enable *continuous curves and polycurves*, we add additional data structures to the BREP, representing the sequence of edges which form the topology of the curve and the polycurve. Euler operators are topological operators only, and process no shape information, so they also have to be extended to handle the additional shape data.

It must be emphasized that the *curve* is not the geometrical counterpart of the topological *edge*, the curve can extend over multiple edges.

For roundings and fillets an extra information block is attached to the curve record, containing the type of curve. The type of a *simple curve* can be generally curved, circular, ellipsoid and hyperbolic.

The *boundary conditions* (in the sense of prescribed tangents or curvatures) at the ends of a curve are also included in the curve information. It also includes the hierarchical relationships which exist between the end conditions of two connected curves. We will call these 'master-slave relations'.

The *singularities* are taken into account by storing information on the kind of continuity or discontinuity occurring when a curve meets another curve. For instance, at a knuckle curve it is stored that intersecting curves have tangency discontinuity at the intersection point.

In HREP a *surface* can be used as the geometrical counterpart of the topological *face*, so that one face corresponds with one surface (a single patch), but it can also correspond to multiple faces, ordered in an orthogonal network (a patch complex). To offer the capabilities for developable surfaces (and possible other special surfaces) in the data model space is also reserved for *special surfaces*.

Figure 4.2 gives an overview of all data elements.

4.2.2 Geometry representation

The *geometry of curves* is represented by 3-D NURBS curves. These offer good free form manipulation characteristics, they are able to represent simple curves (according to [Piegl and Tiller, 1987]) and by choosing this de facto standard we also obtain compatibility with other CAD software.

For *geometry representation of surfaces* of the single patch and the patch complex, we can follow two approaches:
- For single patches and patch complexes use a NURBS surface (see Section 2.1.2), and for N-sided patches a hierarchical decomposition (see Paragraph 2.3.3.3) into 4-sided single patches. Those decomposed 4-sided patches themselves can also be represented by a NURBS surface, so this approach has the advantage of a single representation;
- For single patches and patch complexes use transfinite surface interpolation (as

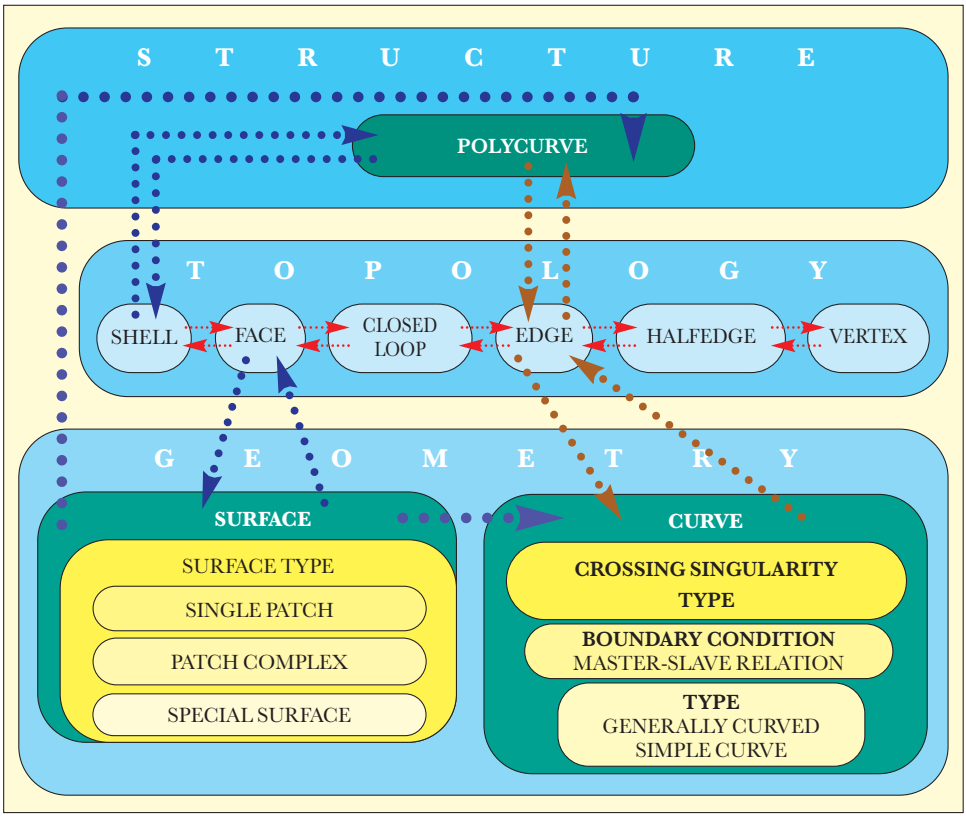

Figure 4.2 *Schematic data model of HREP.*

described in Sections 2.3.1 and 2.3.2) and for N-sided patches one of the methods of Section 2.3.3.

First we judge the fitness of the alternative methods for N-sided patch representation:

- *Refinement by subdivision* is not applicable to our system, because it works on the basis of a polyhedral net of control points. HREP does not have surface control points, it works on the basis of interpolation of a network of curves;
- *Hierarchical decomposition* offers the advantage that 4-sided patches are created, for which we can use the (still to be determined) representation we need anyhow. This would also be an advantage for conversion to general purpose CAD software, IGES for example does support a 4-sided patch, but no N-sided patches (see [IGES, 1993]).

However, these methods need to determine the location of and the tangent plane at the central vertex. This can be left to the user, but that is contrary to our desire to shield the user as much as possible from mathematical details and involvement. Location and tangent plane at the central vertex could also be obtained by an estimation method, based upon some other N-sided patch method, but in that case we could just as easily adopt that other method;

- For the *boolean sum (or convex combination)* method according to [Hahn, 1989a] the

drawback is that 'most interrogation software and NC machine tooling is immediately applicable only to rectangular patches'. However, for our goal this argument does not hold because we wish to develop all software ourselves.

We conclude that the *boolean sum* representation of N-sided patches suits our requirements best.

For 4-sided patches and patch complexes, we can choose between transfinite and composite polynomial (B-spline or NURBS) representation. The merits of these alternative representations are:

- The NURBS surface is the de facto standard in the industry, so its use accommodates easy data exchange with other software. This advantage disappears to some extent, because for N-sided patches we have just selected a non-NURBS representation;
- In HREP the surfaces are not explicitly shaped by the user. They must derive their shape from the curves in the vicinity, a task where transfinite surfaces are designed for;
- Although we use NURBS representation for *curves*, we must keep in mind that a single curve may extend over multiple edges. So curve parameters along the opposite boundaries of a single patch may not be compatible. In [Fenqiang Lin and Hewitt, 1994] a method is presented to approximate a transfinite patch by a NURBS surface, which involves the steps of:
  - Converting all cross sectional curves to B-spline representations, so that one single B-spline is obtained for each curve. In general this step involves curve interpolation;
  - Making all curves compatible;
  - Similar to the Coons-Gordon approach, constructing a B-spline surface by the boolean sum of two ruled surfaces and one tensor product interpolant.

  Because in this process the original curves are approximated by B-spline curves, the transfiniteness character is removed. Besides, for an arbitrary representation of the boundary curves, possibly a high number of control points must be used.

  We conclude that the single fact that NURBS are used for *curve* representation, does not imply usage for *surfaces* too, because additional conversion and processing is necessary;
- Even if NURBS curves are used, each of those curves may have its own parametrization, number and distribution of vertices, which may lead to very local shape characteristics. To represent these accurately, when using a NURBS surface for a N.M network of surface patches, a N.M network of NURBS vertices will not be sufficient.

Considering all these aspects, we conclude that for our purposes the composite polynomial surface is not attractive, so for the representation of single patches and patch complexes, transfinite surface interpolation will be used.

4.2.3 Specification of functionality

Figure 4.1 also contains *functions*, which are implemented as follows:

The *curve fairing algorithm* must meet two requirements; it must enable global as well as local smoothing, and work with a smoothing criterion with a geometrical sense.

According to Sub-Chapter 2.4 there are three candidates for a fairing algorithm: knot removal / knot reinsertion; complete smoothing according to the algorithm discussed in [Pigounakis et al, 1996]; and complete smoothing according to the Dierckx algorithm. The

knot removal / knot reinsertion algorithm cannot meet both requirements. Smoothing according to Pigounakis can be local as well as global, but has no geometric smoothing criterion, it uses a dimensionless overall weight factor instead. The Dierckx algorithm meets both requirements. Its smoothing criterion expresses the total deviation (in m) between original and faired data points. So this algorithm will be implemented for fairing. *Interpolation* can be seen as a special case of fairing, with a deviation of zero.

The functions to support *special surfaces* generate developable and pseudo-surfaces. A pseudo-surface is not really a surface with a surface representation, but it is a region which is governed by a certain user-specified regime. An example is the stem rounding, where waterlines are prescribed to be parabolic.

SAC support and *hull form transformation* can be treated as additional functions, quite separate from the underlying data structure. Both functions can be implemented with standard methods. As transformation methods are selected: linear scaling, Lackenby frame shifting ([Lackenby , 1950]), lengthening or shortening parallel midbody and inflation or deflation of ordinates.

Lines plan generation and *export functions* can also be viewed as additional to the core functionality. HREP contains all data to accommodate both functions, the details are a matter of implementation.

For the *import functions* no specific added functionality is needed. The fact that one of the core elements of HREP is the 3-D curve, which can be parametrized independently from all other curves in the model, accommodates the import of 3-D curves complexes very well. To import surfaces, they can be converted to curve complexes.

For *rapid prototyping* one has to take into account that the dimensions of the model may be considerable, because:

- A ship is quite large, so even a small scale model may be up to 1 m in length;
- For hydrodynamic experiments models of more than several metres are required.

Considering the dimensions, the best choice seems to be three-axis milling and Thick Layer Object Manufacturing. The motivation is that three-axis milling offers a relatively simple, inexpensive way of manufacturing small models, for the purpose of visualizing the hull design. TLOM can be considered a very attractive modelling technique for large scale models for model basins, or for 1:1 prototyping of curved parts of the hull or the superstructure.

Still, model requirements may exceed machining capabilities (not only concerning the workspace, but also due to manufacturing technology), so for both techniques we have to develop a procedure for automatic or guided decomposition into building blocks, which can be assembled after milling or cutting.

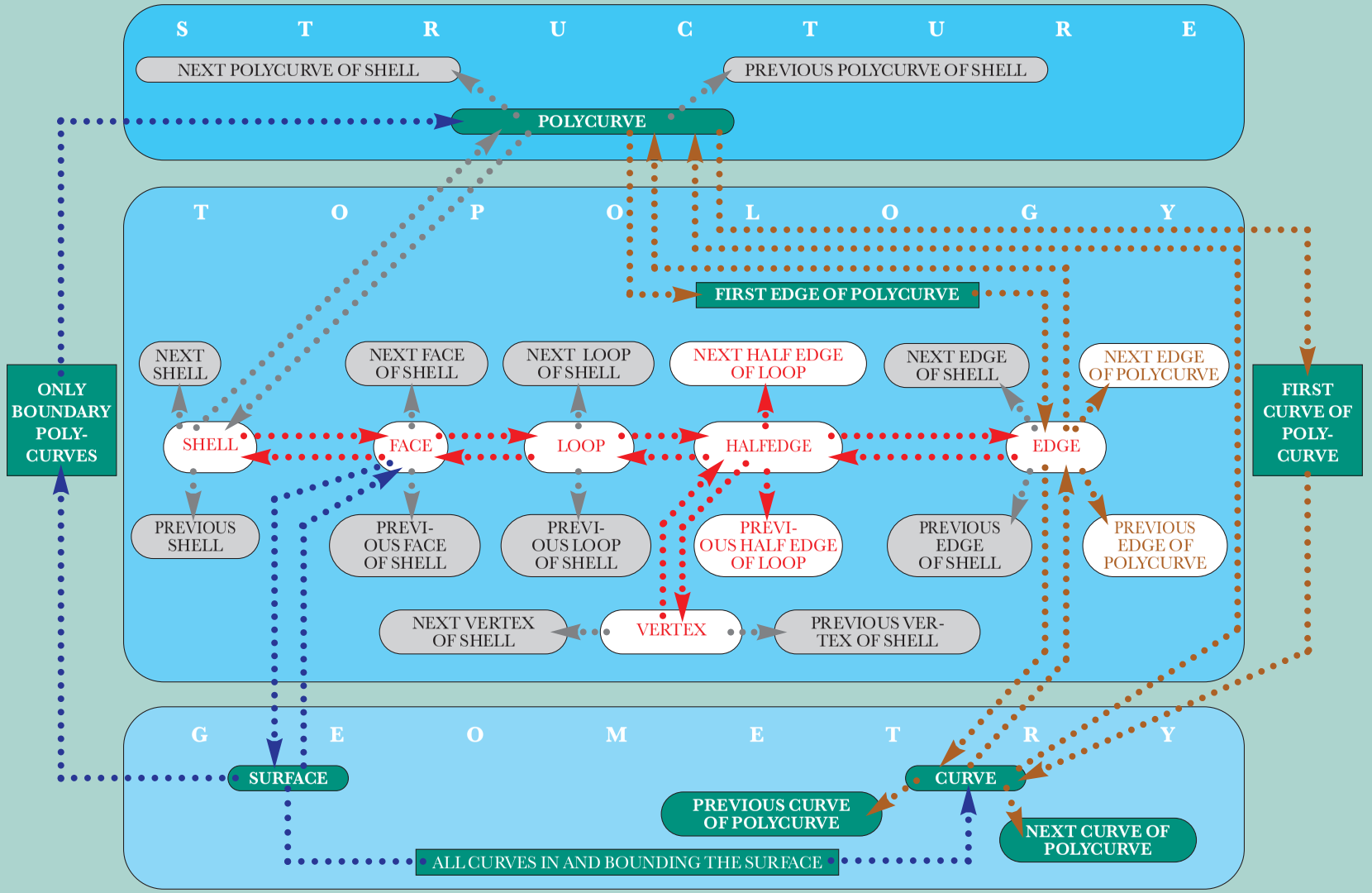

Figure 5.1 *Data model of HREP.*

5 Elaboration of the shape design system

In the previous chapter the conceptual design of data and functions of HREP have been presented. In this chapter these concepts will further be elaborated, and all implementation details will be discussed.

5.1 Data management

Figure 4.2 presented a rudimentary data model of the HREP. The complete model is shown in Figure 5.1, and is much more extended, it even contains more relations than strictly necessary. The additional relations contain linked lists of entities of the solid, and are used, among other things, for searching and batch processing. For example to search a specific vertex (after the user pointed to a screen location) or to draw all curves, it is convenient to have the possibility to traverse all members of a specific category.

In the data model of fig. 5.1 three main streams of relations can be discovered:

- Indicated in red: the traditional shell – face – loop – halfedge – vertex relation of [Mäntylä, 1988], as discussed in Section 2.2.2;
- Indicated in brown: the polycurve – edge – curve relation;
- Indicated in blue: the face – surface – curve – polycurve relation.

While the additional, non-critical, relations are marked in grey.

Both the 'shell – face – loop – halfedge – vertex' and the 'polycurve – edge – curve' relations are always maintained by means of Euler operators, or by structural operators. The 'face – surface – curve – polycurve' relations cannot be maintained permanently; the reason, and the algorithms to construct this relation on demand, will be explained in Section 5.1.4.

5.1.1 Fundamental modelling entities

In this section all data and pointers from the data elements will be discussed, where the following conventions are used:

- Each element must be viewed as a (Pascal-style) record or a (C-style) structure. For each element its data content is listed: first the data type (in bold); then the name of the identifier (in italics, with a name that describes the content shortly); and finally, if necessary, an explanation (in a normal font):
  Data type *identifier* explanation;
- When an **enumerate** data type is used, after its first use follows the list of enumerations;
- The → symbol means 'Pointer to';
- All vectors are called 4-D, because they are in homogeneous coordinates, which contain respectively: longitudinal position from APP (m); transverse position from CL (m, SB

being positive); vertical position from baseline (m); and a dimensionless weight factor. Homogeneous coordinates are used because they simplify transformations and projections;

- A curve has a left and a right side. Given the outward pointing surface normal **N**, and tangent vector **T** of the curve, left is the side pointed to by $\mathbf{N} \times \mathbf{T}$;
- In order to avoid a limitation for a NURBS curve, a linked list of control points and parameters is maintained. One element of this list is called *NURBS atom*.

SHELL	
→face	*First_face_of_shell* First face of the linked list of faces
→edge	*First_edge_of_shell* First edge of the linked list of edges
→polycurve	*First_polycurve_of_shell* First polycurve of the linked list of polycurves
→vertex	*First_vertex_of_shell* First vertex of the linked list of vertices
→shell	*Next_shell* In the current implementation only one shell is used
→shell	*Previous_shell*

FACE	
Boolean	*Face_active* Is used in several functions, or in postprocessing. See e.g. Section 5.1.4.
Boolean	*Phantom_face* In shipbuilding practice it is custom to model one side of CL only, and assume the other side is mirrored. Essentially, in HREP a closed shell is modelled, so without precautions the CL plane would be part of the hull model, which is undesirable because it hampers visibility. To avoid this, one face, called the 'phantom face', is invisible. It is part of the shell, but never displayed.
→surface	*Surface_which_includes_this_face* Surface (or a part of it) which represents the shape of this face.
→loop	*Outer_loop_of_face*
→shell	*Backpointer_to_shell*
→face	*Next_face_of_shell* Next face in the linked list of faces of this shell
→face	*Previous_face_of_shell* Previous face in the linked list of faces of this shell

LOOP	
→halfedge	*First_halfedge_of_loop*
→face	*Backpointer_to_face*
→loop	*Next_loop* Next loop of the face to which this loop belongs
→loop	*Previous_loop* Previous of the face to which this loop belongs

EDGE

→halfedge	*Halfedge_one_end* Halfedge at one end of this edge
→halfedge	*Halfedge_other_end* Halfedge at the other end of this edge
→polycurve	*Backpointer_to_polycurve*
→curve	*Backpointer_to_curve*
→edge	*Next_edge_of_shell* Next edge in the linked list of edges of this shell
→edge	*Previous_edge_of_shell* Previous edge in the linked list of edges of this shell
→edge	*Next_edge_of_polycurve* Next edge of the polycurve to which this edge belongs
→edge	*Previous_edge_of_polycurve* Previous edge of the polycurve to which this edge belongs

HALFEDGE

→vertex	*Vertex* Vertex at this halfedge's side of the edge
→edge	*Backpointer_to_edge*
→loop	*Backpointer_to_loop*
→halfedge	*Next_halfedge_of_loop* In clockwise direction
→halfedge	*Previous_halfedge_of_loop* In anti-clockwise direction

VERTEX

4D vector	*Homogeneous_coordinates*
→halfedge	*Backpointer_to_halfedge*
→vertex	*Next_vertex_of_shell* Next vertex in the linked list of vertices of the shell
→vertex	*Previous_vertex_of_shell* Previous vertex in the linked list of vertices of the shell

POLYCURVE

String	*Name_of_polycurve* The name has no significance for the software. It is specified by the user, and is used for human identification only
Enumerate	*Type_of_plane_which_anchors_polycurve* Polycurve is constrained to remain in this plane. Enumerations are:
	ordinate
	waterline
	buttock
	diagonal — 45° diagonal
	oblique_planar — a plane, not being one of the previous ones.
	completely_free_3D_polycurve — No constraints

4D vector	*Coefficients_of_implicit_plane_equation* Four coefficients A,B,C and D of equation Ax + By + Cz + D = 0 (if polycurve ≠ completely_free_3D_polycurve)
Boolean	*Knuckle_polycurve* Indicates whether this polycurve itself is a knuckle in the surface (a chine)
→shell	*Backpointer_to_shell*
Boolean	*Polycurve_consists_of_a_single_vertex_only* This is not very important for the system. Its sole purpose is to identify the exceptional situation that a polycurve consists of one vertex only
→edge	*First_edge_of_polycurve*
→curve	*First_curve_of_polycurve*
→polycurve	*Next_polycurve_of_shell* Next polycurve in the linked list of polycurves of the shell
→polycurve	*Previous_polycurve_of_shell* Previous polycurve in the linked list of polycurves of the shell

CURVE	
Enumerate	*Boundary_specification_at_start_of_curve* This field specifies for one end of the curve whether tangent or curvature are specified manually or, alternatively, the hierarchical relations (called master-slave relations) between adjacent curves. Enumerations are: tangent_manual curvature_manual tangentcontinuity_neighbouring_curve_is_master tangentcontinuity_neighbouring_curve_is_slave curvaturecontinuity_neighbouring_curve_is_master curvaturecontinuity_neighbouring_curve_is_slave
Enumerate	*Boundary_specification_at_end_of_the_curve* This field specifies the same information for the other end of the curve. Enumerations: equal to *boundary_specification_at_start_of_curve*
Enumerate	*Boundary_specification_crossing_curves* This field specifies the boundary information which is copied into the *Boundary_specification_at_start_of_curve* or *Boundary_specification_at_end_of_curve* field of newly added curves which are crossing this curve. Enumerations are: tangentcontinuity_left_curves_are_master tangentcontinuity_right_curves_are_master curvaturecontinuity_left_curves_are_master curvaturecontinuity_right_curves_are_master
→curve	*Next_curve_of_polycurve*
→curve	*Previous_curve_of_polycurve*
→polycurve	*Backpointer_to_polycurve*
→NURBS curve	*NURBS_for_curve_geometry* Contains the shape of this curve

→**NURBS curve**	*NURBS_for_tangent_ribbon_left* Ribbon of tangent vectors at left side of curve
→**NURBS curve**	*NURBS_for_tangent_ribbon_right* Ribbon of tangent vectors at right side of curve

NURBS CURVE

Enumerate	*Type of curve* This field is used to store the type definition, as supplied by the user. Enumerations are:

general_3D_curve	arbitrary curved
straight_line	straight
circular_2points_radius	circular arc
circular_2points_tangent	circular arc
circular_3points	circular arc
parabola_2points_2tangents	parabola
hyperbola_2points_2tangents_shapefactor	hyperbola

Boolean	*Rational* This variable indicates whether this curve is rational
Floating point	*Radius* Radius of circular arc
Floating point	*Mean deviation* Mean deviation between original and faired points (used in the fairing algorithm)
4D vector	*First_derivatives_start_of_NURBS* First derivative at the start of the NURBS
4D vector	*Second_derivatives_start_of_NURBS* Second derivative at the start of the NURBS
4D vector	*First_derivatives_end_of_NURBS* First derivative at the end of the NURBS
4D vector	*Second_derivatives_end_of_NURBS* Second derivative at the end of the NURBS
Integer	*Degree_of_NURBS* Degree of the NURBS curve
→**NURBS atom**	*First_NURBS_atom* First atom of the NURBS curve
→	*Piecewise_polynomial_representation* For its use see Section 5.2.2

NURBS ATOM

4D vector	*Control_point_location*
Floating point	*Parameter* Parameter value at this vector
→**NURBS atom**	*Next_atom_of_NURBS* Next atom of NURBS curve, nil if it is the end of the list

SURFACE
Integer *Number_of_boundary_curves*
Array of →polycurve
Polycurves_around_surface Array [1..1,1..15] with pointers to all polycurves bounding this surface
Enumerate *Type_of_surface* This field is to store the type of surface, as defined by the user. Enumerations are:

single_patch	
multi_patch	
generic_developable	developable, without further constraints (see definitions in Paragraph 5.4.3.1)
developable_cone	cone
developable_cylinder	cylinder
Pseudo_surface	surface governed by 1 defining curve. See Paragraph 5.4.3.2 for details

Case *type_of_surface*
Single_patch
Integer *Number_of_sides* Number of sides of the patch
Generic_developable
→curve *bounding_curve_1* The first bounding, defining curve
→curve *bounding_curve_2* The second bounding, defining curve
Developable_cone
→curve *Directrix* Curve which forms the directrix
4D vector *Cone_top* Top of the cone
Developable_cylinder
→curve *Directrix* Curve which forms the directrix
4D vector *Generatrix* Direction of generatrix
Pseudo_surface
→polycurve *One_example_curve* Curve which forms an example for this surface

5.1.2 Conventional Euler functions

As pointed out in Section 2.2.2, to maintain the validity of the Euler-Poincaré relation, in a software implementation of a BREP it is good practice to use Euler operators. Another advantage of the use of Euler operators is that they separate the high-level layer of the application from the low-level pointer management in the data structure.

For our HREP these operators have to be extended, however, to include polycurve and curve information. The low-level functions used in HREP represent the conventional Euler operators on manifolds. Quite similar to those used in [Mäntylä, 1988] they are:

MAKE VERTEX, FACE AND SHELL
Creates a shell, a face consisting of one vertex, and a vertex. On input the location of the vertex must be specified.

MVFS (out →**shell** *new_shell*
out →**face** *new_face*
out →**vertex** *new_vertex*
in **4D vector** *coordinates_ of_the_vertex*)

MAKE EDGE AND VERTEX
Creates an edge, two halfedges and a vertex. On input the curve and the polycurve to which the new edge belongs must be specified, as well as the location of the new vertex.

MEV (in →**halfedge** *halfedge1*, *halfedge2* Two halfedges, both pointing to one existing vertex, which will be situated at one end of the edge to create
out →**halfedge** *new_halfedge1*, *new_halfedge2* Two halfedges of the new edge
out →**vertex** *new_vertex*, situated at the other end of the edge created
in **4D vector** *coordinates_of_vertex* Coordinates of the new vertex
in →**polycurve** *polycurve_to_which_edge_belongs*
in →**curve** *curve_to_which_edge_belongs*)

MAKE EDGE AND FACE
Creates an edge, two halfedges and a face. On input the curve and the polycurve to which the new edge belongs must be specified.

MEF (in →**halfedge** *halfedge1*, *halfedge2* Two halfedges, pointing to two vertices to be connected by the new edge
out →**halfedge** *new_halfedge1*,*new_halfedge2* Two halfedges of the new edge
out →**face** *new_face*
in →**polycurve** *polycurve_to_which_edge_belongs*
in →**curve** *curve_to_which_edge_belongs*)

KILL EDGE, MAKE LOOP
Deletes one edge, two halfedges and creates a loop. After this function, *halfedge1* belongs to the existing loop, *halfedge2* to the new one.

KEML (in →**halfedge** *halfedge1*, *halfedge2* Two halfedges of edge to kill)

KILL FACE, MAKE LOOP AND HOLE
Deletes a face and makes a loop and a hole. *Face2* is removed, the new loop is created in *face1*.
KFMLH (in →**face** *face1*, *face2* Two faces to be merged)

KILL EDGE AND VERTEX
Deletes an edge, two halfedges and the vertex pointed to by *halfedge1*.
KEV (in →**halfedge** *halfedge1*, *halfedge2* Two halfedges of the edge to kill)

KILL EDGE AND FACE
Deletes an edge and the face *halfedge2→backpointer_to_loop→backpointer_to_face*.
KEF (in →**halfedge** *halfedge1*, *halfedge2* Two halfedges of the edge to kill)

KILL VERTEX, FACE AND SHELL
Deletes a shell consisting of only one face and a vertex, and that face and vertex.
KVFS (in →**shell** *shell_to_kill*)

MAKE EDGE, KILL LOOP
Creates an edge, two halfedges and deletes a loop.
MEKL (in →**halfedge** *halfedge1*, *halfedge2* Two halfedges, pointing to vertices in two different loops, which will be connected by the new edge
out →**halfedge** *halfedge1*, *halfedge2* Two halfedges of new edge
in →**polycurve** *polycurve_to_which_edge_belongs*
out →**curve** *curve_to_which_edge_belongs*)

MAKE FACE, KILL LOOP AND HOLE
Creates a face and deletes a loop and a hole.
MFKLH (in →**loop** *loop_to_ kill*
out →**face** *new_face*)

5.1.3 Additional structure forming functions

Complementary to the Euler functions, there is a set of functions which operate on the data structure. These are structural functions, and are also required to shield the high-level layer of the application from handling the pointers of structural items themselves.

MAKE POLYCURVE
Creates an empty polycurve.

MAKE_PCURVE	(out	→**polycurve**	*new_ polycurve*
	in	**string**	*name_ of_ polycurve* The descriptive, non-unique name
	in	**enumerate**	*type_of_plane_which_anchors_polycurve* For the enumerations, see the polycurve data type
	in	**real**	*location_of_plane* The input value depends on the value of *type_of_plane_which_anchors_polycurve*: ordinate distance from APP; waterline height above baseline; buttock distance from CL; diagonal height above baseline, at CL; oblique_planar,completely_free_3D_ polycurve not applicable
	in	→**shell**	*shell_to_which_polycurve_belongs*)

KILL POLYCURVE
deletes a polycurve.

KILL_PCURVE	(in	→**polycurve**	*polycurve_to_kill*)

MAKE CURVE
Creates an empty curve.

MAKE_CURVE	(in	→**polycurve**	*polycurve_to_which_curve_belongs*
	in	→**curve**	*curve_after_which_the_new_one_must_be_inserted*
	out	→**curve**	*new_curve*)

KILL CURVE
Deletes a curve.

KILL_CURVE	(in	→**polycurve**	*polycurve_to_which_curve_belongs*
	in	→**curve**	*curve_to_kill*)

Finally, there is one higher-level function GENERATE_POLYCURVE which intersects the surface with an infinite plane, and adds the generated curves to the HREP model. This function uses the methods and algorithms which will be discussed in Sub-Chapters 5.2, 5.3 and 5.4:

GENERATE POLYCURVE

generate_pcurve (in →**4D vector** *intersection_plane* The four coefficients A,B,C and D of the plane equation Ax + By + Cz + D = 0.
out →**polycurve** *generated_polycurve*)

The generation process comprises six steps:

1. For all edges, find the intersections with the intersection plane, in pseudocode:

```
for all edge of the shell do
    intersect the curve edge → backpointer_to_curve with intersection plane;
    if there is an intersection then
        if there are multiple intersections then
            select that one which belongs to edge;
        end if
        store the pointer to the intersection vertex in temporary array V[1..1,1..300]
    end if
end do;
```

2. Create new edges between the new vertices:

```
for all vertex1 of V do
    for all vertex2 of V do
        if vertex1 ≠ vertex2 then
            if vertex1 and vertex2 are in the same loop (to be checked with their
               backpointer_to_halfedge → backpointer_to_loop) then
                make a new edge between vertex1 and vertex2 with MEF();
                store new edge in temporary array E[1..1,1..300];
            end if
        end if
    end for
end for;
```

3. Sort all edges of *E*, so they become adjacent to each other;

4. Generate additional new points on the new edges, by intersecting the original curved surface around each new edge with the intersection plane:
   for all *edge* of *E* **do**
   intersect *edge→halfedge_one_end→backpointer_to_loop→backpointer_to_face→surface_which_includes_this_face* with *intersection plane*
   end do;

5. Create new curves, and assign the new points to these curves. Curves begin or end where they cross polycurves which have been declared by the user to be chines (that means, those which have the *knuckle_polycurve* field set true);

6. With the fairing algorithm, which will be discussed in Sub-Chapter 5.3, generate a smooth curve for each new *edge → backpointer_to_curve → NURBS_for_curve_geometry* on basis of the list of new points.

5.1.4 Construction of the face – surface – curve relationship

The records of the surface data type contain no pointers to connected faces, curves or polycurves. So the relations between those entities are not maintained permanently, but constructed when needed, because of two reasons:
- Surfaces are not necessarily present initially, but may be added or modified in a later design stage;
- To recognize a surface patch complex, or a surface with an auxiliary representation, is more a procedural task than a matter of data organization.

Therefore, links to and from the surface entity are constructed from topological data, each time processing of a surface is required. For each of the three surface types there is a constituent:
- For *single patches*, each surface coincides with the one single face;
- The construction of a *patch complex* will be discussed in Paragraph 5.1.4.3;
- For a *special surface* the user has to specify the boundaries of that surface. The boundaries could be specified with curves or with polycurves. However, in HREP the curves have no name and consequently they are not easy to be identified by a human in printed lists and in alphanumerical menus, so we have chosen to specify the boundaries with polycurves, which are named. For this reason a surface contains an array of polycurves surrounding the surface. A function to determine whether a user-specified surface entity is valid, will be presented in Paragraph 5.1.4.2. However, before we can discuss that procedure, we first have to define some utility functions.

5.1.4.1 *Utility functions*

In this paragraph we define some basic functions, which will be used in the sequel.

MATE

Given one halfedge of edge *he*, MATE returns the other halfedge.

function MATE(*he* :→**halfedge**): →**halfedge**
This function identifies the edge *he* → *backpointer_to_edge*, and chooses the other one of the pointers to both halfedges in the **edge** record.

CLOCKWISE

Given *he*, CLOCKWISE returns the next halfedge about the common vertex, in clockwise direction. It uses the previous halfedge of the loop, as saved in the **halfedge** record.

function CLOCKWISE(*he* : →**halfedge**): →**halfedge**
start of function
CLOCKWISE = MATE(*he* → *previous_halfedge_of_loop*)
end of function

NEXT_HALFEDGE_ON_PCURVE

Given *he*, this function returns the other halfedge on the common polycurve, sharing a vertex, if it exists at all.

function NEXT_HALFEDGE_ON_PCURVE(*he* : →**halfedge**): →**halfedge**
Local variable
next_halfedge_on_pcurve : →**halfedge**
start of function
next_halfedge_on_pcurve := *he*
repeat
next_halfedge_on_pcurve := CLOCKWISE(*next_halfedge_on_pcurve*);
until (*next_halfedge_on_pcurve* = *he*) **or**
(*next_halfedge_on_pcurve* → *backpointer_to_edge* → *backpointer to polycurve* =
he → *backpointer_to_edge* → *backpointer_to_polycurve*)
if *he*=*next_halfedge_on_pcurve* **then** *next_halfedge_on_pcurve* := nil
end of function

DETERMINE_COUNTERPCURVE

This function is used to construct a list of polycurves, crossing the vertex pointed to by *he*.

```
function DETERMINE_COUNTERPCURVE(he : →halfedge)
Local variables
        counterpcurves : array[1..1,1..30] of →polycurve
        number_of_counterpcurves : integer
        he0 : →halfedge
start of function
 he0 := he
 number_of_counterpcurves := 0
 repeat
    if he→backpointer to edge→backpointer to polycurve ≠
       he0→backpointer_to_edge→backpointer_to_polycurve then
      number_of_counterpcurves := number_of_counterpcurves + 1
      counterpcurves[number_of_counterpcurves] := he→backpointer_to_edge→backpointer
      to polycurve
    end if
    he := CLOCKWISE(he)
 until he0=he
end of function
```

HALFEDGE_BELONGS_TO_BOUNDING_PCURVE

This is a function which indicates whether a halfedge belongs to one of the boundary polycurves.

```
function HALFEDGE_BELONGS_TO_BOUNDING_PCURVES(
                    he : →halfedge
                    number_of_bounding_polycurves : integer
                    bounding_polycurves : array[1..1,1..30] of →polycurve
                                                          ): boolean
Local variable
        n : integer
start of function
 halfedge_belongs_to_bounding_pcurves := false
 for n := 1 to number_of_bounding_polycurves do
  if he→backpointer_to_edge→backpointer_to_curve = bounding_polycurves[n] then
       halfedge_belongs_to_bounding_pcurves := true
  end if
 end for
end of function
```

ACTIVATE_FACE

This recursive function marks all faces inside the bounding polycurves, and returns the count of those faces.

```
function ACTIVATE_FACE(f:→face
                       number_of_bounding_polycurves : integer
                       bounding_polycurves : array[1..1,1..30] of →polycurve
                       ): number_of_faces_in_surface: integer
Local variables
        he,he0 : →halfedge
start of function
 number_of_faces_in_surface := 0
 if not f→face_active then
  f→face_active := true
  number_of_faces_in_surface := 1
  he0 := he := f→outer_loop_of_face→first_halfedge_of_ loop
  repeat
   if not HALFEDGE_BELONGS_TO_BOUNDING_PCURVES(
        he,number_of_bounding_polycurves,bounding_polycurves) then
        number_of_faces_in_surface := number_of_faces_in_surface +
        ACTIVATE_FACE(MATE(he)→backpointer_to_loop→backpointer_to_face)
   end if
   he := he → next_halfedge_of_loop
  until he=he0
 end if
end of function
```

5.1.4.2 *Algorithm for recognition of a valid surface area*

The user of the ship hull design system may specify a surface domain, by specifying the boundary polycurves. The computer system must first be able to verify the correctness of this definition, and second, it must notify the faces in that domain that they are part of a surface. Both tasks are performed simultaneously, with the function called VALID_SURFACE, which recognizes all kinds of patches:

1. For a given *surface*, one intersection between two of the bounding polylines must be determined. By traversing the *polycurve→first_edge_of_polycurve→next_edge_of_polycurve* relation of one (arbitrary) polycurve, all edges are visited and, via the *halfedge_one_end* and *halfedge_other_end* pointers from the **edge** record, all halfedges are accessed. For each halfedge with function DETERMINE_COUNTERPCURVE all crossing polycurves are available. If one of them is one of the other bounding polycurves, the intersection is found, otherwise the surface specification is not valid;

2. Starting with the halfedge of the intersection vertex which has *surface* at its right hand, the boundary of *surface* is traversed with the function NEXT_HALFEDGE_OF_SURFACE_BOUNDARY which takes on input one halfedge on the boundary, and returns the next halfedge of the surface boundary:

```
function NEXT_HALFEDGE_OF_SURFACE_BOUNDARY(
                                        he : →halfedge): →halfedge
Local variables
      h,he0 : →halfedge
start of function
  h := MATE(he)
  he0 := h
  he := nil
  repeat
    h := CLOCKWISE(h)
    if h ≠ he0 then
      if HALFEDGE_BELONGS_TO_BOUNDING_PCURVES(
                        h,surface→number_of_boundary_curves,
                        surface→polycurves_around_surface) then
        he := h
      end if
    end if
  until h = he0
end of function;
```

3. If within the (arbitrary) maximum number of traversed polylines of 30 the start polycurve has not been reached, VALID_SURFACE is false;

4. If the surface is valid, only faces inside the surface are marked:

```
Local variables
      f : →face
      number_of_faces : integer
begin of function
      f := shell→first_face_of_shell
      while f <> nil do
       f→face_active := false
       f := f→next_face_of_shell
      end while
      number_of_faces :=
      ACTIVATE_FACE(hstart→backpointer_to_loop→backpointer_to_face)
end of function.
```

The verification phase of the algorithm is demonstrated with the aid of the network of Figure 5.2, where five polycurves, L1..L5, are specified as boundaries of the surface.

1. Assume we start with polyline L1, and the first step of the algorithm finds the intersection with L2;
2. The halfedge at the intersection which has *surface* to its right is H1;
3. The first line of the second step finds MATE(H1), which is H2;
4. The remainder of the second step finds H4 as next halfedge;
5. Even so H9 will be found as subsequent halfedge;
6. In this way L3, L4 and L5 are found, until L1 is reached again.

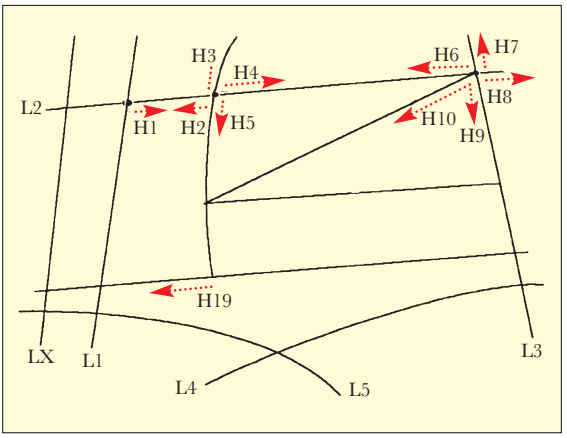

Figure 5.2 *Network.*

5.1.4.3 *Recognition of regular patches complexes*

At this stage we already reveal that in Section 5.4.2 a surface representation for regular patch complexes will be given, so it will be necessary to recognize them. Having adopted a representation for patch complexes, it is advantageous to use complexes as large as possible, because large areas will then be GC^2 continuous. So additionally an algorithm to enlarge a patch complex is desirable. In order to accomplish this, the function VALID_SURFACE from the previous paragraph is extended in two ways:

- For each of the bounding polycurves on output, one possible candidate for extension of the surface is given. That candidate is the polycurve which replaces the polycurve under consideration, and enlarges the surface. An example of such a candidate is LX to replace L1 in Figure 5.2. The candidate is simply selected by taking the polycurve outside and parallel to the boundary polycurve under consideration. With the use of the HREP data structure, given halfedge *he* at the end of the part of the polycurve bounding the surface (for instance H19): $L_{CANDIDATE}$ = NEXT_HALFEDGE_ON_PCURVE(*he*)→ *next_halfedge_of_loop*→*backpointer_to_edge*→*backpointer_to_polycurve*;
- The function specifies on return whether the surface area is regular. For this purpose, for each bounding polycurve a counter is maintained, which counts the number of halfedges which are orthogonal to the bounding polycurve. An orthogonal halfedge is obtained by a call to MATE(*he*→*previous_halfedge_of_loop*) (for the first halfedge on a boundary) or by *he*→*next_halfedge_of_loop* (for the other halfedges). In Figure 5.2 H5 and H9 are orthogonal to L2. A surface is regular if three conditions are met:
  - number of halfedges ⊥ to first polycurve bounding the surface = number of halfedges ⊥ to third polycurve;
  - number of halfedges ⊥ to second polycurve = number of halfedges ⊥ to fourth polycurve;
  - number of faces in surface = (number of halfedges ⊥ to first polycurve – 1) . (number of halfedges ⊥ to second polycurve – 1).

To recognize regular patch complexes the following steps are performed:
1. Select, as bounding polycurves, those of an arbitrary face of the shell;
2. Perform VALID_SURFACE with the bounding polycurves;
3. If the surface is valid and regular, try to replace one bounding polycurve with the candidate, as generated by the VALID_SURFACE procedure. Go to the previous step;
4. If replacing any of the four bounding polycurves by their candidate does not result in a valid and regular surface, the largest regular surface in this area has been found. Continue with the first step, with a face not already lying in any surface.

5.2 Mathematical tools for curve description

5.2.1 Considerations on simple curves

For curve representation NURBS are used. If the only aim was the representation of free form curves, we would just have selected (non-rational) B-splines, because after preliminary experiments it appeared that curve shape modification can more intuitively be done by adding and moving vertices of a B-spline, rather than by the modification of the NURBS' weight factors.

However, the NURBS offers one very nice feature in the context of hull form modelling, that is the ability to represent an exact conic. It is this ability that justifies the inclusion of the NURBS curve in the HREP.

To represent conics we adopted the approach as discussed in [Piegl and Tiller, 1987]. We define the quadratic function

$$\mathbf{C}(t) = \frac{B_0(t)w_0\mathbf{P}_0 + B_1(t)w_1\mathbf{P}_1 + B_2(t)w_2\mathbf{P}_2}{B_0(t)w_0 + B_1(t)w_{11} + B_2(t)w_2},$$

where B are the second-degree Bézier or NURBS coefficients. The w_i's are the weight factors and the $\mathbf{P}_i$'s the vertices of a triangle, according to Figure 5.3. If $w_0=w_2=1$, and $w_1 = \mathbf{MS}/\mathbf{SP}_1$, the whole family of conics can be represented by the rational Bézier or NURBS curve. If $w_1>0$, **C** represents a hyperbola, if $w_1=1$ a parabola and if $w_1<1$ an ellipse.

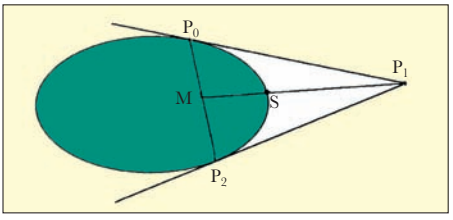

Figure 5.3 *Rational quadratic representation of ellipse.*

It is very attractive to have one mathematical formulation for multiple types of curves. However, there is one disadvantage, which is the lack of agreement between the traditional representation of conics, and the NURBS-based representation. In the classical approach, a conic is the intersection between a plane and a cone (see Figure 5.4), and it is represented by its eccentricity, foci and directrix (see for instance [Jennings, 1994]). This classical representation, which has direct geometrical meaning, is more appealing to many ship designers than the NURBS-based representation, which is only based on coefficients

of a mathematical equation. For example, one might know that an ellipse did result by the intersection of a cylinder with a plane, in which case the cylinder radius and the intersection angle could be specified, but not the NURBS curve coefficients.

In order to be able to present the classical representation towards the system user, a formal conversion between the classical and the NURBS-based representation is desirable, but neither our own imagination, nor a survey of literature provided a solution for this task.

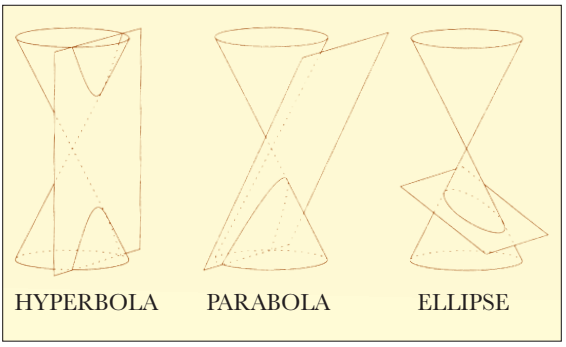

Figure 5.4 *Classical way of viewing a conic (reprint from [Jennings, 1994]).*

According to our knowledge, only numerical methods are available to convert between the representations, for instance:

- An algebraic elaboration of classical metrics of the conics. [Liming, 1979] gives numerous examples of such procedures;
- The brute force approach: with the NURBS representation, generate five distinct points on the conic, and determine the classical metrics of the conics by solving the system of five equations with five unknowns.

5.2.2 Representation of the NURBS curve

So for curve representation in HREP an ordinary NURBS curve is used. Of course the order of the NURBS could be laid into the hands of the user. However, with our aim to shield the user from mathematical involvement, the order has been fixed for each type of curve:

- Straight line segments are of second order, and other simple curves are of third order;
- Free form curves with no constraints for the end point derivatives, or with only the first order end point derivative specified, are of fourth order. It is generally accepted, and it is also our own experience, that fourth-order splines offer sufficient freedom, and are 'stiff' enough to avoid undesired inflections;
- A fourth-order spline has too little degree of freedom to accommodate second order derivative constraint for the begin and the end points of the curve. For free form curves with two second order derivative constraints a spline of the sixth order must be chosen, which is, because of its higher order, unfortunately rather 'flexible' in its behaviour.

An interactive computer system needs a fast algorithm for the evaluation of the spline and its derivatives. The de Boor-Cox recurrency algorithm is available for evaluation, but as suggested by many authors, if a large number of evaluations occurs inside one knot interval a polynomial expression is more efficient. For evaluation we selected the piecewise polynomial (PP) representation of [de Boor, 1978], which provides the requested efficiency. In the **NURBS curve** record a pointer to the piecewise polynomial representation of a NURBS is maintained, so the conversion from control point representation to piecewise polynomial representation is a one-off event.

Evaluation of the piecewise polynomial is performed with the function CALCULATE_PP, which calculates the value of the D^{th} derivative at *Parameter*, using two instruments:

- For the actual evaluation of one derivative, a function which is algorithmically identical to the FORTRAN function PPVALU in [de Boor, 1978] is used;
- To take the weight factor of the NURBS into account, the chain rule is applied, as discussed in [Farin, 1990].

In pseudocode the function is:

```
function CALCULATE_PP (Piecewise_polynomial : Pointer to PP record
                       Rational : Boolean
                       Derivative : Integer
                       Parameter : Floating_point) : 4d vector
Local variables
        N,J : Integer
        Dxdt : array[0..6] of 4d vector
        tmp : 4d vector
begin of function
 if rational then
  for N := 0 to Derivative do
   Dxdt[N] := PPVALU(Piecewise_polynomial,N,Parameter)
    for J := 1 to N do
     tmp :=  Dxdt[N-J] * Dxdt[J] [weight] * binomial_coefficient(N,J)
     Dxdt[N] := Dxdt[N] - tmp
    end for
   Dxdt[N] := Dxdt[N] / Dxdt[0] [weight]
  end for
  calculate_pp := Dxdt[Derivative]
  calculate_pp[weight] := 1
 else
  calculate_pp := PPVALU(Piecewise_polynomial,Derivative,Parameter)
 end else
end of function
```

5.2.3 Boundary conditions for the curves

According to the system architecture of Sub-Chapter 4.1, end point derivative constraints may be specified by the user or, alternatively, they may be declared to be equal to the corresponding derivative of an adjacent curve. This is an inheritance of the end-boundary conditions, and these relations between derivatives of adjacent curves have been baptized *master – slave relations*.

The manual definition of a tangency constraint is rather straightforward. The user simply supplies the tangent information (for example by pointing in the desired direction

with a rotatable vector). The manual definition of specific second order derivatives or curvatures is not practical; the only second-order constraint the user can specify is zero curvature.

To specify a master-slave relation, for each extremity of the curve the user may specify that the tangent or the curvature of the curve adjacent to that end governs. So the curve under consideration will always be slave, and the adjacent curve master. A curve which is on one extremity slave can, on the other one, be master for the adjacent curve.

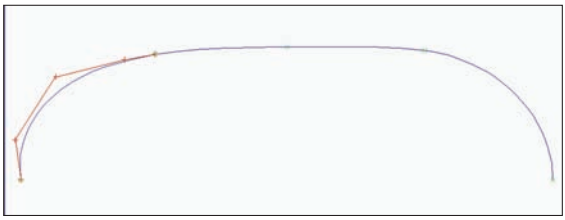

Figure 5.5 *Four curves with tangency dependancy.*

In this way a whole chain of relationships can be defined, and at any time the actual curve geometry for all curves concerned will be calculated according to these relationships. This is illustrated in Figure 5.5 where four curves with the tangent of the left side of the curves are slave of the connected curve. When moving a spline vertex of the leftmost curve, as demonstrated in Figure 5.6, the tangency relations cause all four curves to be updated. The type of relationship between curves is stored in HREP in the *boundary_specification_at_start_ of_curve* and *boundary_specification_at_end_of_curve* fields of the **curve** record, and the actual values of user-defined tangent or curvature are stored in the 'derivatives' fields of the **NURBS curve** record.

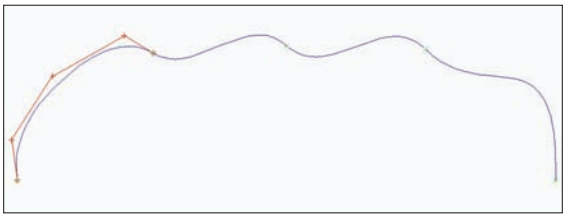

Figure 5.6 *Manipulating leftmost curve updates all curves.*

We applied a recursive function, called GENERATE_NURBS_GEOMETRY, for the interpolation of a NURBS through known data points, in order to keep track of nested master-slave relations. This function utilizes the CALCULATE_PP function of the previous section, and the function INTERPOLATE_SPLINE, which generates a spline curve through a number of data points, with possible constraints on end point derivatives. INTERPOLATE_SPLINE is identical to the curve fairing algorithm, which will be discussed in the next sub-chapter, albeit with a maximum allowed mean deviation of zero.

```
function GENERATE_NURBS_GEOMETRY(c:Curve)
begin of function
 if c→boundary_specification_at_start_of_curve =
       (tangentcontinuity_neighbouring_curve_is_slave or
       curvaturecontinuity_neighbouring_curve_is_slave) then
       GENERATE_NURBS_GEOMETRY(c→previous_curve_of_polycurve)
 end if
```

```
if c→boundary_specification_at_end_of_curve =
      (tangentcontinuity_neighbouring_curve_is_slave or
      curvaturecontinuity_neighbouring_curve_is_slave) then
      GENERATE_NURBS_GEOMETRY(c→next_curve_of_polycurve)
end if
if c→boundary_specification_at_start_of_curve =
      tangentcontinuity_neighbouring_curve_is_master then
    c→NURBS_for_curve_geometry→first_derivative_start_of_curve :=
      CALCULATE_PP( c→previous_curve_of_polycurve→NURBS_for_curve_geometry→
                        piecewise_polynomial_representation,
                    c→previous_curve_of_polycurve→NURBS_for_curve_geometry→
                        rational,
                    1,
                    last parameter of c→previous_curve_of_ polycurve→
                        NURBS_for_ curve_geometry)
end if
if c→boundary_specification_at_start_of_curve =
    curvaturecontinuity_neighbouring_curve_is_master then
     c→NURBS_for_curve_geometry→second_derivative_at_start_of_curve :=
      CALCULATE_PP(
c→previous_curve_of_polycurve→NURBS_for_curve_geometry→
                        piecewise_polynomial_representation,
                    c→previous_curve_of_polycurve→NURBS_for_curve_geometry→
                        rational,
                    2,
                    last parameter of c→previous_curve_of_ polycurve→
                        NURBS_for_ curve_geometry)
end if
if c→boundary_specification_at_end_of_curve =
    tangentcontinuity_neighbouring_curve_is_master then
     c→NURBS_for_curve_geometry→first_derivative_end_of_curve :=
      CALCULATE_PP( c→next_curve_of_polycurve→NURBS_for_curve_geometry→
                        piecewise_polynomial_representation,
                    c→next_curve_of_polycurve→NURBS_for_curve_geometry→
                        rational,
                    1,
                    first parameter of c→next_curve_of_ polycurve→
                        NURBS_for_ curve_geometry)
end if
if c→boundary_specification_at_end_of_curve =
    curvaturecontinuity_neighbouring_curve_is_master then
    c→NURBS_for_curve_geometry→second_derivative_at_end_of_curve :=
      CALCULATE_PP( c→next_curve_of_polycurve→NURBS_for_curve_geometry→
                        piecewise_polynomial_representation,
```

```
                    c→next_curve_of_polycurve→NURBS_for_curve_geometry→
                      rational,
                    2,
                    first parameter of c→next_curve_of_ polycurve→
                      NURBS_for_ curve_geometry)
  end if
  INTERPOLATE_SPLINE(c→NURBS curve)
end of function
```

5.3 Implementation of the fairing algorithm

For the reasons outlined in Section 4.2.3, we have selected Dierckx's approach (see [Dierckx, 1982] and [Dierckx, 1993]) as basis for the HREP fairing/smoothing method. The implementation is as follows, given:

- N data points $\mathbf{q}_i$, with the corresponding weight factors w_i and parameter values t_i;
- a user-specified maximum allowed mean deviation M;
- the degree of the spline K=3.

The task is to find a smooth spline $\mathbf{f}_g(t)$, with g the number of knots, for which the sum of squared deviations is less than S, with

$$S = M^2 \sum_{i=1}^{N} w_i .$$

Furthermore we use fairness function $\mathcal{J}(\mathbf{f})$ of Equation (2.5), which expresses the square of the jump of the second order derivatives:

$$\mathcal{J}(\mathbf{f}) = \sum_{i=2}^{g-1} \left(\mathbf{f}_g^{(k)}(t_i+) - \mathbf{f}_g^{(k)}(t_i-) \right)^2 ,$$

and the function $E(\mathbf{f})$ of Equation (2.4), which represents the closeness of fit, and is defined as the sum of the weighted squared deviations:

$$E(\mathbf{f}) = \sum_{i=1}^{N} w_i \, | \mathbf{f}_g(t_i) - \mathbf{q}_i |^2 .$$

To find the balance between fairness and closeness of fit $\mathcal{J}(\mathbf{f})$ must be minimized, under the condition that $E(\mathbf{f}) \leq S$. This constrained minimization problem is solved by minimizing $\mathcal{J}(\mathbf{f}) + pE(\mathbf{f})$, where p must be chosen so that $F_g(p) = E(f) = S$.
It was proved by Dierckx that when $p \to \infty$, $\mathbf{f}(t)$ tends to the least-squares spline with g knots, and when $p=0$, $\mathbf{f}(t)$ becomes the weighted least-squares polynomial of degree K.

Apparently, the Dierckx algorithm consists of two basic steps:
1. Find the minimum number of knots g, and a knot distribution, for which $F_g(\infty) < S$;
2. Find for this number of knots and for this knot distribution the value p^* for which $f_g(p) = S$.
This process is illustrated in Figure 5.7.

For the first step, let us assume that we have knots $\lambda_1..\lambda_g$, and calculate the least squares spline $\mathbf{f}_g(t)$ determined by the N data points $\mathbf{q}_i$. When writing out the equations we obtain the familiar over-determined linear system of equations, which can be solved in several ways. Two applicable methods are (see [Gentleman, 1966]):

- The most commonly used method of multiplying the observation matrix with its transpose, so that the *normal equations* are obtained, which can be solved with an ordinary matrix solver. According to Dierckx, the matrix of the normal equations is banded as well as positive definite, so the efficient Cholesky solver for band matrices can be used successfully. For large systems, however, the normal equations may become ill-conditioned, which may cause numerical instability;
- Orthogonalization, where an upper triangular matrix is constructed which can easily be solved by backsubstitution.

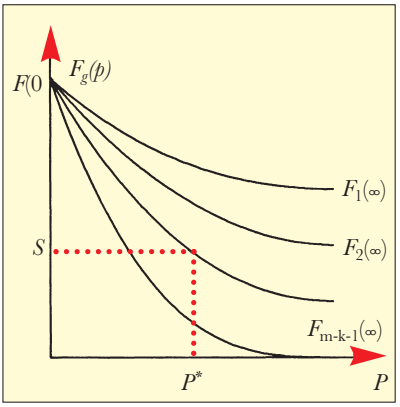

Figure 5.7 *The smoothing function $F_g(p)$, (reprint from [Dierckx, 1993], by permission of Oxford University Press).*

To maintain computational stability, the last method has been selected. Originally Dierckx selected the orthogonalization method with Givens transformations, based on an alternative computation without square roots. In later implementations, conventional Givens transformations with square roots have been used. The transformation with square roots is computationally less efficient, because of the relatively long evaluation of a square root, and in order to make our computer system as efficient as possible we have been experimenting with an implementation without square roots. No problems with stability have been encountered.

When the spline $\mathbf{f}_g(t)$ has been determined on a knot sequence $\lambda_1..\lambda_g$, and $E(\mathbf{f})$ is still greater than S, a new knot λ must be added to the knot sequence, and a new iteration has to be executed to determine $\mathbf{f}_{g+1}(t)$. Dierckx's implementation does provide for multiple simultaneous knot insertions, in order to decrease the number of iterations. Our experiments sometimes showed a large number of inserted knots, and a rather asymmetrical distribution of the new knots. So in the HREP implementation only one single knot is added at a time.

To determine the placement of the new knot, for all spans $\lambda_{j-1}..\lambda_j$, j=2..g, we calculate the errors δ_j:

$$\delta_j = \sum_{r=u}^{v} w_r \,|\, \mathbf{f}_g(t_r) - \mathbf{q}_r \,|^2 \qquad \text{with } \lambda_{j-1} < t_u < t_{u+1} < t_{u+2} \ldots < t_v < \lambda_j .$$

The new knot is added in the middle of the span j with the largest error δ_j.

Once the number of knots g has been found for which $F_g(\infty) = f_g(t) < S$, $F(0)$ is determined, and by means of an iterative numerical procedure the value p^* is found for which $F(p^*) = S$.

The obtained control points of the spline are saved in HREP's **NURBS CURVE** record, with a weight factor of unity for all control points, so actually a non-rational, non-uniform spline is generated.

The last subject we have to discuss in this sub-chapter is fairing with end point derivatives. A formal description of the method is given in [Dierckx, 1993], and will not be repeated here. Described briefly, the method comprises the following steps:

- Suppose we have S derivative constraints at the start of the curve and E at the end;
- A spline $\mathbf{p}(t)$ is generated, to satisfy the end point derivatives only. The minimum order for $\mathbf{p}(t)$ is 2 + S + E;
- All the N data points $\mathbf{q}_i$ are decreased by the corresponding values of $\mathbf{p}(t)$:

  for $i = 1$ to N **do**
  $\mathbf{q}_i^* := \mathbf{q}_i - \mathbf{p}(t_i)$
  end for;
- With the method described above, taking into account the specified mean deviation, a fair spline $\mathbf{s}^*(t)$ is generated for the data points $\mathbf{q}^*$. As an additional constraint, $\mathbf{s}(t)$ has to be zero for the end point derivatives. This additional constraint is taken into account by setting the (S+1) first and (E+1) last spline coefficients to zero;
- The final spline $\mathbf{s}(t)$ is obtained by adding the two constituents: $\mathbf{s}(t) := \mathbf{p}(t) + \mathbf{s}^*(t)$.

5.4 Implementation of the surface model

5.4.1 Specification of surface patches

5.4.1.1 *Relations between adjacent patches*

In order to obtain some kind of continuity between adjacent patches, certain positional and derivative conditions have to be fulfilled. In this respect the topology of the patches is important. According to [Varady, 1987] there are two kinds of relationships between adjacent patches. One of them is PFP (Patch Facing Patch), when each patch boundary coincides with exactly one boundary of the opposite patch, and the other one is MPFP (Multi-Patch Facing Patch), when one patch boundary coincides with multiple boundaries of adjacent patches. In Figure 5.8 two examples are shown.

It might be clear that PFP connection is easier to handle than MPFP, because with PFP for each boundary only the single adjacent patch has to be taken into consideration, while for MPFP multiple patches play a role. For MPFP even a chain of mutually affecting

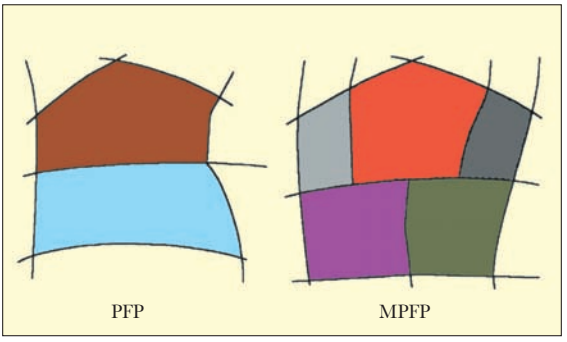

Figure 5.8 *Two kinds of relations between patches.*

patches may occur. On the other hand, it is clear that a ship hull modelling system would be of little use when it covers PFP only. We have to take MPFP into consideration.

5.4.1.2 *Construction of tangent ribbons*

Because we are not restricted to PFP, to ensure GC^1, i.e. tangent-plane continuity between adjacent patches, tangent ribbons over all boundary curves must be constructed. In the case of surface modelling, the tangent ribbons will supply tangency information for interfacing the patches. This is illustrated in Figure 5.9, which shows a network of curves C_i and D_j. For example, the tangent ribbon of curve C_1 is computed by evaluating the connections with the curves $D_1..D_5$. The tangent ribbons are constructed by means of a set of cross boundary vectors, which are calculated at each intersection between two curves. Of course, cross boundary vectors must be independent from curve parametrization, therefore, as is suggested in [Jensen, 1987] and [Jensen, 1991], the Gram-Schmidt orthogonalization process is applied. In Figure 5.10 a magnification of a part of a network is sketched, in order to illustrate how to construct cross boundary vector **T**, with the aid of the tangents to the curves $\mathbf{C}(s)$ and $\mathbf{D}(t)$. The component of vector

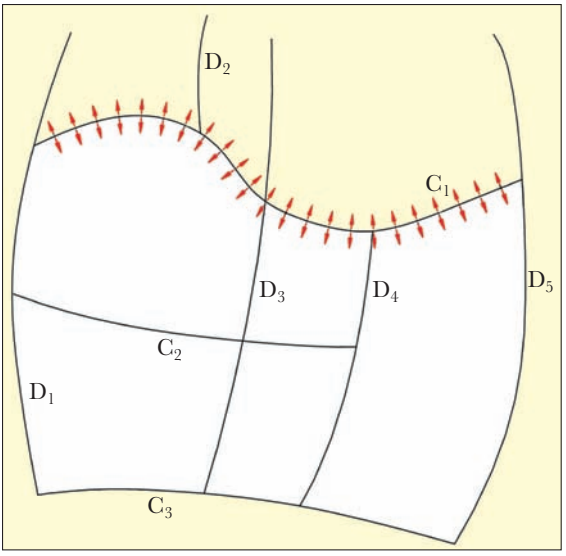

Figure 5.9 *Patch layout and tangent ribbon.*

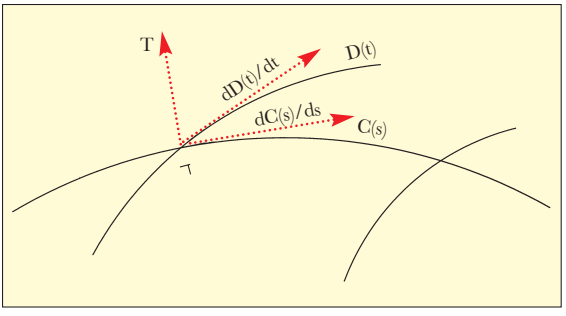

Figure 5.10 *Construction of cross boundary vector. (Please note that* T *is co-planar with the two boundary vectors).*

$$\frac{d\mathbf{D}(t)}{dt} \text{ along } \frac{d\mathbf{C}(s)}{ds} \text{ is}$$

$$\frac{\left(\dfrac{d\mathbf{D}(t)}{dt} \cdot \dfrac{d\mathbf{C}(s)}{ds}\right)}{\left|\dfrac{d\mathbf{C}(s)}{ds}\right|} \cdot \frac{\dfrac{d\mathbf{C}(s)}{ds}}{\left|\dfrac{dC(s)}{ds}\right|}$$

so

$$\mathbf{T} = \frac{d\mathbf{D}(t)}{dt} - \frac{\left(\frac{d\mathbf{D}(t)}{dt} \cdot \frac{d\mathbf{C}(s)}{ds}\right)}{\frac{d\mathbf{C}(s)}{ds} \cdot \frac{d\mathbf{C}(s)}{ds}} \cdot \frac{d\mathbf{C}(s)}{ds} \cdot \tag{5.1}$$

The cross boundary vectors **T** are normalised, to make them independent from parametrization. When programmed, Equation (5.1) gives rather small cross boundary vectors **T** for small angles between the tangents

$$\frac{\mathbf{C}(s)}{ds} \quad \text{and} \quad \frac{\mathbf{D}(t)}{dt}$$

By the normalization of **T** the possible approximation error is magnified, giving rise to undesired shape anomalies, further on in the surface generation process. This problem could be solved by ignoring **T** when the angle between the tangents is smaller than a minimum value. In some of our practical tests it appeared that a minimum value of 15° gives satisfactory results.

The tangent ribbon's construction process could be implemented rather efficiently, based on our HREP. Due to the availability of the polycurve-edge-curve relation, the list of crossing curves can simply be traversed for each curve. The tangent ribbons are calculated for each curve $\mathbf{D}_j$ crossing $\mathbf{C}(s)$ at s, and saved as $\mathbf{T}(s)$ in the curve record (in the *NURBS_for_tangent_ribbon_left* and *NURBS_for_tangent_ribbon_left* of the **curve** record of Section 5.1.1).

The information delivered by the tangent ribbons will be used to create cross boundary surface derivatives when the surface patch is constructed. In order to make curves of a network compatible, we use surface parameters u and v, so curves $\mathbf{C}(s)$ and $\mathbf{D}(t)$ are reparametrized to $\mathbf{C}(u)$ and $\mathbf{D}(v)$, by means of the reparametrization functions $u=u(s)$ and $v=v(t)$. If for $u(s)$ and $v(t)$ ordinary polynomial functions are used, there is the risk of oscillation, so that the unique mapping of s on u, and of t on v may be lost. To avoid this danger, the strictly monotonic interpolation function presented in [Delbourgo and Gregory, 1983] is used.

Because each

$$T \perp \frac{d\mathbf{C}(s)}{ds},$$

for each curve $\mathbf{D}(t)$ crossing $\mathbf{C}$ at (s,t) there exist scalars α and β for which

$$\frac{d\mathbf{D}(v)}{dv} = \alpha \frac{\frac{d\mathbf{C}(u)}{du}}{\left|\frac{d\mathbf{C}(u)}{du}\right|} + \beta \frac{\mathbf{T}(u)}{|\mathbf{T}(u)|}, \tag{5.2}$$

where α is the projection of $\frac{d\mathbf{D}(v)}{dv}$ upon $\frac{d\mathbf{C}(u)}{du}$:

$$\alpha = \frac{d\mathbf{D}(v)}{dv} \cdot \frac{\frac{d\mathbf{C}(u)}{du}}{\left| \frac{d\mathbf{C}(u)}{du} \right|},$$

and β is the projection of $\frac{d\mathbf{D}(v)}{dv}$ upon $\mathbf{T}(u)$:

$$\beta = \frac{d\mathbf{D}(v)}{dv} \cdot \mathbf{T}(u) .$$

For each curve $\mathbf{D}_j$ which crosses $\mathbf{C}$, α_j and β_j are determined, and subsequently two scalar interpolating splines $\alpha(u)$ and $\beta(u)$ are constructed, which are used in (5.2) to determine the first derivative function for all patch boundaries.

Subsequently, we may have to face the situation in which multiple curves meet at one vertex. For instance, if we want to determine the tangent ribbon of curve C0 of Figure 5.11, at vertex V3, we encounter four different curves, which all can be utilized in (5.2) to determine the tangent ribbon.

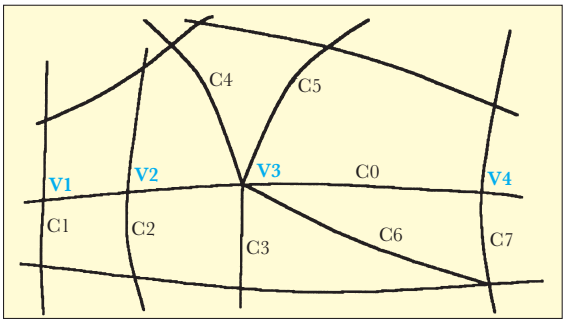

Figure 5.11 *Multiple curves meet at common vertex.*

In these situations the first thing to do is to create *two* ribbons, one for the left side of the curve, and one for its right side. This is also a sensible feature to handle knuckle curves (chines), because the tangent ribbons on both sides of a knuckle curve are not coplanar.

Also when separate tangent ribbons for the left and the right side are used, multiple curves still may meet at one vertex. If we traverse curve C0 in the sequence V1..V4, to determine the tangent ribbon at the right side of C0, at vertex V3 we meet curves C3 and C6 which both might be used to calculate the tangent ribbon at V3. Theoretically C3 and C6 should be compatible, i.e. they should result in equal tangent ribbons. Practically both curves might not be compatible, and the question remains how to determine the tangent ribbon in that case. Instead of solving this problem with mathematical rigour, we followed a pragmatic approach:

- Only one of the curves is used. The choice of curve is arbitrary. Therefore, we select the first curve encountered. As far as Figure 5.11 is concerned, for the tangent plane at the right side of C0, C3 is used (and not C6), and for the tangent plane on the left side C4 is used (and not C5);
- For $\alpha(u)$, $\beta(u)$, $\mathbf{C}(u)$ and $\mathbf{T}(u)$ interpolating splines of the fourth order are used, so interpolation of the α_i's, β_j's, C_j's and T_j's, as determined at the vertices, is performed with

second-order continuous derivatives. In non-compatible configurations this might lead to undulations of the tangent plane. In this case the user of the software can select linear interpolation. When using linear interpolation, only the α_i's and β_j's determined at both ends of the edge to process are used. So in the figure the tangent ribbon at the right side of the edge V2..V3 is based on C2 and C3, and the ribbon of the edge V3..V4 is based on C6 and C7.

We are not very satisfied with this 'solution', because it confronts the user of the software with a mathematical detail, which is contradictory to our requirement to hide those as much as possible. Fortunately, in practice it appears that linear interpolation only seldom has to be used.

Finally, two aspects of the construction of the tangent ribbons can be noted:

- As discussed in [Hoschek and Lasser, 1992] and [Nowacki et al, 1997], the weighting factor functions $\alpha(u)$ and $\beta(u)$ cannot be chosen arbitrarily, because they must comply with the orders of the patch and the vector-valued interpolating splines $\mathbf{C}(u)$ and $\mathbf{T}(u)$. We have not explicitly taken these considerations into account, but in practical tests our approach as described above performs well;
- In the implementation of the described mechanism the curves are evaluated numerically to obtain

  $$\mathbf{C}(u) \text{ and } \frac{d\mathbf{C}(u)}{du},$$

  This makes the surface construction method quite independent of the actual curve representation (which is NURBS in the current implementation), and has the benefit that possible future alternative curve representations can be accommodated.

5.4.1.3 *Processing of N-sided patches*

In Section 4.2.2. we concluded to use the boolean sum approach for N-sided patches. After implementation of the method of Paragraph 2.3.3.2, it indeed appeared that the twists were not always compatible, which led to surface instability, so Equation (2.3) is replaced by the Gregory-like rational combination

$$\mathbf{p}_j(\mathbf{X}) = \mathbf{T}_1(\mathbf{X}) + \mathbf{T}_2(\mathbf{X}) - \left[\frac{v_j}{(u_j+v_j)}.\right]\mathbf{T}_{12}(\mathbf{X}) - \left[\frac{u_j}{(u_j+v_j)}.\right]\mathbf{T}_{21}(\mathbf{X}),$$

with $\mathbf{T}_{21}(\mathbf{X})$ identical to $\mathbf{T}_{12}(\mathbf{X})$ of Equation (2.2), except for the twist term

$$\frac{\partial \mathbf{t}_j(0)}{\partial u} \text{ which is replaced by } \frac{\partial \mathbf{t}_{j-1}(0)}{\partial v}.$$

Our experiments with a variety of ship hull forms have shown that for $5 <= N <= 10$ this method works well, in the sense that a satisfactory surface was obtained. In Sub-Chapter 6.3 an example will be presented.

For N = 4 this method is not applied, because a standard Coons - Gregory patch, as described in Section 2.3.1 is used.

[Gregory, 1982] gives a different solution for N = 3, but instead of using that one we have

applied the boolean sum / convex combination method discussed above also for triangles. It proved to work well.

N = 2 patches can also occur in practice, as shown in Figure 3.6 of Section 3.2.2. We have tried to incorporate it in several ways:

- As a 4-sided patch with two degenerated sides. However, experiments showed problems with twist compatibility, while the rational twist estimator cannot be used along the degenerated sides. This is because the Gregory-like twist estimator interpolates incompatible twists over *corners* of the patches, while incompatible twists over the (degenerated) *edge* occur;
- Similar to the method for triangles of [Gregory, 1984] we could construct two 'corner patches'. For the reason that in parameter space the distances from every point in the patch to the sides are zero, the convex combination is useless, so this approach also failed;
- For our implementation we have split one side, thus creating a 3-sided patch with one degenerated side. Conceptually, we don't like this solution because it is not symmetrical. Experiments, however, show satisfactory performance.

All positional and derivative information which is needed for the patch equations is generated with the tangent ribbon method discussed in the previous paragraph.

5.4.2 Implementation of surface patch complexes

The Gordon patch discussed in Section 2.3.2 is supposed to serve for representation of patch complexes. However, there are two drawbacks to eliminate:

- It can only be applied to a regular network of curves;
- With a network of GC^2 curves, by construction the patch is internally GC^2 in the internal domain. Because derivatives at the boundaries are not taken into account, it is only GC^0 continuous at the patch boundaries.

The first drawback is inherent to this kind of patch, and in Paragraph 5.1.4.2 we have developed a method to recognize regular patch complexes as large as possible. The second drawback is removed with an extended version, the GC^1 Gordon patch, as described in [Jensen, 1991].

The GC^1 Gordon patch for a network of $m \times n$ curves can be described as

$$\mathbf{F}(u,v) = \mathbf{F}_1 + \mathbf{F}_2 - \mathbf{F}_{12}\,, \tag{5.3}$$

with

$$\mathbf{F}_1(u,v) = \sum_{i=0}^{m+1} \mathbf{F}(u_i,v)\,.\,C_i(u)\,,$$

and

$$\mathbf{F}_2(u,v) = \sum_{j=0}^{n+1} \mathbf{F}(u,v_j)\,.\,C_j(v)\,,$$

where $\mathbf{F}(u_0,v)$ is a shorthand notation for

$$\frac{\partial \mathbf{F}(u_1,v)}{\partial u}, \mathbf{F}(u_{m+1},v) \text{ for } \frac{\partial \mathbf{F}(u_m,v)}{\partial u}, \mathbf{F}(u,v_0) \text{ for } \frac{\partial \mathbf{F}(u,v_1)}{\partial v} \text{ and } \mathbf{F}(u,v_{n+1}) \text{ for } \frac{\partial \mathbf{F}(u,v_n)}{\partial v}.$$

In order to satisfy the boundary conditions, $C_j(v)$ must be a cardinal function with the properties:

$$C_j(v_k) = \delta_{j,k} \quad , \quad \frac{dC_j(v_1)}{dv} = 0 \quad , \quad \frac{dC_j(v_n)}{dv} = 0 \quad j{=}1..n \, , \, k{=}1..n \, ,$$

$$C_0(v_j) = 0 \quad , \quad \frac{dC_0(v_1)}{dv} = 1 \quad , \quad \frac{dC_0(v_n)}{dv} = 0 \quad j{=}1..n \, ,$$

$$C_{n+1}(v_j) = 0 \quad , \quad \frac{dC_{n+1}(v_1)}{dv} = 0 \quad , \quad \frac{dC_{n+1}(v_n)}{dv} = 1 \quad j{=}1..n \, .$$

With a similar expression for $C_i(u)$.

Following [Jensen, 1991], the tensor product surface $\mathbf{F}_{12}$ is made compatible, by a method comparable to the *Brown's square* (see [Barnhill, 1977]), rather than by a Gregory construction. The surface equation is:

$$\mathbf{F}_{12}(u,v) = \frac{u^2(1\text{-}u^2)\mathbf{G}_1(u,v) + v^2(1\text{-}v^2)\mathbf{G}_2(u,v)}{u^2(1\text{-}u^2) + v^2(1\text{-}v^2)},$$

where

$$\mathbf{G}_1(u,v) = \sum_{j=1}^{n} \mathbf{F}_1(u,v_j) \, . \, \mathbf{C}_j(v) + \frac{\partial \mathbf{F}_1(u,v_1)}{\partial v} \, . \, \mathbf{C}_0(v) + \frac{\partial \mathbf{F}_1(u,v_n)}{\partial v} \, . \, C_{n+1}(v) \, ,$$

and

$$\mathbf{G}_2(u,v) = \sum_{i=1}^{m} \mathbf{F}_2(u_i,v) \, . \, \mathbf{C}_i(u) + \frac{\partial \mathbf{F}_2(u_1,v)}{\partial u} \, . \, \mathbf{C}_0(u) + \frac{\partial \mathbf{F}_2(u_m,v)}{\partial u} \, . \, C_{m+1}(u) \, .$$

This patch type was also implemented with derivative information based on the tangent ribbons, as determined with the method discussed in Paragraph 5.4.1.2.

5.4.3 Description of special surfaces

It was mentioned at the discussion of the system architecture of Sub-Chapter 4.2 that developable surfaces and pseudo-surfaces require special attention. The implementation of both types is the subject of this section.

5.4.3.1 *Developable surfaces*

According to [Redont, 1989] there are at least three ways of defining developable surfaces:

- Developable is any surface that can be obtained by bending a plane, where bending is a transformation which preserves arc lengths;

- A ruled surface is a family of straight lines, $\mathbf{x}(t,v) = \mathbf{p}(t) + v\mathbf{q}(t)$, with $\mathbf{q}(t)$ the generatrix. This surface is developable if one of these conditions is met :
  - $\mathbf{p}$ is constant, then the surface is a cone;
  - $\mathbf{q}$ is constant, then the surface is a cylinder;
  - $\mathbf{q} = \frac{d\mathbf{p}}{dt}$, then the surface is generic developable, and $\mathbf{p}(t)$ is called the convolute of the surface. This definition is illustrated in Figure 5.12.
- A developable surface is a one-parameter family of planes. Each tangent plane touches the surface along a straight line, called the ruling, see Figure 5.13 for an illustration.

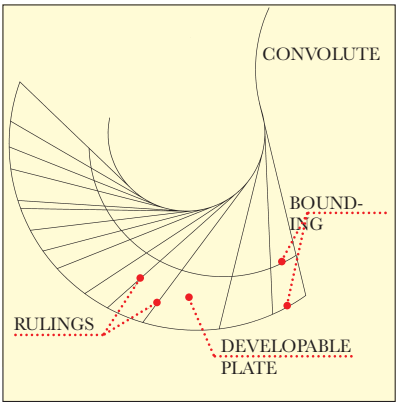

Figure 5.12 *Developable surface.*

The first definition has the most physical nature. The second definition is very concise mathematically, and it is the most efficient one, in terms of data storage and algorithmic complexity. The third definition is a geometric one, and, consequently, the most appealing to a ship designer. This definition has also been used in specific naval architectural publications on developable surfaces, [Nolan, 1971] and [Clements, 1981].

Initially, it was our intention to use the second method for internal representation, because of its efficiency, but after some preliminary experiments it became obvious that for a nearly cylindrical surface the convolute may be located at quite a large distance from origin, giving rise to possible numerical instability. Even worse, for a nearly cylindrical surface the convolute may switch from one side of the surface to the other side, causing enormous instabilities, when $\mathbf{p}(t)$ was represented by a GC^2 spline.

Based on these considerations, and taking into account our desire that the user should be shielded from all mathematics as much as possible, in our system we have decided to use the third definition to *present* the developable surface mechanism to the user.

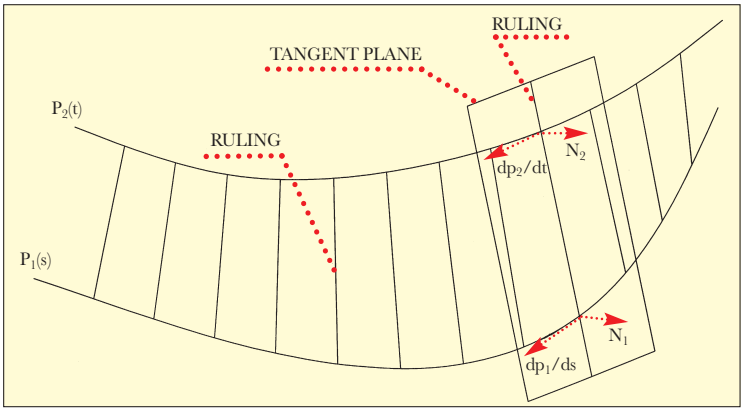

Figure 5.13 *Tangent plane of developable surface.*

For *internal* representation we decided to use the equation

$$\mathbf{x}(t,v) = \mathbf{p}(t) + v\mathbf{q}(t)\,,$$

where $\mathbf{p}$ is one of the bounding curves defined by the user. $\mathbf{q}$ is to be determined, to fit the boundary condition of a developable surface. This calculation of $\mathbf{q}$ is done in three steps:

1. Given two boundary curves, $\mathbf{p}_1(s)$ and $\mathbf{p}_2(t)$, select some arbitrary points on $\mathbf{p}_1$;

2. For each point on $\mathbf{p}_1$ find the corresponding point on $\mathbf{p}_2$, that is the point where

$$\mathbf{N}_1 \times \mathbf{N}_2 = 0. \text{ Here } \mathbf{N}_1 = (\mathbf{p}_1(s) - \mathbf{p}_2(t)) \times \frac{\mathrm{d}\mathbf{p}_1(s)}{\mathrm{d}s}, \text{ and } \mathbf{N}_2 = (\mathbf{p}_1(s) - \mathbf{p}_2(t)) \times \frac{\mathrm{d}\mathbf{p}_2(t)}{\mathrm{d}t}.$$

   An iterative solution to $\mathbf{N}_1 \times \mathbf{N}_2 = 0$, as suggested in [Nolan, 1971] and [Clements, 1981], requires the minimization of the function $\mathbf{N}_1 \times \mathbf{N}_2 = |\mathbf{N}_1|\,|\mathbf{N}_2|\,\sin(\theta)$, where θ is the warp angle. Experiments with this solution showed that this function may become very flat in the minimum region, so it proved to be difficult to find a stable and accurate solution with one of the standard minimization techniques.

   A computationally more efficient approach is based on the consideration that, to be developable,

$$\frac{\mathrm{d}\mathbf{p}_1(s)}{\mathrm{d}s}, \frac{\mathrm{d}\mathbf{p}_2(t)}{\mathrm{d}t} \text{ and } \mathbf{q} = \mathbf{p}_1(s) - \mathbf{p}_2(t)$$

   all must lie in one plane. So the aim to achieve is $\mathbf{q}.\mathbf{N}=0, \Rightarrow \mathbf{q}.(\mathbf{T}_1 \times \mathbf{T}_2) = 0$. In general the Newton-Raphson root finding procedure, based on a numerical determination of derivatives, finds the solution in a few iterations;

3. For every t and $\mathbf{p}(t)$ a corresponding $\mathbf{q}$ is found. For this set a C^2 interpolating spline $\mathbf{q}(t)$ is generated.

The geometric representation of the developable surface is not stored permanently in HREP. It has been implemented as a software function, and on demand it is calculated from the actual curve geometry. The processing speed of the algorithm is sufficiently high to enable interactive surface manipulation on a low-end PC.

With the presented scheme the major part of the surface area between the two defining polycurves is covered. However, parts at the beginning and the end of this area will remain undefined, because the extreme rulings of the surface never extend from the endpoint of one curve to the endpoint of the other curve. See e.g. Figure 5.14 where the red rulings are the limits

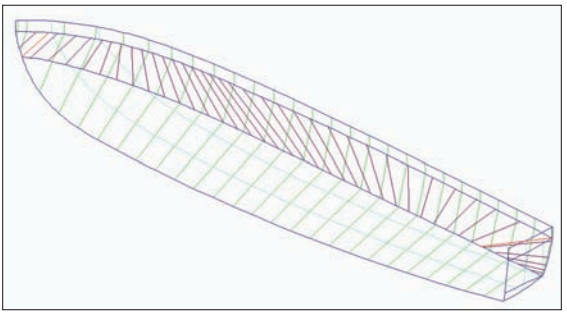

Figure 5.14 *Extreme rulings.*

of the surface determined with the method described above. To integrate the undefined areas into the developable surface two constructions can be used:

- Cylindrical: extrapolate the convolute, under the condition that

$$\mathbf{q}(t) = \frac{\mathrm{d}(\mathbf{convolute}(t))}{\mathrm{d}t}, \text{ while keeping } \mathbf{q}(t) \text{ constant;}$$

- Conical: freeze the convolute, and create a conic over the remaining boundary curve.

Both options imply a discontinuity, either in $\mathbf{q}(t)$, or in $\mathbf{p}(t)$. We voted for the second method, because it does not require an extrapolation of curves, so it is easier to implement.

5.4.3.2 *Pseudo-surfaces*

Now we reach the end of this section, the 'pseudo-surface' is still left for discussion. This entity is a pragmatic solution to pass certain geometric qualities of a single polycurve to other polycurves in the same surface area. As can be recognized in the **surface** record of Section 5.1.1, it consists of a surface area to be specified by the user, and one polycurve which serves as an example. All polycurves lying in planes parallel to that example will, as far as they lie within the specified surface area, inherit its definitions of simple curves.

This mechanism can provide shortcuts to define specific regions of a vessel's hull. For example, as sketched in Figure 5.15, a pram-type aftbody may have an increasing bilge radius. One of the ordinate curves in the bilge area is selected as *one_example_curve* in the **SURFACE** record, defined as type 'circular_2points_tangent' in the **NURBS curve** record, while the bottom tangent is defined to be slave from the neighbouring bottom frames. All other ordinate curves in that region will inherit the type and the tangent definition, and their shape will be modified accordingly.

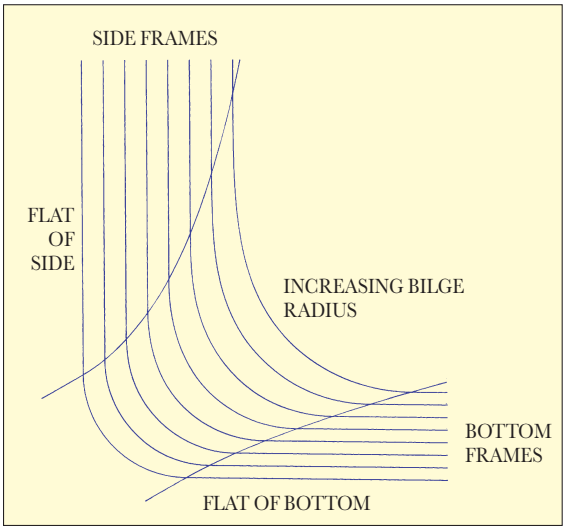

Figure 5.15 *Pseudo-surface defined by one bilge curve.*

5.4.4 Continuity considerations for surfaces

So far we have constructed transfinite patches and patch complexes, which are internally GC^2, but on the boundaries GC^1 only. Conceptually, this might be considered as a disadvantage, but on the other hand none of the system requirements specifically ask for GC^2 continuity. It must even be taken into consideration that in the neighbourhood of two intersecting GC^2 curves, the surface inherits the shape of the curves, and implicitly inherits its GC^2 character. The question whether the lack of GC^2 continuity poses a serious limitation, and when, will be addressed in Section 6.4.2.

5.5 Processing of the shape model for rapid prototyping

In Section 4.2.3 it was concluded that in the context of ship hull modelling two types of physical models are needed: Physical Concept Models, and Rapid Prototyping models.

PCM needs a low-cost and desktop solution, so for this purpose we have selected a basic three-axis NC milling machine. For RP of large scale models our attention goes to the TLOM technique, because it offers finish-free automated fabrication of large size models.

In this sub-chapter it will be described how to divide a hull form, as represented in HREP, into segments. Segmentation can be necessary because of dimensional constraints, or constraints of the applied manufacturing technology.

5.5.1 Fabrication of prototypes by three-axis milling

5.5.1.1 *Principal considerations for the application*

With this technique, milling of an arbitrary object is limited by three constraints:

- Dimensional limitations of the applied milling machine;
- The fact that the prototype is only accessible from one side;
- For easy assembly after milling, we require that the splitting planes between segments coincide with one of the three Cartesian planes. This is also required to have a fixed base plane for positioning a segment on the milling machine. The milling direction will consequently always be perpendicular to one of those planes.

Due to accuracy and mechanical considerations, we exclude repositioning of the prototype. So it will not be possible to mill one side, reposition the prototype and then mill another side.

The keyword for milling is *accessibility*, which is the possibility for the cutter to reach the prototype surface from the milling direction. If the prototype is not fully accessible, it must be divided into accessible segments. Our segmentation algorithm consists of three steps:

1. Create an accessibility map for the surface;
2. Combine areas of equal accessibility into segments;
3. Create output for each segment.

In the first step an accessibility map is created. In HREP we have a perfect data element for this purpose, and that is the *face*. For each face entity it is investigated whether it is accessible from the outside from any of the three Cartesian directions. This is performed by checking if an outward pointing vector, in the direction to investigate, through each corner of the face, intersects with any other face of the object. This step creates an accessibility map, as illustrated in Figure 5.16.

5.5.1.2 *Application of a genetic algorithm for segmentation*

The actual segmentation procedure is implemented with a numerical optimization algorithm. Furthermore, the model will only be split at existing polycurves, for the reason that we want to avoid modifications of topology and geometry *as part of* the optimization process. Because division in between the polycurves is not considered, our optimization domain is not continuous. This discrete character of the optimization problem makes it hard to handle it with traditional optimization techniques, which often require positional

continuity or even continuity of derivatives of the function to optimize. So we have selected the genetic algorithm of [Goldberg, 1989] for optimization, which does not require continuity of function or derivatives.

The genetic algorithm uses a *fitness function*, which will be evaluated during the optimization process, so we have to construct a function representing the *fitness for three-axis milling* of an arbitrary segment of a vessel's hull.

Given a milling direction and a proposed segment, this function returns two values. The first one is a boolean value (OK or NOT OK) indicating whether this segment can be milled from this direction, and the second is the fitness value, which indicates the level of applicability of this milling direction for this segment. A higher value indicates a higher level of applicability.

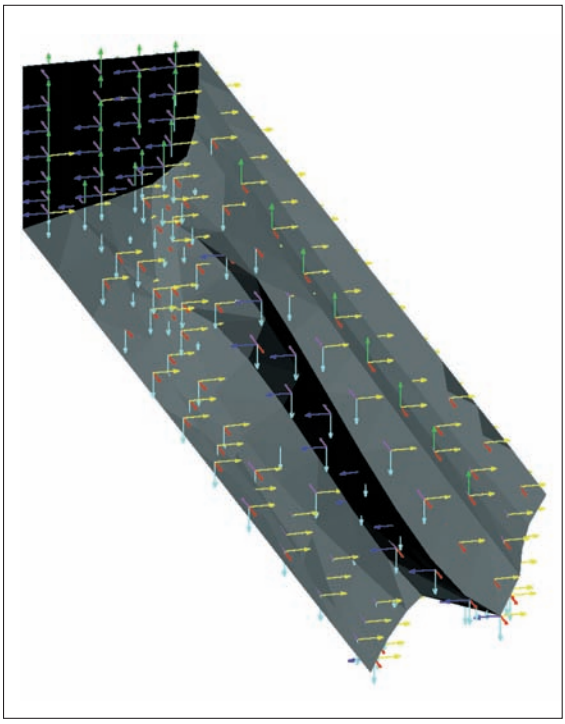

Figure 5.16 *Accessibility map.*

The boolean value is used to determine whether a segment is valid for milling. The fitness value is used in the optimization algorithm, which optimizes the segmentation with the objective to obtain that particular segmentation with the highest applicability, that means, with the maximum fitness value.

After the initialization of the fitness value to (an arbitrary) zero, the fitness function takes the following aspects into account:

- If the segment does not have a plane on the backside that is perpendicular to the considered milling direction (a plane on which the segment must rest during milling), the boolean value is set to NOT OK and the fitness value is decreased by a large number: for example one thousand. This is done to achieve that the optimization procedure, which is searching for maximum fitness, will always reject a segmentation which includes this segment;
- For each face of the segment which is not accessible from the milling direction, the fitness value is decreased by one and the boolean value is set to NOT OK;
- If one of the extreme dimensions (measured in the Cartesian directions) is less than a user-specified minimum, the boolean is set to NOT OK, and the fitness is decreased by 250. This provision is made to avoid segments with an unpractically small dimension;
- If one of the extreme dimensions is more than the maximum dimensions of the milling device, the boolean is set to NOT OK, and the fitness is decreased by 250;

- Finally, to stimulate divisions into segments of comparable size, the fitness is increased by the square root of the (dimensionless) volume of the segment.

Initially the complete vessel is evaluated by the fitness function, for all six possible milling directions. If one of these evaluations returns an OK a solution has been found without segmentation. However, in most cases the function will return NOT OK and segmentation will be necessary. As a preparation for segmentation a sorted list of *internal* polycurves (that are polycurves not lying on a boundary of the segment) is created, as well as unique identifiers to those polycurves. This list of polycurves is used later on to split the model into segments.

The function to be optimized splits the model into two segments, determines the fitness of both segments, and returns the sum of these fitnesses to the genetic algorithm. The genetic algorithm converges to a segmentation with the highest fitness value, which is a segmentation with as many accessible faces as possible, and within the maximum machine limits. If one of the segments is not valid for milling it is further sub-segmented recursively.

The algorithm has been implemented with the parameters as suggested by Goldberg: probability of mutation 0.0333, probability of crossover 0.60 and a population size of 30. With 30 generations, the algorithm performs well. It might be that with a smaller population or with fewer generations the optimum segmentation can also be found, but the processing time is so short (only a minute on a modal PC) that optimization for computation speed is not very useful.

The last step is the transfer of the geometry data of the segments to the milling machine. The first possibility is the simplest, because it works with the industry standard STL file, with its simple structure (see [Bailey, 1996]). It contains a list of coordinates of flat triangular facets, representing the surface. By using recursive subdivision of all faces present in HREP such a list can be created, and with external software based on these STL files a milling path can be generated. This step is validated by checking it with external simulation software, as illustrated in Figure 5.17.

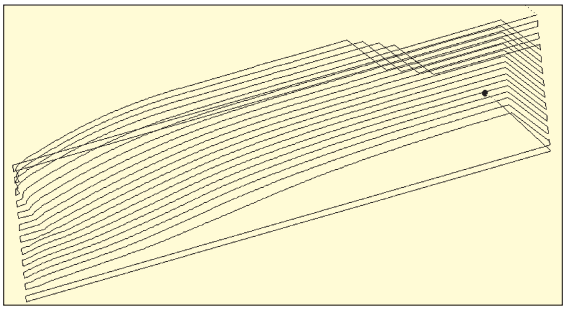

Figure 5.17 *Milling path (created with DeskProto software from Delft Spline Systems of Utrecht).*

However, by the use of the STL file format a lot of topological and geometrical information is lost, and the software receiving the STL information must perform the task to reconstruct and slice the surface.

So it seems sensible to perform slicing on basis of the HREP data. To export the slices the Common Layer Interface file format [CLI, 1994] is used. The CLI file essentially consists of a set of layers of zero order approximation. Alternatively, layers of first order approximation (ruled layers) could be generated on basis of the HREP model, but unfortunately the CLI specification does not provide for ruled layers.

The generation of the slice data is done in three steps:

1. If there is no polycurve available with *polycurve→coefficients_of_implicit_plane_equation* = slicing plane, then create such a polycurve with GENERATE_PCURVE(*slicing plane* , *polycurve*);
2. Calculate sufficient points from *polycurve*, and store them in the CLI output file;
3. If *polycurve* has just been generated then KILL_PCURVE(*polycurve*).

5.5.1.3 *Evaluation of the approach*

This segmentation process can be illustrated by a vessel with propeller tunnels, and with a bulbous bow. The bow area is restricted to access from the sides, and the tunnels to access from below, so (omitting considerations of maximum and minimum dimensions for the time being) the fore and aft body have to be milled separately. If the deck in the aft region is not completely parallel to the base plane, this region also has to be segmented into two parts, one for milling from below and one for milling from above. As shown in Figure 5.18 our algorithm automatically finds this segmentation. The milling path of Figure 5.17 concerns the SB foremost segment of this application example.

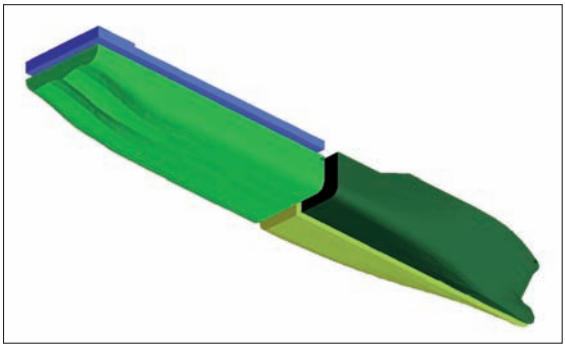

Figure 5.18 *Segmented object.*

However, the implemented segmentation algorithm does have a few drawbacks:

- Accessibility is checked in the first step only. During the segmentation process, initially inaccessible areas may become accessible, because an obstruction may be removed by segmentation. That effect is not yet accounted for.
  This disadvantage can be overcome by creating a new accessibility map in each recursive segmentation step;
- Not all hull forms can be produced in this way, because subdivision on Cartesian planes is insufficient to obtain accessibility of all parts. An example of an impossible geometry is given with a typical stern shape of an inland waterway vessel of Figure 5.19.

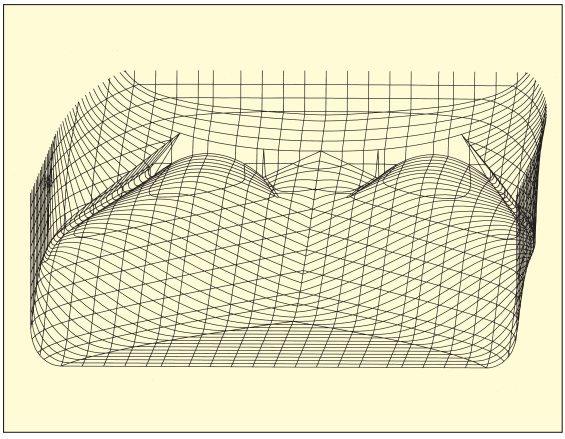

Figure 5.19 *Not possible with 3-axis milling (Courtesy of gebr. van Dijke, Oud Beyerland).*

5.5.2 Thick Layered Object Manufacture

5.5.2.1 *Decomposition strategy*

In [Broek et al, 1998] a decomposition process for TLOM is discussed. Starting with a CAD model this process comprises four phases:

- Functional decomposition, into parts which have a kinematic degree of freedom relative to each other;
- Morphological decomposition, based on considerations of shape characteristics, such as singularities or large flat regions, and the choice of advantageous segment orientation with regard to tool positioning, stacking and manufacturing efficiency;
- Geometric decomposition into thick layers by adaptive slicing;
- Technological decomposition, which is a matter of machining issues, such as tool interference or manageable dimensions.

For a monolitical hull form the aspect of *functional* decomposition is not relevant. The *morphological* decomposition of the hull form is relevant, and has to be accomplished. The problem of *geometrical* decomposition is solved in [Horváth, Vergeest and Juhász, 1998] and for the *technological* decomposition the details and dimensions of the actual apparatus must be available.

We do not intend to present the complete morphological decomposition scheme, including possible segmentation at discontinuities, and to discuss how to take into account added functionality such as hollowing and stacking. We are going to consider a simplified version, which gives preference to two constraints:

- The length of the carving blade may not be too long in order to avoid blade instability. According to [Horváth, Vergeest, Broek, Rusák and de Smit, 1998] stability is sufficient when the angle α between the top/bottom plane of a slice, and the front surface of a layer is greater that 45°;
- For easy stacking of the layers it is required that the top/bottom planes of the layers lie in one of the three Cartesian planes.

5.5.2.2 *Simplified morphological decomposition*

So just as the keyword for three-axis milling was *accessibility*, for our partial morphological segmentation for TLOM the keyword is *carvability*.

As a first step a carvability map is created. For each face element the face normal is calculated, and for each of the three Cartesian planes it is investigated whether the angle between the normal and the plane under consideration is smaller that 45°. If so, there is no danger of blade instability, and that face is marked 'carvible' for layers with the investigated orientation of the top/bottomplane.

Completely similar to the approach of Paragraph 5.5.1.2 a *fitness function for TLOM* is created. Given a plane and a segment, this function returns a boolean value indicating whether this segment can be composed of layers, which all have their top/bottomplane parallel to that plane, and a fitness value which indicates the applicability of this layer orientation for this segment.

The fitness function takes only two aspects into account:

- For each face of the segment which is not carvible with this layer orientation, the fitness value is decreased by one, and the boolean value is set to NOT OK;
- Equal to three-axis milling, to stimulate division into segments of comparable size, the fitness is increased by the square root of the (dimensionless) volume of the segment.

Just as for three-axis milling, this fitness function is used in a recursive subdivision procedure, which uses the genetic algorithm to optimize the segmentation.

5.5.2.3 *Demonstration and discussion of the decomposition*

This algorithm has been implemented, and applied to the bulbous bow of Section 6.2.3. The result of this decomposition for TLOM is shown in Figure 5.20. The aft part consists of vertical layers, the fore part of horizontal layers.

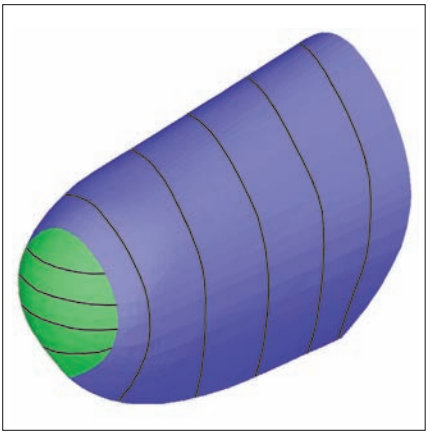

Figure 5.20 *Decomposed for TLOM.*

Our conclusions about the applicability of the proposed approach are:

- Considerations of minimum and maximum layer dimensions are aspects of the technological decomposition, and not taken into account in the proposed decomposition method. However, we could easily incorporate them in the morphological decomposition phase;
- For application on a ship hull, it is questionable whether an object should be decomposed at discontinuities, or at the edges of flat areas. For example a typical midship section (FOB – bilge – FOS) has second order discontinuities at both sides of the bilge, but with the top/bottom planes of the layers in ordinate planes, carving would be a continuous motion;
- Even segmentation at first order discontinuities (crest singularities) is only necessary when the (external) edge is less than 180° (Figure 5.21). After all discontinuities with a greater angle can be carved in multiple passes;
- Similar to segmentation for three-axis milling, this algorithm subdivides only at existing polycurves. So in general the segment thickness will not be an integer multiple of layer thicknesses;
- It appeared that the carvability criterion of 45° leads to a staggering effect; a face is carvible either for layers with one orientation, or with another orientation, there is no transition area. The relaxation to a maximum angle of 50° for the angle between the face normal and the top/bottomplane gave better results.

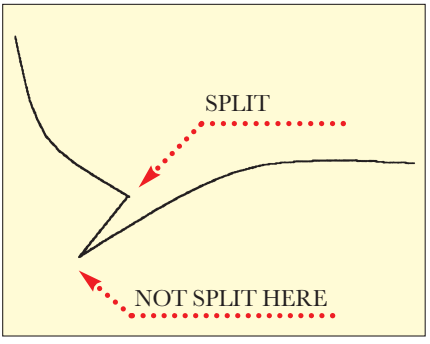

Figure 5.21 *Split at knuckle < 180°.*

5.6 Implementation of SAC support functions

According to the system design of Figure 4.1 we need additional *SAC support* and *hull form transformation* functions. In HREP the two have been combined (see Figure 5.22), in a way to fulfil the enhanced freedom requirement of Section 1.2.2.

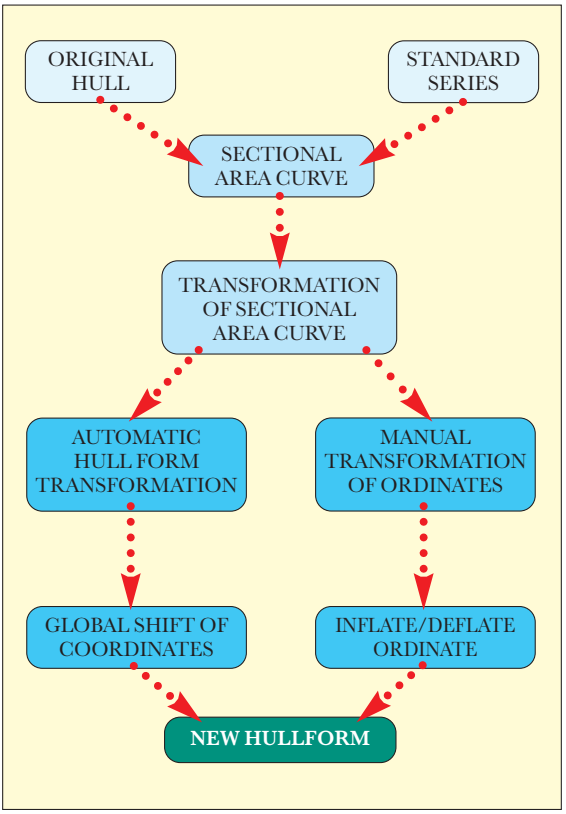

Figure 5.22 *SAC support & Hull form transformation.*

The transformation process starts with an initial SAC, which is either derived from the hull form to be transformed, or obtained from a standard series of SAC data. In HREP the SAC diagrams according to [Lap, 1954] have been included as standard series.

To obtain the required block coefficient and longitudinal centre of buoyancy, the SAC is transformed with the Lackenby-type operators of [Lackenby, 1950], where each longitudinal coordinate is shifted. Assumed that x denotes the longitudinal location (with x=–1 at APP, x=0 at midship, x=1 at FPP) we have two shifting functions:

- Shifting towards one end of the vessel: $\text{shift}_A = A \,.\, x \,.\, \sin(\pi \,.\, \text{abs}(x))$; (5.4)
- Shifting proportionally to the area of the section: $\text{shift}_B = B \,.$ sectional area, (5.5)

where A and B are scaling parameters. For each SAC transformed with a different combination of A and B a different block coefficient ($Cb = f_1(A,B)$) and longitudinal centre of buoyancy ($LCB = f_2(A,B)$) is obtained, so we have two functions

$$\begin{aligned} f_1(A,B) - Cb_{required} &= 0 \text{ and} \\ f_2(A,B) - LCB_{required} &= 0\,. \end{aligned} \tag{5.6}$$

This system of nonlinear equations can be solved with standard techniques. We have chosen the Newton-Raphson method.

The transformed SAC can be utilized in two ways:

- Manually, where a user can select an ordinate of interest, and automatically lets the ordinate area be adapted to the desired area, known from the SAC. In this automatic adaptation, in order to increase the ordinate area, the ordinate is inflated in a direction

perpendicular to the local shape of the ordinate itself. To decrease the area, the ordinate is deflated.

Of course, it is also possible for the user to completely modify the ordinate manually, by manipulation of points or vertices, in order to match the desired area;

- Completely automatically, for the complete vessel. Either each ordinate is adapted with the inflate/deflate method, similar as described for the manual transformation, or each coordinate of the hull is shifted longitudinally, according to (5.4) and (5.5), with scaling parameters A and B determined by solving system (5.6).

These mechanisms can be applied very easily with HREP, because only coordinates of the **vertex** and the **NURBS-atom** records have to be modified; the topology remains unchanged.

5.7 Design of the user interface

5.7.1 Requirements and solutions for the visual interface

According to [Hand, 1997] a visual user interface for 3-D modelling has three tasks:

- Object manipulation, such as scaling or editing the object itself;
- Viewpoint manipulation, to control the view on the object, such as zooming or point-of-view movement;
- Application control, for the communication between the user and the system.

For all these tasks communication between man and machine is necessary, and there exist many new devices for this purpose, such as data gloves, head mount displays and multi-axis joystick controllers. However, the application of these 3-D oriented devices for ship design falls beyond the scope of this thesis, so for the visual interface we stick to the 'desk-top metaphor'.

So object manipulation and viewpoint manipulation are performed with the aid of a 2-D cursor and the mouse, and application control is done with the aid of conventional pull-down menus and dialog boxes.

The implementation of the visual user interface was further led by requirements of Sub-Chapter 1.2.3. For practical reasons during development the user interface has been split in an alphanumerical part and a graphical (windows-based) part. An example of the graphical user interface for DOS is printed in Figure 5.23, and an

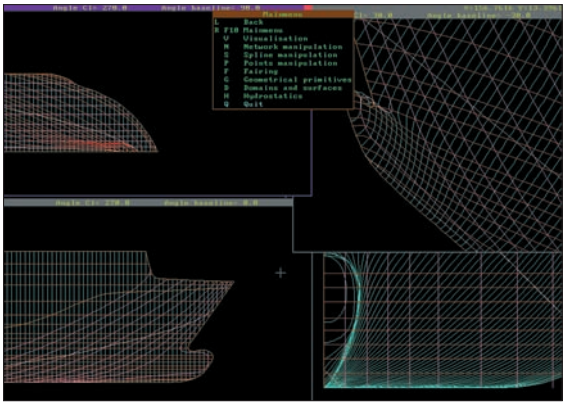

Figure 5.23 *DOS graphical user interface.*

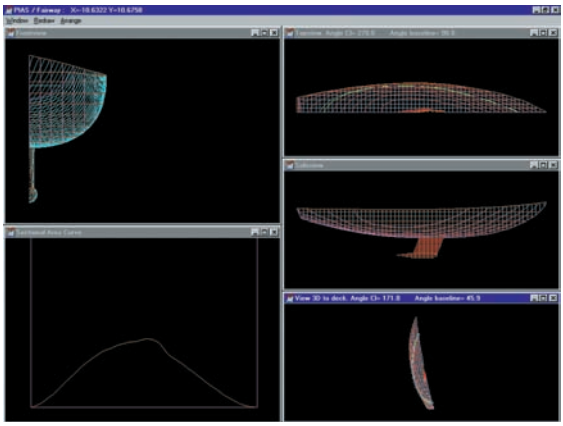

Figure 5.24 *WIN32 graphical user interface.*

example of the MS-Windows interface in fig. 5.24.

The *intuitivity* requirement implies that all actions and commands must be as clear as possible, with as little explanation as possible. In our view, for our kind of applications, this restricts the use of icons or symbols, and for each action or command an appropriate textual description is formulated, which appears on screen whenever necessary. After all, what is more clear for a *fairing* command? The simple word 'fair', or some vague small icon with a primitive sketch of something which might vaguely be connected with the fairing subject?[1].

The *versatility* requirement implies that we need *only a few*, but *powerful* functions. For example, at some stage some users of the new system came with some additional tips to increase functionality:

- One asked for a specific deck-construction function, so that he could draw deck at centerline as a straight line, while the system generates deck at side at a specific height (e.g. $1/50^{th}$ of the local breadth) below deck at centerline;
- Another user asked for a function to generate a sheer strake, at some fixed distance below deck edge;
- A third one, designing a hard-chine planing vessel, required a function for the spray rail. He wanted the vertical coordinates of the two sides of the spray rail always to be equal, even after modification of one side.

If we had implemented these additional functions, the number of functions would have exploded. When we look at the tips, we see that they are all concerned with *relations* between curves of the hull surface. For that reason we have included one general function where (general) relations between curves can be defined. These relations (not shown in the data model of Sub-Chapter 5.1, because it is an additional detail) are always maintained.

The *consistency* requirement is fulfilled with the use of a single underlying representation, where each window may give a different look of the model. When in one window a modification is made, all other windows are signalled, so that they can update their views. The net result is that a modification in one window is directly visible in all other windows.
Unfortunately, because of historical aspects of programming, this mechanism is not yet applied on the alphanumerical manipulation screen, so currently it will not be possible to

[1] We are of the opinion that the emerging use of icons in computer software, and in daily life, is a form of regression. Mankind did not develop the art of reading and writing, only to return to naive pictures in the computer age!

work with mixed graphical and alphanumerical windows. There is no fundamental barrier against mixing however.

Of course all hull form manipulations could be performed by the Euler operators and structure functions of Sub-Chapter 5.1, and by procedures for geometrical processing based on the methods of Sub-Chapters 5.2, 5.3 and 5.4. However, we were of the opinion that this level of abstraction is not appealing to the naval architect, so functions with a more practical orientation have been implemented.

Concerning naming conventions there are some differences between the terminology used in this thesis, which is more oriented towards the CAGD community, and the terminology in the user interface (and in the user manual), which is more naval architect centric[2].

5.7.2 Shape manipulation possibilities

In Chapter Four the concept of the new system was sketched, and indeed the functions as presented in Sub-Chapter 4.1 have been included in the system. On a lower level of abstraction these functions enable manipulation possibilities, from which a number will be discussed in this section. However, it must be stated clearly that neither the selection of possibilities, nor the sequence of discussion, implies a prescribed or recommended working sequence. After all *freedom* was an important system requirement.

Essentially the hull is designed, or modified, by manipulation of the curves, either graphical or alphanumerical. In the graphical mode, every modification of a curve, in one window, is immediately processed and visible in all other windows. The number, location and nature of the curves are dynamically chosen by the ship designer. Planar curves (such as ordinates, waterlines and buttocks) are actually anchored to their plane, other curves are completely free to manipulate. Some important further manipulation options are:

- The designer can choose different viewpoints; orthogonal, isometrical or perspective. Modification of points or spline control points is performed by the mouse-driven cursor, while the 3-D coordinates are determined by the projection of the new location into the plane parallel to the viewing plane (of the window where the manipulation was performed), through the original location. After each modification planar curves are projected into their plane;
- Curve manipulation is performed either by manipulation of the spline control points, or by moving a point of the curve. In the latter case the curve shape is updated by interpolation or fairing;
- Curves can be created by the intersection of the hull with a plane, or by the projection of an arbitrary space curve upon the hull. When curves are not necessary for a proper form definition, they can also be deleted;

[2] Differences of terminology are:
- The *simple curve shapes* of this thesis are called *geometrical primitives* in the user interface;
- The discussed *pseudo surfaces* are called *3-D geometrical primitives* in the user interface;
- A *polycurve* is called *line*, because a naval architect recognizes the waterline, not the waterpolycurve;
- Consequently the *curve* is called *line segment*;
- At fairing, the *mean deviation* is called *smoothing factor*.

- The topological relationships are maintained automatically, without the need or the possibility for the designer to interfere;
- Because the type of surface patch is recognized with the algorithms as described in Sub-Chapter 5.1, the proper method for the representation of the curved surfaces is automatically chosen. This aspect also requires no user interaction.

In appendix A all functions which can be selected in the graphical screen are listed. The functions are grouped into submenus, which are collected in one main menu. This distribution into submenus is only for practical reasons, otherwise the list of functions would become unmanageable.

5.7.3 Conventional output to paper

The drawing of a lines plan on paper can easily be produced by HREP. It is our experience with older hull form software that users, projects and organizations all ask for their own specific layout. To enable some freedom in this respect, the following method is applied:

- In an alphanumerical menu the user specifies which parts of the vessel must be drawn in what views;
- It is also specified on which location on the drawing (relative with respect to other parts) each part must be drawn;
- Additional text or automatic legends can be defined for each part. Legends can be defined in different coordinate systems (metric, on the basis of frame spaces, on the basis of ordinates, all with roman or arabic numerals);
- For each vessel the layout can be defined separately and is stored with the project so that a new lines plan can be drawn for each design variant very quickly.

In Chapter Six examples of lines plans are plotted.

5.7.4 Transfer of the model to CAE and general purpose CAD software

5.7.4.1 *Exchange of pure geometry*

In order to ease the design process, a variety of transfer functions has been implemented on basis of the HREP. Those functions only export geometry and no topology, because contemporary general purpose CAD software does not support the kind of BREP which forms the basis of our HREP. Merits of the exchange of a complete model (including topology and attributes) will be discussed in the next paragraph. From HREP the geometry of hull form can be exported in several ways:

- 3-D curves can be exported as NURBS to IGES and to DXF for Autocad 14+, and as piecewise linear segments to DXF for Autocad prior to Version 14;
- Face geometry (Coons patches) can be converted into IGES format. N-sided patches are decomposed in 4-sided ones with a recursive subdivision algorithm;
- To proprietary formats, such as to NUPAS-Cadmatic (a system for engineering, construction and piping) and Dawson (for potential flow calculations).

Unfortunately, export to other software may pose more problems than might be expected. This might be due to errors in receiving software (such as Autocad 12, not processing surface patches well), unstable import procedures from receiving software (such as in NUPAS-Cadmatic) or limited functionality of the data carrier (such as the inevitable choice of piecewise linear segments for DXF prior to Autocad V14).

5.7.4.2 *Product model exchange*

STEP is intended for the exchange of a complete representation of a product (see [Owen, 1997]). Part 42 contains extensive methods for transfer of geometry, topology and attributes. So it could be an option to transfer the complete HREP model by means of STEP to other software. The relevant application protocol is part 216: 'Ship moulded forms', which is currently under construction. A preliminary version ([ISO, 1995]) of this application protocol proposes the following methods for hull form representation:

- Offset points;
- Collection of planar curves;
- Wireframe representation;
- Surface representation.

In the most recent proposal ([ISO, 1997]) the number of representations has been reduced to three:

- Offset points;
- Wireframe;
- Non-manifold surface.

It is not necessary for all three representations to be supported. In practice, a selection from the allowed representations is made. Take, for instance, the ShipRight product model of [Lloyd's Register of Shipping, 1997] where the 'curve' entity (a collection of 3-D curves) was adopted, as well as the so-called 'uv-surface', which is a collection of curves spanning a regular network. The curves themselves are represented by piecewise linear segments.

To export geometry from the HREP model the non-manifold surface representation of STEP could be used because it is genuinely complete. But if data transfer also has to be accomplished with applications that do not support the non-manifold surface, the HREP has to be converted to a table of offset points and to a wireframe representation. And even within one representation a multitude of representations of underlying geometric entities can be used.

As an example, let us assume that we want to transfer geometry from HREP to the Shipright product model, then the transfer of all curves from HREP, in the NURBS format that HREP uses internally, would not suffice, because the NURBS is not supported by Shipright. It would have to be explicitly converted to piecewise linear segments.

It is recognized within STEP that because of this multitude of representations, the data transfer between two arbitrary systems may be difficult. However, by means of the 'conformance classes', systems can communicate the specific representations they support, systems can recognize incompatible data, and possibly convert it to a representation they can handle.

For a STEP interface to be useful within our system, at least a part of the allowed representations must be supported, which is quite an effort to develop. So, for the time being, an interface with STEP is not considered, but it might be a future option.

6 Application and evaluation of the system

In the previous chapters we have discussed the background and implementation of a novel system for hull form design and engineering. This chapter is of a more practical nature; it is dedicated to experiences with this system. In the first sub-chapter the implementation itself is discussed. In the second sub-chapter ship hulls designed with the system are presented, while in the third sub-chapter a hull created with surface patch capability is included. After these examples a qualitative benchmark is made, where the system is compared with the requirements of Chapter One.

The last two sub-chapters contain results of a user poll, experiences and comments of users of the system, and a discussion of the user-friendliness subject.

6.1 The Fairway software package

The system discussed in the previous chapters has been implemented into a software package called **Fairway**. Fairway is a part of the **PIAS**[1] suit of programs, which was originally developed in the mid-eighties. PIAS contains modules for ship hull definition, compartmentation, and a variety of analysis modules, such as estimation of resistance and propulsion, and calculation of intact stability, damage stability, tank calibration and longitudinal strength. PIAS also contains a module to digitize existing lines plans, a module which also can be used to digitize for Fairway.

Presently Fairway is in use by about 20 organisations.

Originally, Fairway has been implemented under the MS-DOS operating system, making use of a proprietary graphical windowing system. Because RAM demand for a complicated vessel may be a few megabytes, under MS-DOS a Protected Mode memory manager is necessary. Not counting system libraries, Fairway contains 40512 lines of Pascal code, and a few hundred lines of assembler for low-level graphical screen I/O.

A 32-bits Windows version was released in June 1999. Apart from the general benefits of 32 bits linear memory, and the Microsoft Windows API (Application Programming Interface), the Windows version offers no specific benefits.

A preliminary version of Fairway became available in 1995, and since that time approximately 150 objects have been modelled with Fairway. Included are of course many commercial cargo vessels, but also tugs, planing vessels, offshore units, semi-submersibles,

[1] PIAS is an abbreviation for **P**rogram for the **I**ntegral **A**pproach of **S**hipdesign. The program suit is developed by SARC of Bussum, the Netherlands, and is used at more than 150 desktops.

yachts, a tank hatch and a complicated funnel of a 100 m luxury yacht have been designed or engineered with Fairway.

All functions and structures which have been discussed in the fourth and fifth chapters are integrated in Fairway, including support for rapid prototyping. To support three-axis milling an optimum segmentation is created, and for each segment STL or CLI data are generated, which can be fed into (external) control software for the milling device. Also the decomposition for TLOM is included, with output in a propriety file format. Apart from this core functionality, Fairway also contains functions for shell plate development (both for developable and doubly curved shell plates) and for templates (used while forming the curved shell plate).

There are two elements which are fully available inside the Fairway program, but which are not operational:

- The creation of a topological hole, with the 'Kill face, make loop and hole' Euler function of Section 5.1.2, because appropriate functions have not yet been included in the user interface;
- The surface patch complex (the Gordon patch from Section 5.4.2) for reasons to be explained in Section 6.4.2.

6.2 Examples of actual designs of ship hulls

In this section we will discuss some actual ship hulls, designed or faired with the Fairway system. The selection of examples was driven by two considerations:

- To demonstrate many parts of the system functionality, the vessels should cover a wide range of dimensions and shape characteristics;
- To give the presentation a certain amount of objectivity, hull forms designed by, or for, a variety of companies or people must be included.

Based on these considerations, seven examples will be presented:

- The design of a 20 m schooner yacht;
- The design of an 85 m cargo vessel;
- The smoothing and shell plate development of a loose bulb (to be constructed onto bulbless bow);
- The conversion of a 142 m offshore support vessel;
- The modelling of a 31 m harbour tug;
- The modelling of a 12.50 m motor yacht, including superstructures;
- Details in fore and aftship of a 100 m cargo vessel.

6.2.1 Schooner yacht

The design of this vessel with other software was already discussed in the third chapter. The design with Fairway led to the lines plan of Figure 6.1, and consisted of the following design actions:

1. The contourline was defined, including all necessary knuckles;
2. The bulwark line was designed, by judging this line in a three-dimensional window while manipulating in two other windows showing side view and top view;

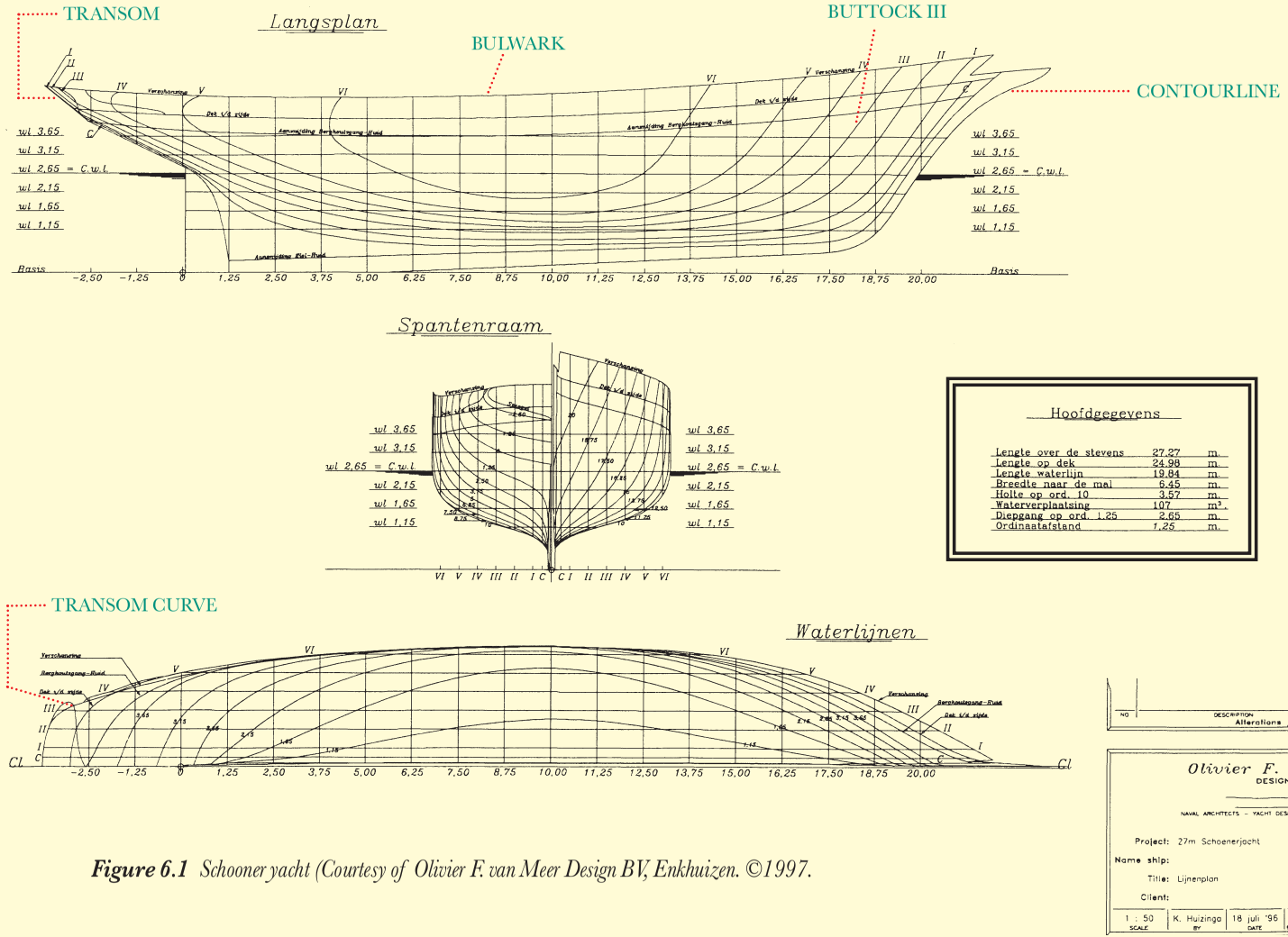

Figure 6.1 *Schooner yacht (Courtesy of Olivier F. van Meer Design BV, Enkhuizen. ©1997.*

3. A limited number of ordinates was drawn, and individually faired;
4. The transom was designed, by specifying a cylindrical transom plane, on which the heart-shaped transom curve was projected;
5. The contourline was copied to a parallel curve, in order to reflect the breadth of the keel bar. This new curve was defined as a 'knuckle' curve, so all subsequent generated curves intersecting this one automatically receive the correct knuckle information;
6. The construction waterline was generated by the system. Where necessary this curve and its neighbourhood were faired;
7. Buttock III was generated, and faired;
8. The sheer strake curve was 'woven' into the model, judged in a three-dimensional view and faired where needed;
9. More waterlines, buttocks and ordinates were generated, and faired where needed;
10. During this process the displacement was monitored. If necessary, the hull form was slightly modified to match the criterion;
11. The hull form was exported via DXF to Autocad, where the labels and notifications of the lines plan have been added.

The complete design process with Fairway took about two days.

6.2.2 Cargo vessel

In Figure 6.2 the Fairway model of the cargo vessel discussed in Section 3.3.3 is shown. This Fairway model was created by the design staff of Ferus Smit Shipyards.
The design process did comprise the following actions:

1. Because a rough sketch of the body plan was available, the first action was to digitize that sketch, and to import it into Fairway;
2. The centerline was split into multiple curves. Specific types (NURBS, straight) were assigned to those curves, and the tangents at some ends of some curves were assigned to be dependent to the tangents of neighbouring curves. For example, the curve from fr. -3 to fr. 4 is a linear one, and the top end of the subsequent curved part, above the stern bulb, has tangent continuity with that linear curve;
3. The deck edge was fixed at the correct height. This was done in an alphanumerical menu, because dealing with exact numerical values is easier to do alphanumerically than graphically;
4. Subsequently, all present ordinates were faired (with the aid of the curvature plot, and by manipulation of the control points);

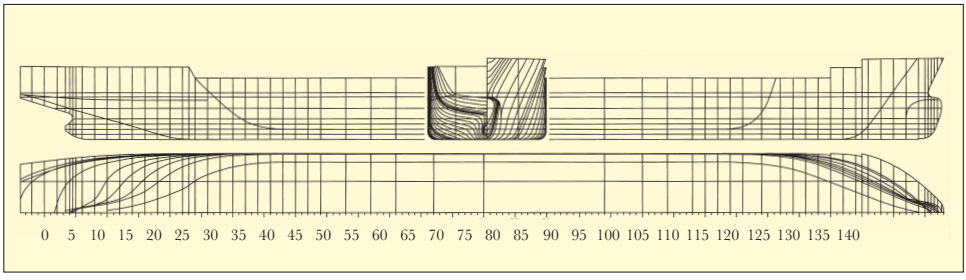

Figure 6.2 *Fairway model of cargo vessel (Courtesy of Shipyard Ferus Smit, Westerbroek).*

5. Because FOS and FOB are curves of (second order) discontinuity, they have been added to the model, simply by interconnecting the available points on the available ordinates. (Today an alternative would be the projection of this curve onto the hull surface, but this option was not available when this ship was designed);
6. At the topside of the bulb a 3-D curve was created in a similar way: by connecting points on the ordinates, which had been created at the desired locations on the ordinates in the first place. This spatial curve is used to determine the shape of the upper part of the bulb. By the way, this is a partial curve (it does not extend over the full surface), clearly visible in the area from fr. 137 to fr. 143.

   A knuckle curve in the aft ship was created in a similar way. In this way the *curves commonly used in shipbuilding* have been fixed, and it was expressed by the designer that this is a very important issue;
7. A waterline was generated at design draft, so hydrostatic properties could be calculated instantaneously. This waterline was immediately faired, which caused unfairness in some connected ordinates. Those affected ordinates were slightly modified and faired;
8. Finally more waterlines, a buttock and additional ordinates have been generated, and faired or adapted when it was necessary.

Just like in Section 3.3.3 the designer did make an estimation of the time involved:

- Two days for the initial design resulting in an accuracy sufficient for hydrostatical calculations, and for visual fairness;
- Two additional days for inclusion of all details;
- About three or four additional days for fairing up to production level.

Finally it was emphasized by the designer that **all** details can be accommodated in the Fairway model, opposite to the method of Section 3.3.3, which required manual post processing.

6.2.3 Bulbous bow

The third example is a bulbous bow, to be attached to an existing bulbless bow. A hand drawn form plan of the bulb, consisting of 5 ordinates and 6 waterlines, was digitized. In Fairway these curves were smoothed, and additional curves were generated, up to 15 ordinates, 12 waterlines and 6 buttocks. Finally, butts and seams were added to the bulb model, with the butts lying in ordinate planes, as usual. The seams lie either in waterline planes, or in some radial plane, specified by the shipyard. For the latter the plane type **oblique_planar** of the **polycurve** record of Section 5.1.1 was used.

All surfaces have been exported from Fairway to IGES, and imported into a general purpose CAD system. Shown in Figure 6.3 is an isometric view of the bulb, generated with that CAD system, with a different colour for each shell plate. The figure also shows the shell plate developments of those plates, including the projections of all present waterlines, frames and buttocks onto the plates.

Because no accurate shape information of the existing bow was available, the intersection between bulb and bow has not been determined with Fairway. The bulb was made fit on board.

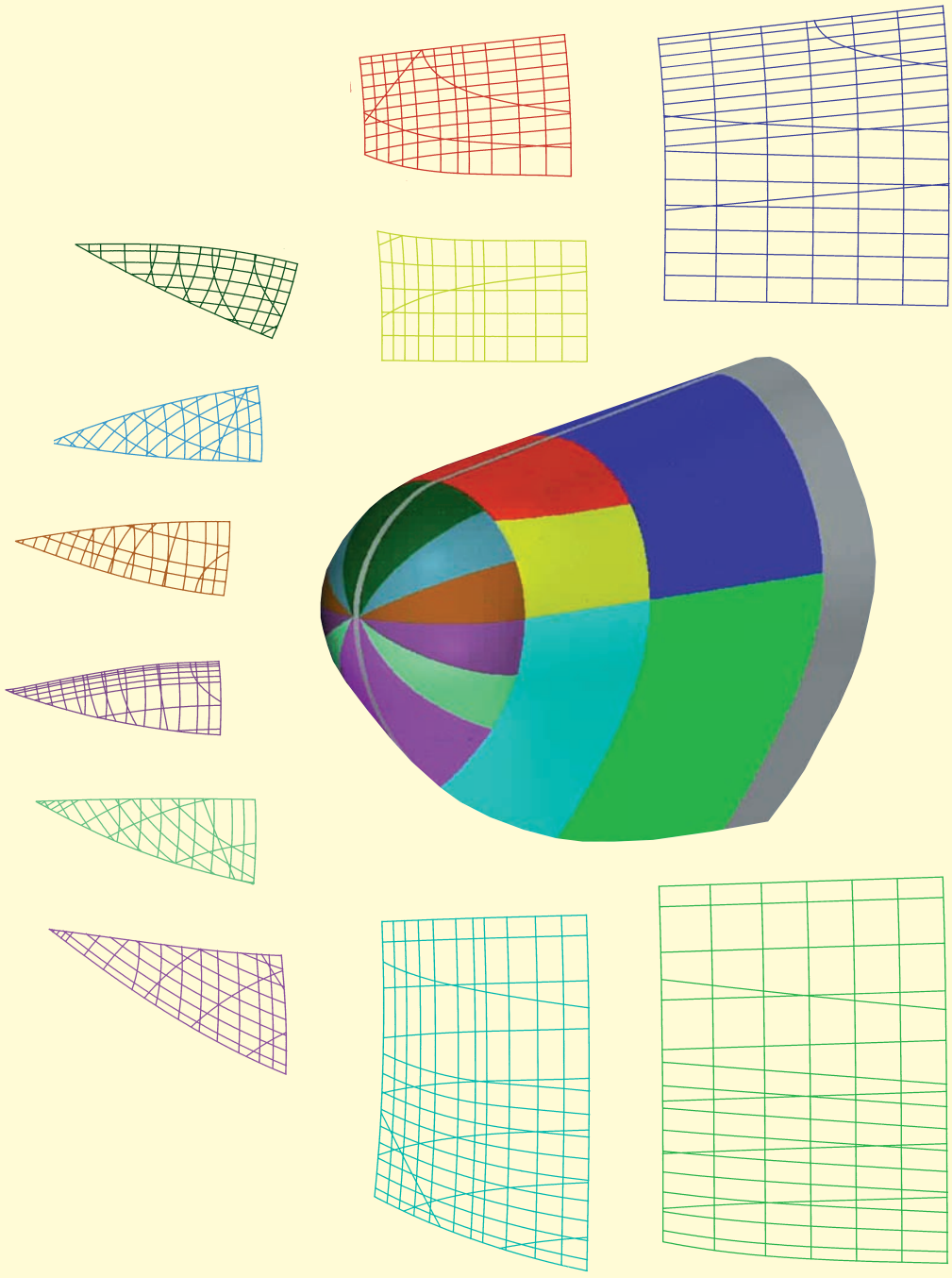

Figure 6.3 *Bulbous bow and shell plate developments (Courtesy of Visser Shipyard, Den Helder).*

6.2.4 Offshore support vessel

An arctic supplyvessel of 159 × 22.40 × 12 m was converted to offshore support vessel. The vessel was built in the Ukraine in 1987, and is characterized by its ice-breaking stem. One of the main purposes of the converted vessel will be to transport heavy deckcargo, so in order to provide the necessary deck area and transverse stability, it was decided that the vessels had to be widened to 30.40 m. For that purpose sponsons were added to the original hull. The conversion took place in 1998 in Dubai, UAE. Figure 6.4 shows the lines plan of the vessel including the newly attached sponsons. Please note the frame numbering, which, according to US convention, starts at FPP, and increases going aftwards.

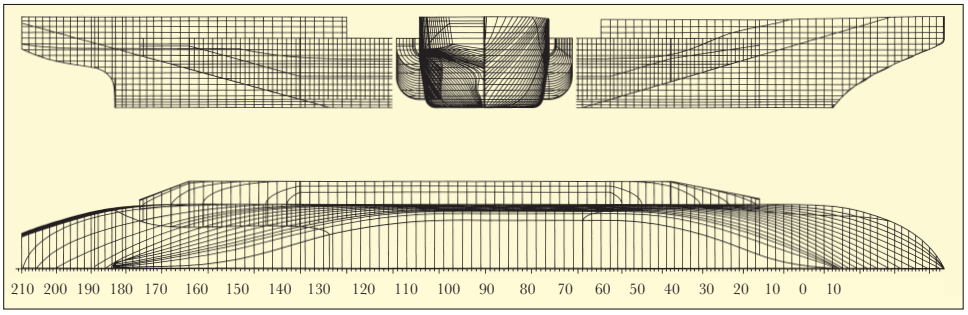

Figure 6.4 *Offshore support vessel, including new sponsons (Courtesy of DMC, Sharjah, UAE).*

To reach this result, the following design actions have been performed:

1. The ordinates of the original hull form have been digitized, from a scale 1/100 lines plan;
2. FOS, knuckle curves, bilge radius have been added, and waterlines and more transverse sections generated;
3. Intersection curves between the shell and sponsons have been constructed by generating the intersection curves between the shell and predetermined planes. The used planes were waterline planes, ordinate planes and oblique planes;
4. Starting and ending at those intersection curves, the midship section was modified to include the sponson;
5. At cross sections of discontinuity (where the waterlines show a knuckle) ordinates have been added;
6. Developable surfaces (consisting of parts of cylinders and conics) in the aft and fore regions of the sponson have been defined;
7. Finally, the hull form including sponsons has been exported to external software, for analysis of hydrostatics and stability;
8. It was reported by the shipyard that during construction it appeared that the maximum deviation between the new sponson and the original hull was 25 mm. Probably this deviation stems from the inaccuracy implied by digitizing the scale 1/100 lines plan, and possibly also from building inaccuracies of hull and sponsons.

The rendered Fairway model is shown in Figure 6.5, a photograph after conversion in Figure 6.6.

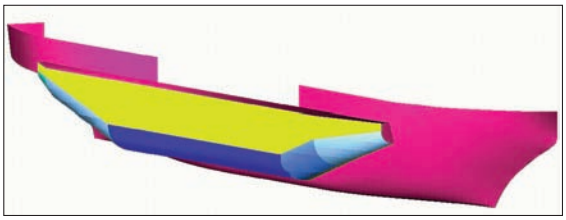

Figure 6.5 *Half-model of vessel with sponsons. The blue regions indicate the developable parts.*

Figure 6.6 *Converted vessel (Courtesy of DMC, Sharjah, UAE).*

6.2.5 Multipurpose tug

In Figure 6.7 a lines plan of a multipurpose tug is presented. This hull form is a variation of an earlier form, also modelled in Fairway. The modification consists of a small linear transformation, and a raised bottom shell at the stern. Because this last modification has a local nature only, some existing ordinates had been removed from the original model, and the three remaining ordinates were modified by means of manipulation in the windowing system. The attached buttocks were automatically updated in accordance with the ordinate modification.

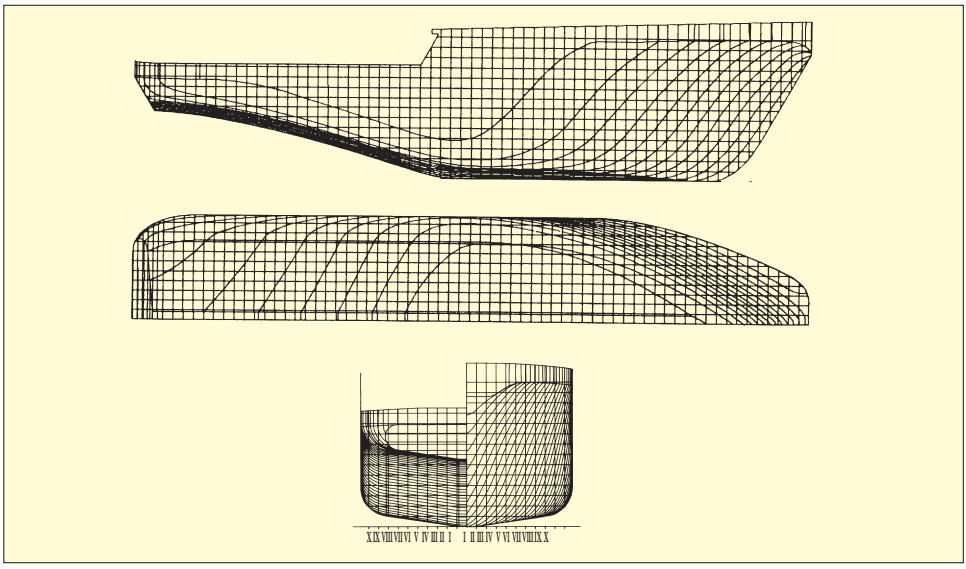

Figure 6.7 *Harbour tug (Courtesy of Wijsmuller Engineering BV, IJmuiden).*

This whole process took two man days, including smoothing, generation of construction frames and preparation of files for Autocad (to be used for construction drawing), PIAS (for hydrostatic analysis) and Dawson/RAPID (for CFD analysis). The result of one CFD calculation (made with Dawson) is shown in Figure 6.8.

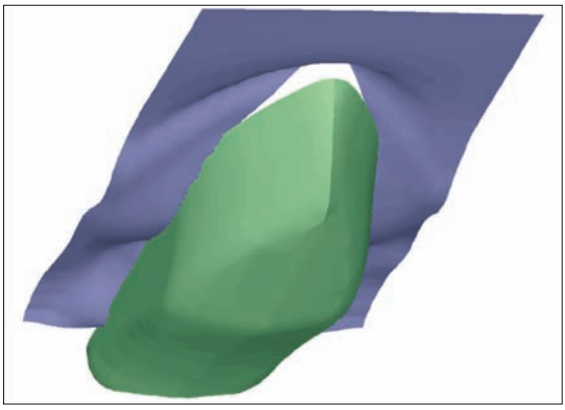

Figure 6.8 *Result of CFD calculation with Dawson.*

6.2.6 Motor yacht

This example shows the 12.50 m 'Goodvaer' motor yacht. Rather than discussing the design actions of this vessel, we present only the results. With this vessel, the designer chose to continue modelling beyond deck edge. As shown in Figure 6.9, the bulwark, deckhouse, swimming platform and even the mast have been modelled with Fairway. It shows that indeed the system offers a lot of freedom, and that shape modelling is not restricted to the hull form.

The curved faces of the model have been exported to IGES, and imported into a general CAD system, which produced the rendered picture of Figure 6.10.

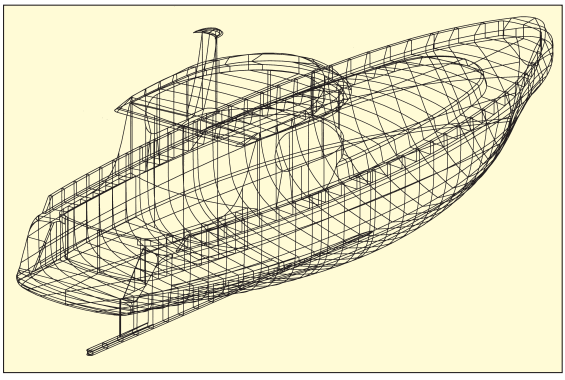

Figure 6.9 *Wireframe representation of 12.50 m. motor yacht (Courtesy of Olivier F. van Meer design BV, Enkhuizen ©1997).*

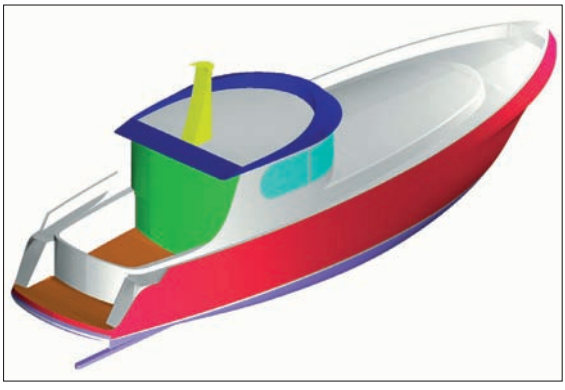

Figure 6.10 *Rendered picture of motor yacht of fig. 6.9 (Courtesy of Olivier F. van Meer design BV, Enkhuizen ©1997).*

6.2.7 Stem and stern details of cargo vessel

In Figures 6.11 and 6.12 we present the stem and stern configuration of a 99.95 × 11.93 × 5.70 m general cargo vessel, designed by de Lint of Nijmegen. This example is not presented for the design aspects of the hull form itself, which is conventional, but for the high degree of detailing.

To enable the Fairway model to be used for production, the following details have been included:

- Bow thruster tunnel, including intersection curve between hull and tunnel;
- Stern tube, including intersection curve between hull and tube;
- Rudder support, which is integrated with the hull;
- Discontinuity at deck level. Above main deck the hull is 160 mm wider than below.

Noted is that the designer used partial curves (not covering the complete hull surface) where appropriate.

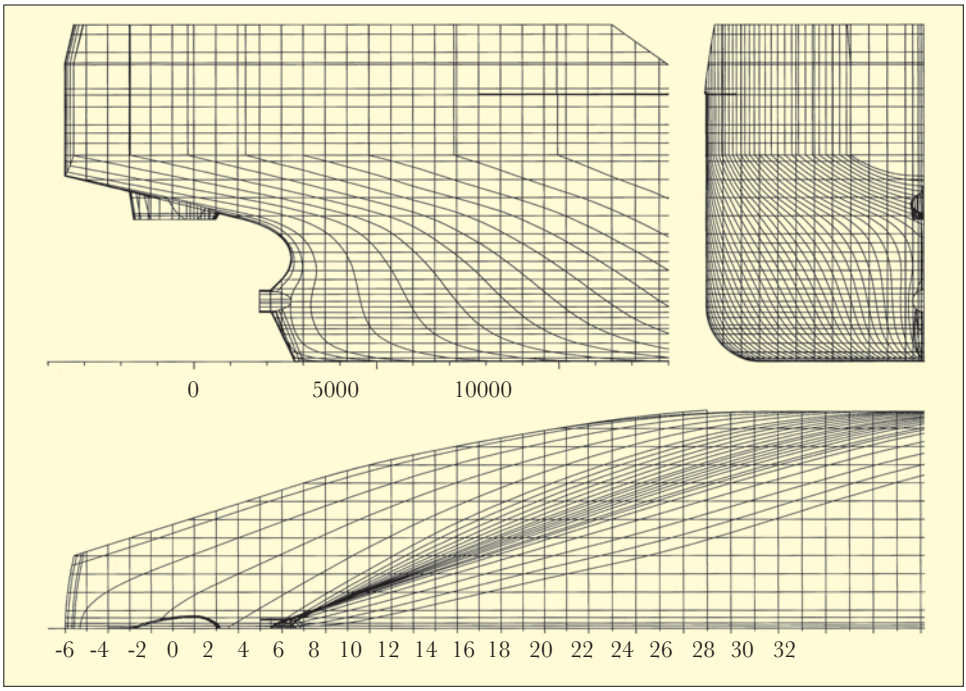

Figure 6.11 *Aftship configuration (Courtesy of E. de Lint, Nijmegen).*

6.3 Design with surface patches

All examples presented in the previous section are designed with Fairway, without the use of the surface representation such as described in Sub-Chapter 5.4. The reason is that the surface capabilities have been implemented in Fairway quite recently, and it is not yet

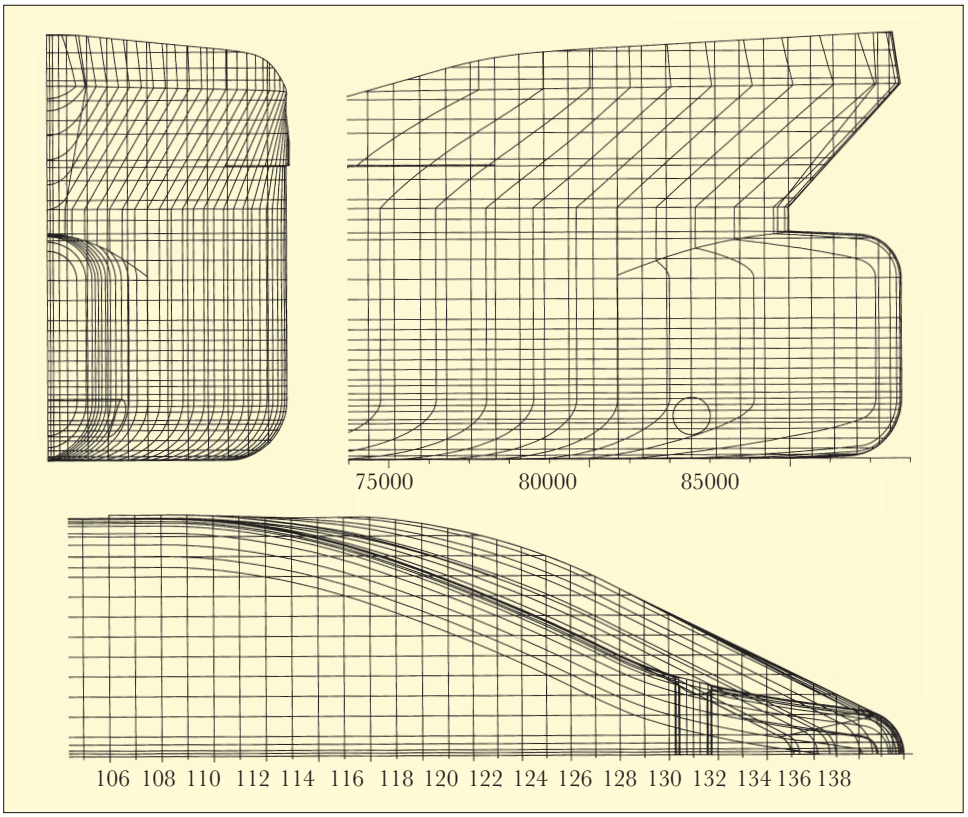

Figure 6.12 *Foreship configuration (Courtesy of E. de Lint, Nijmegen).*

released for production use to its regulars users, so no real projects could have been executed. Because surface capabilities play an important role in the design of the system, in this section we will demonstrate an example using them. This example concerns a frigate-like hull form of 96.00 × 11.50 × 6.00 m, at a draft of 3.25 m. For the sake of brevity we will concentrate on shape design, and will leave other important design considerations (such as form coefficients or stability particulars) out.

In Figure 6.13 a set of curves is shown. These are a few curves, chosen by the designer, which include the important shape aspects. Initially 9 curves are used:

- Stem / stern contour;
- Deck edge;
- Construction waterline;
- Knuckle in foreship;
- Transom;
- Four ordinates.

Two additional curves have already been included for detailing, because when these details are prepared

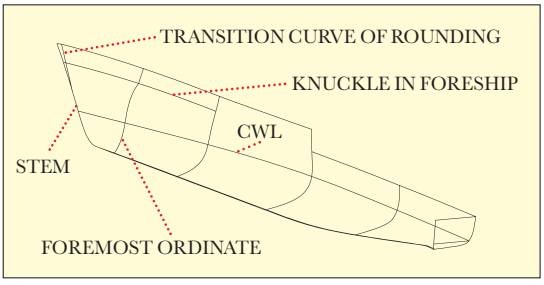

Figure 6.13 *Initial set of curves.*

from the very first start of a design, they quite naturally arise in the design process. Those prepared details are the transitions curve at the stem rounding, and the keel bar. To underline the importance of surface capabilities, in Figure 6.14 the cross sections based on the initial set of curves, *generated without using surface capabilities*, are shown first. In Figure 6.15 the result using the surface capabilities is shown. When we study this figure carefully, we see two areas with undesired shape characteristics:

- Cross sections near the transom are too wide in the middle of the 'vertical' part, and the bilge area is too flat;
- Cross sections in the vicinity of FPP have two unwanted points of inflection. They show a concave and a convex part, where only a straight part (in the lower region) and a concave part (for the flare) are required.

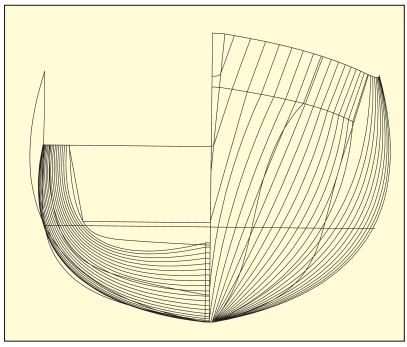

Figure 6.14 *Cross sections, generated without surface representation.*

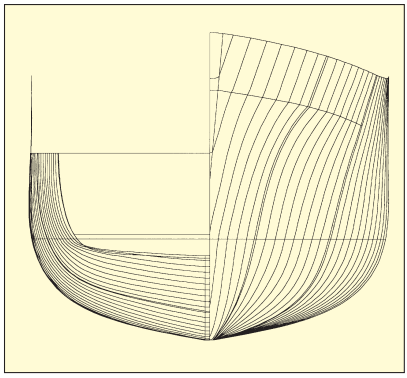

Figure 6.15 *Cross sections, based on surface representation with initial set of curves.*

These effects are not *incorrect*, because they result from a perfectly valid surface interpolation, but they are considered to be *undesired* from the designer's point of view. In order to avoid them, the set of curves was extended, see also Figure 6.16:

- To force the shape of the aftship into a more desired direction, an additional ordinate was generated, and modified to the desired shape;
- To give the sides of the ordinates near the transom more support, an additional partial waterline was generated in the aftship, and modified to the desired shape;
- The foreship undulation could have been countered with an additional waterline, midway between deck edge and CWL, but closer inspection of the set of curves shows that this region is represented by a five-sided patch. Because the transition curve of the rounding ends at the stem, somewhere between CWL and knuckle, this patch is bounded by CWL, stem, transition curve, knuckle and the foremost ordinate. It appeared that a four-sided patch gave a more desired result, so the transition curve was connected with the intersection between CWL and stem.

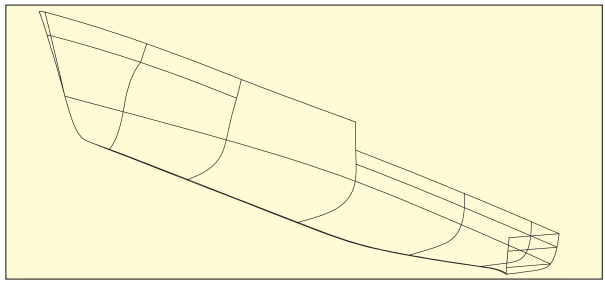

Figure 6.16 *Improved set of curves.*

The cross sections generated on basis of this improved set of curves is shown in Figure 6.17, the top and side view in Figure 6.18, and a rendered view in Figure 6.19.

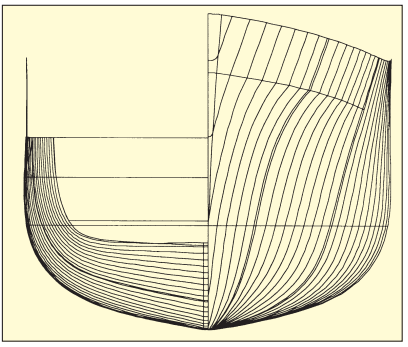

Figure 6.17 *Cross sections, based on improved set of curves.*

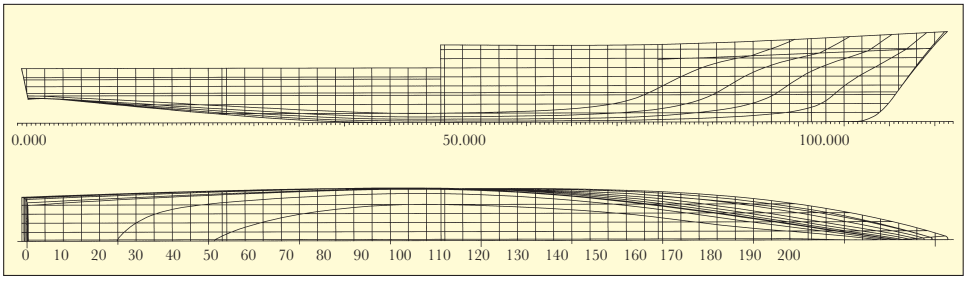

Figure 6.18 *Waterlines and buttocks.*

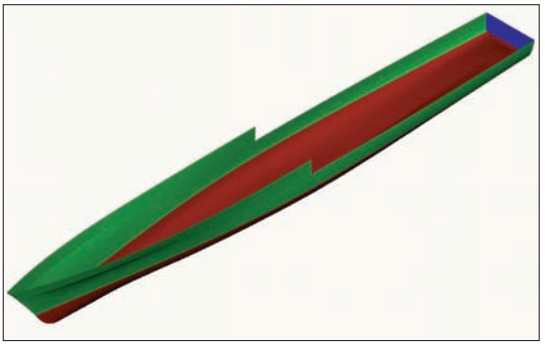

Figure 6.19 *Rendered view of completed hull form.*

6.4 Evaluation of the ship designs

6.4.1 Revisiting the requirements and goals

In this section we will compare the realized possibilities of our new system with the requirements of Section 1.2.2.

- The *draw 2-D & model 3-D* requirement is met. For example, as demonstrated with the motor yacht of Section 6.2.6, curves may be anchored in one of the orthogonal planes (such as the ordinates or the waterlines);
- The *work 3-D* requirement is also met. The motor yacht, for example, also contains spatial curves (such as curves of the deckhouse);

- The *enhanced freedom* requirement is met. If we traverse the examples mentioned at this requirement in Section 1.2.2:
  - As demonstrated by the variety of described actions, there is no prescribed working sequence;
  - Sub-Chapter 6.3 shows the ability to work with surfaces, and the examples of 6.2 show the use of curves;
  - It is not obvious from the results themselves, but during the design processes both NURBS-curve control points, and points of the curve itself have been used;
  - During the design of the yacht and the cargo vessel of Section 6.2.1 and 6.2.2 hydrostatic particulars and block coefficient were permanently monitored;
  - The tug of Section 6.2.5 was initially created by a hull form transformation.
  - The cargo vessel, the bulbous bow and the offshore support vessel have initially been imported from hand-drawn lines plans;
  - Sections 6.2.5 and 6.2.6 give examples of export capabilities to CFD and general CAD software.
- The *applicability* and *precision* requirements are met. After all, the system can be applied for initial design, as demonstrated with the cargo vessel and the frigate, and for fairing up to production precision, as demonstrated with the bulb, the tug and the detailing of the cargo vessel of Section 6.2.7;
- The *integrated data and functions for CAD and CAE* requirement can be met, as demonstrated with the plate development of the bulb (and the templates, which are not shown);
- The capabilities of *integration or data exchange with analytical software* is demonstrated with the export to software for stability and CFD calculations, as used for the offshore support vessel and the tug;
- The *stability* and the *predictability* of the system is a matter of implementation, they cannot be demonstrated by the resulting hull designs. With these aspects will be dealt in the subchapter where user experiences are discussed;
- And finally the system is *processable*, given the various examples of drawings and export.

In Section 1.2.3 three additional practical requirements are formulated which are met by the design of the computer system, but which also cannot be demonstrated by the results:
- Issues of mathematics hidden for the user as far as possible;
- Use global as well as local fairing;
- Work with a smoothing criterion with a geometrical meaning.

The goal formulated in that section, that the system must be suitable for any activity with the hull form, such as ab initio design, design modifications etcetera, is reached, as demonstrated by the variety of backgrounds of the presented examples.

Finally, we notice that in the period in which the presented examples were created, there was no prototyping machine available, so no actual scale model has been created. Therefore, PCM and RP *as methodologies supporting the design process* have not been evaluated.

6.4.2 Aspects of higher order surface continuity

As discussed, the implementation of the surface patches was only completed fairly recently, so all designs of Sub-Chapter 6.2 have been made without surface modelling. The

hull forms are created by gradual and smart addition of curves, controlled by the user. In all cases the design was started with a few curves, and gradually curves were added, at locations where good support from existing curves was available. The results show that working without surface patches works surprisingly well.

When the surface patches were implemented, we were aware of the fact that at the patch boundaries only GC^1 continuity was achieved, as discussed in Section 5.4.4. In practice the users (who were not aware of this aspect) were rather satisfied with the surface capabilities, and did not complain about any derivative continuity aspect. So as an experiment we decided to go a step further, and to disable the Gordon patch completely, without notifying the users. Notably was a performance increase, which is quite understandable because the calculation of a single point on the surface with Equation (5.3) of Section 5.4.2 requires for an $N \times M$ network the evaluation of an order of magnitude of $2 \times N \times M$ curves.

Because users did not notice any different behaviour of the system without the Gordon patches, it was concluded that *explicit* GC^2 surface continuity is not necessary for ship hull form application. Apparently the *implicit* GC^2 continuity, as discussed in Section 5.4.4 created by the GC^2 continuous curves, is sufficient.

6.5 User poll

In order to present an objective evaluation of the system all users of Fairway have been sent a questionnaire. From the 22 questionnaires, 14 have been filled in and returned. The questions to the users fall apart in two categories, one concerning a judgement of the design and implementation of the system, and the other is more involved with spontaneous reactions and tips of users. The first category will be dealt with in this sub-chapter, the second category in the next one.

6.5.1 Backgrounds of respondents

To get an idea of the experience of the users, we have asked in the first place for the number of hull forms designed or handled with Fairway. The 14 users who replied have been involved with more than 100 projects. Per user the minimum number was 1, the maximum about 25.

In the second place the users have been asked for their experience with other CAD software. The majority did only work manually, but some other systems were reported:

- Hull form transformation;
- PIAS Hull form generation;
- Legacy system of the shipyard;
- Multisurf.

Finally, the users were presented our objectives and requirements of section 1.2.2, with the question if they could be underlined. Everybody did agree with this list, and some enhancements to the requirements were proposed:

- Intuitive approach of design;

- Extension of interfacing with other CAD software. Not only import or export capabilities, but a genuine two-way traffic;
- User functions and buttons to be presented in a logical and intuitive way;
- Not only the possibility to import hull forms exactly, but a more transparent ability to design with an eye on an example shape. Proposed was a fixed overlay picture of the example shape, just as picture, not connected to the actual hull form to design;
- Compatible with other software;
- One respondent warned for the contradiction between freedom and consistency. Too much emphasis on freedom could lead to an inconsistent model, and therefore an unpredictable behaviour of the software.

6.5.2 Judgement of efficiency of Fairway

This evaluation touches the core goal of the system design. The question we posed was: 'For one or more hull designs, we kindly ask you to make a comparison for the time involved, with and without Fairway. We understand that this question is hard to answer, because the same hull form will never be designed in two different ways. Therefore we ask you to make a proper estimation of time. For example you could estimate the time involved with a previous design, using Fairway. Or, in retrospective, you could estimate the time that would have been spent in the old days to reach the result now obtained with Fairway.

One of the complicating aspects in this respect will be that Fairway is equipped with a modern windows-oriented user interface, while previously used software might have had a more cumbersome interface. An additional confusing aspect is the enhanced speed of modern hardware. We are specifically interested in the fundamental aspects, and it would not be fair to take into account efficiency differences related to interface or hardware issues.

With an eye on these considerations, we ask you to estimate the time involved with previous CAD methods as if they were running on high-speed hardware, and were equipped with a modern user interface. To give an example: SARC's old PIAS Hull form generation must be imagined with a windows-based graphical interface, which processes modifications of one view in all other windows. (By the way, if we had not entered the Fairway direction, the hull form generation software would really have worked this way).'

The respondents presented the following comparisons:

Fairway experience*	Type of vessel	Previous design method			Design with Fairway	
		Method	man-hours	Preciseness and detailing	man-hours	Preciseness and detailing
25	Roundbilge yacht	PIAS hull-form generation	80	High	32	High

Fairway experience*	Type of vessel	Previous design method			Design with Fairway	
		Method	man-hours	Preciseness and detailing	man-hours	Preciseness and detailing
25	Coaster	PIAS hull-formgeneration	40	Low	16	Medium
1	Barge	Manual	40	Low	80	High
1	Buoy handling	Proprietary CAD	100	Medium	100	High
>3	Coaster	Manual	70	Low	25	Low
>3	Tug	Manual	40	Low	25	Low
7	Tug (Developable)	Manual	60	Low	25	High
7	Addition of sponsons to hull	Manual	80	High	40	High
10	Container	PIAS hullform-generation	16	Low	2	Medium
3	Drillship	Manual	?	Medium	160	Medium
5	Inland tanker	Manual	125	Low	125	High
5	Seagoing Tanker	Manual	15	Low	20	Medium
6	Product Tanker	Manual	40	Low	16	Medium
10	Fishing Vessel	Manual	36	Medium	45	Medium/high
10	Fishing Vessel	Manual	40	Medium	70	High
8	Motor Yacht	–	–	–	50	Very high
8	Sailing Yacht	–	–	–	20	Medium

*The Fairway experience is expressed in the total number of projects a user executed with Fairway

Besides, some respondents made additional remarks:
- One respondent declared not to be able to make a realistic estimation of time. He reported a global efficiency increase for Fairway by a factor 3 or 4, compared with manual design;
- It was noted that a design with Fairway is easier to adapt that a manual one.

From the presented material we may draw the conclusion that the use of Fairway increases the efficiency of the hull design process. Some of the examples show that the users use this increased efficiency to decrease the time spent to hull design, but it is remarkable that in the majority of cases the time is not significantly less when using Fairway. In those cases the increased efficiency is used to make a more accurate (that is a more precise and/or more detailed) design.

6.5.3 Judgement of user-friendliness

The word 'user-friendly' is one of the most disabused words of this decade. Nowadays people talk about 'user-friendliness' when they have opinions about colours, sound playing capabilities, funny icons and screen gadgets. Or they are frustrated about some well considered constraint of a system, and they call the system consequently 'user-unfriendly'.

In our opinion, considering user-friendliness in general, three questions have to be answered:
1. Is the *design* of the system friendly for the user?
2. Is the user friendly for the system?
3. Is the implementation adequate?

The first question, concerning the design of the system, is a sensible one. After all, the design is the basis of a system, and a bad *system design* cannot easily be improved.

A user seldom asks himself the second question, but it is nearly as important as the first one. In a nutshell it implies that a user must be familiar with the backgrounds of the system, and must patiently study its main characteristics before printing a 'friendly' or 'unfriendly' label.

The last question is the least important, not because it is not important to the user, but because a bad *implementation* can be improved.

The three questions have been propounded to the users, and answered as summarized as follows (where the number of respondents who selected a particular choice is printed bold, and the numbers in brackets represent their Fairway experience):

	Full yes	Moderate yes	Moderate no	Full no	No reply
Is system design user-friendly ?	**6** (25,10,6,6, 10,1)	**6** (1,3,7,3,5,8)		**1** (1)	**1** (20)

	Full yes	Moderate yes	Moderate no	Full no	No reply
Do you agree that the user must be friendly for the system ?	**11** (25,1,20,1, 3,7,10,3,5, 6,6)	**1** (8)		**2** (10) (1)	
Is the user-interface user-friendly ?	**7** (25,1,7,5, 6,6,10)	**5** (1,3,10,3,8)		**1** (1)	**1** (20)

On the second question one respondent made the additional remark that the software itself should be sufficiently transparent to recognize the design and background, an aspect which he considers not completely successful in Fairway.

Overall we may conclude that the panel of users is quite positive about friendliness of system design and implementation, and it is not surprising to see that those who rank the system higher, are the more experienced users.

6.6 Experiences and comments of users

The respondents from the poll have been invited to express their remarks and desires. An anthology of these statements is printed in the next sections. Because this sub-chapter is dedicated to views of users of Fairway, it is not the intention of the author to comment each remark or tip extensively. It may be clear, however, that all tips can be implemented in Fairway as an additional function, and it might be expected that future releases will possess much of the desired functionality.

6.6.1 General remarks and views

In Sub-Section 6.4 the author has already drawn his conclusions about the system. Some specific conclusions of other users of Fairway are:

- The Lackenby hull form transformation method is to be preferred above the inflate/deflate method;
- The lines plan definition is very versatile, but confusing because of the great amount of variables;
- Export to general CAD software is often cumbersome (here the author replies that the cause of this phenomenon is the existence of the many different types of geometry representations *within* one exchange standard (DXF, IGES etc.));
- A (part of the) hull can be developable, in which case curvature in one direction is zero. However, there is no provision for a kind of 'practical' developability, which implies that a small amount of double curvature is acceptable;
- Curves with second-order boundary condition are too flexible for common use (here the author remarks that these splines are of sixth order, which is apparently too high);

- Working with N-sided patches, with N>4, sometimes requires understanding of the patch configuration and the surface modelling backgrounds (which is also demonstrated in Sub-Chapter 6.3);
- Relation between parameters of conics and the traditional conics construction is not evident (see Section 5.2.1. for this topic);
- To remain acquainted with Fairway, it is important to use it regularly;
- It is up to the user to keep consistency within the model, because with the 'fairing' functions curves of low precision could be produced, which do not intersect each other correctly. A novel user could be confused by that aspect;
- Anybody acquainted with the manual design of ship hull forms, can learn to use Fairway.

6.6.2 Tips for improvement

Many users came with a variety of specific tips to enhance the system, which are listed below in a random sequence.

- The addition of an UNDO function (was mentioned several times);
- The possibility to interactively modify a curve, by moving a single point of the curve;
- Interactive data exchange with Microstation;
- 'Lock' a curve (so that it cannot be affected anymore);
- The addition of layers, which can be switched on or off;
- Panning of an image;
- Copy and move from within the graphical menu (is now only possible alphanumerically);
- Compose lines plan graphically (is currently only possible alphanumerically, with a graphical preview);
- The possibility to connect a curve with specified boundary conditions to itself (at the other end);
- An additional function to construct a circular arc, tangent to two given curves;
- An alternative surface definition by means of interactive manipulation of NURBS surfaces;
- Rendering options, and Gaussian surface curvature representation;
- The ability to use multiple models, and boolean operations with these models;
- An additional hull form transformation method: rotation and mirroring;
- A visible system of axes;
- Directly plot on paper any view from each window;
- An autobackup facility;
- The possibility to sketch unconnected curves (with topology functionality switched off) in the first preliminary design stage, combined with automatic construction of topological relations for the later design stages;
- A menu-picker on digitizer;
- At export, the possibility to connect arbitrary text labels to entities;
- More integration between graphical and alphanumerical user interfaces;
- More expressive user manual (with more examples).

7 Conclusions and subjects for further research and development

The objectives and requirements as formulated in the first chapter form the basis of this thesis. In Chapter Three we have sufficiently demonstrated the inadequacy of traditional computer methods, and therefore in the fourth chapter a new ship hull system is conceptualized, based on CAD techniques discussed in Chapter Two. The translation of this concept into a practically functional computer system (named **Fairway**) is described in Chapter Five.

Because the system grew goal-directed from requirements via concept to implementation, a *formal* confrontation of Fairway with the requirements would be superfluous. In the sixth chapter it was demonstrated by a variety of hull form designs, produced by different designers, that the system suits the requirements very well. Based on estimations of a panel of users we concluded that Fairway is much more efficient than previously used methods. The use of Fairway resulted either in saving of time, or in a higher level of precision or detailing.

The respondents in the panel also came up with valuable advices for enhancements of the system, which gives us, combined with our own ideas, some work to do in the years to come. It would be subjective to appraise the importance of the enhancements; the sequence of implementation is hard to predict, partly because it also depends on the number of times the users of Fairway remind us of their particular wishes. Not counting simple improvements and additional user-interface functions, it might be expected that our future attention will focus on:

- A less rigid definition of developable surfaces. Currently the surfaces are determined by the theory of geometry, which produces a surface with curvature in one direction only. However, practice is stronger than doctrine, and in general a common designer is not interested in plates of exact mono-curvature, but in a kind of multi-curved plates which behave in practice just as developable ones. An idea would be to specify an allowable warp angle per metre ruling, and use that constraint to determine the shape of the hull plate;

- The implementation of an undo function;

- Decide on a method to convert from the classical definition of a conic to a NURBS-based one (and reverse), and implement it (See Section 5.2.1);

- Interactive data exchange with the outside world (so that Fairway works as a 'hull server', and generates sections and surfaces for application in general CAD systems, the so-called 'hull-clients');

- The inclusion of an additional elementary curve element: the circular arc which is tangent to two given curves (for roundings);

- A facility to import a hull form model in foreign format, e.g. in DXF, IGES or proprietary ASCII tables of offset. Application can be thought in the field of design (for example importing externally faired hull lines, to serve as a parent hull in transformation) and surface re-engineering (e.g. reconstruction of hull form, by photogrammetry). However, one problem to overcome is the recognition of topology. If the topology reconstruction aspects are solved, it would also be possible to initiate a design with loose curves, and convert it to a complete model in a later design stage;

- Additional surface definition, e.g. free to manipulate NURBS surfaces. This is not contradictory to our objections to the use of single-patch NURBS surfaces for ship hull modelling, because within Fairway they would be embedded in a complete geometric model. Another interesting field to explore would be surfaces modelled with fuzzy geometry;

- The use of multiple hull models simultaneously within Fairway, and boolean operation on these models.

Appendix A

List of functions in the visual interface of Fairway

This appendix lists the functions which can be activated from the visual interface of Fairway. The used terminology in the program differs in some occasions from the terminology of this thesis. Differences are listed in the footnote of Section 5.7.1.

All functions are grouped into submenus, which themselves are grouped into one main menu. The complete list of functions is:

Main menu

V	**V**isualisation	go to the visualisation submenu
N	**N**etwork manipulation	go to the network manipulation submenu
S	**S**pline manipulation	go to the spline manipulation submenu
P	**P**oints manipulation	go to the points manipulation submenu
F	**F**airing	go to the fairing submenu
G	**G**eometric primitives	go to the geometric primitives submenu
D	**D**omains and surfaces	go to the domains and surfaces submenu
H	**H**ydrostatics	go to the hydrostatics submenu

Submenu visualisation

3	Rotate **3**-D	rotate the 3-D view
N	Perpendicular (**N**ormal) view	normal view on a spot on the hull form
L	View on **L**ine	view on selected line type
P	**P**erspective parameters	go to the perspective parameters submenu
2	Zoom out (**2**)	zoom out by a factor 2
Z	**Z**oom window	zoom in
B	Zoom **B**ack	zoom back
M	**M**ark	mark point
S	**S**how special points	show special points
1	**1** line visible	make 1 line visible
V	Window **V**isible	all lines in selection box will be visible
I	Window **I**nvisible	all lines in selection box will be invisible
A	**A**ll lines visible	all lines will be visible

Submenu perspective parameters

Specify the longitudinal coordinate of object point
Specify the transverse coordinate of object point
Specify the vertical coordinate of object point
Specify the distance from eye to object point

Submenu network manipulation

3	**3**-Dimensional projection	go to the 3-dimensional projection submenu
P	**P**oints of the network	show points of the network (on or off)
A	**A**ctive for connection	activate a specific line for making connections
N	**N**ew line	add new (3-D) line and make it active
C	**C**onnect	connect active line with a point
D	**D**isconnect	disconnect point from active line
R	**R**emove line	remove a complete line
I	**I**nsert line	insert line (in plane to be specified)
M	**M**erge two lines	merge two lines
S	**S**plit line	split a line into two lines
W	**W**hich line ?	list name and properties of a line

Submenu 3-dimensional projection

Create auxiliary line
Choose the projection type (parallel or point projection)
Specify the longitudinal component of the direction of projection
Specify the transverse component of the direction of projection
Specify the vertical component of the direction of projection
Give the name of the auxiliary line
Project auxiliary line onto ship hull

Submenu spline manipulation

S	**S**elect segment	select a line segment for processing
P	**P**oints	show points of selected line (on or off)
V	**V**ertices	show vertices of selected line (on or off)
C	**C**urvature	show curvature (on or off)
+	Scale of curvature **+**	distance between curvature and line 2× greater
–	Scale of curvature **–**	distance between curvature and line 2× smaller
X	e**X**change vertex	move a vertex
R	**R**emove vertex	remove a vertex
I	**I**nsert vertex	add vertex, without changing shape of the curve
N	**N**ew vertex	add vertex at the position of the cursor

Submenu points manipulation

C	**C**onnections	show connections between points
D	**D**elete internal points	delete internal points of a line segment
X	e**X**change point	move point in space
M	**M**ove point	move point along its line
S	**S**wap point	swap two points
R	**R**emove	remove a point
I	**I**nsert	insert a point
K	**K**nuckle on/off	make a knuckle of a point (on or off)

Submenu fairing

S	**S**moothing factor	show and/or change smoothing factor of selected line
W	**W**eight factors	show weight factors of selected line (on or off)
X	e**X**change 1 weight factor	modify the weight factor of 1 point
M	**M**odify weights in window	modify the weight factors in the selection box
F	**F**air	fair the selected line
I	**I**nterpolate	interpolate a line through all points
E	**E**xtract	extract the selected line
P	**P**rocess all points	process modifications of a line in all connected lines
1	Process **1** point	process modifications of a line in one connected line
+	Increase weight (**+**)	double the weight factor of a point and fair the line
–	Decrease weight (**–**)	halve the weight factor of a point and fair the line

Submenu geometric primitives

M	Choose **M**aster	choose master of selected line
F	**F**ree slave from master	remove relation between master and slave
D	**D**efine master-slave relation	define the relation between master and slave
L	**L**ine type	choose line type for selected segment
I	**I**nput radius or shape factor	input of radius or shape factor of line
R	Adapt **R**adius or shape factor	modify radius or shape factor of line
X	e**X**change tangent	modify tangent at one line end
T	**T**angent dependency	define boundary constraint of tangent at one line end
C	**C**urvature dependency	define boundary constraint of curvature at one line end

Submenu domains

C	**C**lear border lines	clear border lines of the selected domain
A	**A**dd border lines	add border lines to the selected domain
S	**S**elect developable surface	select a developable surface for display
N	**N**umber of rulings	give the number of rulings to display
P	**P**rocess complete surface	project all lines to the developable surface

Submenu hydrostatics

H	Concise **H**ydrostatics	calculate brief hydrostatics
E	**E**xtended hydrostatics	calculate extended hydrostatics
A	Ordinate or SAC **A**rea	display ordinate area or SAC area and centre of buoyancy
F	**F**it ordinate or SAC area	fit ordinate or SAC area (fit to the desired area)
S	**S**how envelop lines	display envelop lines and desired area

Appendix B

Alphabetical list of commercial naval architectural software mentioned in this thesis

Autoship
Autoship Systems Corporation, Vancouver, Canada

Defcar
Defcar Ingenieros, Madrid, Spain

FastShip
Proteus Engineering, Stevensville, Maryland, USA

FORAN
SENERMAR, Madrid, Spain

L/GRAND
Logos, Delft, the Netherlands

Macsurf / Maxsurf
Formation Design Systems, Freemantle, Australia

Mastership
Yachting Consult, Eindhoven, The Netherlands

Multisurf
Aerohydro, Southwest Harbor, Maine, USA

NAPA
Napa Oy, Helsinki, Finland

Nauship
Nauticad, Marina di Carrara, Italy

NUPAS
Numeriek Centrum Groningen (NCG), Groningen, The Netherlands

PIAS
Scheepsbouwkundig Advies en Reken Centrum (SARC), Bussum, The Netherlands

Shipshape
Wolfson Unit, Southampton, UK

Seasafe
Seasafe, Sollentuna, Sweden

Tribon
Kockums Computer Systems, Malmö, Sweden

Glossary

APP	After Perpendicular.
Bilge	Joint between bottom and side, often a circular arc.
Bilge radius	Radius of circular arc in the bilge.
BREP	Boundary REPresentation.
Butt	A transverse boundary of the steel plates of the vessel's shell.
Buttock	Intersection of the hull with a vertical plane parallel to the centerline of the vessel.
CAD	Computer-Aided Design: design supported by computer software.
CAE	Computer-Aided Engineering: production-related modelling or calculations, supported by computer software. Results in printed output, drawings, or computer files.
CAM	Computer-Aided Manufacturing: fabrication process, supported by computer. Results in tactile products.
Cb	Block coefficient = Underwater volume / (length × breadth × draft).
CEM	Concept Exploration Model.
CFD	Computational Fluid Dynamics.
Chine	A curve where crossing lines do have a knuckle.
CL	Centerline. A vertical plane at zero breadth.
Cm	Midship section coefficient = Immersed area of midship section / (breadth × draft).
Cp	Prismatic coefficient, Cp = Cb / Cm.
Cross section	Intersection of the hull with a vertical, transverse plane.
Deadweight	Weight carrying capacity.
Diagonal	Intersection of the hull with a plane normal to cross section planes, but inclined with respect to the baseplane and the centerline.
Displacement	Mass of the vessel (underwater volume multiplied by specific gravity of seawater).
EPM	Expert Parametric System.
First principle	Used in 'first principle calculations': calculations based on a genuine physical model of the phenomenon (contrary to empirical calculations).
FOB	Flat Of Bottom curve, the transition curve between the curved shell, and the bottom plane.
FOS	Flat Of Side curve, the transition curve between the curved shell, and the planar side shell.

FPP	Forward Perpendicular.
Frame	In general a construction frame. In this thesis used as a synonym for *ordinate*, *station* or *cross section*.
GA	Genetic Algorithm.
GC^X	Geometric continuity of the X-th order. A GC^1 curve has tangent continuity, and a GC^2 curve curvature continuity.
Hull form coefficients	Coefficients of geometrical properties of the (underwater part of the) hull. For instance Cb, LCB, Cm etcetera.
LCB	Longitudinal Centre of Buoyancy. The longitudinal position of the centre of gravity of the underwater body.
Lightweight	The weight of the empty vessel.
Lpp	Length between perpendiculars.
Loa	Length over all.
Midship	Location at Lpp / 2.
Moulded hull	Hull form, without shell plates.
Ordinate	In general one of the 21 cross sections used in preliminary ship design. In this thesis used as a synonym for *frame*, *station* or *cross section*.
Parallel body	The amidship portion of a ship in which the shape of the sections remains unchanged.
Parent form	At hull form transformation, the form which is varied to create the new form.
PCM	Physical Concept Modelling.
RP	Rapid Prototyping.
SAC	Sectional Area Curve, the curve which represents the longitudinal distribution of cross sectional area below the CWL.
Seam	A longitudinal boundary of the steel plates of the vessel's shell.
Shell (1)	In naval architectural sense: the (steel, wood, GRP or otherwise) outer skin of the vessel.
Shell (2)	In CAD parlance: a topological entity which bounds a solid object.
Shell plate development	Rubber transformation of curved shell plate from 3-D space to the plane.
TLOM	Thick Layered Object Manufacturing.
Waterline (1)	Intersection of the hull with a horizontal plane.
Waterline (2)	The single waterline (1) at design draft. Also called Construction Water Line (CWL).

References

Abrams, S.L., Bardis, L., Chryssostomidis, C., Patrikalakis, N.M., Tuohy, S.T., Wolter, F.E. and Zhou, J., (1995), 'The geometric modeling and interrogation system praxiteles'. *Journal of Ship Production*, Vol. 11, No. 2, May, pp. 117-132.

Alef, W.E. and Collatz, G., (1976), 'Computer aided design of ship's lines by nonlinear distortion of parent forms', in: Jacobsson et al., editors. Proceedings of *Computer Application in the Automation of Shipyard Operation and Ship Design*, June 8-11, Gothenburg, Sweden.

Alfeld, P. (1989) 'Scattered data interpolation in three or more variables', in: Lych, T. and Schumaker, L.L., editors. *Mathematical Methods in Computer Aided Geometric design*, Academic Press, San Diego, USA, pp. 1-33.

Anantha Subramanian, V. and Suchithran, P.R. (1999) 'Interactive curve fairing and biquintic surface generation for ship design'. *International Shipbuilding Progress*, Vol. 46, No. 446, July, pp. 189-208.

Bailey, M. (1996) 'The use of solid rapid prototyping in computer graphics and scientific visualization'. Web *http://www.sdsc.edu/~mjb*.

Bardis, L. and Vafiadou, M. (1992) 'Ship-hull geometry representation with B-spline surface patches'. *Computer-Aided Design*, Vol. 24, No. 4, April, pp. 217-222.

Barnhill, R.E. (1977) 'Representation and approximation of surfaces', in: Rice, J.R., editor. Proceedings of the *Mathematical Software III symposium*, Madison, USA, March 28-30, pp. 69-120.

Baumgart, B.G. (1974) '*GEOMED – A Geometric Editor*'. Stanford artificial intelligence laboratory, Computer science department report no CS-414.

Birmingham, R.W. and Smith, T.A.G. (1998) 'Automatic hull form generation: a practical tool for design and research', in: Proceedings of *Practical Design of Ships and Mobile Units*, September 20-25, The Hague, The Netherlands.

Boor, C. De (1978) '*A Practical Guide to Splines*'. Springer-Verlag New York.

Broek, J.J., Horváth, I., de Smit, B., Lennings, A.F. and Vergeest, J. 'Aspects of shape decomposition for thick layered object manufacturing of large sized prototypes', in: Proceedings of the *7èmes Assises Européennes du prot. rap.*, Paris, November 19-20.

Charrot, P. and Gregory, J.A. (1984) 'A pentagonal surface patch for computer aided geometric design'. *Computer Aided Geometric Design*, No. 1, pp. 87-94.

Chiyokura, H., Takamura, T., Konno, K. and Harada, T. (1991) 'G1 surface interpolation over irregular meshes with rational curves', in: G. Farin, editor. *NURBS for Curve and Surface Design*, SIAM, Philadelphia, USA, pp. 15-34.

Chomsky, N. (1976) '*Reflections on Language*'. Pantheon books, New York.

Clements, J.C. (1981) 'A computer system to derive developable hull surfaces and tables of offsets'. *Marine Technology*, Vol. 18, No. 3, July, pp. 227-233.

CLI V2.0 (1994). 'Common Layer Interface version 2.0 specification'. Web *http://www.cranfield.ac.uk*.
Coastdesign (1992). '*Marine Design Software*'. Commercial brochure of Coastdesign UK, July.
Coons, S.A. (1974) 'Surface patches and B-spline curves', in: Barnhill, R.E. and Riesenfeld, R.F., editors. *Computer Aided Geometric Design*, March 18-21, Utah, USA, pp. 1-16.

Delbourgo, R. and Gregory, J.A. (1983) 'C^2 rational quadratic spline interpolation to monotonic data'. *IMA Journal of Numerical Analysis*, No. 3, pp.141-152.
Dierckx, P. (1982) 'Algorithms for smoothing data with periodic and parametric splines'. *Computer Graphics and Image Processing* 20, pp. 171-184.
Dierckx, P. (1993) '*Curve and Surface Fitting with Splines*'. Oxford University Press, Oxford, UK.
Dijk, C.G.C. van (1994) '*Interactive Modeling of Transfinite Surfaces with Sketched Design Curves*'. Ph.D. thesis, Delft University Press, Delft, The Netherlands.
Dolenc, A. and Mäkelä, I. (1994) 'Slicing procedures for layered manufacturing techniques'. *Computer-Aided Design* Vol. 26, No. 2, February.
Dong, J. (1998) '*Rapid Response Manufacturing*'. Chapman & Hall, London, UK.

Earnshaw, J.L. and Yuille, I.M. (1971) 'A method of fitting parametric equations for curves and surfaces to sets of points defining them approximately'. *Computer-Aided Design*, winter 1971, pp. 19-22.

Farin, G., Rein, G., Sapidis, N. and Worsey, A.J. (1987) 'Fairing cubic B-spline curves'. *Computer Aided Geometric Design* 4, pp. 91-103.
Farin, G., (1990) 'C*urves and Surfaces for Computer Aided Geometric Design*'. Academic Press, San Diego, USA.
Faux, I.D. and Pratt, M.J. (1979) '*Computational Geometry for Design and Manufacture*'. Ellis Horwood Ltd, Chichester, UK.
Fenqiang Lin and Hewitt, W.T. (1994) 'Expressing Coons-Gordon surfaces as NURBS'. *Computer-Aided Design*, Vol. 26, No. 2, February.
Ferguson, E.S. (1992) '*Engineering and the Mind's Eye*'. MIT press, Cambridge, Massachusetts, USA.
Fog, N.G. (1984) 'Creative definition and fairing of ship hulls using a B-spline surface'. *Computer-Aided Design*, Vol. 16, No. 4, July.

Gentleman, W.M. (1973) 'Least squares computations by Givens transformations without square roots'. *J. Inst. Maths Applics* 12, pp. 329-336.
Gertler, M. (1954) 'A Reanalysis of the Original Test Data for the Taylor Standard Series', *DTMB report 806*.
Ginnis, A. and Wahl, S. (1998) 'Benchmark results in the area of curve fairing', in Nowacki, H. and Kaklis, P.D., editors. *Creating Fair and Shape-Preserving Curves and Surfaces*, B.G. Teubner, Stuttgart-Leipzig, Germany, pp. 29-54.
Goldberg, D.E. (1989) '*Genetic Algorithms in Search, Optimization and Machine Learning*'. Addison Wesley, Reading, USA.
Gomes, A. and Middleditch, A. (1997) 'Synthesis of a unified approach to shape

modeling', in: Strasser, W., Klein, R. and Rau, R., editors. *Geometric Modeling: Theory and Practice*, Springer-Verlag, Berlin, Germany, pp. 226-246.
Gordon, W.J. (1969) 'Spline-blended surface interpolation through curve networks'. *Journal of Mathematics and Mechanics*, Vol. 18, No. 10, pp 931-951.
Gregory, J.A. (1982) 'C1 rectangular and non-rectangular surface patches', in: Boehm, W. and Hoschek, J., editors. *Proceedings of Surfaces in Computer Aided Geometric Design*, April 25-30, Oberwolfach, Germany, pp. 25-34.
Gregory, J.A. (1984) 'N-sided surface patches', in: Gregory, J.A., editor. Proceedings of *The Mathematics of Surfaces*, Manchester, September 17-19, pp. 217-232.
Gregory, J.A. (1989) 'Geometric continuity', in: Lyche,T. and Schumaker, L.L., editors. *Mathematical Methods in CAGD*, Academic Press, San Diego, USA, pp. 353-372.
Gregory, J.A., Hahn, J.M. (1989) 'A C2 polygonal surface patch'. *Computer-Aided Design*, No. 6, pp. 69-75.
Gregory, J.A., Lau, V.K.H. and Zhou, J. (1989) 'Smooth parametric surfaces and N-sided patches', in: Dahmen, W., Gasca, M, and Micchelli, C.A., editors. Proceedings of *Computation of Curves and Surfaces*, July 10-21, Tenerife, Spain, pp. 457-498.
Gregory, J.A., Lau, V.K.H. and Hahn, J.M. (1993) 'High order continuous polygonal patches', in: Farin, G., Hagen, H. and Noltemeier H., editors. *Geometric Modeling*, Springer-Verlag, Wien, Austria, pp. 117-132.
Groot, D.J. de (1977) 'Designing curved surfaces with analytical functions'. *Computer-Aided Design*, Vol. 9, No. 1, January.
Guldhammer, H.E. (1969) '*Formdata IV* '. Danish Technical Press, Denmark.

Hagen, H. and Schulze, G. (1987) 'Automatic smoothing with geometric surface patches'. *Computer Aided Geometric Design* 4, pp. 231-235.
Hahn, J. (1989a) 'Filling polygonal holes with rectangular patches', in: Strasser, W and Seidel H.P., editors. *Theory & Practice of Geometric Modeling*, Springer-Verlag, Heidelberg, Germany, pp. 81-92.
Hahn, J.M. (1989b) 'Geometric continuous patch complexes'. *Computer-Aided Design* No. 6, pp. 55-67.
Hand, C. (1997) 'A survey of 3D interaction techniques'. *Computer Graphics Forum*, Vol. 16, No. 5, pp. 269-281.
Harries, S. (1998) '*Parametric Design and Hydrodynamic Optimization of Ship Hull Forms*'. Mensch & Buch Verlag, Berlin.
Harries, S, and Abt, C. (1998) 'Parametric curve design applying fairness criteria', in: Nowacki, H. and Kaklis, P.D, editors. *Creating Fair and Shape-Preserving Curves and Surfaces*, Teubner, Stuttgart-Leipzig, Germany, pp. 68-77.
Hees, M. van, (1997) '*Quaestor: Expert Governed Parametric Model Assembling*'. Ph.D. thesis Delft University of Technology, Delft, The Netherlands, February 10.
Hills, W. and Welsh, M. (1986) 'An effective method of preliminary hull form design using a workstation'. *International Shipbuilding Progress*, October, pp. 187-194.
Holtrop, J., (1984) 'A statistical re-analysis of resistance and propulsion data'. *International Shipbuilding Progress*, Vol. 31, No. 363, November.
Horváth, I. and Juhasz, I. (1997) '*Computer Aided Mechanical Design*'. Technical Book Publisher, Budapest (in Hungarian).
Horváth, I. and Vergeest, J.S.M. (1998) 'Theoretical fundamentals of natural representa-

tion of shapes generated with gestural devices', in: Horváth, I. and Taleb-Bendiab, A., editors. *Proceedings of TMCE'98 Tools and Methods of Concurrent Engineering Symposium*, April 21-23, Manchester, UK, pp. 393-409.

Horváth, I., Vergeest, J.S.M., Broek, Rusák, Z., J.J., Smit, B. de (1998) 'Tool profile and tool path calculation for free-form thick-layered fabrication'. *Computer-Aided Design*, Vol. 30, No. 14, pp. 1097-1110.

Horváth, I., Vergeest, J.S.M., Juhász, I. (1998) 'Finding the shape of a flexible blade for free-form layered manufacturing of plastic foam objects', in: *Proceeding of the ASME 1998 Computer in Engineering Conference*, Atlanta, Georgia, USA, September 13-16.

Horváth, I. (1999) Lecture notes. Web *http://www.io.tu.delft.nl/research/ICA/*.

Hoschek, J. and Lasser, D. (1992) '*Grundlagen der geometrischen Datenverarbeitung*'. B.G. Teubner, Stuttgart, Germany.

Hubka, V., (1982) '*Principles of Engineering Design*'. Translated and edited by Eder W. E., Butterworth Scientific, London.

IGES (1993) '*Initial Graphics Exchange Specification*'. U.S. Product Data Association.

Isaacson, E. and Bishop Keller, H. (1966) '*Analysis of Numerical Methods*'. John Wiley & Sons, New York.

ISO (1995) '*ISO TC184/SC4/WG3 N 395 (T12): Validation Report for the Ship Moulded Forms Application Protocol, AP216*', May 1.

ISO (1997) 'ISO TC184/SC4/WG3 N 571 (T12): Application Protocol: Ship Moulded Forms'. Web *http://www.nist.gov/sc4/*, 14 February.

Jager, P.J. de, Broek, J.J and Vergeest, J.S.M. (1996) 'Rapid prototyping: extending the layer concept'. Proceedings of the *2nd International Conference on Rapid Product Development (ICRPD)*, June 10-11, Stuttgart, Germany.

Jennings, G.A. (1994) '*Modern Geometry with Applications*'. Springer-Verlag, New York, USA.

Jensen, T. (1987) 'Assembling triangular and rectangular patches and multivariate splines', in: Farin, G.E., editor. *Geometric Modeling: Algorithms and New Trends*, SIAM, Philadelphia, USA, pp. 203-220.

Jensen, T.W., Petersen, C.S. and Watkins, M.A. (1991) 'Practical curves and surfaces for a geometric modeler'. *Computer Aided Geometric Design* 8, 357-369.

Jorde, J.J (1997) 'Mathematics of a body plan'. *The Naval Architect* January, pp. 38-41.

Kasteren, J. van (1997) 'Printen in drie dimensies'. *NRC Handelsblad*, December 6.

Kato, K. (1991) 'Generation of N-sided surface patches with holes'. *Computer-Aided Design*, Vol. 23 No. 10, December.

Keuning, J.A., Gerritsma, J., and Terwisga, P.F. van (1993) 'Resistance tests of a series planing hull forms with 30° deadrise angle and a calculation model based on this and similar systematic series'. *International Shipbuilding Progress*, Vol. 40, No. 424, December.

Kim, S.Y., Kim, H.C. and Lee, Y.S. (1996) 'Initial hull form design using fuzzy modeling'. *Schiffstechnik*, Bd. 43, pp. 175-180.

Kjellander, J.A.P. (1983) 'Smoothing of cubic parametric splines'. *Computer-Aided Design*, Vol. 15, No. 3, May, pp. 175-179.

Kovachey, Sv and Yovev, Y. (1983) 'Computer aided synthesis of ship form', in: Proceedings of *The 2nd International Symposium of Practical Design in Shipbuilding*, Tokyo and Seoul.

Koelman, H.J. (1978) '*Scheepsvormgeneratie*'. Report Haarlem Polytechnical School, Haarlem, The Netherlands, pp. 1-76 (in Dutch).
Koelman, H.J. (1985) '*Scheepsvormgeneratie*'. Master thesis, Delft University of Technology, Delft, The Netherlands, pp. 1-21 (in Dutch).
Koelman, H.J. (1997a) 'Hull form design and fairing: tradition restored', in: Sen, P. and Birmingham, R., editors. The proceedings of *The Sixth International Marine Design Conference*, June 23-35, Newcastle upon Tyne, UK.
Koelman, H.J. (1997b) 'A topological approach to hull form design', in: proceedings of the *Twelfth International Conference on Hydrodynamics in Ship Design*, September 17-19, Szklarska Poręba, Poland.
Kosko, B. (1997) '*Fuzzy Engineering*'. Prentice-hall. New Yersey, USA.
Kouh, J.S. and Chau, S.W. (1990) 'Design and representation of hull form using rational cubic Bézier curves', in: van Oortmerssen, G., editor. *Proceedings of the International Symposium on CFD and CAD in Ship Design*, September 25-26, Wageningen, The Netherlands.
Kuiper, G. (1970) 'Preliminary design of ship lines my mathematical methods'. *Journal of Ship Research*, Vol. 14, No. 1, pp. 52-66.
Kulkarni, P., and Dutta, D. (1996) 'An accurate slicing procedure for layered manufacturing'. *Computer-Aided Design*, Vol. 28, No. 9. pp. 683-697.
Kuo, C. (1971), '*Computer Methods for Ship Surface Design*'. Longman Group Limited, London, UK.
Kuriyama, S. (1994) 'Surface modeling with an irregular network of curves via sweeping and blending'. *Computer-Aided Design*, Vol. 26, No. 8, August.

Laansma, K. (1993) 'L/GRAND: A new generation ship design system'. *Schip & Werf de Zee*, pp. 554-555.
Lackenby, H. (1950) 'On the systematic geometrical variation of ship forms'. *Trans. INA*, Vol. 92, pp. 289-316.
Lap, A.J.W. (1954) 'Diagrams for determining the resistance of single screw ships'. *International Shipbuilding Progress*, Vol. 1, No. 4, pp. 179.
Lennings, A.F. (1998) 'DeskProto - redefining concept modelling'. *Prototyping Technology International '98*, pp. 110-113.
Liming, R.A. (1979) '*Mathematics for Computer Graphics*'. Aero Publishers, Fallbrook, California, USA.
Lloyd's Register of Shipping (1997) '*ShipRight Product Model*'. SDG Report 95/50/3.1, August 1.

Ma, W. and Kruth, J.P. (1995) 'Parametrization of randomly measured points for least squares fitting of B-spline curves and surfaces'. *Computer-Aided Design*, Vol. 27, No. 9, September.
Mäntylä, M. (1988) '*An Introduction to Solid Modeling*'. Computer Science Press, Mayland, USA.
Michelsen, J., Baatrup, J. and Jensen, J.J. (1993) 'Development of a CAD system for ship design in a university environment', in: Bogdanov, P.A., editor. *Proceedings of IMAM'93 (VI)*, November 15-20, Varna Bulgaria.
Michelsen, J. (1994). '*A Free-Form Geometric Modeling Approach with Ship Design Applications*'. Ph.D. thesis, Department of Ocean Engineering, Technical University of Denmark, Copenhagen, Denmark.

Mortenson, M.E. (1985) '*Geometric Modeling*'. John Wiley & Sons, New York, USA.
Munchmeyer, F.C., Schubert, C. and Nowacki, H. (1979) 'Interactive design of fair hull surfaces using B-splines', in: Kuo, C., MacCallum, K.J. and Williams, T.J., editors. *Computer Applications in the Automation of Shipyard Operation and Ship Design III*, June 18-21, Glascow, Scotland.
Muuss, M.J. and Butler, L.A. (1991) 'Combinatorial solid geometry, boundary representations, and non-manifold geometry', in: Rogers, D.F. and Earnshaw, R.A., editors. *State of the Art in Computer Graphics*, Springer-Verlag, New York, USA, pp. 185-223.

Nasri, A.H. (1987) 'Polyhedral subdivision methods for free-form design'. *ACM Transactions of Graphics*. Vol. 6, No. 1, January.
Nasri, A.H. (1991) 'Surface interpolation on irregular networks with normal conditions'. *Computer Aided Geometric Design*, Vol. 8, pp. 89-96.
Nolan, T.J. (1971) 'Computer-aided design of developable hull surfaces'. *Marine Technology*, April, pp. 233-242.
Nowacki, H. and Xinmin Lü (1994) 'Fairing composite polynomial curves with constraints'. *Computer Aided Geometric Design* 11, pp. 1-15.
Nowacki, H., Feng Jin and Xiuzi Ye (1997) 'A synthesis process for fair free-form surfaces', in: Strasser, W., Klein, R. and Rau, R., editors. *Geometric Modeling: Theory and Practice*, Springer-Verlag, Berlin, Germany, pp. 32-51.

Oossanen, P. and Pieffers, J.B.M. (1985) 'NSMB-systematic series of high speed displacement ship hull forms'. *MARIN Workshop on Developments in Hull Form Design*, October, Wageningen, The Netherlands.
Owen, J. (1997) '*STEP: an Introduction*'. Information Geometeers Ltd, Winchester, UK.

Peacock, D., Smith, W.F. and Kumar Pal, P. (1997) 'Hull-form generation using multi-objective optimisation techniques', in: Sen, P. and Birmingham, R., editors. The proceedings of *The Sixth International Marine Design Conference*, June 23-25, Newcastle upon Tyne, UK.
Peters, J. (1990a) 'Local smooth surface interpolation: a classification'. *Computer Aided Geometric Design*, Vol. 7, pp. 191-195.
Peters, J. (1990b) 'Smooth mesh interpolation with cubic patches'. *Computer-Aided Design*, Vol. 22, March, pp. 109-120.
Peters, J. (1994) 'Constructing C1 surfaces of arbitrary topology using biquartic/bicubic splines', in: Sapidis, N.S., editor. *Designing Fair Curves and Surfaces*, pp. 277-294.
PIAS (1985): *'User Manual of the Program for the Integral Approach of Ship Design*'. SARC, Bussum, The Netherlands.
Piegl, L. and Tiller, W. (1987) 'Curve and surface constructions using rational B-splines'. *Computer-Aided Design*, Vol. 19, No. 9, November.
Piegl, L. (editor) (1993). '*Fundamental Developments of Computer Aided Geometric Modeling*'. Academic Press, San Diego, USA.
Piegl, L. and Tiller, W. (1996). 'Algorithm for approximate NURBS skinning'. *Computer-Aided Design*, Vol. 28, No. 9, pp. 699-706.
Pigounakis, K.G., Sapidis, N.S. and Panagiotis, D.K. (1996) 'Fairing spatial B-spline curves'. *Journal of Ship Research*, Vol. 40, No. 4, December, pp. 351-367.

Rabien, U. (1979) 'Ship hull surface design by transforming given mesh representations', in: Kuo, C., MacCallum, K.J. and Williams, T.J., editors. *Computer Applications in the Automation of Shipyard Operation and Ship Design III*, Glascow, Scotland, June 18-21.
Redont, P. (1989) 'Representation and deformation of developable surfaces'. *Computer-Aided Design*, Vol. 21, No. 1, January/February.
Requicha, A.A.G. (1980) 'Representations for rigid solids: theory, methods and systems'. *Computing Surveys*, Vol. 12, No. 4, December, pp. 437-463.
Reuding, T. (1989) 'Bézier patches on cubic grid curves - an application to the preliminary design of a yacht hull surface'. *Computer Aided Geometric Design* 6 (1989) pp. 11-21.
Rogers, D.F. and Satterfield, S.G. (1982) 'Dynamic B-spline surfaces', in: Rogers, D.F, Nehrling, B.C. and Kuo, C., editors. *Computer Applications in the Automation of Shipyard Operation and Ship Design IV*, June 7-10, Annapolis, USA.
Rogers, D.F. and Fog, N.G. (1989) 'Constrained B-spline curve and surface fitting'. *Computer-Aided Design*, Vol. 21, No. 10, December.
Roulier, J. and Rando, T. (1994) 'Measures of fairness for curves and surfaces', in: Sapidis, N.S., editor. *Designing Fair Curves and Surfaces*, SIAM, Philadelphia, USA, pp. 75-121.

Sarkar, B. and Menq, C.H. 'Parameter optimization in approximating curves and surfaces to measurement data'. *Computer Aided Geometric Design* 8, pp. 267-290.
Schneekluth, H. and Bertram, V. (1998) '*Ship Design for Efficiency and Economy*'. Butterworth-Heinemann, Oxford, UK.
Standerski, N.B. (1989) 'The generation of ship hulls with given design parameters using tensor product surfaces', in: Strasser W. and Seidel, H.P., editors. *Theory and Practice of Geometric Modeling*, Springer-Verlag, Heidelberg, Germany.
Stroobant, G. and Mars, D. (1982) 'Ship hull form fairing', in: Rogers, D.F, Nehrling, B.C. and Kuo, C., editors. *Computer Applications in the Automation of Shipyard Operation and Ship Design IV*, June 7-10, Annapolis, USA.
Su Bu-qing and Liu Ding-Yuan (1989) '*Computational Geometry*'. Academic Press, San Diego, USA.

Taylor, D.W. (1933) '*The Speed and Power of Ships*'. Ransdell Incorporated, Washington D.C., USA.
Todd, F.H. (1963) 'Series 60, methodical experiments with models of single-screw merchant ships'. *DTMB Research and Development Report 1912*.
Tuohy, S., Latorre, R., Munchmeyer, F. (1996) 'Developments in surface fairing procedures'. *International Shipbuilding Progress*, Vol. 43, No. 436, December, pp. 282-313.

Varady, T. (1991) 'Overlap patches: a new scheme for interpolating curve networks with N-sided regions'. *Computer Aided Geometric Design* no. 8 (1991) pp. 7-27.
Ventura, M. and Soares, C.G. (1998) 'Hull form modelling using NURBS curves and surfaces', in: Proceedings of *Practical Design of Ships and Mobile Units*, September 20-25, The Hague, The Netherlands.

Wake, M. (1997) 'Major innovations planned for FORAN'. *The Naval Architect*, July/August, p. 48.
Warren, J. (1992) 'Creating multisided rational Bézier surfaces using base points'. *ACM Transactions on Graphics*, Vol. 11, No. 2, April, pp. 127-139.

Weiler, K. (1985) 'Edge-based data structures for solid modeling in curved-surface environments'. *IEEE Computer Graphics and Applications*, January, pp. 21-40.

Weiler, K. (1986a) 'The radial edge structure: a topological representation for non-manifold geometric boundary modeling', in: Wozny, M.J., McLaughlin, H.W. and Encarnaçao, J.L., editors. *Geometric Modeling for CAD Applications*, May 12-16, Rensselaerville, USA. pp. 3-36.

Weiler, K. (1986b) 'Boundary graph operators for non-manifold geometric modeling topology representation', in: Wozny, M.J., McLaughlin, H.W. and Encarnaçao, J.L., editors. *Geometric Modeling for CAD Applications*, May 12-16, Rensselaerville, USA. pp. 37-66.

Weinblum, G. (1953) 'Systematische Entwicklung von Schiffsformen'. *Jahrbuch der STG 1953*, pp. 186-215.

Woodward, C.D. (1987) 'Cross-sectional design of B-spline surfaces'. *Comput. & Graphics* Vol. 11, No. 2, pp. 193-201.

Woodward, C.D. (1988) 'Skinning techniques for interactive B-spline surface interpolation'. *Computer-Aided Design*, Vol. 20, No. 8, October, pp. 441-451.

Woodward, C.D. (1990) '*Methods for Computer-Aided Design of Free-Form Objects*'. Ph.D. thesis, Department of Computer Science, Helsinki University of Technology, Finland.

Yamaguchi, Y. and Kimura, F. 'Nonmanifold topology based on coupling entities'. *IEEE Computer Graphics and Applications*, January.

Yuille, I.M. (1979) 'Interactive program for the design of ship hull forms', in: Kuo, C., MacCallum, K.J. and Williams, T.J., editors. *Computer Applications in the Automation of Shipyard Operation and Ship Design III*, June 18-21, Glascow, Scotland.

Zeid, I. (1991) '*CAD/CAM Theory and Practice*'. McGraw-Hill, New York, USA.

Summary

Computer Support for Design, Engineering and Prototyping of the Shape of Ship Hulls

This thesis describes the design and development of a computer system which is useful for all shape aspects of a ship's moulded hull form, such as design, fairing, design modification, visualization, physical modelling and preparation for export to other general or specific CAD, CAE or CAM software.

Based on an inventory of hull design methodology, the requirements for the system are formulated. Subsequently mathematical methods for representation of free form surfaces are investigated, and an overview is made of the methods used by existing hull form modelling software. It appears that the single-patch B-spline/NURBS surface method (or straightforward multi-patch extensions of it) is dominant, but a theoretical as well as a practical analysis of the capabilities of that method show that it is less appropriate for our purposes. The most important objection is the regularity of the network, which imposes severe limitations on the freedom of the system user.

Furthermore, the system requirements imply a complete geometric model, which includes both geometry and topology.

So a novel system is designed, based on three core elements:

- A Boundary REPresentation, to represent topology. This BREP is extended with the capability to include continuous free form curves, and sequences of curves. For geometry representation this extended BREP uses NURBS-curves.
  This construction enables one of the most important properties of the new system, which is the use of an arbitrary, and coherent, composition of curves on the hull surface. The location and orientation of the curves are completely free;
- Surface generation based on transfinite interpolation of the curves, either with four-sided, or with N-sided (N<>4) patches;
- Interactive and semi-automatic fairing capabilities, with a weighted least-squares algorithm with automatic knot selection. The user can determine the balance between fairness and shape constraints, and the algorithm fairs the curve globally, taking into account the user's choices.

To accommodate all required activities with the hull form, the system is equipped with many supporting entities or functions, such as:

- Elementary curve shapes, e.g. to be used for waterline roundings;
- Interpolation of curves and surfaces;
- Developable surface representations;
- Export functions, for example to CFD or general CAD software;
- Visualization functions, such as lines plan generation;

- Physical modelling (rapid prototyping), including automatic decomposition into processable segments;
- Hull form transformation, and support for the Sectional Area Curve;
- Hierarchical geometrical relationships between curves.

The system has been implemented on a PC, and was baptized 'Fairway'. It has been applied to many designs, and eight of those, created by different designers, are presented in this thesis. These examples demonstrate the versatility of the system, and the variety of hull forms which can be handled. Using Fairway it also became apparent that satisfactory results could be obtained without explicit GC^2 surface continuity.

The value of the approach is further investigated by an opinion poll among users of the system. One of the most important issues of the questionnaire was a comparison between a design made with and without the Fairway system. Grosso modo it appeared that either a Fairway design took less time for a comparable result, or it took comparable time for a more fair and a much more detailed result. In general the users reported quite positively about the system, but they also came back with tips for improvement, mainly concerning the user-interface and additional support functions.

Finally, it is concluded that the methodology of our new system is a better solution for the ship hull subject than conventional approaches.

H.J. Koelman, 1999

Samenvatting

Een computersysteem ter ondersteuning van het voorontwerp, het detailontwerp en het maken van tastbare modellen van scheepsvormen.

Dit proefschrift beschrijft het ontwerp en de ontwikkeling van een computersysteem wat gebruikt kan worden bij alle vormaspecten van de scheepsromp, zoals ontwerp, stroken, ontwerpwijzigingen, visualisering, fysiek modelleren en het doorsturen van de gegevens naar andere algemene of specifieke ontwerp- of fabricageprogrammatuur.

De aan het systeem te stellen eisen zijn opgesteld aan de hand van een inventarisatie van ontwerpmethodologieën voor de scheepsromp. Vervolgens worden wiskundige methodes voor het weergeven van gekromde oppervlakken onderzocht, en wordt een overzicht gegeven van de technieken die gebruikt worden in bestaande programmatuur voor het modelleren van scheepsvormen. Het blijkt dat de *single-patch B-spline/NURBS* oppervlaktebeschrijving (of eenvoudige *multi-patch* uitbreidingen daarvan) de boventoon voert, maar zowel een theoretische als een praktische analyse van de mogelijkheden daarvan toont aan dat die minder geschikt voor onze doeleinden is. Het belangrijkste bezwaar is de regelmatigheid van het netwerk, die ernstige beperkingen oplegt aan de vrijheid in het gebruik.

Daarnaast impliceren de systeemeisen het gebruik van een compleet geometrisch model, een model wat zowel geometrie als topologie bevat.

Daarom is een nieuw systeem ontworpen, wat gebaseerd is op drie elementen:

- Een *Boundary REPresentation*, om de topologie weer te geven. Deze *BREP* is uitgebreid met de mogelijkheid om doorlopende krommen en ketens van krommen te bevatten. De vorm van deze krommen wordt met *NURBS* weergegeven.
  Deze constructie maakt het mogelijk een willekeurig en samenhangend samenstel van krommen op de scheepsromp te gebruiken, waarbij de plaats en oriëntatie van de krommen volkomen vrij zijn. Dit is een van de belangrijkste eigenschappen van het nieuwe systeem.
- Het genereren van de vorm van de gekromde oppervlaktes, gebaseerd op *transfinite* interpolatie van de krommen, hetzij met vierzijdige, of met N-zijdige (N<>4) *patches*.
- Interactieve, en gedeeltelijk automatische strooktechnieken, die gebaseerd zijn op een gewogen kleinste kwadraten algoritme met automatische knooppuntbepaling. De gebruiker kan daarbij het evenwicht tussen gladheid enerzijds en vormvereisten anderzijds bepalen, waarbij het algoritme de kromme in globale zin strookt, rekening houdend met de opgegeven voorkeuren.

Om alle vereiste activiteiten met de scheepsvorm mogelijk te maken, is het systeem uitgerust met een aantal ondersteunende middelen of functies:

- Eenvoudige krommen, bijvoorbeeld te gebruiken voor waterlijnafrondingen.
- Interpolatie van lijnen en oppervlakken.
- Ontwikkelbare oppervlakken.

- Mogelijkheden om de vorm te exporteren, bijvoorbeeld naar programmatuur voor omstromingsberekeningen, of naar algemene CAD paketten.
- Visualiseringsfuncties, zoals het genereren van lijnenplannen.
- Het maken van fysieke scheepsmodellen, inclusief het automatisch opdelen in segmenten die ook daadwerkelijk gefabriceerd kunnen worden.
- Rompvorm transformatie, alsmede ondersteuning door, en gebruik van de Kromme Van Spantoppervlakken.
- Hierarchische verhoudingen tussen de vorm van lijnen.

Het systeem, genaamd 'Fairway', is geïmplementeerd op een PC. Het is reeds gebruikt voor vele ontwerpen, en acht daarvan, gemaakt door verschillende ontwerpers, worden in dit boek gepresenteerd. Deze voorbeelden tonen de veelzijdigheid van het systeem aan, en de variëteit aan rompvormen die behandeld kan worden. Tijdens het gebruik van Fairway kwam ook aan het licht dat bevredigende resultaten kunnen worden behaald zonder dat oppervlakken onderling per se expliciet GC^2 continu hoeven te zijn.

De relevantie van onze benadering is nader onderzocht met een enquête onder gebruikers van het programma. Eén van de belangrijkste vragen was daarbij een vergelijking te maken tussen een ontwerp gebruikmakend van, en een ontwerp zonder Fairway. Het bleek grosso modo dat hetzij een ontwerp met Fairway minder tijd in beslag nam voor een vergelijkbaar resultaat, of dat een vergelijkbare hoeveelheid tijd werd gespendeerd voor een gedetailleerder of strokender resultaat. In het algemeen waren de gebruikers behoorlijk positief over het systeem, maar zij kwamen ook met nuttige tips voor verbetering, die voornamelijk betrekking hebben op de gebruikers *interface* en op aanvullende ondersteunende functies.

Tenslotte wordt geconcludeerd dat de aanpak van ons nieuwe systeem geschikter is voor het ontwerpen van en werken met scheepsvormen dan de conventionele methodes.

H.J. Koelman, 1999

Acknowledgements

This Ph.D. thesis is the result of years of research, discussion and hard labour. Without the cooperation and enthusiasm of many persons during all these years this thesis would not be, at least not in this satisfactory form.

In the first place I am grateful to Martin van Hees, who ignited the idea of a promotion, in a smoky Newcastle pub.

Without the enthusiasm and encouragement of my colleagues, Martin Pronk, Bart Soede, Guido Vijn and Mark Visser, and my former colleague Robert Kleijweg, the Fairway system would certainly be less handy, and less advanced.

However, the long and principal discussions with the ship designers Klaas Huizinga and Ad van der Horst also contributed to Fairway, and to this thesis.

I am also grateful to Gebr. van Dijke, Engelaer Scheepsbouw, Delmon Marine Consultantants, Egbert de Lint, Olivier van Meer Design, Ferus Smit Shipyard, Visser Shipyard and Wijsmuller Engineering for allowing their specific designs to be presented and discussed.

Then of course my first promoter, Arie Aalbers. His knowledge of contemporary ship design methodologies, but always with a strong historical awareness, was very motivating.

Equally inspiring was the very conscious and critical support of Imre Horváth, who disclosed me the latest CAD methods.

Finally, my compliments go to Ad Vredenduin, for the design and realization of all figures in this thesis and for the design of the layout, and to Marion Goddijn for the accurate improvements in my English grammar.

Herbert Koelman
September 1999

Biography

Herbert Koelman was born on November 17, 1957 in Amsterdam in The Netherlands. He attended secondary school in Hilversum. From 1975 to 1979 he studied naval architecture at the Polytechnical School of Haarlem. During practical work at MARIN of Wageningen in 1978 he wrote a first simple computer program for hull form generation.

From 1979 to 1985 he continued his naval architectural study at the Delft University of Technology, where he graduated at the section of ship hydrodynamics. The subject of his final project in Delft was *Computer-Aided Design of Hull Forms*.

During his study, in 1980, he founded the company SARC (which is an abbreviation of *Scheepsbouwkundig Advies en Reken Centrum*, which means *Centre for Naval Architectural Design, Calculations and Consultancy*). Ever since he has been engaged in ship design, and in the research on, and the development and marketing of software for maritime use. This software is dedicated to the fields of ship design, engineering calculations and maritime safety.

The quest for alternative, and better methods for hull form modelling began in the early nineties. Ultimately, in close cooperation with Delft University, it led to the software and methods as described in this thesis.

Stellingen bij het proefschrift *Computer Support for Design, Engineering and Prototyping of the Shape of Ship Hulls*

1. In de literatuur die handelt over het vormgeven van schepen met de computer richt men zich veelal eerder op het haalbare dan op het wenselijke. Dat remt de vooruitgang.

2. De populariteit van de NURBS oppervlaktebeschrijving op het gebied van scheepsvormmodellering is meer ingegeven door het mistige acroniem en het najagen van this year's model, dan door z'n capaciteiten.

3. Dit proefschrift toont aan dat een bevredigende oplossing kan worden bereikt door de integratie van bestaande technieken voor geometrisch modelleren, zonder per se de 1001ste nieuwe te hoeven ontwikkelen.

4. Hoewel we over geavanceerde technieken voor virtueel modelleren beschikken, kan het maken van tastbare modellen niet worden verwaarloosd.

5. Het is niet te verwachten dat mijn kindskinderen een semantisch betekenisvol produktmodel van een schip zullen meemaken.

6. De acceptatie van de regels voor de probabilistische lekstabiliteit van schepen zou aanmerkelijk stijgen door het consequent vervangen van het woord 'kans' door 'index'.

7. De probabilistische lekstabiliteitsregels voor schepen zouden geherformuleerd moeten worden, waarbij de elegante grondslagen van de methode vanzelfsprekend behouden dienen te blijven.

8. De wereld mag softwaremakers dankbaar zijn, want hun gebrekkige producten zijn goed tegen de werkeloosheid.

9. Maatgevend voor de effectiviteit van het PC gebruik is heden ten dage de reboot snelheid.

10. In de softwarewereld houdt het begrip 'technologie' meestal niet meer in dan wat conventies en administratieve procedures.

11. Gelet op de algemeen toenemende aandacht voor presentatie van een werk boven inhoud heb ik steeds meer profijt van mijn drie jaar kleuterschool dan van mijn zes universitaire jaren.

12. In Nederland biedt het belastingrecht meer middelen om misdadigers te straffen dan het strafrecht. Dit illustreert de zwakke rechtspositie van de gewone burger in het fiscale rechtssysteem.

H.J. Koelman
December 1999

Propositions of thesis *Computer Support for Design, Engineering and Prototyping of the Shape of Ship Hulls*

1. The literature dealing with the design of the shape of ship hulls is more dedicated to attainability than to desirability. That restraints progression.

2. The popularity of the NURBS-based surface description for modelling ship hull forms is more a consequence of the misty acronym and chasing this year's model, rather than of its capabilities.

3. This thesis proved that adequate solutions can be obtained by integrating some existing geometric modelling techniques, instead of developing the 1001th new one.

4. Contrary to the fact that we have methods for sophisticated virtual modelling, the physical model making cannot be neglected.

5. It is not to be expected that my children's children will see a product model of a ship which carries all semantics.

6. The acceptance of the legislation for probabilistic damage stability of ships can be significantly increased by the consequent substitution of the word 'probability' by 'index'.

7. The code for probabilistic damage stability of ships should be completely reformulated, while keeping its elegant foundations.

8. The world has to be grateful to software producers, because their defective products reduce the rate of unemployment.

9. A tangible measure of the effectiveness of the use of a personal computer nowadays is the rebooting speed of the system.

10. In the software world 'technology' often means nothing more than a few conventions and some administrative procedures.

11. Due to the present trend to put more emphasis on the presentation of a work than on its content, I benefit more and more from my kindergarten exercises than from my university years.

12. The Dutch tax laws provide more powerful means to punish criminals than the criminal laws. This illustrates the weak legal status of ordinary civilians in our tax system.

H.J. Koelman
December 1999