

Alla mia Famiglia
A Giulia

UNIVERSITÀ DEGLI STUDI DI PISA

Facoltà di Ingegneria

Corso di Laurea in Ingegneria Aerospaziale

Tesi di Laurea

**A Code for Surface Modeling and Grid
Generation Coupled to a Panel Method for
Aerodynamic Configuration Design**



Relatori:

Prof. Aldo Frediani

Laureando:

Rauno CAVALLARO

Dott.Ing. Giovanni Bernardini

ANNO ACCADEMICO 2007-2008

Summary

An integrated platform has been developed which features a geometric, a grid generation and an aerodynamic analysis module. The main intent is to execute a quick though reliable preliminary aerodynamic analysis on a generic complex aerodynamic configuration and, at the same time, provide a mean of exporting the defined geometry or grid to leading CAE/CAD, meshing and analysis softwares, for deep detail modifications or more accurate, although time consuming, analysis.

In the geometric module, the process of shape definition is easily and intuitively achieved with the aid of specific features and tools. The geometric description relies on NURBS, a flexible, accurate and efficient parametric form. Once the configuration has been defined, the user is ready to move on the grid generation module, or to export it to IGES standard format in order to use CAE/CAD, meshing or aerodynamic analysis programs.

The grid generation module is capable to build structured or unstructured meshes. Both of the processes are automatized, even if the user can easily set and control grid parameters. The structured grid generator is oriented to LaWGS description standard, while the unstructured grid can be exported to different formats.

The user is now ready to launch Pan Air, a panel method, as the aerodynamic solver. The preprocessor and postprocessor aid to the definition of the flow parameters and to the graphical visualization of the results.

One of the strength of this code is the user friendly GUI organization of each module: the user is aided throughout all the steps. Besides this, every module relies

on fast computational algorithms to speed up the overall process.

For all these reasons, this code has a natural lean to be used in pair with an optimization tool.

Acknowledgements

Alla fine di questo *lungo* percorso, è giusto fermarsi un momento e guardarsi intorno, ma soprattutto indietro ed un dovuto e sentito pensiero va a chi mi è stato vicino, a chi mi ha aiutato ed a chi ha contribuito in modo importante al raggiungimento di questo traguardo.

Volendo percorrere la strada a ritroso, la prima persona alla quale va un caloroso ringraziamento è il Professor Aldo Frediani, in lui ho trovato entusiasmo, coinvolgimento ed ha riacceso in me una passione per lo studio negli ultimi anni un po' affievolitasi.

Ringrazio anche il mio correlatore Giovanni Bernardini, per l'umanità dimostrata e il tempo e la pazienza che mi ha dedicato, ma cosa ancora più importante lo ringrazio infinitamente per il grande bagaglio di conoscenze che è riuscito a trasmettermi.

Non posso certo dimenticare la Professoressa Maria Vittoria Salvetti che tra una battuta e l'altra mi ha sempre sostenuto, dimostrando di credere nelle mie capacità, anche se spesso volubili.

Spostandomi in ambito ludico un ruolo determinante in questi anni lo hanno ricoperto i miei migliori amici, che hanno contribuito a preservarmi dal logorio dello studio, facendomi così arrivare fresco, ma un po' in ritardo, a questo appuntamento!

Sicuramente un ringraziamento speciale va alla mia dolce metà Giulia che senza lasciarsi scoraggiare dal mio carattere impossibile mi è stata vicina negli ultimi due anni, aiutandomi in maniera decisiva e donandomi la stabilità e la serenità di cui

avevo bisogno per affrontare questo percorso.

Il ringraziamento finale va alla mia famiglia, alla quale non sono mai riuscito ad esprimere direttamente la mia gratitudine, ma alla quale spero di aver regalato in questa giornata grande gioia e soddisfazione.

Table of contents

Summary	I
Acknowledgements	III
Introduction	1
1 ASD: Aerodynamic Shape Design	5
1.1 Introduction	5
1.2 ASD Surface Generator	6
1.2.1 The Main Window	6
1.2.2 The Features and the Feature Windows	6
1.2.3 The Body Feature	6
1.2.4 The Wing Feature	9
1.2.5 The Bulk Feature	15
1.2.6 The Wingbody Feature	16
1.2.7 The Inlet/Outlet Feature	18
1.2.8 The Fillet Feature	23
1.2.9 The Tfillet Feature	27
1.2.10 The Wround Feature	29
1.2.11 Viewing the Results	30
1.3 ASD Surface Mesher	33
1.4 The ASD tools	37

1.4.1	Airfoil Manager	37
1.4.2	NACA Airfoil Generator	39
1.4.3	Section Sketcher	40
1.4.4	Flap Sketcher	41
1.5	ASD Limitations	45
2	NURBS (Non Uniform Rational B-Spline)	51
2.1	A Brief Historical Survey	51
2.2	Curve and Surface Basics	54
2.2.1	Implicit and Parametric Forms	54
2.2.2	Advantages and Disadvantages	55
2.2.3	Requirements for the parametric forms	56
2.2.4	Power Basis Form of a curve	57
2.2.5	Bézier Curves	57
2.2.6	Tensor Product Surfaces	62
2.3	B-Splines	65
2.3.1	Shortcoming of polynomial and Bézier forms	66
2.3.2	B-Spline Basis Functions	67
2.3.3	B-Spline Curves	74
2.3.4	B-Spline Surfaces	81
2.4	Rational B-Splines	87
2.4.1	Nurbs Curves	88
2.4.2	Nurbs Surfaces	91
2.5	Fundamental Geometric Algorithms	92
2.5.1	Knot Insertion	93
2.5.2	Knot Removal	93
2.5.3	Degree Elevation	95
2.5.4	Degree Reduction	96
2.5.5	Other Advanced Geometric Algorithms	96
3	Free Form Surface Design	97
3.1	Introduction	97

3.2	Fitting	98
3.2.1	Global Interpolation	99
3.2.2	Local Interpolation	103
3.2.3	Transfinite interpolation	105
3.3	Parameterization	107
3.3.1	Uniform Parameterization	107
3.3.2	Chord Length Parameterization	108
3.3.3	Centripetal Parameterization	108
3.3.4	Performance of the Different Parameterizations	108
3.3.5	Knot Vector Selection	109
3.3.6	Parameterization and Knot Vector Selection for Surfaces . . .	110
3.4	Notion of Continuity	111
3.4.1	Continuity of Curves	111
3.4.2	Continuity of Surfaces	115
3.5	Visual Aspects of Continuity	117
3.6	Implementation of G^2 or C^2 Continuity with Local and Global Algorithms	118
3.6.1	Limitations of the Local Approach for G^2 or C^2 Continuity . .	118
3.6.2	Limitations of the Global Approach for C^2 continuity	119
3.7	Variational Analysis and Modeling of Free Forms	121
3.7.1	Fairness of Curves and Surfaces	121
3.7.2	Constrained Optimization	123
3.7.3	Conclusions	126
4	Surface Modeling in ASD	127
4.1	The old Algorithms featured in ASD	127
4.1.1	ASD Curve Algorithm - <code>local_interp_crv</code>	127
4.1.2	ASD Bicubic Surface Algorithm - <code>local_interp_sfc</code>	132
4.2	The new Geometric Algorithms	134
4.2.1	The new Curve Algorithm - <code>global_interp_crv</code>	134
4.2.2	The new Surface Algorithm - <code>global_interp_sfc</code>	141
4.2.3	Lofting Algorithm	142

4.3	The ASD Advanced NURBS GUI	145
4.3.1	C^1 Algorithm Parameter	145
4.3.2	C^2 Algorithm Parameter	146
4.3.3	Lofting	147
4.4	Some Results with the new Geometric Engine	147
4.5	Future Improvements	151
5	Structured Grid Generation Module	153
5.1	Introduction	153
5.1.1	Grid Connectivity-Based Classification	154
5.1.2	An Overview of Structured Mesh Generation	155
5.2	The New Grid Generation Module	157
5.3	Requirements for the Structured Grid Generation	158
5.3.1	ASD and the Mesh	158
5.3.2	Pan Air and the Mesh	159
5.4	Logics and Mesh Organization	160
5.4.1	Subdivision in Logical Subsets	160
5.4.2	Meshing the Logical Subsets	162
5.4.3	Connections between Logical Subset Grids	165
5.5	The Structure Grid Generation Interface	169
5.5.1	Wing and Body Feature Parameters	169
5.5.2	Mesh Generation	170
5.5.3	Mesh Analysis and Stats	170
5.5.4	Plotting Tools	170
5.5.5	Grid Storing	170
5.5.6	Pan Air Preprocessor Launcher	171
5.6	Examples of Structured Grid Generated	171
5.7	Limitations and Future Improvements	172
6	Panel Method: Pan Air	177
6.1	Panel Methods	178
6.1.1	The Prandtl-Glauert equation	178

6.1.2	Panel Method Theory	182
6.1.3	Limits of Application of Panel Method	188
6.2	Pan Air	189
6.2.1	Pan Air Capabilities	190
6.2.2	Pan Air Technology	191
6.2.3	Pan Air Geometry Input	192
6.2.4	An overview of configurations analyzed with Pan Air	194
7	Pan Air Pre/Post Processor Module	201
7.1	Pan Air Preprocessor	201
7.2	Pan Air Postprocessor	210
7.3	A simple testcase	217
8	Conclusions	223
	List of figures	227
A	Evaluation of the <i>stiffness matrix</i>	235
	Bibliography	237

Introduction

Nowadays the preliminary design optimization holds an important role in the overall design process. To be efficient, optimization needs a high degree of automation, leaving to the designer only small and fast tasks. The efficiency would be enhanced if the different modules are part of the same environment, that is, with a unique integrated platform.

The preliminary aerodynamic property estimations represent the basis from where to start further investigations and modifications. Although theoretical prediction or lower order aerodynamic solver, like vortex lattice softwares, may be enough accurate for an early preliminary design, when investigating innovative configurations, like the *PrandtlPlane* one, the order of approximation may be unacceptable, mainly because its effect can lead to unreliable results. For example, PrandtlPlane stability is highly sensitive to small aerodynamic load variations, thus more refined analysis should be undertaken. Moreover, the gained experience with the traditional configurations in aeronautics, may not always help.

This and other related problems lead to the need of an efficient and fast code for preliminary evaluation, capable also to export the geometrical shapes to international standard formats, in order to eventually submit more accurate analysis. Since the steps for obtaining an aerodynamic analysis consist in a geometrical configuration definition, a grid generation and a program pre and postprocessing, the main idea is an integrated environment where preliminary aerodynamic design and optimization could be easily and quickly undertaken.

In the first chapter an overview of ASD capabilities and tools is given. ASD (aerodynamic shape design) is a tool written in Matlab language, for geometric shape design of aerodynamic surfaces purposes. Further, it features an internal unstructured mesher. In this chapter, the process of generating configurations is analyzed step by step, mainly from the user perspective; also the use of the tools is shown. A brief description of the integrated unstructured mesher is also given. Finally, the geometric limitations of the code are pointed out: they concern the continuity of the generated NURBS surfaces. The practical drawbacks are discussed, showing the need of a geometric engine improvement.

Second chapter deals with NURBS, a parametric form. First, a brief historical survey is given, followed by the definition of some noteworthy parametric forms, like Bézier and B-spline, fundamental to NURBS understanding. After, NURBS are analyzed, and some important algorithm important for their manipulation and implementation are briefly discusses.

The third chapter is of main importance, since it discusses how NURBS are effectively built over a set data points. The interpolation problem is analyzed in depth, as well as parameterization problems. Then, continuity is discussed from three different perspectives; the effects of continuity on shape fairness are briefly shown. Finally, implementation of new algorithms, based on fairness and variational modeling, and overcoming the previous limitations, is carried out.

Next chapter shows more in detail the capabilities of the new algorithms compared with the old ones. The new GUI, where the interpolation parameters are controlled, is presented. The chapter ends with some pictures of configuration generated with the new algorithms.

Chapter five treats grid generation. After a short introduction on grid generation topic, and especially on the structured mesh, the requirements of the aerodynamic solver Pan Air on the input networks are pointed out. Thus, the logics adopted in the grid generation process and all its underlying reasons are discussed in depth. The parameters which control the mesh generation, and the graphical interface are finally presented, as well as some examples of networks generated.

Chapter six introduces the panel method theory, from the equations to the field

of application. Then Pan Air, the panel method developed at Boeing and NASA, and chosen to be integrated with this code, is described; in particular an overview of its capabilities, its requirements, and some of its applications are presented.

The integration of Pan Air within the platform is pointed out in chapter seven. Both the preprocessor and the postprocessor are presented in details, with the aid of pictures of analyzed configurations. A final simple test case is submitted, with the aim of checking Pan Air predictions and its grid sensitivity.

1

ASD: Aerodynamic Shape Design

1.1 Introduction

ASD (Aerodynamic Shape Design) is a fully parametric, modular, scriptable aerodynamic surface generator. One of ASD peculiarity is the easy and fast new configuration generation and existing configuration modification capability, due to the parametric approach and the user friendly graphical interface. Another trademark is the ability to generate non-conventional configurations, such as the *PrandtlPlane* one, whose aerodynamic shapes can not be easily designed using conventional CAD softwares. Together with this geometric capabilities, ASD is provided of a meshing tool, capable of producing triangular meshes and hybrid tri-quad meshes suitable both for aerodynamic and electromagnetic simulations and featuring automatic wakeline recognition for aerodynamic panel solvers. ASD surface mesher can export various formats from panel neutral *.dat* files to standard *.stl* tri mesh. The output of the surface generator can be exported as set of trimmed NURBS into an IGES file or fed into meshing tool built into the code. Some useful tools aid the users to sketch the geometric shapes, such as the airfoil manager, the section manager, the Naca airfoil manager and the flap sketcher.

Due to its characteristics, ASD main field of application is the early stage of aerodynamic design. In this stage, in fact, ASD can quickly generate a huge number of configurations, making it possible to analyze many different layouts during the

initial optimization process.

The ASD code has been written in Matlab, and requires, a part from Matlab, the Spline Toolbox.

Whereas more details are required, refer to [1; 2; 3; 4].

1.2 ASD Surface Generator

1.2.1 The Main Window

As soon as the program interface starts, the main window (fig.1.1) is brought on focus. The main window is made up of 8 list boxes and a menu bar. The list boxes show the features currently in memory, and allow the user to remove or modify existing features or to add new ones. Double clicking on a feature opens the corresponding modify window. In every list box feature are listed showing the feature tag and ordered as they have been defined. The features that contain surface data are marked with a “ ==> ” right before the tag. Every list box allows multiple selections. The *Generate selected* and *Mesh selected* pushbuttons pass the selected surfaces to the surface generation function and to the meshing tool respectively. It is to be noted that only feature that contain surface data can be meshed. The *View* pushbutton opens the *Surface Viewer*.

1.2.2 The Features and the Feature Windows

It is worth a note that only body, wing and inlet/outlet features are independent, being the remaining features just connections. Each feature can be added and edited through its own window, which appears selecting the *Edit* voice on the menu, or just double-clicking on the feature.

1.2.3 The Body Feature

The *Add/Modify Body* interface enables to specify all the parameters involved in the generation of a Body feature. The interface for this feature is shown in fig. 1.2. Using these parameters, the code generates a skeleton of the body and then

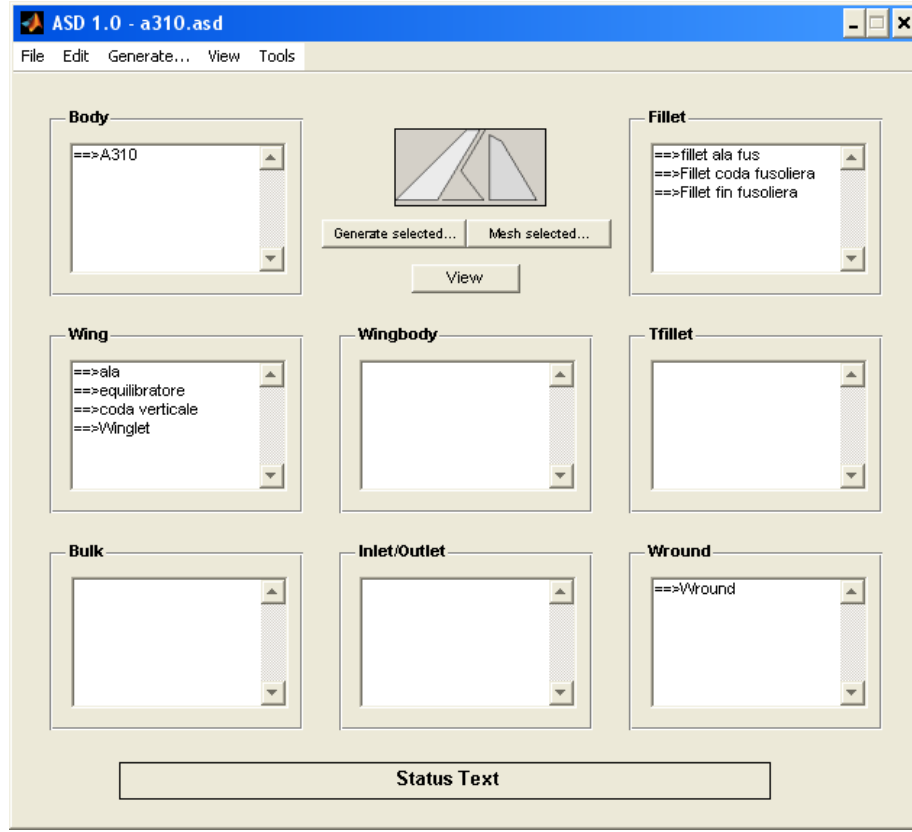


Figure 1.1: The ASD main window.

interpolates it using a bi-cubic NURBS. A scheme of the skeleton along with the body lines is shown in the picture 1.3. The *Tag* field lets the user specify a tag for the feature which will be stored along with all the parameters and will be shown in the ASD main window list boxes. The *Upper/Lower Section* list box contains the list of used body sections. Using the *Add* and *Remove* pushbuttons the user can add or remove body sections selecting appropriate *.dat* files. Next to each list box there is the number of section files listed. The number of upper and lower sections must be equal for the surface generator to work. The fields *X0*, *Y0*, *Z0* are the coordinates of the first section of the list. The coordinate system used in ASD has the positive x-axis in the direction of the fuselage, starting from the nose, the positive y-axis is directed spanwise from the fuselage toward the wing tip, while the positive z-axis is normal to both x and y and directed upward. If *Y0* is set to 0 the body is assumed

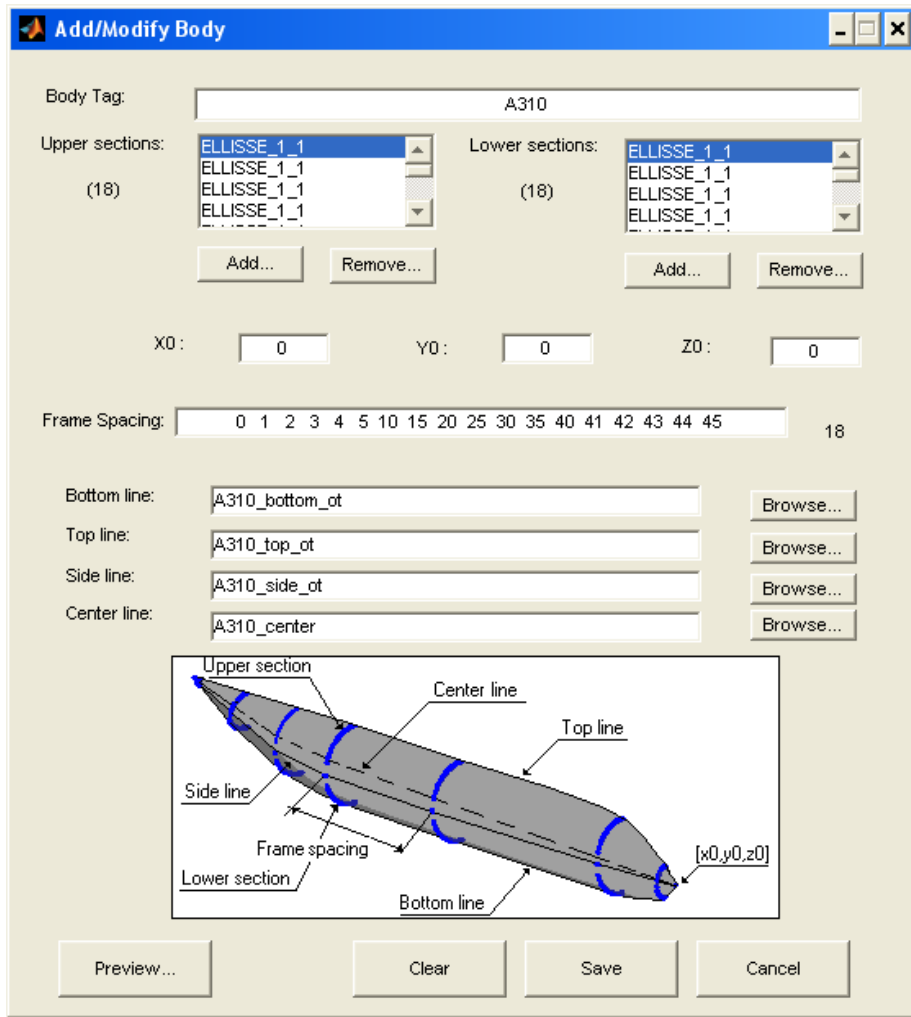


Figure 1.2: The body feature window.

to be on the symmetry plane and no other action is taken, otherwise the shape is symmetrized on the plane $y=Y0$. The field *Frame Spacing* contains the x coordinates of the body sections relative to $X0$. The fields *Body lines* are used to specify the .dat file containing body lines data. The user can both browse for the file or input the complete path. Pressing the *Preview* pushbutton will plot on a separate figure a preview of the skeleton and of the resulting interpolated surface, as shown in fig. 1.4.

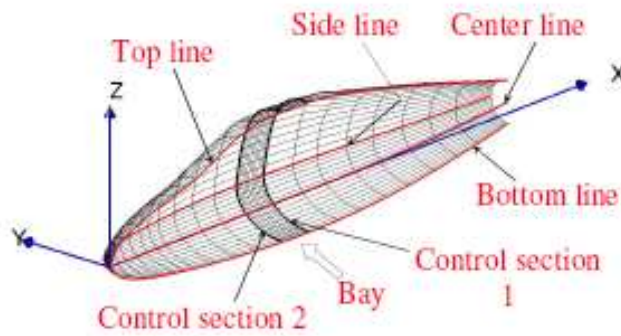


Figure 1.3: Parameters involved in the definition of the body feature

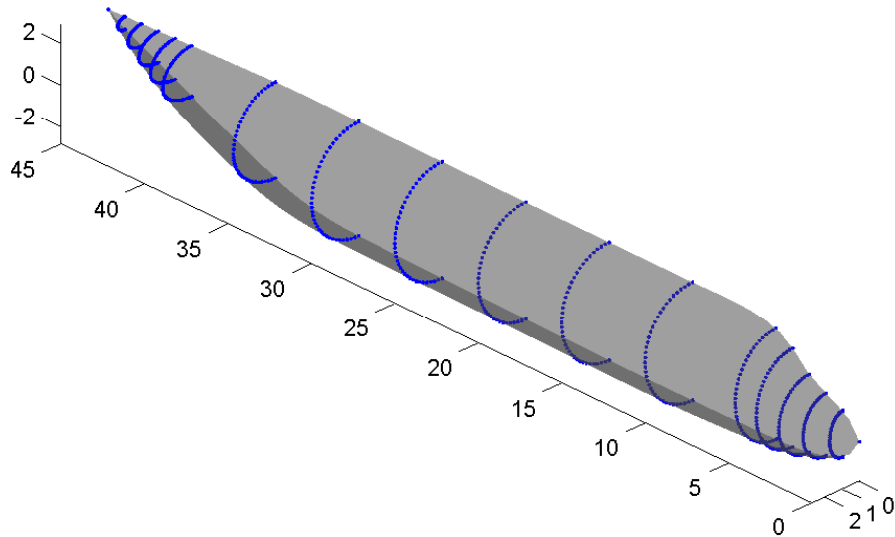


Figure 1.4: Preview of the skeleton and resulting interpolated surface for a body feature.

1.2.4 The Wing Feature

The *Add/Modify Wing* interface lets to specify all the parameters involved in the generation of a Wing feature. The interface for this feature is shown in fig.1.5. Using these parameters the ASD code generates a skeleton of the wing which is then interpolated with a linear-cubic NURBS where the linear direction is spanwise.

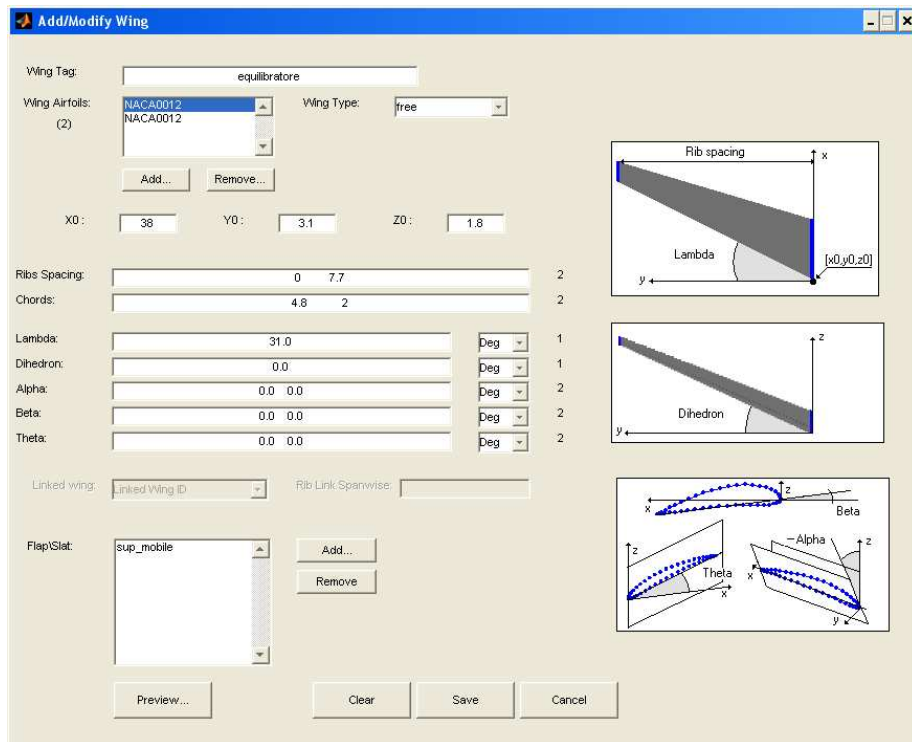


Figure 1.5: The wing feature window.

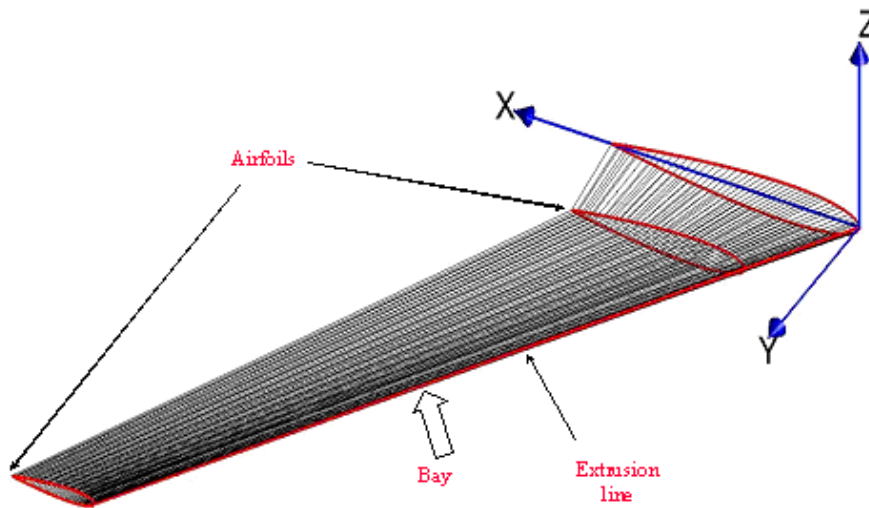


Figure 1.6: Wing components

An example is shown in fig. 1.6. The *Wing Airfoils* list box contains the list of used airfoils. Using the *Add* and *Remove* pushbutton the user can add or remove airfoils selecting appropriate *.dat* file. The *Add* pushbutton opens a standard get dir dialog which defaults on the standard UUC AIRFOIL DATABASE directory. Upon selecting a directory, the *Airfoil Viewer* window opens and lists all the airfoils

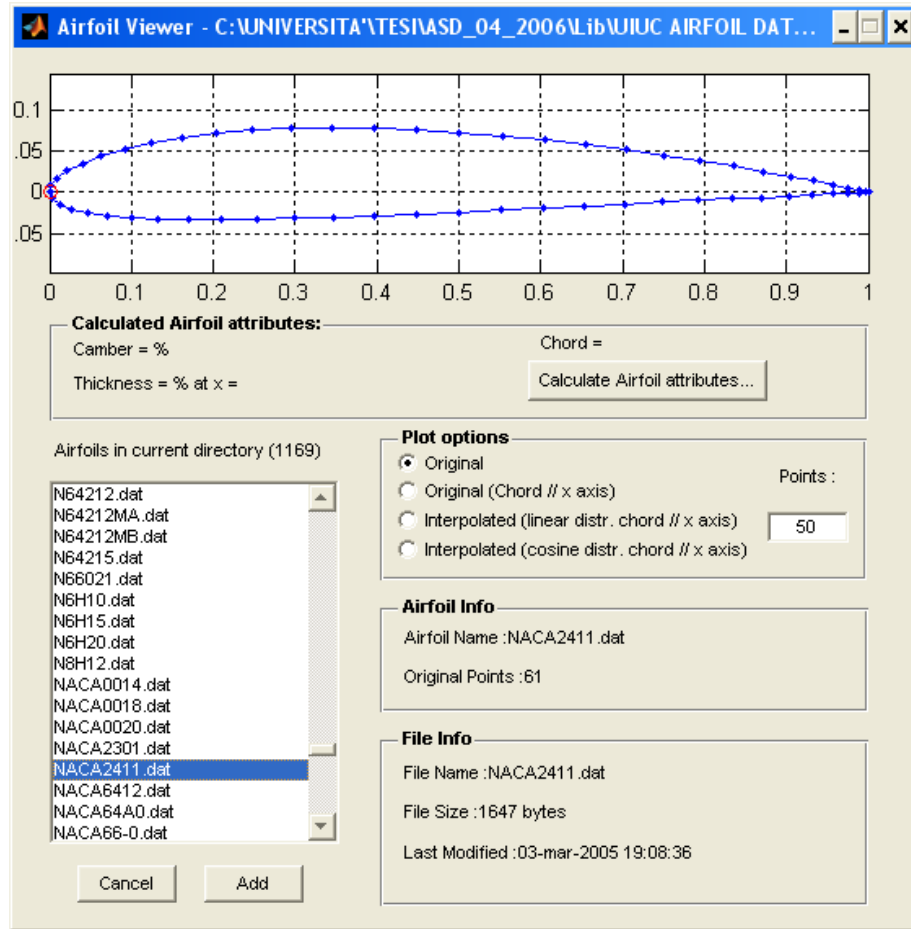


Figure 1.7: The Airfoil Viewer window

found in this directory (fig.1.7). The Airfoil Viewer enables the user to see original points in the file selected, re-interpolated airfoil using an arbitrary number of points. It also enables the user to calculate camber and maximum thickness of the airfoil selected. All of the plot options act only on the visualization of the airfoil

on this window. The *airfoil loader* function loads the original set of points, closes the trailing edge if found open, locates the leading edge, sets the chord parallel to the x-axis and of unitary length and re-interpolates the points using a chord-wise cosine distribution of 50 points. The same set of action is performed when the **Interpolated (cosine distr. chord // x axis)** radiobutton is selected (see also the *Airfoil Generator* and the *Airfoil Manager* sections). The popup *Wing type* is used to select if the wing being created is to be freely placed, or is to be connected with other wings. When the wing is free, the position and shape of the wing is derived from the information of **X0**, **Y0**, **Z0**, **dihedron** and **sweep angle**. If the feature is linked to another wing then the last airfoil is linked to a spanwise section of another wing. The user must thus specify which airfoil on the linking wing is to be matched, along with the ID of the linked wing. This is used to constrain a wing to follow the wing it is linked to. This feature is useful to define T-tails.

The **X0**, **Y0**, **Z0** are the coordinates of the nose of the first airfoil on the list (assumed to be the root airfoil). If **Y0** is set to 0 and the dihedron angles are all 90° or -90° then the wing is assumed to be on the symmetry plane, so only the upper or lower side of the airfoil is considered.

The *Ribs Spacing* field contains the spanwise position of the airfoils relative to the first airfoil. The *Chords* field contains the chords of the airfoils listed, *Lambda* and *Dihedron* contain the sweep and the dihedron angle of the defined bays; *Alpha*, *Beta*, *Theta* contain information on the rotation angle of the airfoils around the three principal axes, *x*, *y* and *z*, respectively.

If the wing type is set to “linked”, then the *Linked Wing ID* popup menu is used to select which wing is to be linked to. Obviously there must be some wing defined to define a linked wing, as self referencing is not allowed. The *Rib Link Spanwise* field is used to select the spanwise section position on the wing is to be linked to. The user can make use of expressions involving other wing feature, as long as they have been already defined, which are evaluated before saving the result. For example the expression

$$wing(2).y0 + wing(2).ribs_spacing(2)$$

evaluates to the y position of the second airfoil of wing 2. Next to the field there is

a textbox where the result of evaluation is displayed.

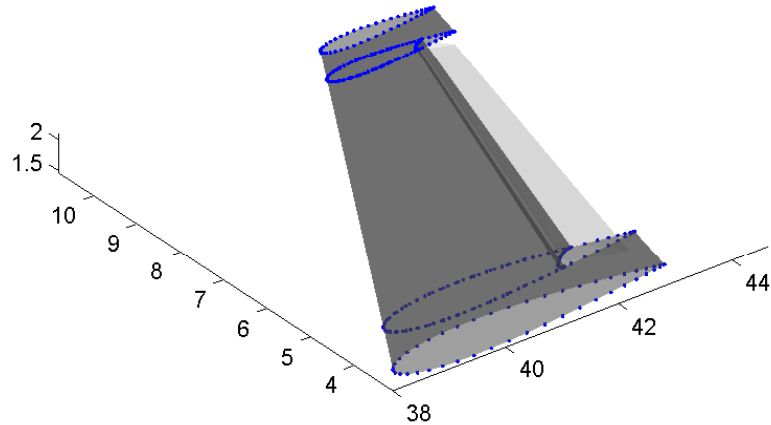


Figure 1.8: The Preview wing feature

Pressing the *Preview* pushbutton will plot on a separate figure a preview of the skeleton and of the resulting interpolated surface, as shown in fig.1.8

Every wing feature can have a variable number of control surfaces defined. They are all listed in the *Flap/Slat* list box. Double-clicking on one of the listed features opens the corresponding *Add/Modify Mobile Surface* window (fig.1.9).

Within the aim of this window, it is possible to generate mobile surfaces on a wing, both on LE (slat) and TE (flap, aileron etc). The mobile surface must be defined within one single bay. Parameters describing the mobile surface are the *ribs spacing*, to set the spanwise position, the *gap*, which specifies the relative gap, in percentage of the length of the mobile surface, to leave between the mobile surface and the main wing surface, the *chords*, to set the chords at fist and last spanwise section.

The mobile surface sectional geometry is created by a tool called *Flap Sketcher* (section 1.4.4).

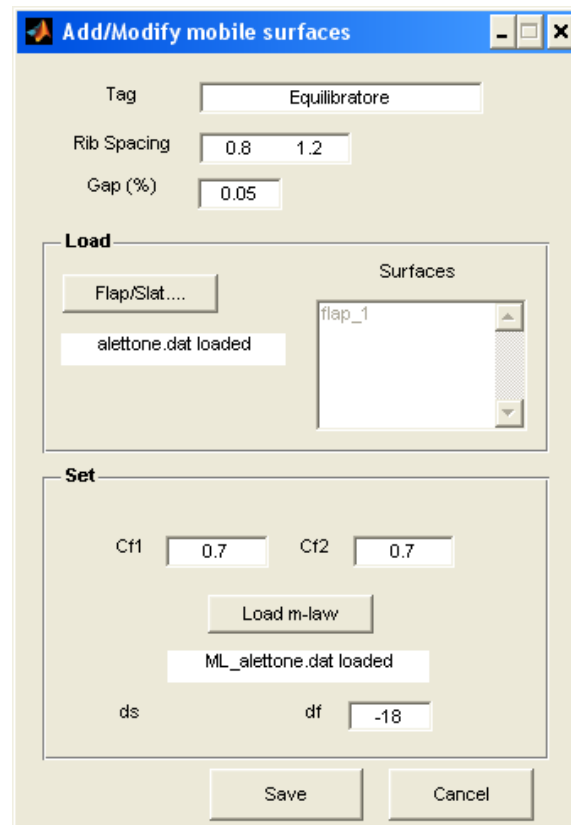


Figure 1.9: The *Add/Modify Mobile Surface* window



Figure 1.10: Mobile surface deflection

Finally, a flap/slat motion law is stored in a plain text *.dat*, so that when selecting a slat and flap deflection angle, the mobile surface configuration is properly achieved.

1.2.5 The Bulk Feature

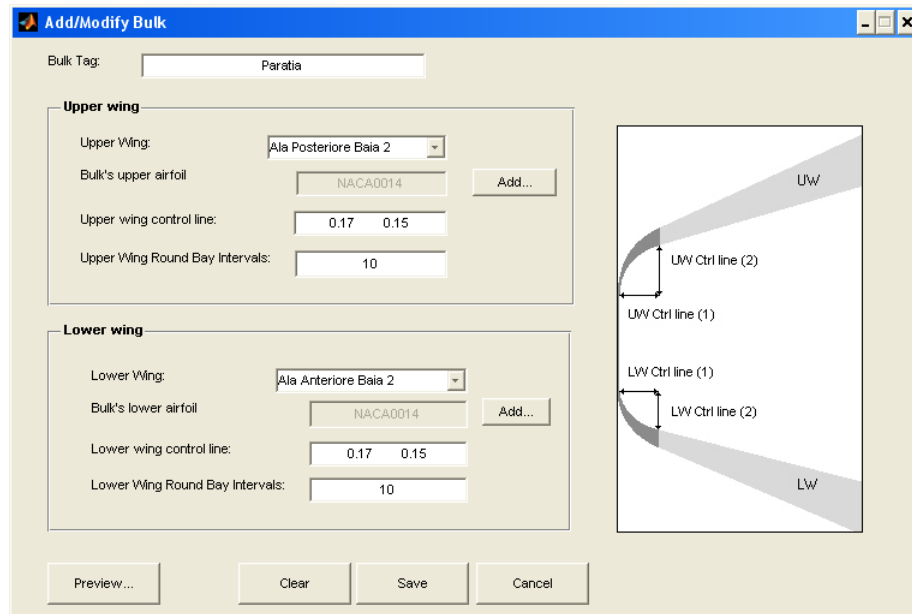


Figure 1.11: The bulk feature window

The *bulk* feature is specifically designed to link the two wing tips of a biplane configuration as shown in fig.1.12. The bulk feature creates a connection element

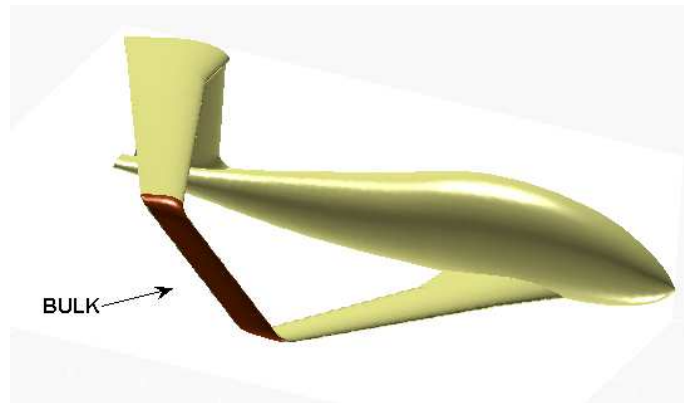


Figure 1.12: A bulk example

between two wings, by smoothly transform one airfoil to the other, giving the user the ability to select the bulk start and end airfoil. The airfoil transformation is linear along the bulk midline (see fig.1.13).

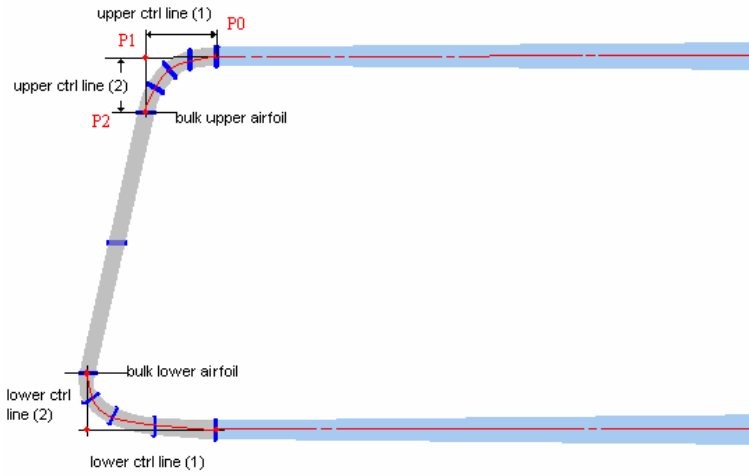


Figure 1.13: The bulk control parameters

The popup menu *Upper Wing* and *Lower Wing* are used to select which wing is to be the start(end) wing. In the *Bulks upper (lower) airfoil* the user specify the airfoil to be used for the upper (lower) end of the bulk feature. It can be selected using the *airfoil viewer*. The *Upper (Lower) Wing Control Line* specify the vertical and horizontal distance of the linear part of the bulk feature from the wing last airfoil. Finally, the *Upper(Lower) Wing Round Bay Intervals* contains the number of sections that have to be created along the round from the start wing to the beginning of the linear part of the bulk. The minimum number is two. The intermediate airfoils are created linearly spacing and morphing the first airfoils to the last one on the other side of the bulk.

Pressing the *Preview* pushbutton will plot on a separate figure a preview of the skeleton and of the resulting interpolated surface in dark transparent gray, and the two wings that take part at the generation in light transparent gray, as shown in fig.1.14

1.2.6 The Wingbody Feature

The WingBody feature is a wing-like surface to blend to a generic root section of a wing. The result is a smoothly blended surface that extends the wing to the

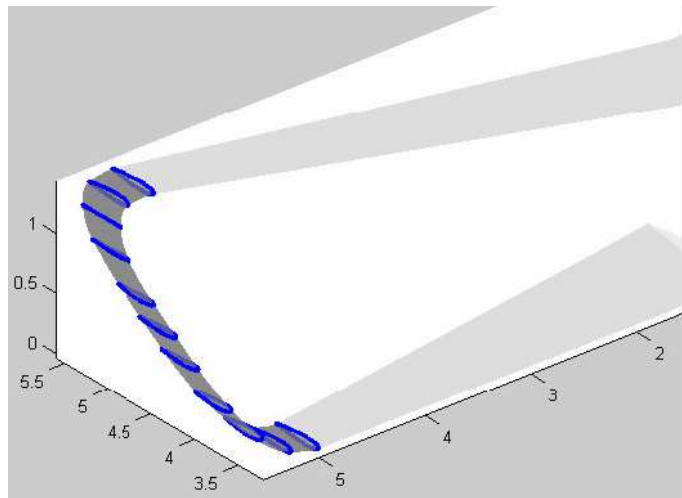


Figure 1.14: The Preview Bulk figure

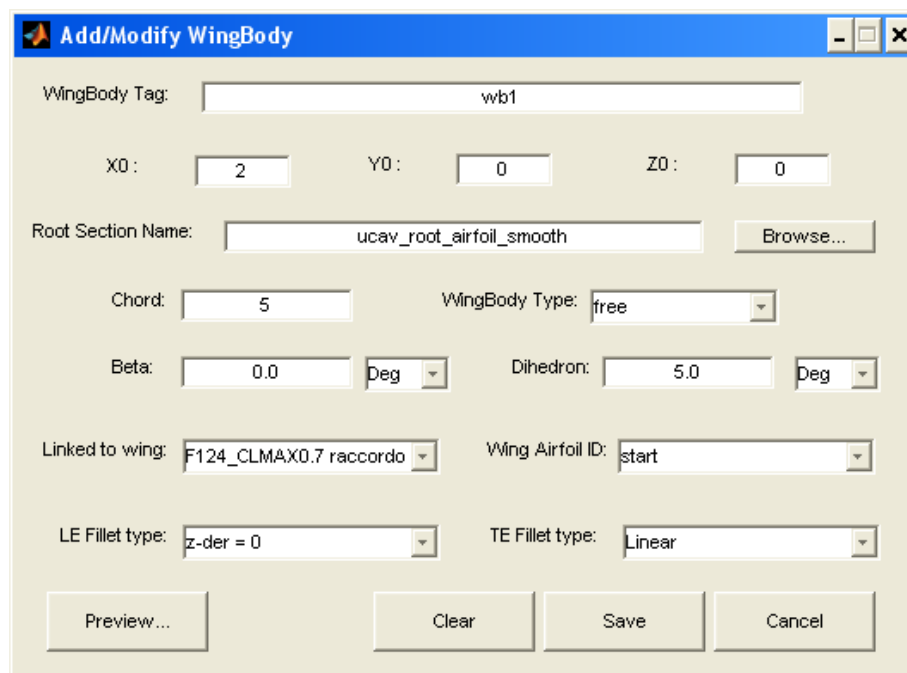


Figure 1.15: The wingbody feature window

section specified by the user using different options to control section smoothness and generation. As a wing-like feature, it retains several definitions and parameters typical of the Wing feature.

The *Add/Modify WingBody* interface (fig.1.15) enables to specify all the parameters involved in the generation of a wingbody feature. The $X0$, $Y0$, $Z0$ are the coordinates of the foremost point of the root section. The editable field *Root Section Name* is used to specify the *.dat* file containing root section data. The file should contain an airfoil-like set of bi-dimensional coordinates, with unit chord. The chord of the root section is specified in the following field. The *Beta* and *Dihedron* editable fields contain information on the rotation angle of the airfoils around the y axis and the dihedron angle of the wingbody respectively.

The *Linked to Wing* and *Linked Airfoil ID* popup menu list all the wings currently defined and lets the user select the one to which the wingbody feature is to be linked to, and lets the user select which airfoil is the one the wingbody feature is to be linked to. Finally, the *LE Fillet Type* and *TE Fillet Type* popup menus allow the user to control the type of leading edge and trailing edge connection. The two conditions are independent and are linearly blended one into the other moving from the LE to the TE. The 3 types of connections are:

- 1 $z\text{-der} = 0$: the derivatives at the wing interface are preserved, thus preserving the smoothness of the surface; the derivatives at the root section are the same as the wing derivatives except for the z component which is set to zero;
- 2 $z\text{-der} = 0, x\text{-der} = 0$: the derivatives at the wing interface are preserved, thus preserving the smoothness of the surface; the derivatives at the root section are the same as the wing derivatives except for the z and x component which are set to zero.
- 3 Linear: all derivatives both on the wing interface and on the root section are set equal to the vector connecting the two.

Fig. 1.16 and 1.17 clarify the connection modality.

1.2.7 The Inlet/Outlet Feature

The inlet/outlet feature is a body-like feature specifically designed to generate the cavity of an inlet or outlet. The inlet/outlet surface is built by interpolation of a set

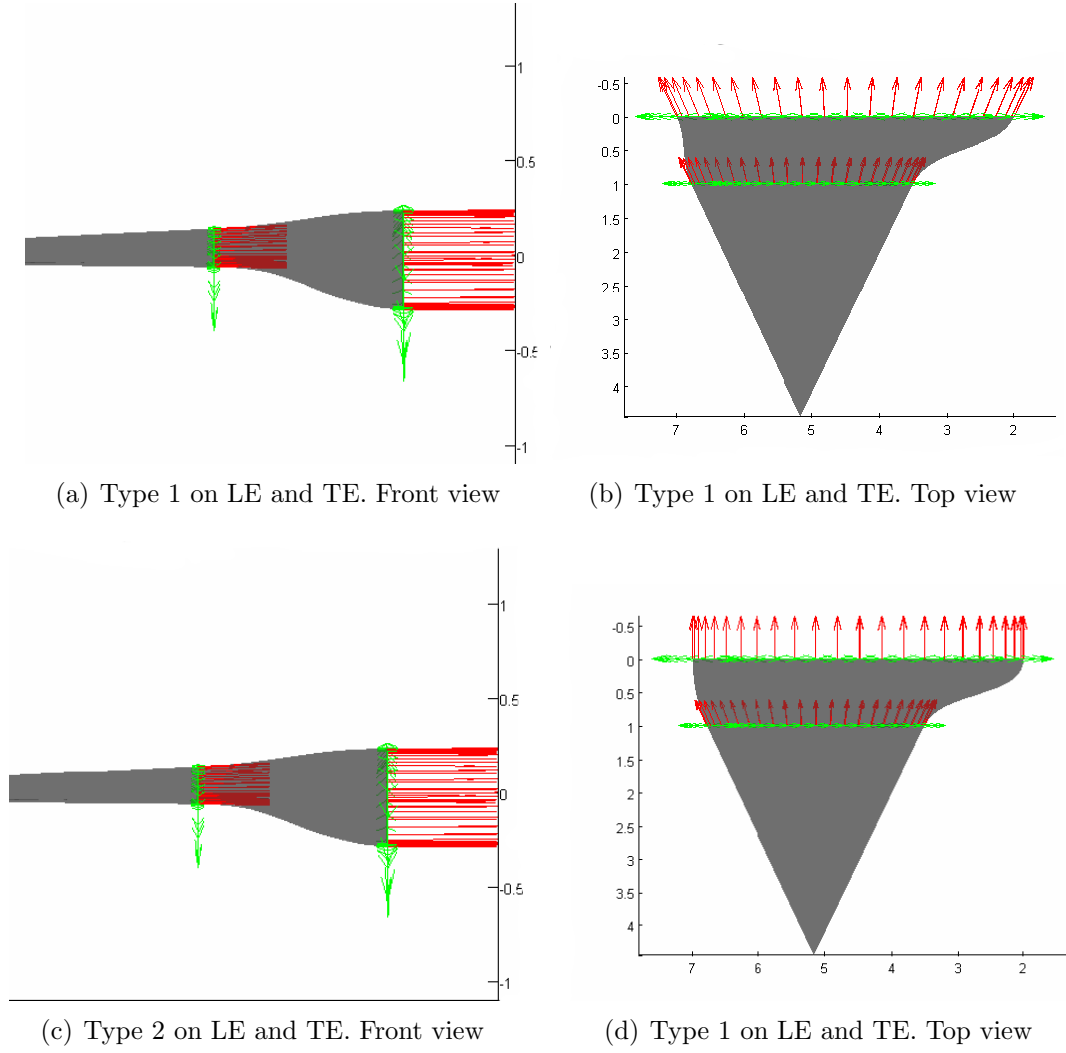


Figure 1.16: WingBody creation, type 1 (a) and (b), type 2 (c) and (d).

of frames. The interface gives the user the control over the spatial position of these frames, allowing a quick and easy way to parametrically modify the surface. The generation of the inlet/outlet feature also takes care of intersecting the feature with the $y = 0$ coordinate plane and body, wing, wingbody features. All other features are not checked for intersection.

The *Add/Modify Inlet/Outlet* interface lets to specify all the parameters involved in the generation of a the feature. Using these parameters, the code generates a skeleton of the surface and then interpolates it using a bi-cubic NURBS. A scheme

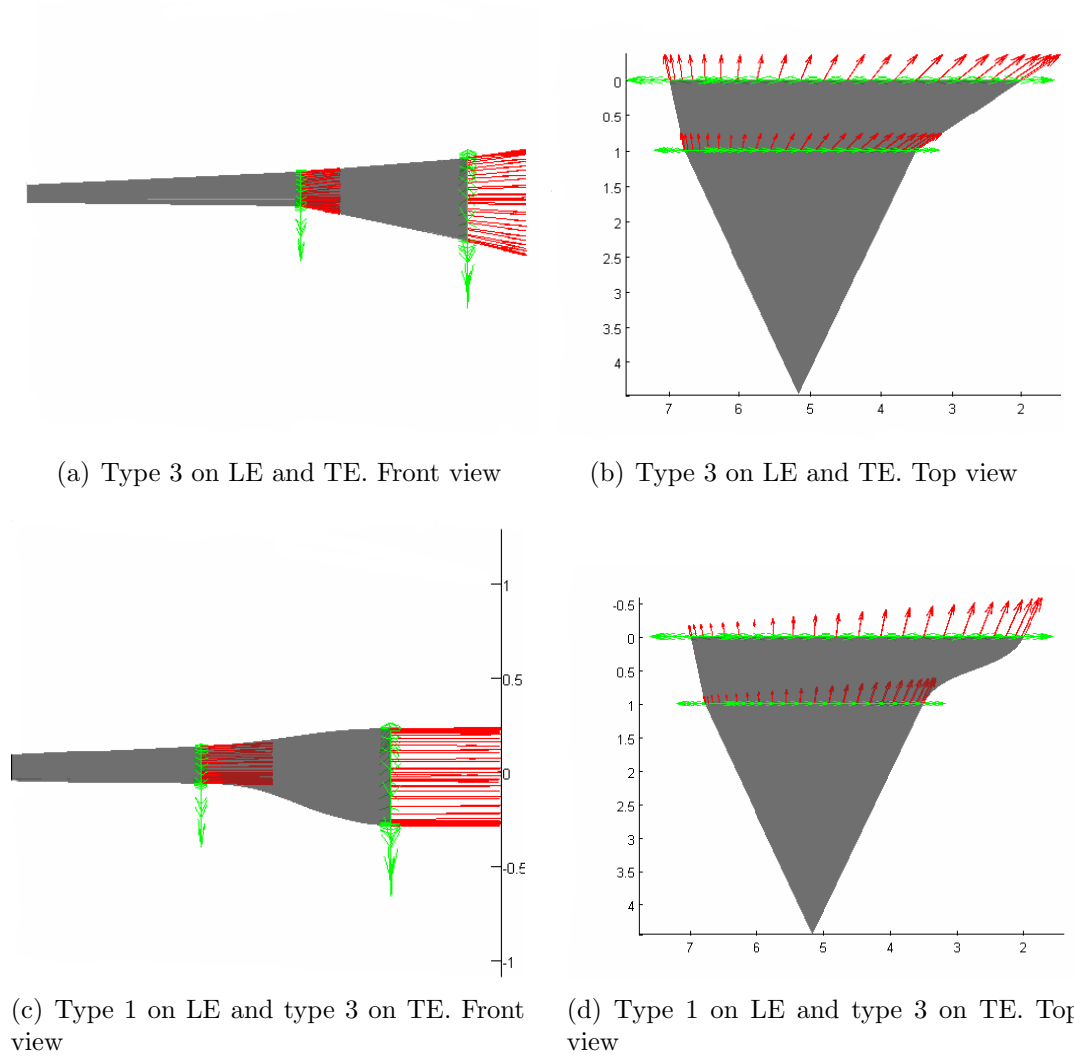


Figure 1.17: WingBody creation, type 3 (a) and (b), type 2 on leading edge and type 3 on trailing edge (c) and (d).

of the skeleton along with the center line is shown in fig.1.19. The *Section* list box contains the list of sections in use. Using the *Add* and *Remove* pushbuttons the user can add or remove sections by selecting appropriate *.dat* files. The section files are normalized plane shapes in the $[-1,1] \times [-1,1]$ domain. The actual shape of the section is derived from these by applying the scale factor independently on each semi-axis. (see fig.1.21).

The $X0$, $Y0$, $Z0$ fields represent the coordinates of the first section of the list.

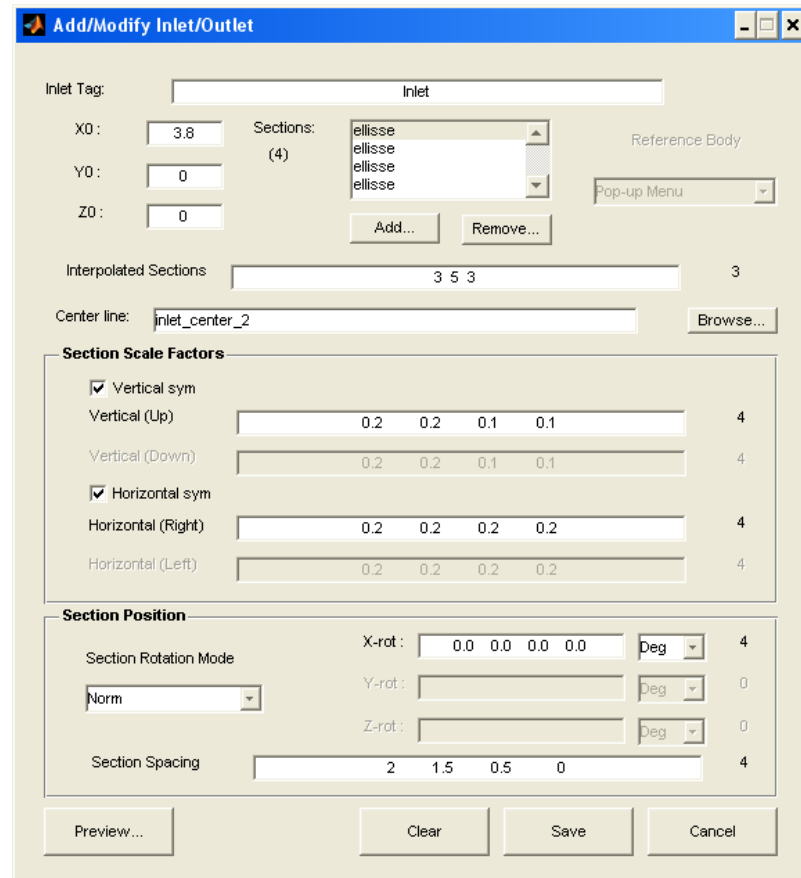


Figure 1.18: The inlet/outlet feature window

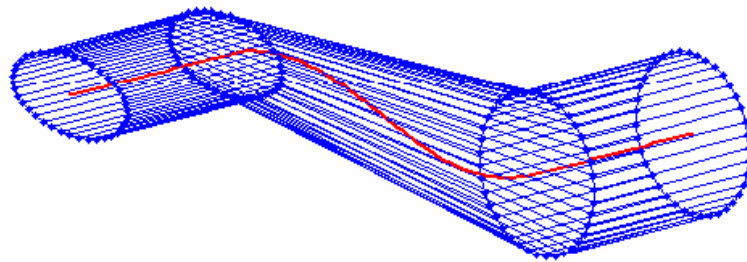


Figure 1.19: The inlet/outlet skeleton and center line

Unlike the body feature, the inlet/outlet always generates the complete surface of the inlet, regardless of its relative position to the symmetry plane. The surface generator function takes care of all the intersection both with the symmetry plane

and the other features, and writes the inlet/outlet surface as a trimmed NURBS. The *Interpolated Sections* field contains the number of extra sections the code has to add between each pair of sections. The effect of this automatic adding feature is to force the feature to follow the centerline without changing shape. In fig.1.20 the effect of changing the number of interpolated section is clearly visible, being the red sections the ones defined by the user, and the blue the ones added by the code. The *Center*

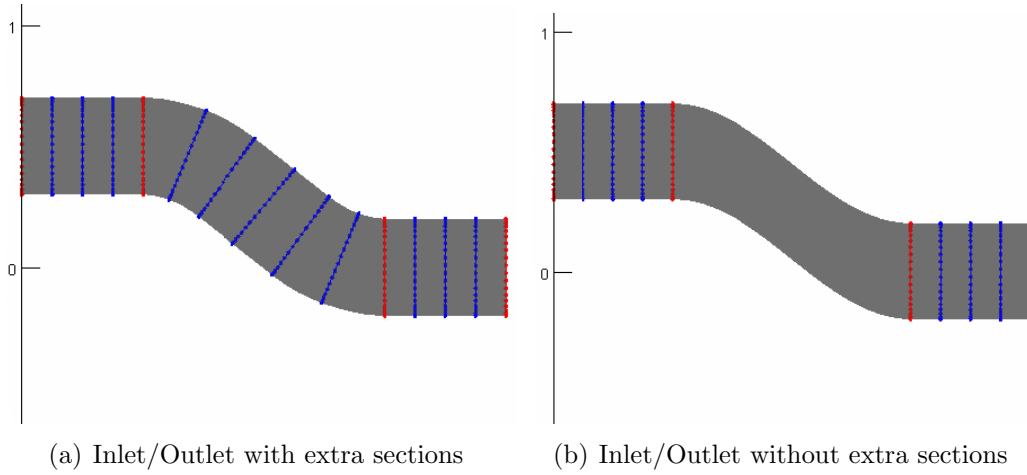


Figure 1.20: Effect of the interpolated sections in the geometry of the Inlet/Outlet.

line editable field is used to specify the *.dat* file containing center line data. The centerline does have a direction. Inside the code it is essential to know which side is the one to keep in order to write trims properly. As a convention, the centerline is supposed to start from the inside and going outside, so that in the case of an engine inlet and outlet, the first section is the one on the engine face, for both the inlet and the outlet. The center line has another peculiarity that the body lines do not have: its defined by 3D points, thus allowing a single line to define a 3-dimensional path for the feature.

The *Section Scale Factors* panel give access to the section scale factors. Since the sections are defined in a normalized square $[-1,1] \times [-1,1]$ the scale factors give the user the possibility to obtain very different shapes of section starting from a simple *.dat* file. The scale factors must be specified for both axes and for both the positive and negative semi-axis (fig.1.21).

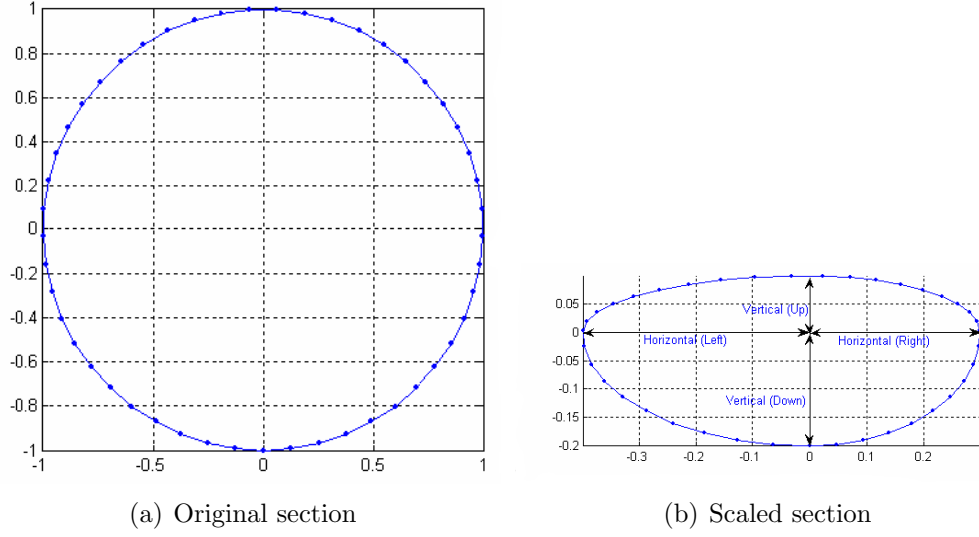


Figure 1.21: Inlet/Outlet Section Scale Factors.

The *Section Position* panel groups together all the information about section rotation and spacing. In the *Section Rotation Mode* the user select the mode of rotation of the sections: the **User** mode requires the user to specify for every section the 3 rotation angles around the 3 principal axes, the **Norm** mode require the user to input only the x-rotation (the rotation of the section in its plane), while the other two rotation are calculated automatically to keep the section plane normal to the center line, the **x** mode requires the user to input only the x-rotation (the rotation of the section in its plane), while the other two rotation are calculated automatically to keep the section plane normal to x-axis line. The rotation angles are specified in the *X-rot*, *Y-rot*, *Z-rot* editable fields.

Finally, in the *Section Spacing* field the x position of the center of the sections relative to the center line are inserted.

1.2.8 The Fillet Feature

The fillet feature is used to create the fillet between a wing and a body. The *Fillet type* field contains the type of fillet to create. If set to **smooth** then the fillet will be a round type fillet tangent to the fuselage on the leading edge, and linearly rotating

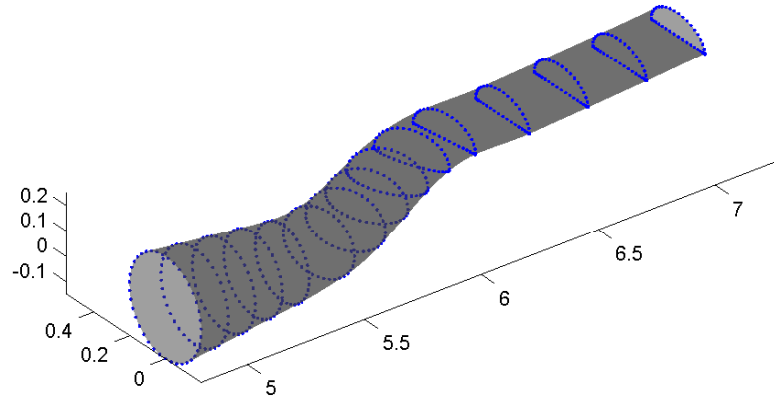


Figure 1.22: The inlet/outlet preview

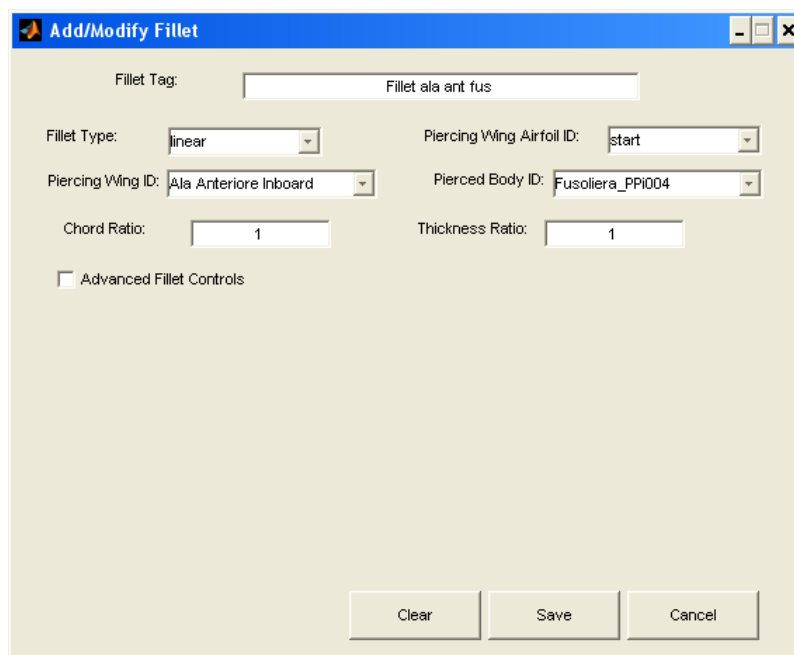


Figure 1.23: The fillet feature window

into perpendicular intersection towards the trailing edge (fig.1.24). If set to `linear`,

the fillet is just a linear blending surface from the wing to the surface.

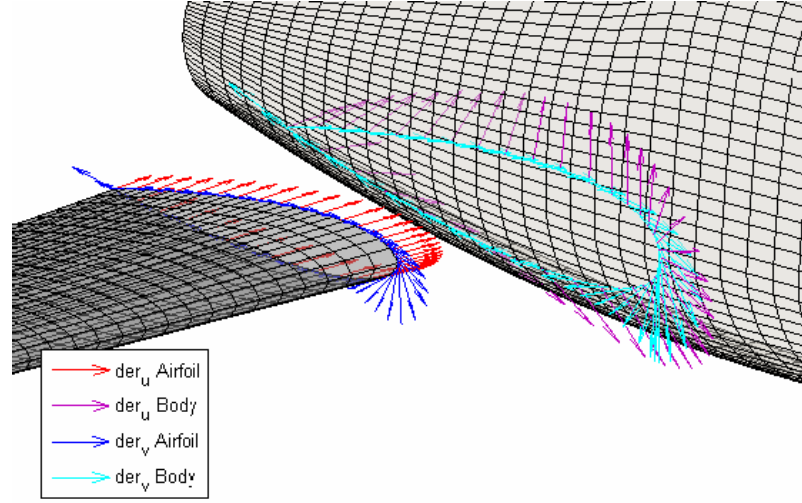


Figure 1.24: Smooth fillet derivatives on wing and body

In the *Piercing wing ID/Pierced body ID* the user selects which are the bodies involved in the fillet creation. The *Piercing wing airfoil ID* is used to specify if the fillet starts from the root or the tip airfoil. It is useful when creating double fuselage configuration, where the fuselage is connected to the center wing by a fillet starting from the tip of the middle wing. There are two ways to control the geometry of a fillet feature: a basic and an advanced.

The basic controls strips down to only 2 parameters: a chord ratio and a thickness ratio. If the **advanced Fillet Controls** checkbox is checked, advanced control parameters are visible. With the **Basic** fillet parameters, the user directly controls the dimension of the hole on the body, while its shape is controlled by the base wing airfoil (fig.1.25).

With the **Advanced** fillet parameters, the user directly controls the dimension and the shape of the hole on the body, by providing an auxiliary airfoil and its position (fig.1.26). Both these methods affect the fillet creation by giving different ways to locate the intersection points on the body surface. Once these points are determined, the fillet creation follows the same path, by creating a smooth or linear surface depending on the user selection. The *Chord Ratio* is a scaling parameter

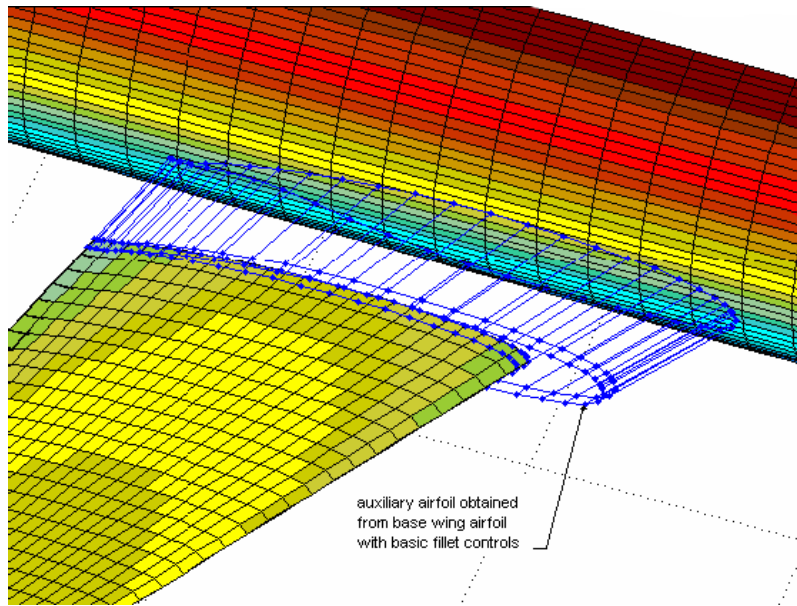


Figure 1.25: Creation of fillet with Basic controls

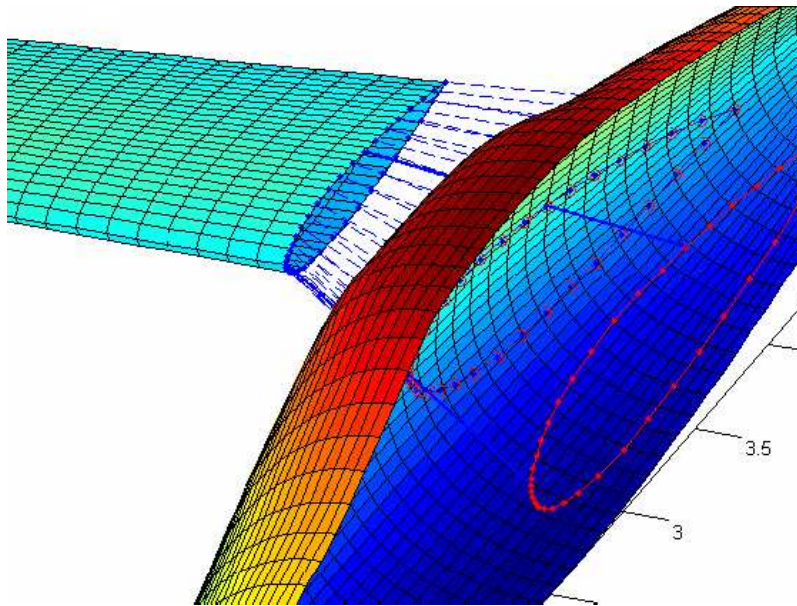


Figure 1.26: Creation of fillet with Advanced controls

used to define an auxiliary airfoil to intersect with the body feature. This parameter is the chord-wise scaling factor to be applied to the base wing airfoil to obtain the new airfoil (fig.1.25). The *Thickness Ratio* is a scaling parameter used to define an

auxiliary airfoil to intersect with the body feature. This parameter is the thickness-wise scaling factor to be applied to the base wing airfoil to obtain the new airfoil. In the *Advanced Fillet parameters* the user is requested to set up the auxiliary airfoil that defines the shape of the fillet. The fillet is indeed defined connecting the wing to this auxiliary airfoil located according to these parameters. Figure 1.26 shows the wing airfoil, the wing extension to intersect the fuselage, and the auxiliary bay defined by the wing airfoil and the auxiliary airfoil. The result of the intersections between this bay and the fuselage is the starting point to construct the fillet NURBS surface.

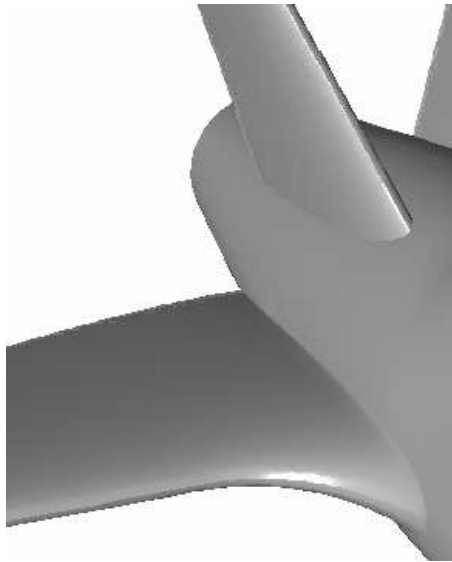


Figure 1.27: Example of smooth and linear fillet on the same configuration.

1.2.9 The Tfillet Feature

The Tfillet feature is used to create the fillet between two wings and works as the fillet feature, except that only the basic controls are available. The field *TFillet type* contains the type of fillet to create. If set to **smooth** then the fillet will be a round type fillet tangent to both wings, with a variable radius from the leading edge to the trailing edge (fig. 1.29). If its set to **linear**, the fillet is just a linear blending surface from the intersecting wing to the other.

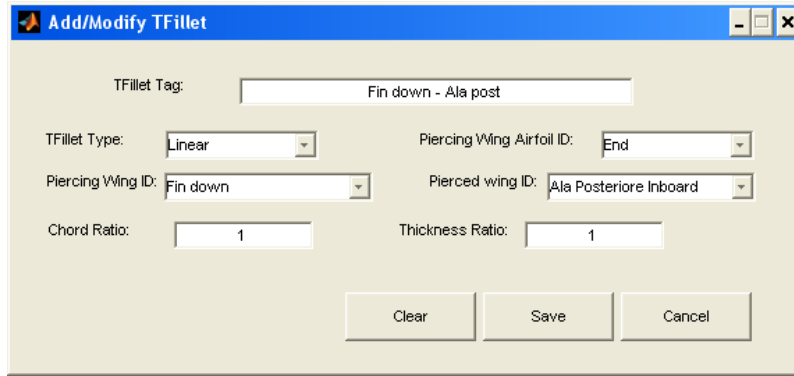


Figure 1.28: The TFillet feature window

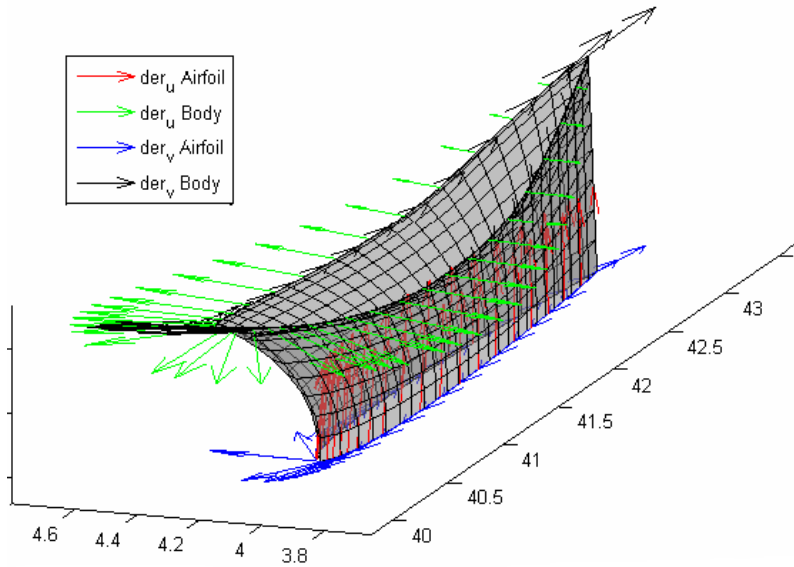


Figure 1.29: Smooth fillet derivatives on wing TFillet

The *Piercing wing ID*/*Pierced wing ID* specify which are the wings involved in the fillet creation, and the *Piercing wing airfoil ID* distinguish if the fillet starts from the root airfoil or the tip airfoil.

The *chord ratio* and *thickness ratio* are scaling parameters used to define an auxiliary airfoil to intersect with the wing feature; they are respectively the chord-wise and thickness-wise scaling factor to be applied to the base wing airfoil to obtain the new airfoil (fig.1.30)

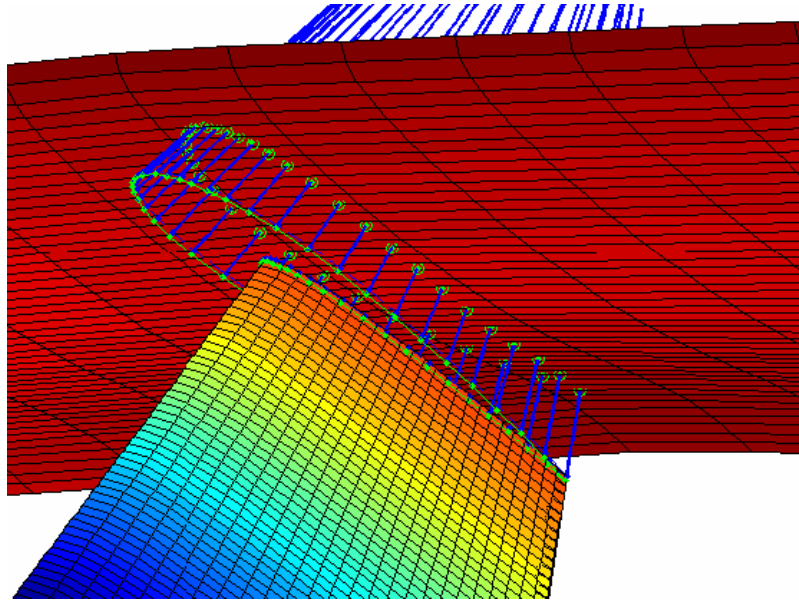


Figure 1.30: Auxiliary airfoil definition and surface intersection

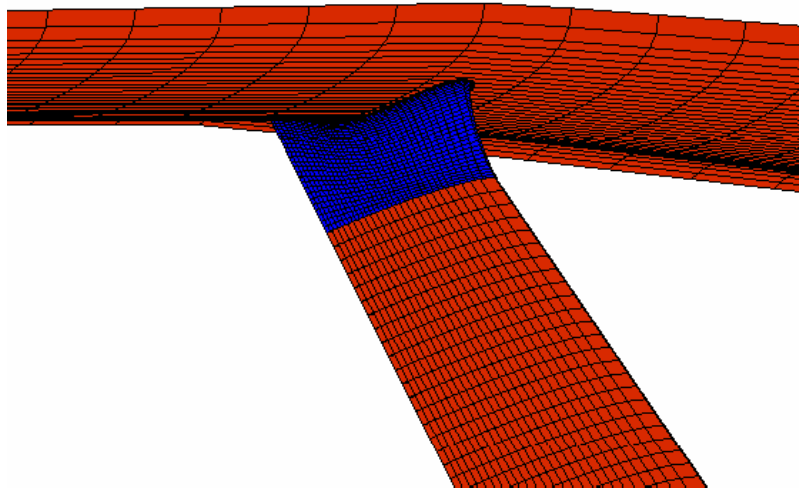


Figure 1.31: Smooth TFillet on T-tail configuration

1.2.10 The Wround Feature

The wround feature is used to create a connection element between two wings, connecting their extreme airfoil with a smooth continuous surface. There is no control over the shape of the wround, depending only on the position of the two airfoils

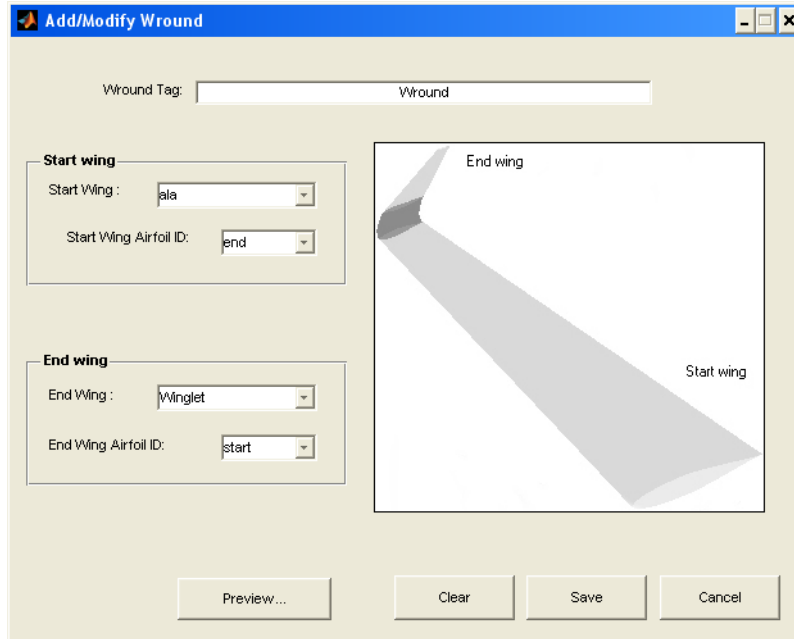


Figure 1.32: The Wround feature window

involved and the surface derivatives on them. To create a Wround feature at least 2 wings must be defined. In the *Start Wing panel* the user selects from a dropdown menu the first wing and the airfoil to be connected to the second wing. The same is done with the *End Wing panel* for the second wing. Pressing the *Preview* push-button will plot on a separate figure a preview of the skeleton and of the resulting interpolated surface in dark transparent gray, and the two wings that take part at the generation in light transparent gray, as shown in fig.1.33.

1.2.11 Viewing the Results

Every surface generated is passed to the Surfaces Viewer window and is automatically selected for plotting. The user can now select which features to plot and what kind of plot to perform. The options are:

1. **Render** : the surfaces are rendered in different color depending on the type of surface using a metal like look (fig.1.35).

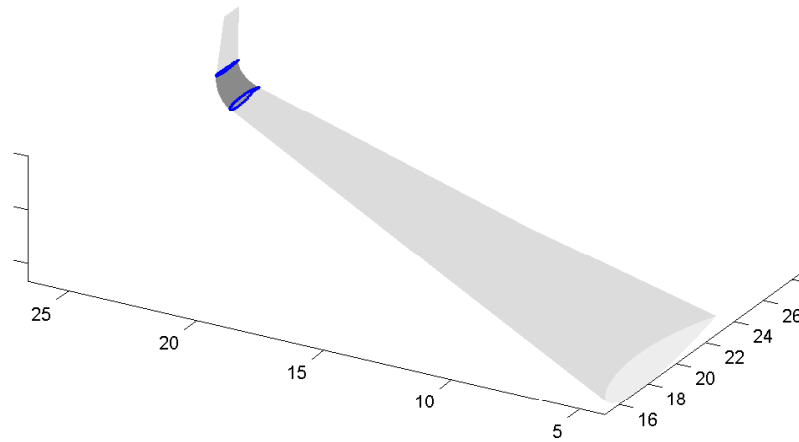


Figure 1.33: The wound feature window

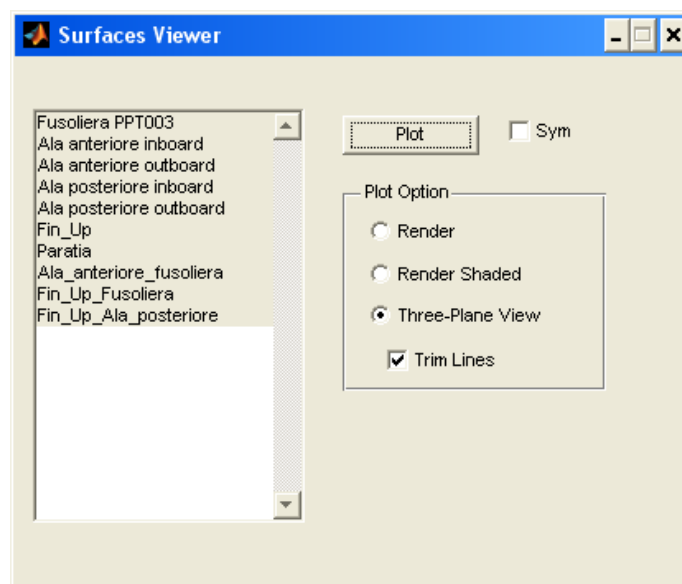


Figure 1.34: The *Surface Viewer* window

2. **Render Shaded** : the surfaces are rendered in different color depending on the type of surface using a metal-like look and a half transparent effect(fig. 1.36).
3. **Three Plane View**: the surfaces are rendered in gray using a metal-like look and displaced on the figure in 4 views, 3 principal plus an isometric view (fig.

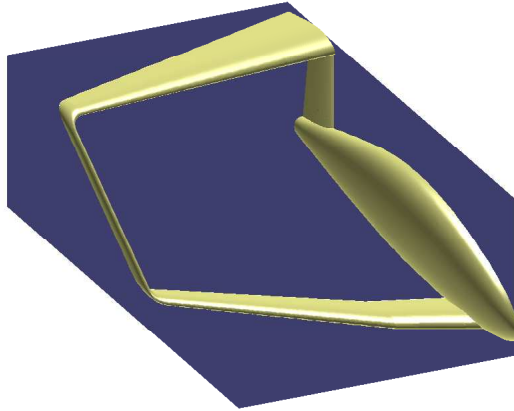


Figure 1.35: Surface Viewing: Render

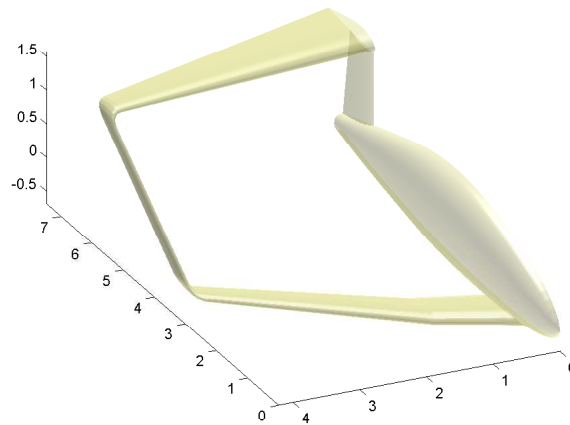


Figure 1.36: Surface Viewing: Render Shaded

1.37).

By default the plots are not symmetrized, and the trim lines calculated during surface generation are not shown. The user can turn on the symmetization or the trim lines plot by clicking on the corresponding checkbox (see fig.1.34). However, only surfaces are symmetrized, while trim lines are plotted only for the $y > 0$ plane.

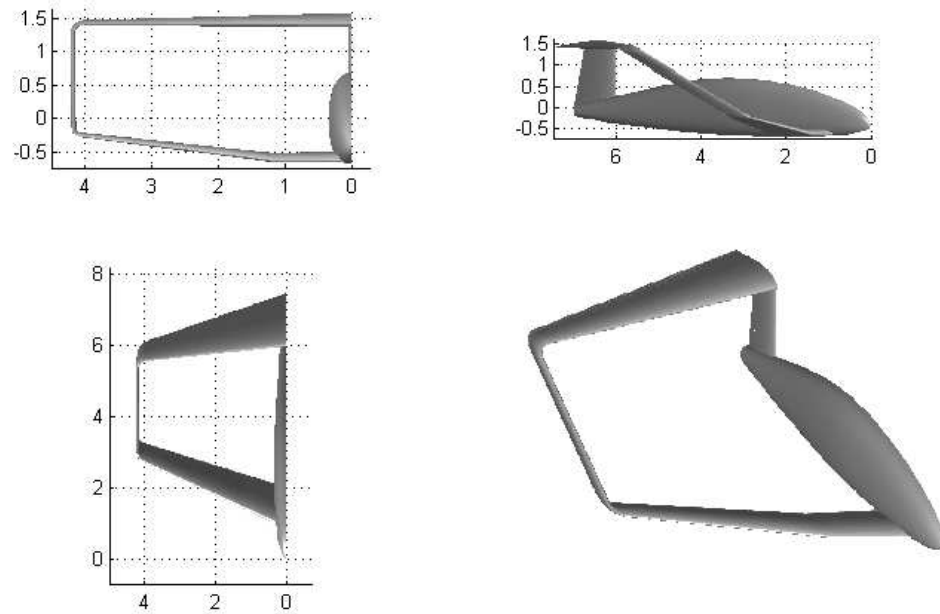


Figure 1.37: Surface Viewing: Three-Plane View

1.3 ASD Surface Mesher

From the ASD main window it is possible to launch the *ASD Surface Mesher* on the selected features. Obviously the features selected must contain surface data. The ASD Surface Mesher is a GUI designed to help the user manage all the parameters and functions underlying the process of mesh generation. The ASD Surface Mesher sub-function are specifically designed to generate triangular meshes with maximum chord distance and maximum element size specified by user. The interface also enables the user to perform different tasks on the mesh, ranging from Delaunay flip to triangular-to-mixed mesh transform, plotting and analyzing meshes. In this section only a brief illustration is given, for more details on the surface mesher refer to [1], for more details on grid topics see also section 5.

The *Mesh Data* and *Mesh stats* panel (active only after a mesh has been analyzed) summarize the mesh properties. The mesh generation controls are grouped in the *Mesh Parameters* panel. From this panel its possible to define and generate a mesh. The *Max Chord Distance* parameter specifies the maximum distance from the

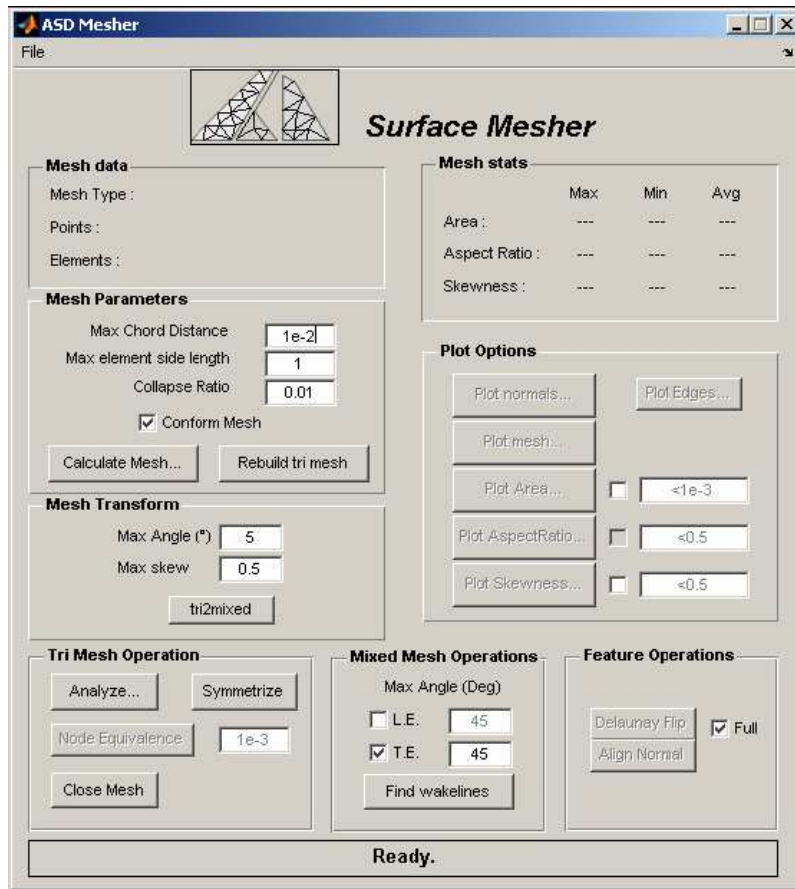


Figure 1.38: ASD Surface MESHER, the interface window

real surface to the mesh (fig.1.39), whereas the *Max Element side length* parameter

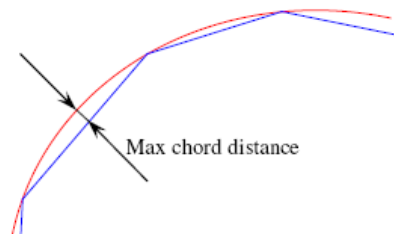


Figure 1.39: Mesh parameters: max chord distance

specifies the maximum element side length. The *Collapse Ratio* can only assume values greater than 0 and lower than 0.5. When the *conform mesh* box is checked

the conform algorithm uses this value to decide whether to move a point on the interface of two surfaces or to add a triangle. The ratio is the maximum value of the ratio `point_to_add_distance/segment_length` that the user allows to move points. Above that value a new triangle is added. This conform procedure is only applied to the inlet/outlet and wingbody mesh creation, and will thus not affect all other surface meshes.

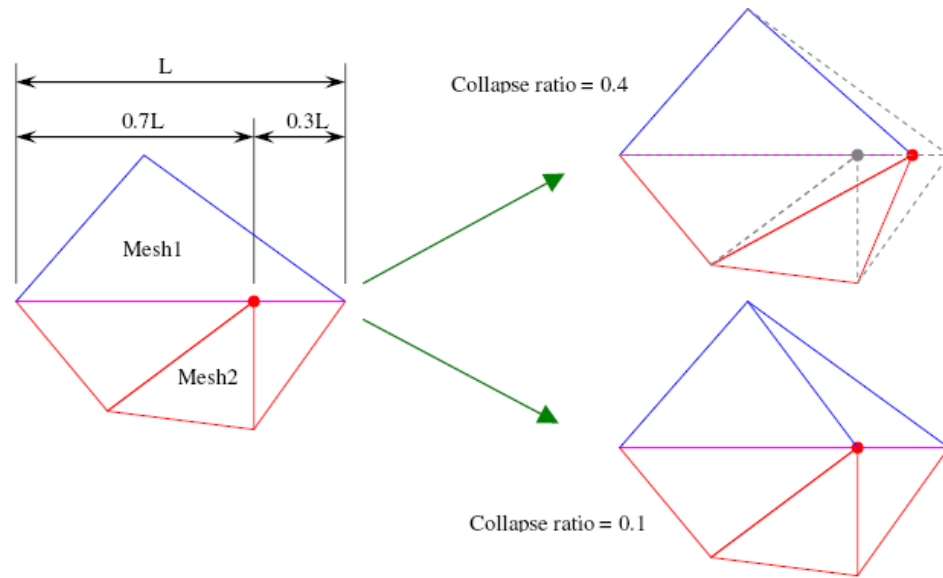


Figure 1.40: Mesh parameters: collapse ratio

The ASD Surface Mesher meshes all the surfaces separately, then trims the meshes created according to the trim lines calculated by ASD Surface Generator. The meshes so created might not be correctly connected at the interface of two surfaces (see fig.1.41). When correct mesh connection is desired the *Conform Mesh* box should be checked.

When a mesh is calculated, all the surfaces are meshed independently. All these partial meshes are stored inside the relative feature structure. At the end of the meshing procedure, whether it has been conformed or not, all the partial meshes are concatenated into one single mesh. Most of the following function work on this mesh, few of them work on the element mesh.

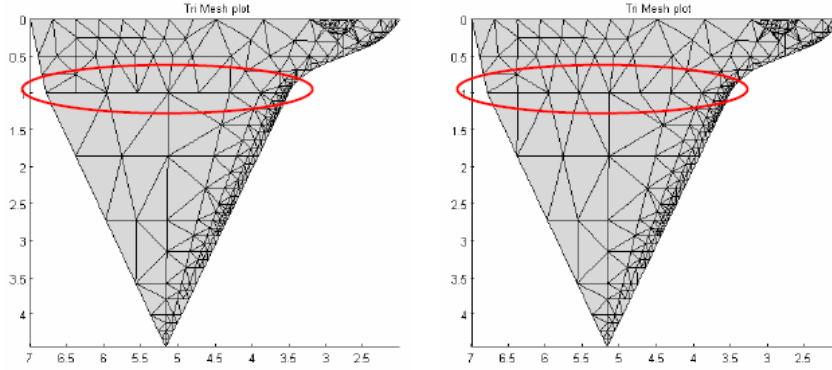


Figure 1.41: Non conform mesh and conform mesh

The *Rebuild tri mesh* function rebuilds the complete mesh starting from the mesh contained into single features. This is useful to recover the original mesh.

The set of tools within *Plot Option Panel* are used to plot the full mesh, calculate and plot the edges and plot the results of the mesh analysis. All these functions work on the full mesh and do not modify it. The interface is similar to the one of the new structured mesher (section 5.5) thus it won't be reported here.

The *Mesh Transform Panel* groups all the mesh transform options. All the functions inside this panel work on the complete mesh and only if the type is “tri”. Triangular meshes are converted to mixed (quad-tri) meshes by recursively bonding together two adjacent triangles which meet the requested characteristics. The control parameters are the maximum angle between the triangles normal to allow union of the two triangles into one quadrilateral element, and the maximum value of skewness allowed for the resulting quadrilateral element. Finally, the *Tri2mixed* button starts the conversion, which modifies only the complete mesh.

The *Tri Mesh Operations* panel groups all the mesh operation options, which are *mesh analysis*, to perform a mesh geometric analysis, *mesh symmetrization*, in order to symmetrize the mesh taking care of normal reversing wherever needed, *node equivalence* to perform a node equivalence based on the specified tolerance with the intent of collapsing all the points that are closer, and *mesh closure*, to close open meshes with a simple triangulation. The last function is useful to prepare meshes for softwares that do not allow open surfaces.

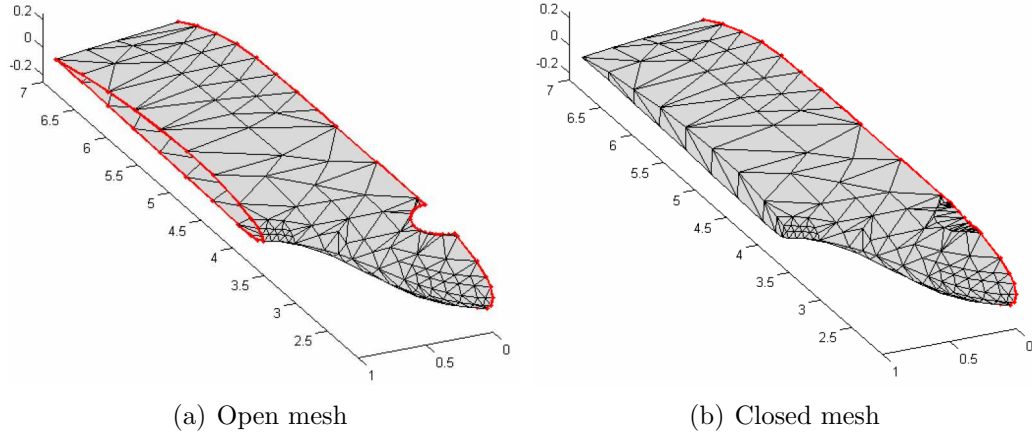


Figure 1.42: Tri-mesh operation: closure of a mesh.

The *Mixed Mesh Operations* panel groups all the mixed mesh operation options, aimed to automatic LE and TE wake lines search. This functions are used when preparing an input file for an aerodynamic panel method, where wake separation lines must be specified into the code. And, finally *Feature Operations panel* groups the *Delaunay Flip*, which perform a cycle of Delaunay flips in 3D space (if the *Full* checkbox is selected the cycles are repeated until no flips are left), and the *Align Normal* functions, which aligns the element normals according to the sign of the global feature normal determined during surface generation.

1.4 The ASD tools

1.4.1 Airfoil Manager

Airfoil Manager is a graphical user interface (GUI) designed to manage airfoil databases stored in *.dat* files. The interface is currently capable of reading a folder containing the *.dat* files, displaying it in a listbox and plotting the selected airfoil along with the information on the file and the airfoil. The interface can operate on plain coordinate file as well as labeled coordinate files, as long as the points are stored starting from the trailing edge and ending on the trailing edge itself.

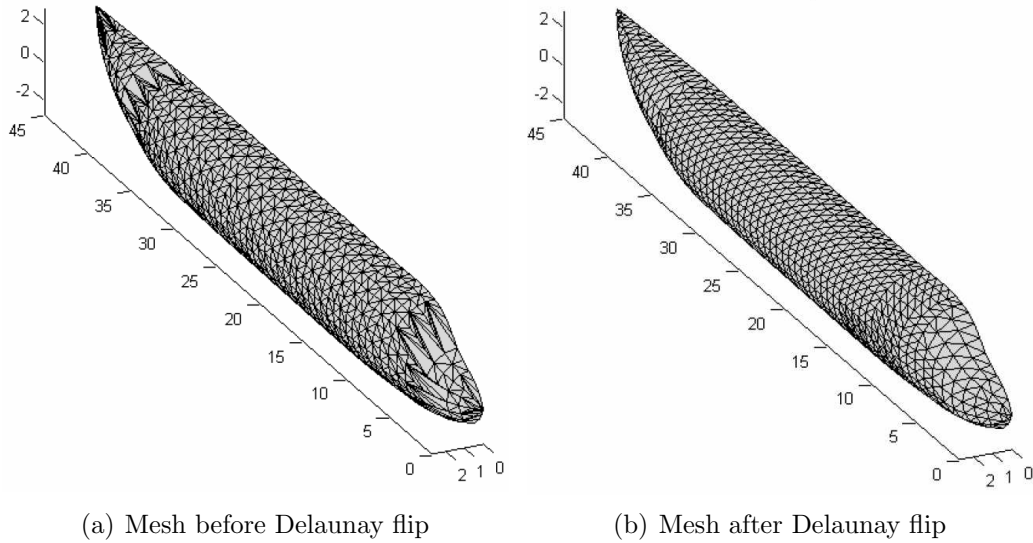


Figure 1.43: Feature operations: full Delaunay flip.

The only section accessible by the user the *Plot Option* panel, where the user can select if and how the airfoil must be interpolated and how many points must be used for interpolation. This number is the same number the airfoil is saved, so it does not only affect the quality of the plot, but also the quality of the exported airfoil .dat file. The options are among the original airfoil coordinates as loaded from the .dat file, the original airfoil coordinates with the chord forced parallel to the x axis, the interpolated airfoil with linear distribution and cosine distribution. The editable text box allows the user to set the number of points for the interpolated airfoil.

The *airfoil attributes* panel is used to estimate camber, thickness information of the currently selected airfoil, based on the interpolated curve calculated on the original points. There may be a little disagreement with the actual thickness and camber values.

In the *airfoil info* panel are collected all the information regarding the selected airfoil: the name and the number of points stored in the .dat file.

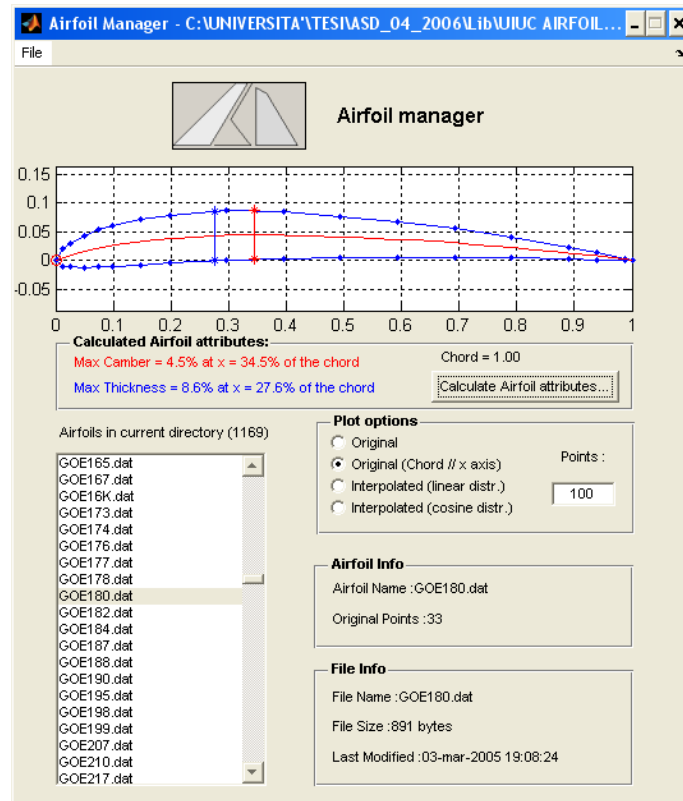


Figure 1.44: Airfoil Manager GUI

1.4.2 NACA Airfoil Generator

NACA Airfoil Generator is a graphical user interface (GUI) designed to create NACA 4-digit and NACA 5-digit airfoil series.

The interface gives the user the capability to explore different airfoils by simultaneously plotting thickness distribution, camber line and the airfoil itself on two graphical windows. The tool is currently capable of dealing with 4-digit and 5-digit standard (non-modified) NACA airfoils such as, for example the NACA4415 or the NACA23012 airfoil.

The user could only introduce the name of the airfoil and the number of points to generate. This number is the same number the airfoil is saved, so it does not only affect the quality of the plot, but also the quality of the exported airfoil *.dat* file. The points are calculated every time the user changes any of the two editable texts,

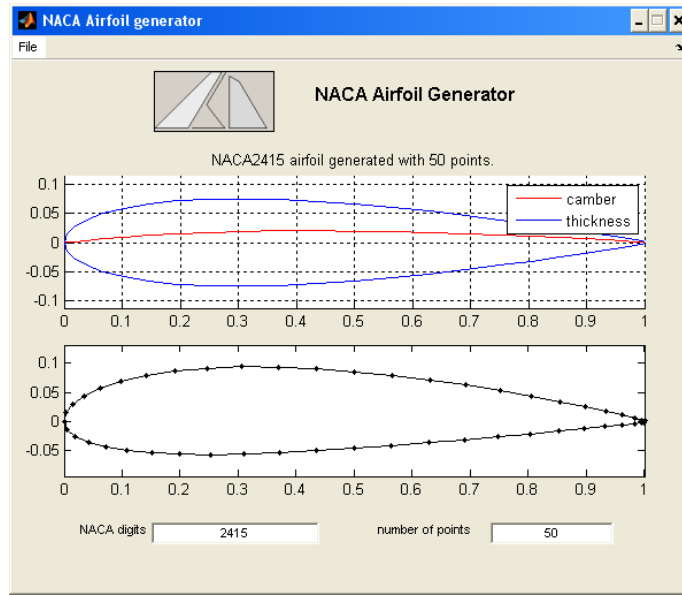


Figure 1.45: NACA Airfoil Generator GUI

and are distributed along the chord using a cosine-law. For equations of the NACA 4 or 5-digit series refer to [5].

1.4.3 Section Sketcher

The interface gives the user the capability to create a section with any shape, or to explore a section in a database by plotting the section coordinates as stored in the *.dat* file, on a graphical window. Finally, the interface includes a Save As command to save the result as a plain coordinate *.dat* ASCII file.

The interface is very simple and consists of only one window. This window is divided in a plot box where the section is shown along with the current parametrization, one modify box where the user can modify the section inserting, adding or moving some control points, one visualization box where the users decide the representation of the section, one zoom box containing the details about the zoom of the file representation, one interpolation box where the user controls interpolation parameter, one export box.

The *simplify curve* pushbutton causes the curve to be simplified. Recursively

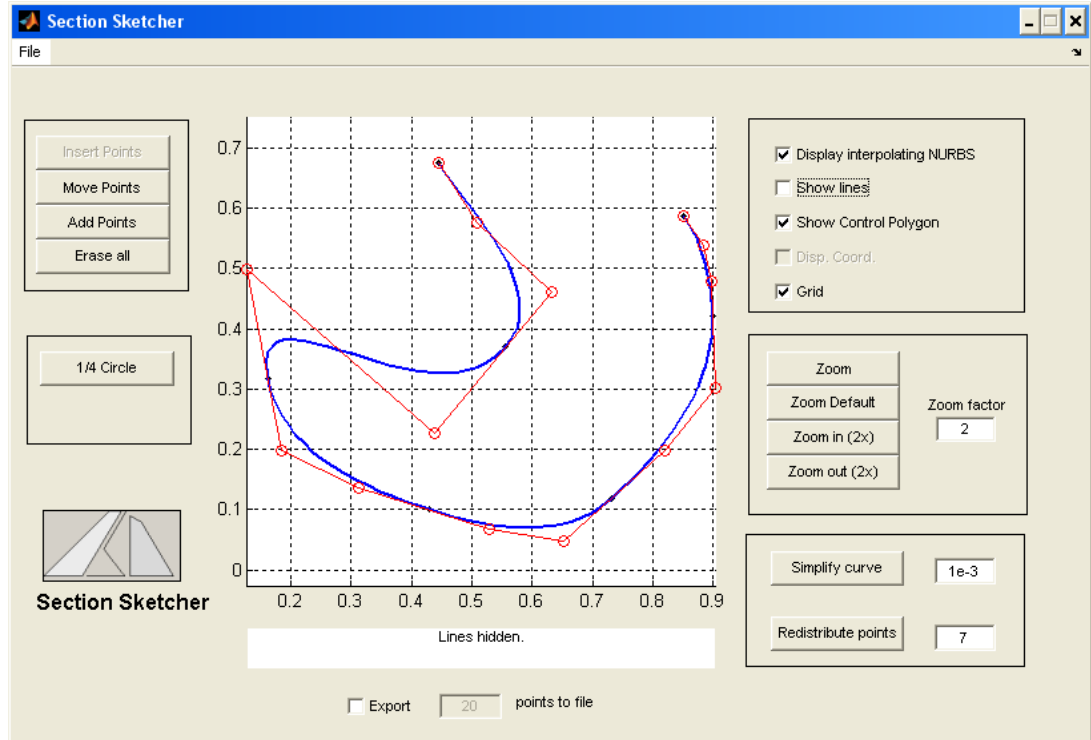


Figure 1.46: Section Sketcher GUI

all the points in the curve are checked if removable maintaining the curve into the prescribed tolerance. The *Redistribute points* redistributes the interpolation points of the NURBS uniformly in the parametric space. The number of points used in redistribution process is set by the user in the dialog box.

1.4.4 Flap Sketcher

Flap Sketcher is a graphical user interface designed to help defining 2-dimensional mobile curves on different airfoils by means of NURBS. The interface gives the user the capability to create any Flap/Slat configuration starting from the base airfoil, or to modify an existing configuration.

The idea is to create a description of the mobile surfaces that is, to some extent, independent from the original airfoil it was created on. To achieve this, the mobile surfaces are saved in a parametric format and automatically adjusted to the loaded

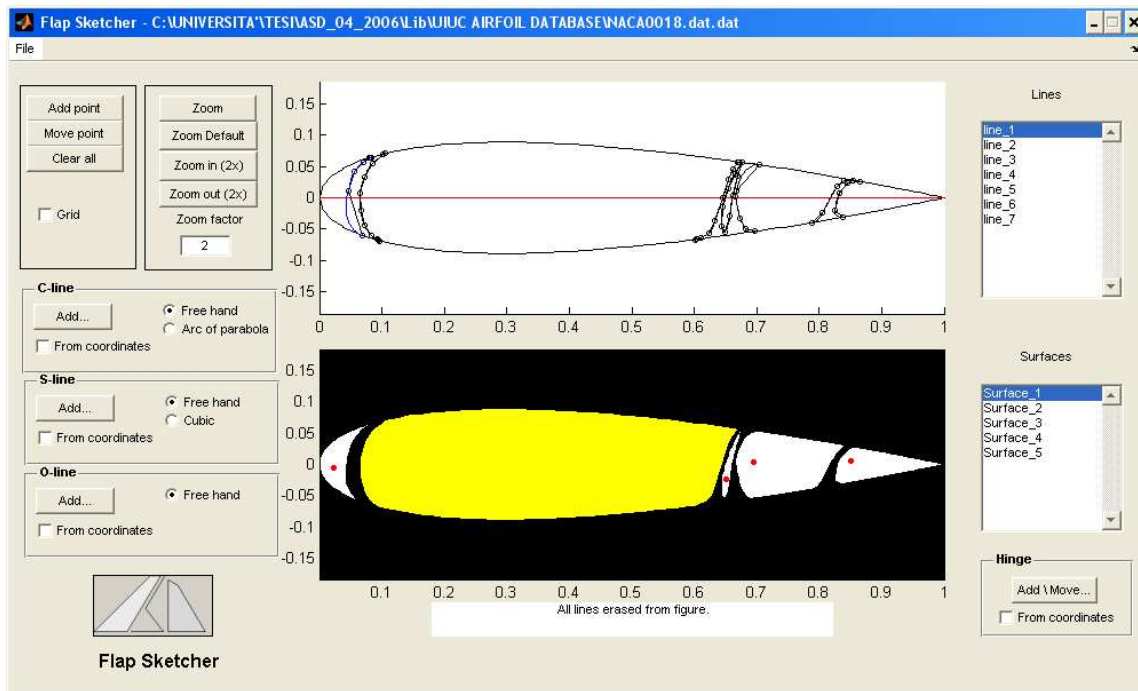


Figure 1.47: Flap Sketcher GUI

airfoil.

The Flap/Slat creation process consists of 3 basic steps:

- select the base airfoil,
- draw the flap/slat lines using the sketch assist tools of the flap sketcher
- position the center of rotation of every surface.

In the first step, an airfoil is easily loaded by means of the menu. Once loaded the airfoil is shown on both windows. The upper window shows the lines and should be used to sketch, while the lower window shows the results of the operations. Besides this, both windows can be used to enter points.

There are several methods to input the lines that define a flap/slat surface. Every line added modifies the flap configuration and the result is displayed on the lower window. There are 3 different types of lines, that differ essentially for the end derivatives.

The C type line is tangent to the airfoil surface and both end derivatives point towards the trailing edge. The first points to input are the ones on the airfoil border, upper and lower, then all the points that define the line between these two. The user can input an unlimited number of points in any order, but usually 3 to 5 are sufficient. In the lower window the user sees the result of the splitting. If the *free-hand* radiobutton is selected the user can freely input the points that define the line, while if the *arc of parabola* radiobutton is selected the user must input only one intermediate point, and the parabola arc will be automatically generated.

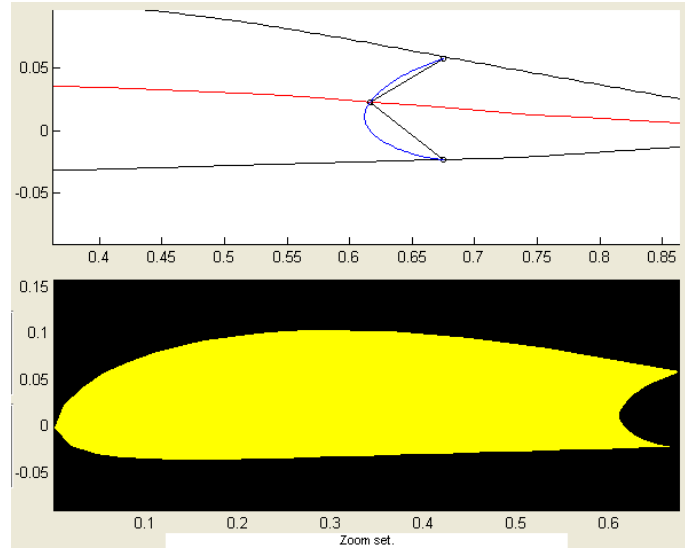


Figure 1.48: Flap Sketcher: C type line

The S type line is tangent to the airfoil surface with the upper end derivative pointing towards the trailing edge and the lower pointing towards the leading edge. The process is identical to the C type line, however a cubic curve instead of an arc of parabola is given.

The O type line is a closed line used to define those parts of the control surface that completely lay inside the airfoil. To define these type of surfaces the user must first provide the point that will be treated as a TE of the surface, then the one that will be treated as the LE, then all the points on the left and finally the points on the right. The sketch of this surface is currently only available in free-hand mode only.

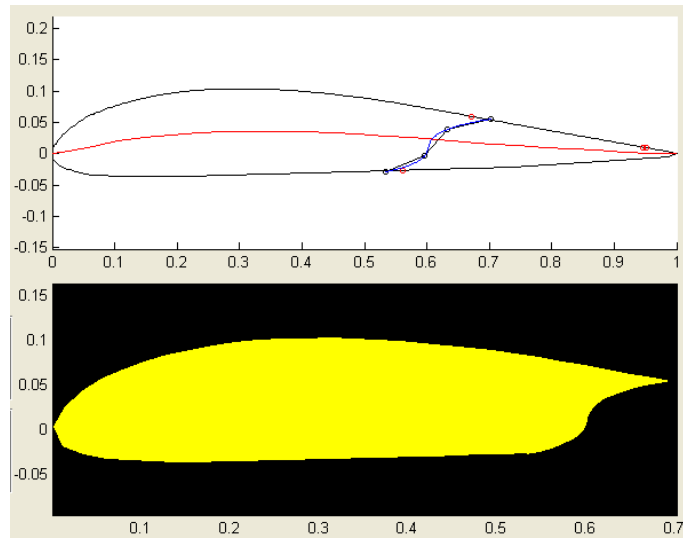


Figure 1.49: Flap Sketcher: *S* type line

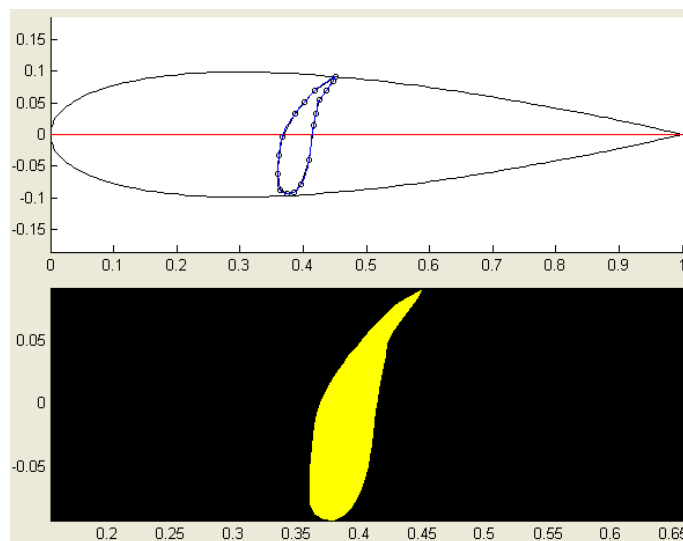


Figure 1.50: Flap Sketcher: *O* type line

To input the points is also available a coordinate mode, that asks the user to manually input the coordinates of the points.

Every line added is given a default name and it is listed on the left of the window. The program automatically creates the surfaces that are defined by the current lines, starting from the LE to the first line, from the second line to the third line and so

on.

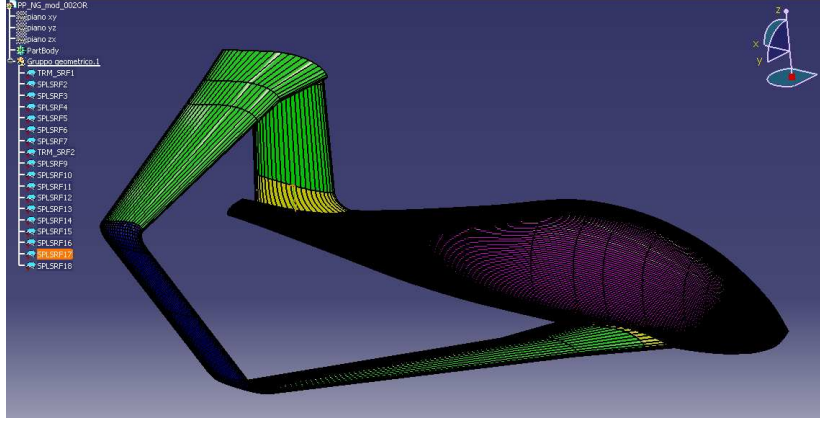
To complete the flap/slat creation, all the movable surfaces must be assigned a hinge point; this is achieved by selecting the point directly on the plotting window, otherwise by entering the coordinates in a popup window.

1.5 ASD Limitations

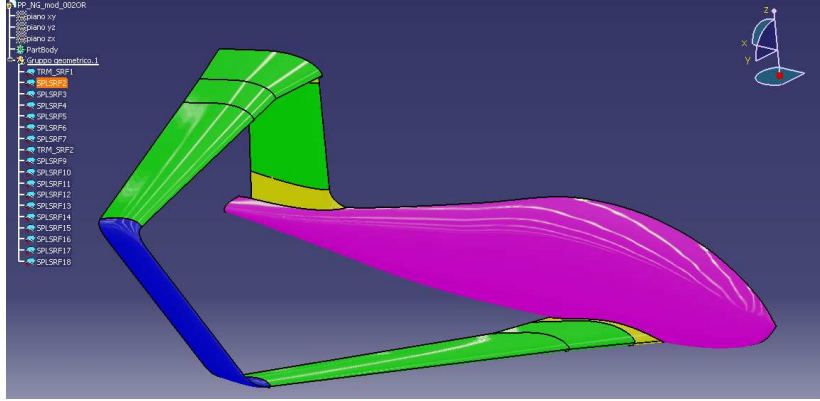
ASD limitations are documented and reported in [1; 2]. However, different problems arises during program utilization in conjunction with other softwares. In fact, when exporting the surfaces generated from ASD to an *IGES* format, and then importing this file with the multi platform CAE/CAD/CAM commercial software CATIA, or other similar softwares, an unexpected difficulty occurs. Advanced softwares like CATIA and Pro/ENGINEER, being oriented also to automotive and industrial design where the freeform surfaces should meet strict requirements to near perfect aesthetical reflection quality, implement a severe surface analysis. As will be shown in section 3.5, high quality surfaces require curvature and tangency alignment. On the other hand, ASD geometrical engine relies on an interpolation surface algorithm capable of generating surfaces with just tangency alignment. Thus, in the process of importing an ASD generated surfaces, the advanced softwares recognizes the curvature discontinuities, and splits the shape in a multitude of patches at the internal whom also the curvature is continuous.

On the visual side, this inconvenient could be overcome, at least on CATIA, by selecting the appropriate visualization filters (see fig.1.51 and 1.52). But, on the practical side, this split process create a multitude of entities which not only slow down the program, but cause the work to be impractical.

The most sever drawback arises when other programs import the CATIA or Pro/ENGINEER configuration in order to do some operations, like grid generation. A typical usage is the CATIA and GAMBIT pairing in order to create an high quality mesh for an accurate CFD analysis with FLUENT software, for example. When GAMBIT recognizes all the patches, the grid generation problem becomes prohibitive. Further, the CFD postprocessing is cumbersome: to know the forces



(a) Visualization option: shading with edges



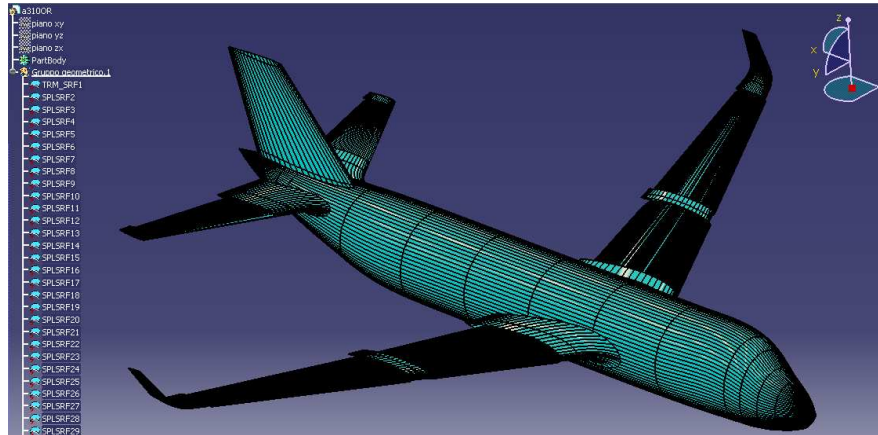
(b) Visualization option: shading with edge without smooth edges

Figure 1.51: A surface generated with ASD and imported with CATIA. Note the patch subdivision.

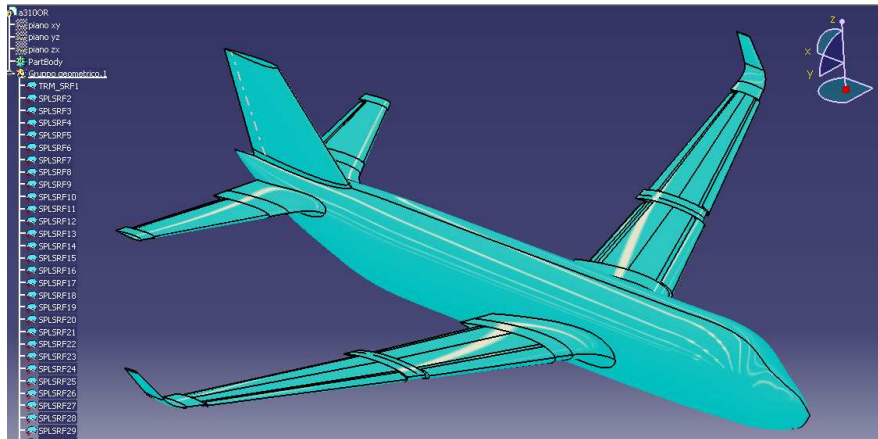
acting on a wing, the patches should be merged. An optimization, which relies on automation, is thus not possible for grid generation and CFD analysis processes.

To overcome this first limitation, modifications on the geometric engine are needed. Referring to figure 1.53 and figure 1.54, the modifications should focus on the NURBS subblock.

Other main ASD limitation is the inability to generate a structured grid. In order to interface the grid generated from the internal mesher to a preliminary aerodynamic analysis program, like a panel method code, some restrictions on the mesh should be met. These programs, and in particular *Pan Air*, require a structured grid as input geometry.



(a) Visualization option: shading with edges



(b) Visualization option: shading with edge without smooth edges

Figure 1.52: A surface generated with ASD and imported with CATIA. Note the patch subdivision.

Thus, close to the actual mesher, fig.1.55, a structured mesher module should join the grid generation process.

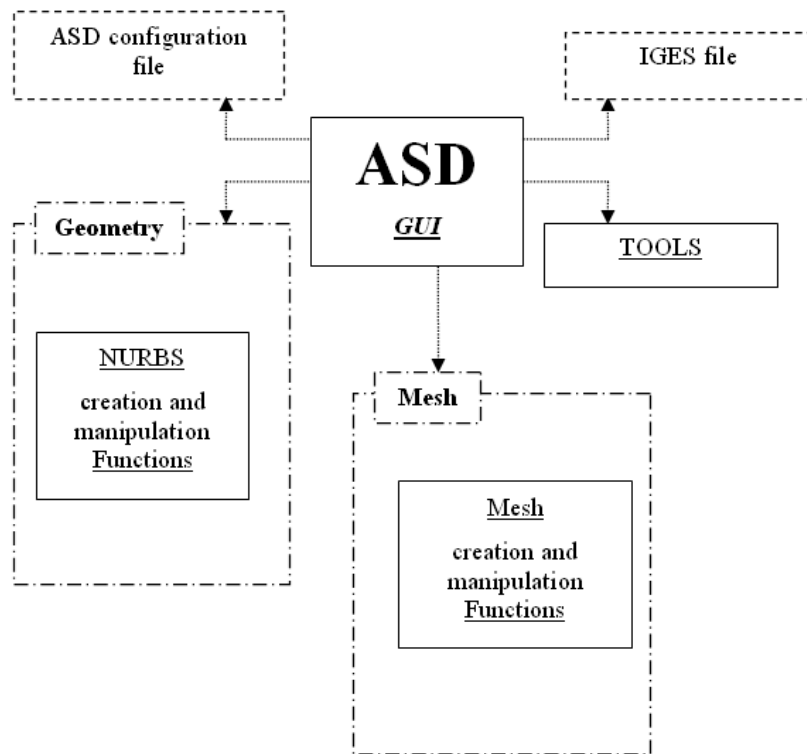


Figure 1.53: Main decomposition of ASD code

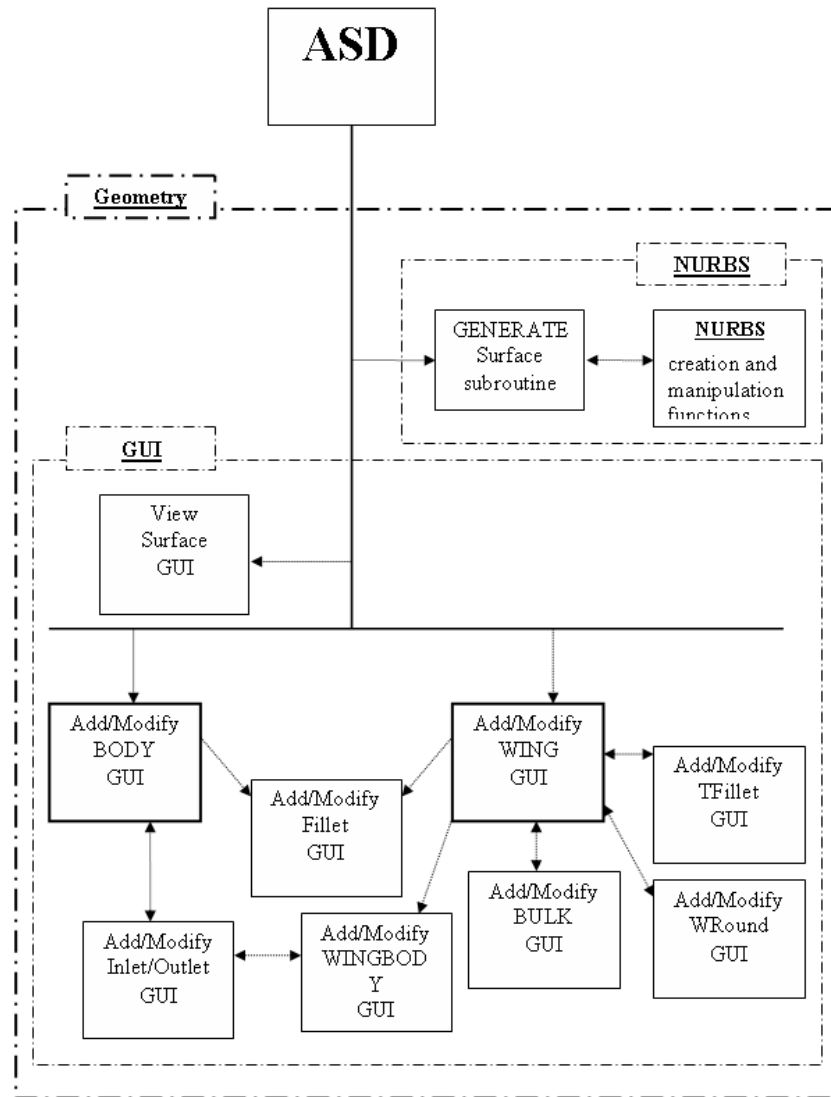


Figure 1.54: Decomposition of the ASD geometrical engine

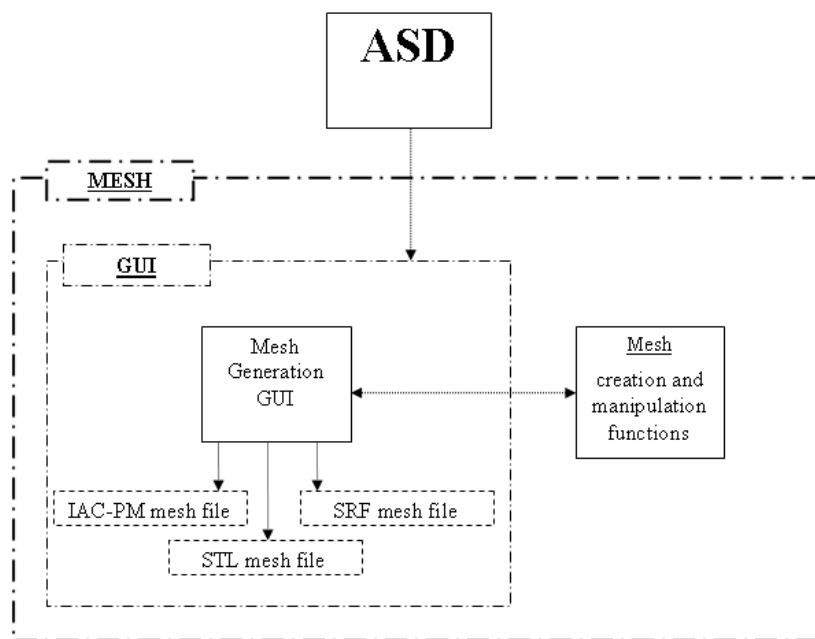


Figure 1.55: Decomposition of ASD mesher

NURBS (Non Uniform Rational B-Spline)

Starting from a brief historical summary on CAGD (computer aided geometric design), this chapter deals with parametric forms. For the most important of them, detailed mathematical model are reported, main characteristics are depicted, benefit and drawbacks are analyzed; especially for NURBS (non uniform rational B-Spline). Most outstanding algorithms aimed to NURBS manipulation are also briefly presented. Lastly, a short treatise on *continuity* is provided.

2.1 A Brief Historical Survey

The earliest recorded use of curves in a manufacturing environment seems to go back to early AD Roman times, for the purpose of shipbuilding. A ship's ribs were produced based on templates which could be reused. Later Venetians perfectionate these technique. Drawing became popular only in the 1600s in England, the classic *spline* was probably invented then. This connection between drawing and manufacturing in shipbuilding was the earliest use of constructive geometry to define free-form shapes.

Another keypoint originated in aeronautics, with Liming (NAA, North American Aviation): he realized that to store a design in terms of numbers was more efficient instead of manually traced curves. Thus he translated the classical drafting constructions into numerical algorithms. Liming's work became very influential in the 1950s when it was adopted by U.S. aircraft companies.

The turning point was the advent of numerical control in the 1950s: machining of 3d shapes out of blocks of wood or steel became reality. Soon was found the need of adequate software, in terms of producing a computer compatible description of shapes. The most promising description was found to be in terms of parametric surfaces. At that time the theory of parametric surfaces was well understood in differential geometry, but their potential for the representation of surfaces in a Computer Aided Design (CAD) environment was not known at all. This exploration can be viewed as the origin of Computer Aided Geometric Design (CAGD) ¹. Only large corporations could afford the computers capable of performing the calculations, so the developments were internal at each company and kept secrets.

In the 1960s-1980s one of the major contribution to CAGD development was the work of of Bézier (at Renault), who defined a particular polynomial parametric form, milestone for the subsequent development; this leads in 1971 to *UNISURF*, car body design and tooling.

Stand-alone from Bézier's work was mathematician de Casteljau activity at Citroën. He adopted the use of Bernstein polynomials for his curve and surface definitions from the very beginning, together with what is now known as the de Casteljau algorithm; another of the breakthrough insight of his work was to use control polygons, a technique that was never used before. After it was found that his parametric curve is mathematically identical to Bézier's one. His work was publicized only in 1967.

In the US, the development was linked mainly to the aerospace industry. Ferguson (at Boeing) used piece cubic curves together so that they formed composite curves which were overall twice differentiable; these curves, referred as *spline curves* could easily interpolate to a set of points. The meaning of the term *spline curve* has since undergone a subtle change. Instead of referring to curves that minimize certain functionals, spline curves are now mostly thought of as piecewise polynomial (or rational polynomial) curves with certain smoothness properties. Ferguson's first works was printed in 1964.

¹This term was introduced by Barnhill and R. Riesenfeld in 1974, during the first famous conference on that topic, at the University of Utah.

Coons (MIT researcher and FORD consultant) developed the patch theory using cubic piecewise polynomials in the Hermite form. Coons devised a simple formula to fit a patch between any four arbitrary boundary curves. His famous report were publicized in the 1967. Improvements about this topic (called *transfinite interpolation*) were done by Gordon (at General Motors) who developed a generalization, capable of interpolating a rectangular network of curves, and Gregory.

De Boor (at General Motors) was the first to introduce the tensor product surface. He also was the first to use B-Splines (short for Basis Splines) as a tool for geometric representation. The recursive evaluation of B-spline curves is due to him and is now known as the de Boor algorithm: it is based on a recursion for B-splines. It was this recursion that made B-splines a truly viable tool in CAGD. Before its discovery, B-splines were defined using a tedious divided difference approach which was numerically very unstable.

Spline functions are important in approximation theory, but in CAGD, parametric spline curves are much more important. These were introduced in 1974 by Riesenfeld and Gordon who realized that de Boor's recursive B-spline evaluation was the natural generalization of the de Casteljau algorithm. B-spline curves include Bézier curves as a proper subset and soon became a core technique of almost all CAD systems. A first B-spline-to-Bézier conversion was found by W.Boehm. Several algorithms were soon developed that simplified the mathematical treatment of B-spline curves.

The generalization of B-spline curves to NURBS has become the standard curve and surface form in the CAD/CAM industry. They offer a unified representation of spline and conic geometries: every conic as well as every spline allows a piecewise rational polynomial representation. The development at Boeing is exemplary for the emergence of NURBS. The company realized that different departments employed different kinds of geometry software; worse, those geometries were incompatible. Thus NURBS were adopted as a standard since they would allow a unified geometry representation. Companies such as Boeing, SDRC, or Unigraphics soon initiated

making NURBS an IGES standard ².

In the last twenty-thirty years the technological development brings CAD tools area of application outside industry and brought them till inside the medium user computer. Such a change hauled with him CAGD research and development.

Today in totally different from industrial areas arises typical geometric modeling problems. For example, subjects like geology, weather forecast, medicine are concerned with *surface fitting* problems.

For a more detailed discussion about the history and the development of CAGD refer to [6; 7].

2.2 Curve and Surface Basics

The two most common methods of representing curves and surfaces in geometric modeling are implicit equations and parametric functions. Sometimes other representations can be used (for example *explicit* representation), but the previously mentioned are predominant in the academic, industrial and commercial world. For more details refer to [7; 8; 9].

2.2.1 Implicit and Parametric Forms

Implicit Forms

In the implicit forms, one or two equation describes a relation between the spatial coordinates of the points of the form. For an implicit surface: $f(x,y,z) = 0$. Every form has his unique representation, a part for a multiplier constant. The description of a curve in the three dimensional space is obtained as intersection of two implicit surfaces: $f_1(x,y,z) = f_2(x,y,z) = 0$.

²Initial Graphics Exchange Standard, developed to facilitate geometry data exchange between different companies.

Parametric Forms

In the parametric forms, each of the coordinates of a point is represented separately as an explicit function of one or more independent parameters. For a surface: $C(u,v) = (x(u,v), y(u,v), z(u,v))$, with u and v being the two independent parameters: hence, it is essentially a mapping of a domain $\mathbb{D} \subset \mathbb{R}^2$ in \mathbb{R}^3 (usually the domain \mathbb{D} is normalized to $[0, 1] \times [0, 1]$). The description is not unique for a form. In the curve case, the independent parameter is only one.

2.2.2 Advantages and Disadvantages

There is no answer to the question of which form is better. Every form can be more appropriate, depending from the application. Anyway, a comparison follows about capabilities and limits of the previously mentioned forms:

- The parametric form is more suited for simultaneous two and three dimensional representation: adding or removing third coordinate switches between the cases. On the other hand implicit aren't so flexible: a curve in two-dimensional spaces is represented with an implicit equation, but the generalization to the three dimensional space need adding, apart from the third coordinate, also another implicit equation.
- It is cumbersome to represent bounded curve segments or surface patches with the implicit form. However, boundness is built into the parametric form (through the bound of the parameter value). On the other implicit forms are well suited for unbounded geometry; the opposite states for parametric forms
- Parametric form introduce a direction; for the curves the natural direction should be defined concordant to the parameter. Hence, it is easy to generate ordered sequences of point along a parametric curve, and meshes of points on surfaces.
- The parametric form is more natural for designing and representing shape in a computer. The coefficients of the most used parametric forms possess

considerable geometric significance. This leads to an intuitive design method and numerically stable algorithms with a distinctly geometric flavor.

- The complexity of many geometric operations depends greatly on the method of representation. For example, to compute a point on a curve or surface is difficult in the implicit form, but determine if a given point lies on the curve or surface is difficult in the parametric form.
- Sometimes in the parametric form one must deal with anomalies, unrelated to true geometry. A classic example is the unit sphere: here the parametric calculations of the pole is critical, even if geometrically this points aren't different from the other points on the surface.

Since we are concerned almost exclusively with bounded surfaces, computer use and the geometric insight of the coefficient is important, the parametric form will be the preferred one.

2.2.3 Requirements for the parametric forms

In order to fulfill the CAD/CAE demands, a parametric representation should be efficiently implemented on the computer, should allow the description of the geometries of interest and should allow the local post-editing of part of that shape (that is modify, after created, the shape in a specific point should not modify the rest of the shape). In particular the choice would be restricted to the representation capable of satisfying the following points:

1. ability to describe geometries like straight lines, continuous curves and piecewise curves, aerodynamic surfaces with mathematical accuracy
2. capability of easy processing in a computer contest, in particular:
 - (a) easy and efficient points and derivatives evaluation,
 - (b) calculation insensitivity to trunk and round-off errors,
 - (c) small memory allocation request for storage;
3. simplicity and mathematically well understood.

2.2.4 Power Basis Form of a curve

A widely used class of functions is the polynomials. Although they satisfy the second and third point of the previous list, they fail to represent precisely a number of important curves. There are two common methods of expressing polynomial functions, the first is power basis (the other one is Bézier). A n th-degree power basis curve is given by:

$$\mathbf{C}(u) = [x(u), y(u), z(u)]^T = \sum_{i=0}^n \mathbf{a}_i u^i \quad \text{with } 0 \leq u \leq 1 \quad (2.1)$$

where \mathbf{a}_i are vectors, u the parameter. In matrix form it holds:

$$\mathbf{C}(u) = [A] \cdot [u] = [\mathbf{a}_i]^T [u^i] \quad (2.2)$$

where $A = [\mathbf{a}_0 \ \mathbf{a}_1 \ \dots \ \mathbf{a}_n]$ and $u^i = [1, u, u^2, u^3 \dots]^T$ are the basis functions. The power basis form has the following disadvantages:

- is not well suited with interactive shape design; the coefficients \mathbf{a}_i carry very little geometric insight about the shape of the curve;
- the algorithms for processing power basis (e.g., Horner's method [8]) are algebraic rather than geometric oriented;
- the algorithms are prone to round-off error if the coefficients vary greatly in magnitude.

2.2.5 Bézier Curves

The Bézier curves were developed independently by Bézier (at Renault) and de Casteljau (at Citroën). From a mathematical point of view they are exactly the same as power basis, but they remedy the latter's shortcomings. A n th-degree Bézier curve is defined by

$$\mathbf{C}(u) = \sum_{i=0}^n B_{i,n}(u) \mathbf{P}_i \quad \text{with } 0 \leq u \leq 1 \quad (2.3)$$

where \mathbf{P}_i , the geometric coefficients, are the *control points*, and $B_{i,n}(u)$, the basis (or blending) functions, are the classical n th-degree Bernstein polynomials, given by:

$$B_{i,n}(u) = \frac{n!}{i!(n-i)!} u^i (1-u)^{n-i} \quad (2.4)$$

The union of the control points is called *control polygon*. In addition Bézier curves

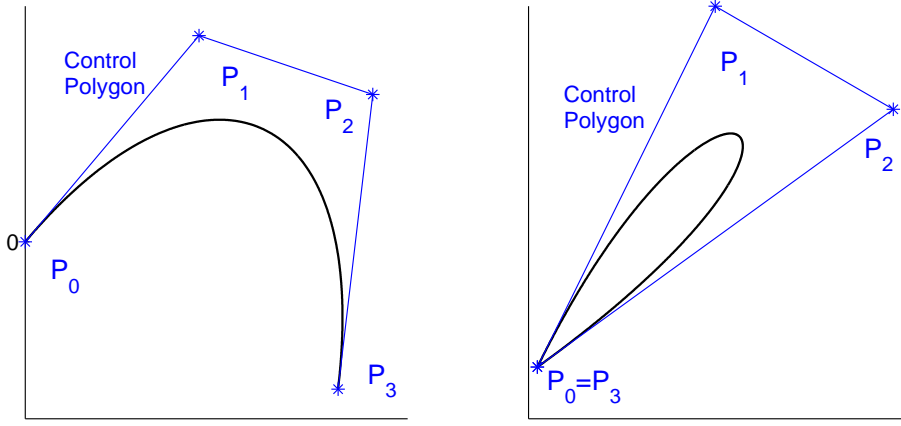


Figure 2.1: Two examples of cubic Bézier curves.

are invariant under the usual transformations such as rotations, translations and scalings; that is, one applies the transformation to the curve by applying it to the control polygon.

In any representation scheme, the choice of the basis functions determines the geometric characteristics. These functions have these properties:

P.1.1 nonnegativity: $B_{i,n}(u) \geq 0$ for all i,n and $0 \leq u \leq 1$;

P.1.2 partition of unity: $\sum_{i=0}^n B_{i,n}(u) = 1$ for all $0 \leq u \leq 1$;

P.1.3 $B_{0,n}(0) = B_{n,n}(1) = 1$;

P.1.4 $B_{i,n}(u)$ attains exactly one maximum on the interval $[0,1]$, exactly at $u = \frac{i}{n}$;

P.1.5 symmetry: the set of polynomials $B_{i,n}(u)$ is symmetric with respect to $u = \frac{1}{2}$

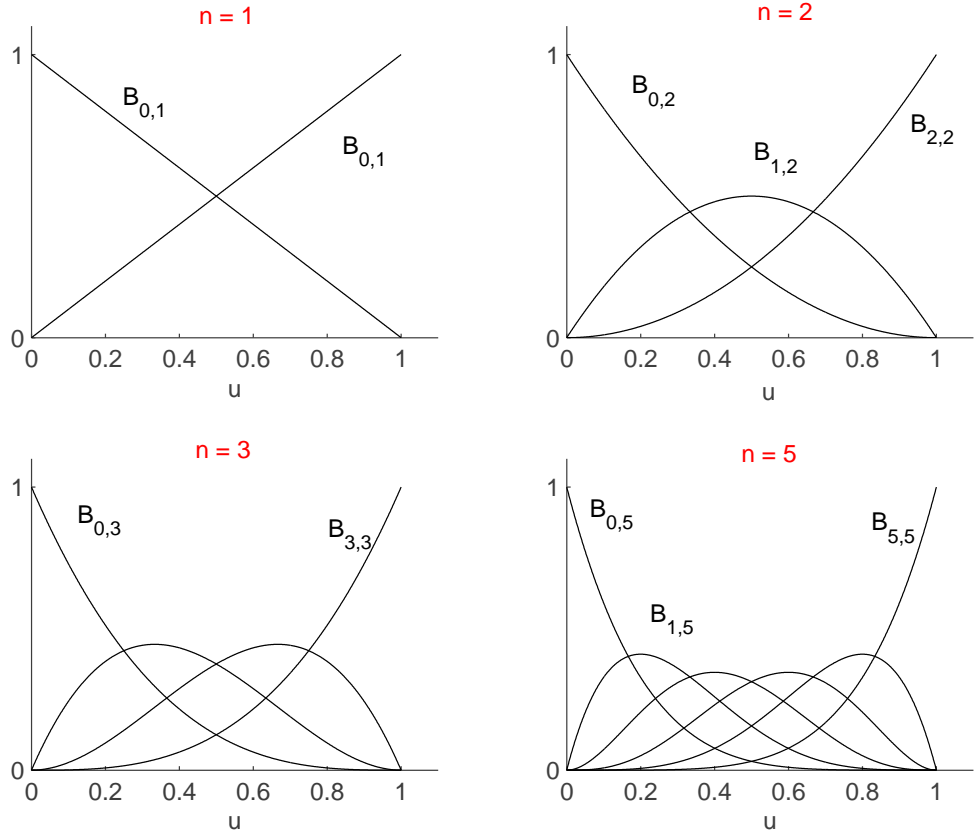


Figure 2.2: The Bernstein polynomials

P.1.6 recursive definition: $B_{i,n}(u) = (1 - u)B_{i,n-1}(u) + u B_{i-1,n-1}(u)$; if $i < 0$ or $i > n$ then it is set $B_{i,n}(u) \equiv 0$;

P.1.7 derivatives:

$$B'_{i,n}(u) = \frac{dB_{i,n}}{du} = n (B_{i-1,n-1}(u) - B_{i,n-1}(u)) \quad (2.5)$$

with

$$B_{-1,n-1}(u) \equiv B_{n,n-1}(u) \equiv 0$$

The sixth properties yields simple and efficient algorithm to compute values of the Bernstein polynomials at fixed values of u . Combining the above mentioned linear interpolation and the Bézier mathematical definition (eq.(2.3)), fixing $u = u_0$ and

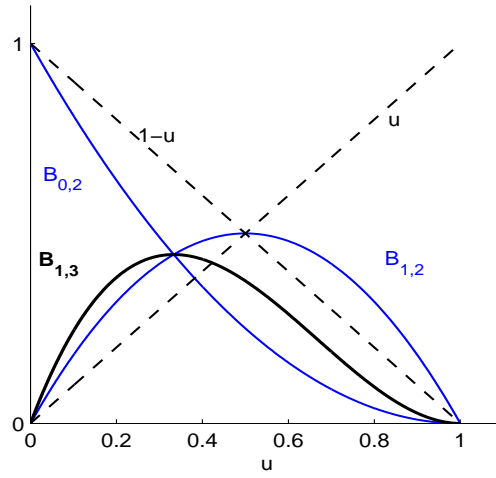


Figure 2.3: The recursive definition of the Bernstein polynomial, $B_{1,3}$

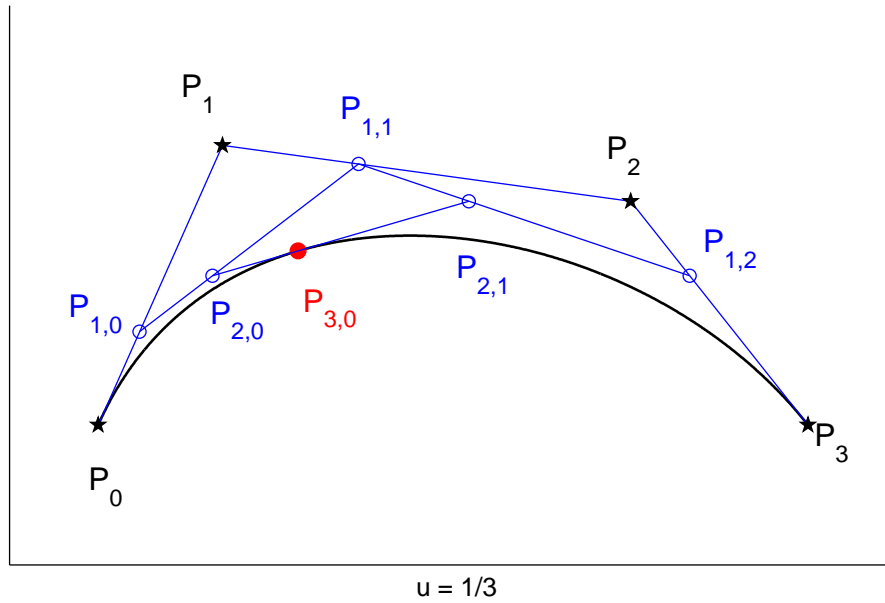


Figure 2.4: Evaluation of a point at $u = \frac{1}{3}$ with repeated linear interpolation, i.e. deCasteljau algorithm

denoting \mathbf{P}_i by $\mathbf{P}_{0,i}$ yields the *deCasteljau Algorithm* :

$$\mathbf{P}_{k,i}(u_0) = (1 - u_0) \mathbf{P}_{k-1,i}(u_0) + u_0 \mathbf{P}_{k-1,i+1}(u_0) \quad \text{for} \quad \begin{cases} k = 1, \dots, n \\ i = 0, \dots, n \end{cases} \quad (2.6)$$

This is a corner cutting process, as shown in fig.2.4.

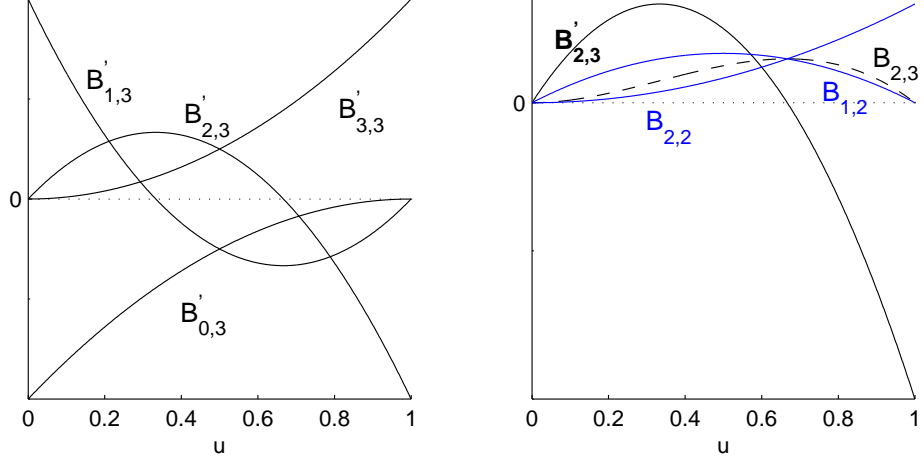


Figure 2.5: Derivatives: (a) the derivatives of the cubic Bernstein polynomials; (b) the derivative $B'_{2,3}$ in terms of $B_{1,2}$ and $B_{2,2}$

Using property seven:

$$\begin{aligned} \mathbf{C}'(u) &= \frac{d(\sum_{i=0}^n B_{i,n}(u)\mathbf{P}_i)}{du} = \sum_{i=0}^n B'_{i,n}(u)\mathbf{P}_i \\ &= n \sum_{i=0}^{n-1} B_{i,n-1}(u)(\mathbf{P}_{i+1} - \mathbf{P}_i) \end{aligned} \quad (2.7)$$

and

$$\begin{aligned} \mathbf{C}'(0) &= n(\mathbf{P}_1 - \mathbf{P}_0) & \mathbf{C}''(0) &= n(n-1)(\mathbf{P}_0 - 2\mathbf{P}_1 + \mathbf{P}_2) \\ \mathbf{C}'(1) &= n(\mathbf{P}_n - \mathbf{P}_{n-1}) & \mathbf{C}''(1) &= n(n-1)(\mathbf{P}_n - 2\mathbf{P}_{n-1} + \mathbf{P}_{n-2}) \end{aligned} \quad (2.8)$$

That is, from eq.(2.7) and eq.(2.8)

- the derivative of an n th-degree Bézier curve is an $(n-1)$ th-degree Bézier curve;
- the expressions for the end derivatives are symmetric (this is a consequence of symmetry of the blending functions);
- the k th derivative at an endpoint depends only on the $k+1$ control points at that end.

As a polynomial, even the Bézier form can't represent precisely some geometric shapes, e.g., circles, hyperbolas, ellipses, cylinders, cones, spheres. This limitation is overcome using rational basis functions, which yields to the definition of *Rational Bézier* forms. For more detail refer to [7; 8]. Anyway, conceptually is the same as for *B-Splines* and *Rational B-Splines*, section(2.4).

2.2.6 Tensor Product Surfaces

The curve $\mathbf{C}(u)$ is a vector valued function of one parameter. It is a mapping of straight line into Euclidean three dimensional space. A surface is a vector valued function of two parameters and represent a mapping of a region, \mathcal{R} , of the uv plane (with u and v being the two parameters) into Euclidean three dimensional space. There are many schemes for representing surfaces. The most simple, and the one most widely used in geometric modeling applications, is the *tensor product* scheme. A drawback is the inability to model complex topologies, problem overcome with other techniques (triangular patches, n-sided patches, hierarchical approaches ...). This scheme is the unique used in the present work. The tensor product surfaces were first investigated from de Casteljau, even if the popularity of this type of surfaces is due to work of Bézier. Initially Bézier patches were only used to approximate a given surface. Later it was found that any B-Spline surface can be written in piecewise Bézier form (patches). This method is basically a bidirectional curve scheme. It uses basis functions and geometric coefficients. The basis functions are bivariate functions of u and v , which are constructed as products of univariate basis functions. The geometric coefficients are arranged (topologically) in a bidirectional, $n \times m$ net. Thus, a tensor product surface has the following form:

$$\mathbf{S}(u,v) = (x(u,v), y(u,v), z(u,v)) = \sum_{i=0}^n \sum_{j=0}^m f_i(u) g_j(v) \mathbf{b}_{i,j} \quad (2.9)$$

where $\begin{cases} \mathbf{b}_{i,j} = (x_{i,j}, y_{i,j}, z_{i,j}) \\ 0 \leq u, v \leq 1 \end{cases}$

The domain (u, v) of the mapping is a square (a rectangle, in general), that's why the tensor product surfaces can be also called rectangular patches. The matrix form:

$$\mathbf{S}(u, v) = [f_i(u)]^T [\mathbf{b}_{i,j}] [g_j(v)] \quad (2.10)$$

with $[f_i(u)]^T$ being a $(1) \times (n+1)$ row vector, $[g_j(v)]$ a $(m+1) \times (1)$ column vector, $[\mathbf{b}_{i,j}]$ a $(n+1) \times (m+1)$ matrix of three dimensional points. Fixing one parameter, for example $u = u_0$:

$$\begin{aligned} \mathbf{C}_{u_0}(v) = \mathbf{S}(u_0, v) &= \sum_{j=0}^m \left(\sum_{i=0}^n \mathbf{b}_{i,j} f_i(u_0) \right) g_j(v) = \sum_{j=0}^m \mathbf{c}_j(u_0) g_j(v) \\ \text{where} \quad \mathbf{c}_j(u_0) &= \sum_{i=0}^n \mathbf{b}_{i,j} f_i(u_0) \end{aligned} \quad (2.11)$$

\mathbf{C}_{u_0} is a curve, lying on the surface \mathbf{S} , and is called *isoparametric curve*.

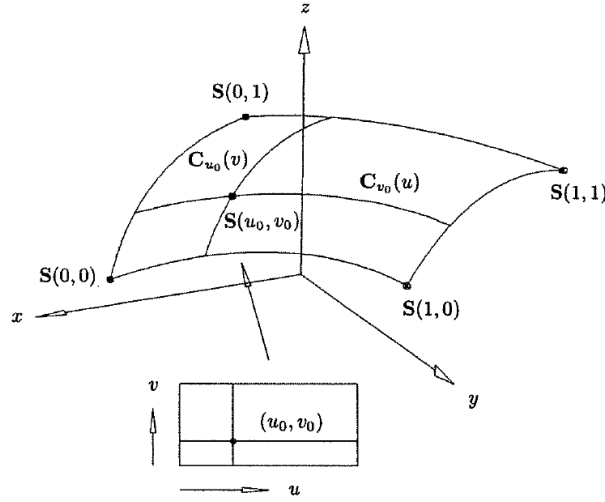


Figure 2.6: A tensor product surface showing isoparametric curves (from [8])

The Bézier surfaces are obtained by taking a bidirectional net of control points and products of the univariate Bernstein polynomials:

$$\mathbf{S}(u, v) = \sum_{i=0}^n \sum_{j=0}^m B_{i,n}(u) B_{j,m}(v) \mathbf{P}_{i,j} \quad 0 \leq u, v \leq 1 \quad (2.12)$$

Fixing $u = u_0$ leads to a Bézier curve lying on the surface.

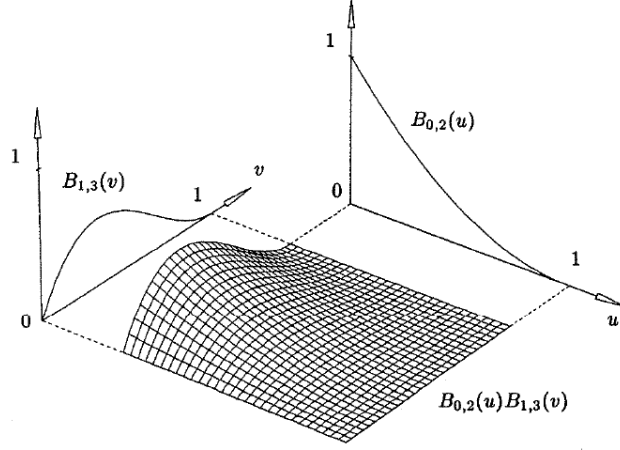


Figure 2.7: The Bézier tensor product basis function $B_{0,2}(u) B_{1,3}(v)$ (from [8])

$$\begin{aligned}
 \mathbf{C}_{u_0}(v) &= \mathbf{S}(u_0, v) = \sum_{i=0}^n \sum_{j=0}^m B_{i,n}(u_0) B_{j,m}(v) \mathbf{P}_{i,j} \\
 &= \sum_{j=0}^m B_{j,m}(v) \left(\sum_{i=0}^n B_{i,n}(u_0) \mathbf{P}_{i,j} \right) \\
 &= \sum_{j=0}^m B_{j,m}(v) \mathbf{Q}_j(u_0)
 \end{aligned} \tag{2.13}$$

$$\text{where } \mathbf{Q}_j(u_0) = \sum_{i=0}^n B_{i,n}(u_0) \mathbf{P}_{i,j} \quad j = 0, \dots, m$$

Since the basis functions remain the same, Bézier curve properties transfer to surface. Also the *deCasteljau* algorithm can be easily extended to compute points on a Bézier surface. Fixing u_0, v_0 , and applying deCasteljau algorithm to the j_0 row of control points, i.e. to \mathbf{P}_{i,j_0} with $i = 0, \dots, n$, creates the points $\mathbf{Q}_{j_0}(u_0)$. Therefore, applying deCasteljau Algorithm $(m+1)$ times yields $\mathbf{C}_{u_0}(v)$. Then, applying it again to $\mathbf{C}_{u_0}(v)$ with $v = v_0$ yields $\mathbf{C}_{u_0}(v_0) = \mathbf{S}(u_0, v_0)$. This process requires

$$\frac{n(n+1)(m+1)}{2} + \frac{m(m+1)}{2} \tag{2.14}$$

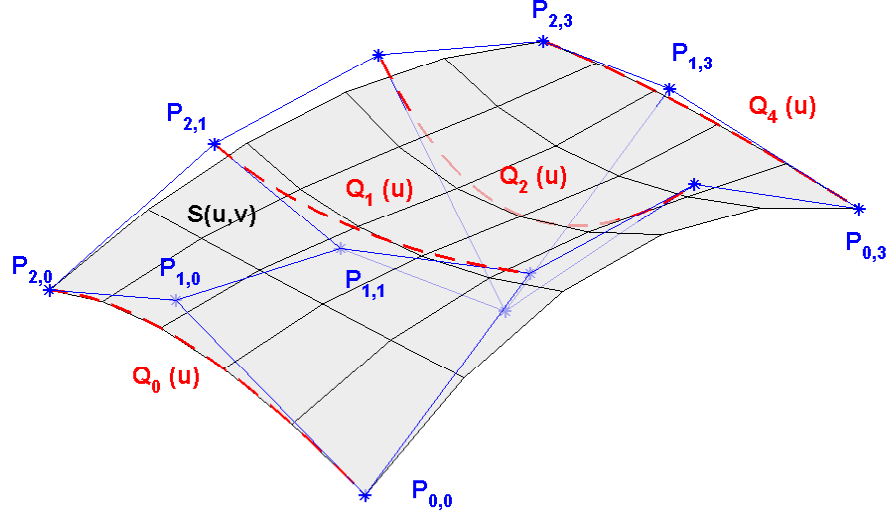


Figure 2.8: Bézier surface: note that the Q_j (eq.(2.13)) don't lie on the surface.

linear interpolations. By symmetry, computing $C_{v_0}(u)$ first and then $C_{v_0}(u_0) = S(u_0, v_0)$ requires

$$\frac{m(m+1)(n+1)}{2} + \frac{n(n+1)}{2} \quad (2.15)$$

linear interpolations. As a consequence is more economical to compute first $C_{v_0}(u)$ or $C_{u_0}(v)$ depending which one between n and m is larger. More about surfaces will be analyzed in sections 2.3.4 and 2.4.2. However for a detailed treatise on surfaces, even with generic topology, refer to [7].

2.3 B-Splines

There are different way to treat B-splines, due their historical different development. In fact B-splines were first investigated in the statistical an probability field (since 1940s). After de Boor, Cox and Mansfield independently discover the recurrence relation. It was this recursion that made B-splines a truly viable tool in CAGD.

Before its discovery, B-splines were defined using a tedious divided difference approach which was numerically very unstable. An important step was the parametric use of B-splines, 1974 by Riesenfeld and Gordon who realized that de Boor's recursive B-spline evaluation was the natural generalization of the deCasteljau algorithm. B-spline curves include Bézier curves as a proper subset and soon became a core technique of almost all CAD systems. A first B-spline-to-Bézier conversion was found by W.Boehm. Several algorithms were soon developed that simplified the mathematical treatment of B-spline curves; these include Boehm's knot insertion algorithm, the Oslo algorithm by Cohen, Lyche, and Riesenfeld, and the introduction of the blossoming principle by Ramshaw and deCasteljau [7; 8].

The adopted approach here is the classic de Boor one, yet probably not the easiest to understand but surely with the most computing oriented flavor.

2.3.1 Shortcoming of polynomial and Bézier forms

The shortcomings of curves of just one polynomial or rational segment are:

- a high degree is required in order to satisfy a large number of constraints: a Bézier curve interpolating n points should be of $n - 1$ degree. High degrees leads to inefficiency and numerical instability;
- complex shapes require high degrees;
- a powerful interactive shape design requires local control which is not sufficiently achieved with single-segment curves (surfaces), even Bézier.

The natural solution is to piece together many segments obtaining a *piecewise polynomial* or *piecewise rational polynomial*, as depicted in fig.2.9. However the quality of this forms is not satisfying since:

- they show a lack of efficiency (storing more coefficients than required, think at the coincident points)
- little flexibility in control point positioning while maintaining continuity

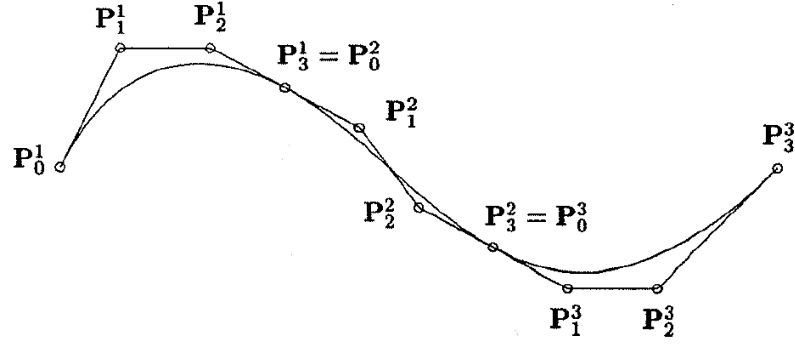


Figure 2.9: A piecewise cubic polynomial curve with three segments represented in Bézier form

- many computations are needed for determining the continuity of the geometric shape

What one is looking for is, in the case of a univariate shape, a curve representation of the form

$$\mathbf{C}(u) = \sum_{i=0}^n f_i(u) \mathbf{P}_i \quad (2.16)$$

where \mathbf{P}_i are *control points*, and f_i are generic piecewise polynomial functions forming a basis for the vector space of all piecewise polynomial functions of the desired degree and continuity. Here continuity is a matter of only the basis functions. Furthermore the blending functions should have all the previous seen analytic properties, which will transfer to nice geometric properties.

2.3.2 B-Spline Basis Functions

Let $U = (u_0, \dots, u_m)$ be a nondecreasing sequence of real numbers ($u_i \leq u_{i+1}$, $i = 0, \dots, m-1$). The u_i are the *knots*, U is the *knot vector*. The i -th basis function of

p -degree, denoted by $N_{i,p}(u)$, is recursively defined as :

$$N_{i,0}(u) = \begin{cases} 1 & \text{if } u_i \leq u \leq u_{i+1} \\ 0 & \text{otherwise} \end{cases} \quad (2.17)$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u)$$

The half-open interval $[u_i, u_{i+1})$ is called the i th *knot span*; it can have zero length since knots need not be distinct. A few basis functions are shown in fig.(2.10) for different degrees and knot vectors. Is important to notice that:

- $N_{i,0}$ is a step function, zero everywhere except on the i th knot span;
- $N_{i,p}$ is a **linear combination** of two $(p-1)$ -degree basis (fig.(2.11));
- if the eq.(2.17) yields the quotient $\frac{0}{0}$, then his value will be set to zero;
- the computation of the generic $N_{i,p}$ passes through the computation of zero-th degree basis from $N_{i,0}$ to $N_{i+p,0}$. It generates a truncated triangular table, shown in fig.(2.12).

Even if the basis function are defined on the whole real line, generally only the knot vector interval $[u_0, u_m]$ is of interest. The blending functions' properties are:

P.2.1 Local support: $N_{i,p}(u) = 0$ if u is outside the interval $[u_i, u_{i+p+1})$. For a proof look at the triangular scheme of fig.2.12.

P.2.2 In a given knot span $[u_j, u_{j+1}]$ at most $p+1$ of the $N_{i,p}$ are nonzero, namely the functions $N_{j-p,p}, \dots, N_{j,p}$. Again refer to the mentioned scheme for a proof.

P.2.3 Non negativity: $N_{i,p} \geq 0$ for every p, i and u .

P.2.4 Partition of unity: for an arbitrary knot span $\Delta U = [u_i, u_{i+1}]$ it holds:

$$\sum_{j=i-p}^i N_{j,p}(u) = 1 \quad \forall u \in \Delta U$$

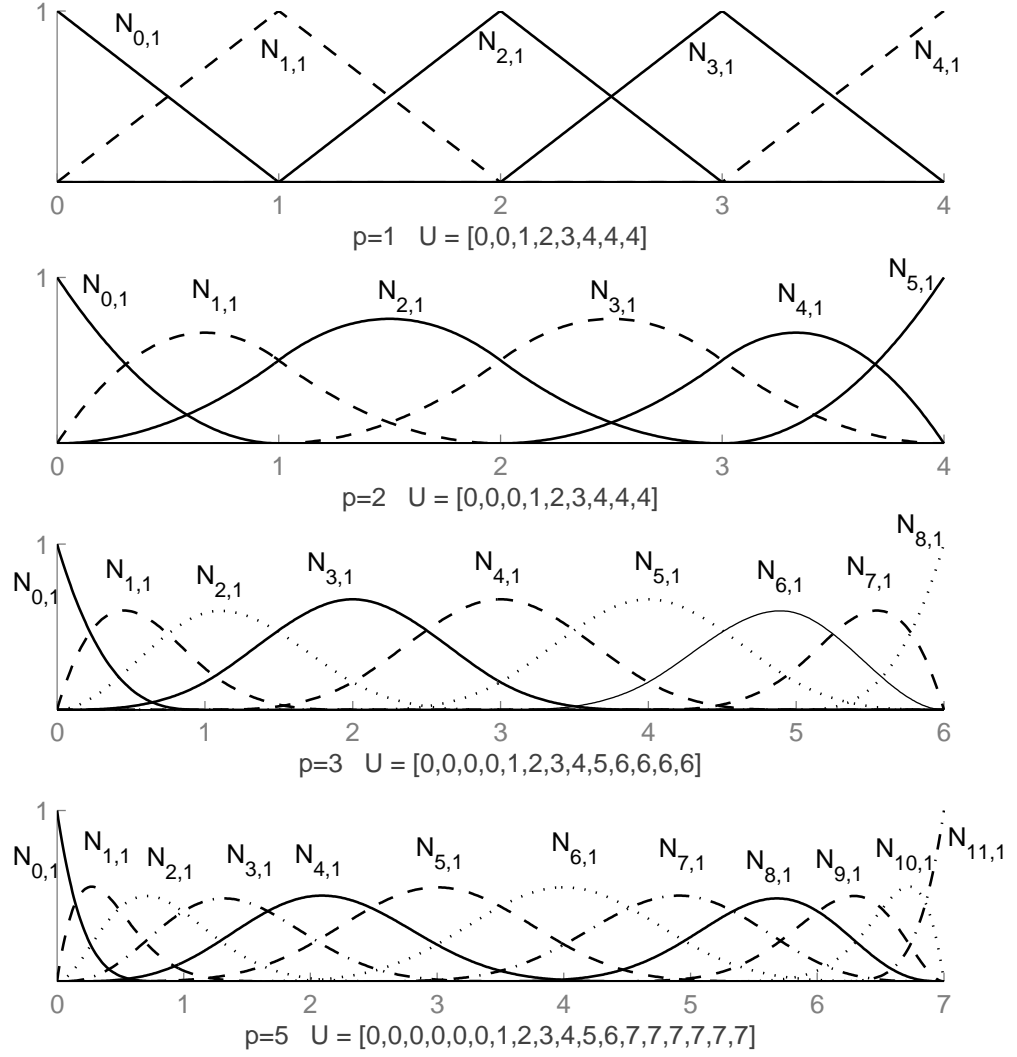


Figure 2.10: Non zero basis functions for different degrees and knot vectors

P.2.5 All derivatives of $N_{i,p}$ exist in the interior region of a knot span. At a knot, the basis function is $p - k$ times continuously differentiable, where k is the multiplicity of the knot.

P.2.6 A part from $p = 0$ (zeroth degree basis), $N_{i,p}$ attains exactly one maximum

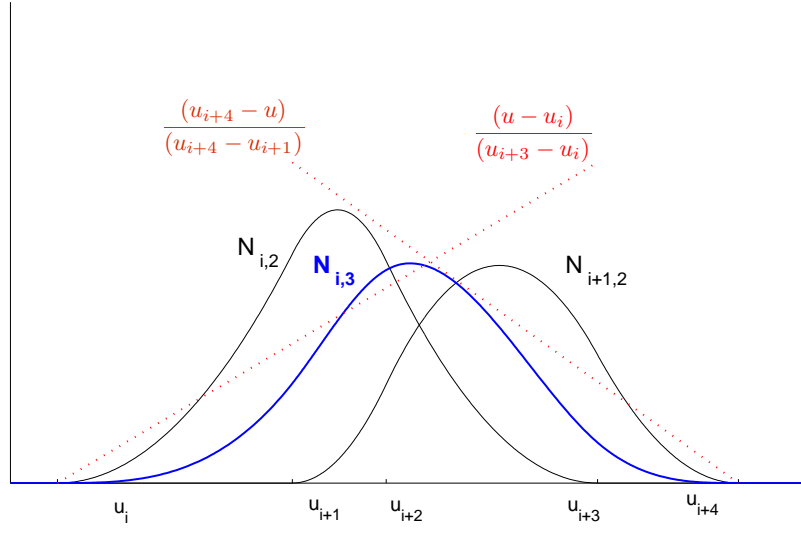


Figure 2.11: The recursive definition of B-spline basis: $N_{i,3}$ obtained as linear interpolation of $N_{i,2}$ and $N_{i+1,2}$

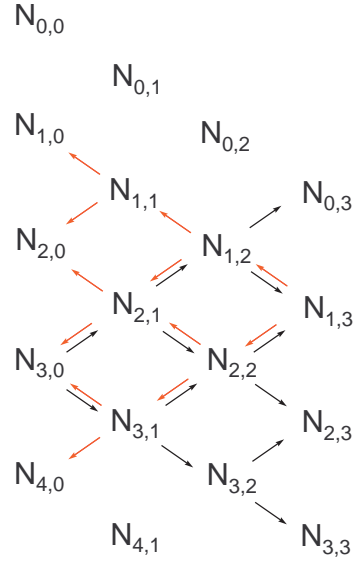


Figure 2.12: Dependencies between the basis functions

value.

Once the degree is fixed the knot vector completely determines the basis $N_{i,p}$. In literature there are more than one kind of knot vectors. The *non periodic* (or *clamped*,

or *open*) knot vectors have the following form:

$$U = \{\underbrace{a, \dots, a}_{p+1}, u_{p+1}, \dots, u_{m-p}, \underbrace{b, \dots, b}_{p+1}\} \quad (2.18)$$

in simple words, the first and last knots have multiplicity $p + 1$. As will be shown in the next section, the use of repeated knots ensures that the end points of the spline coincide with the end points of the control polygon. This representation is the most used in the CAGD, and will be implicitly assumed in the rest of the dissertation³. Additionally a knot vector is *uniform* if all interior knots are equally spaced, otherwise it is *nonuniform*. For a nonperiodic knot vectors there are two additional properties of the basis functions:

P.2.7 A knot vector of the form

$$U = \{\underbrace{0, \dots, 0}_{p+1}, \underbrace{1, \dots, 1}_{p+1}\}$$

yields the Bernstein polynomials of the same degree p .

P.2.8 If the number of knots is $m + 1$, then there are $n + 1$ basis functions, where $n = m - p - 1$; $N_{0,p}(a) = N_{n,p}(b) = 1$.

Derivatives of the basis functions

The derivative of a basis function is

$$N'_{i,p} = \frac{p}{u_{i+p} - u_i} N_{i,p-1}(u) - \frac{p}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u) \quad (2.19)$$

The derivative expression, like eq.(2.17), leave space for recursive definition; an examples is depicted in fig.2.14.

³is important to point out that the algorithms still hold, with small modifications, in case of *periodic* or *unclamped* knot vector

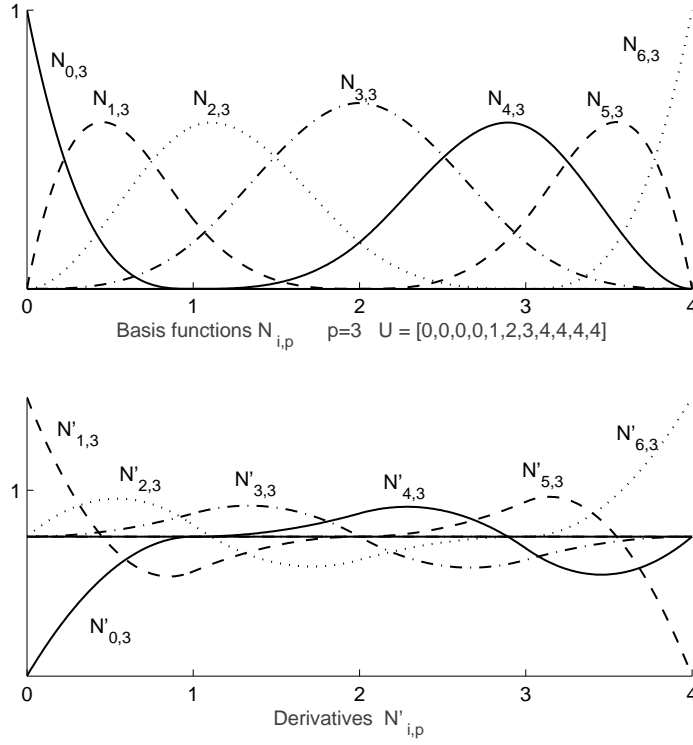


Figure 2.13: Cubic basis functions and corresponding derivatives

The repeated differentiation leads to the general formula

$$N_{i,p}^{(k)}(u) = p \left(\frac{N_{i,p-1}^{(k-1)}}{u_{i+p} - u_i} - \frac{N_{i+1,p-1}^{(k-1)}}{u_{i+p+1} - u_i} \right) \quad (2.20)$$

Another expression, giving the k th derivative of $N_{i,p}(u)$ in terms of k th derivative of $N_{i,p-1}$ and $N_{i+1,p-1}$ is

$$N_{i,p}^{(k)} = \frac{p}{p-k} \left(\frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}^{(k)} + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}^{(k)} \right) \quad (2.21)$$

Effects of multiple knots

Is important to understand the effect of multiple knots. Recalling property P.2.5 of the basis functions, let's take a deeper look with an easy but straightforward

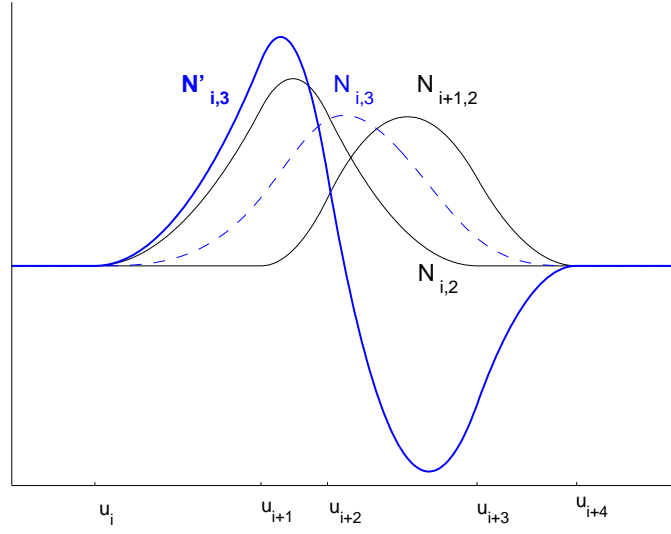


Figure 2.14: The recursive definition of B-spline derivatives: $N'_{i,3}$ as combination of $N_{i,2}$ and $N_{i+1,2}$

example. Let's assume there is just a double interior knot, $u_i = u_{i+1}$. From the recursive basis definition (eq.2.17) it follows that the basis function $N_{i,0}$ is set to zero, thus $N_{i-1,1}$ reaches the unity at his end and $N_{i,1}$ starts from the unity. The second degree basis function, $N_{i-1,2}$ (which has the double knot inside his support) is the linear interpolation of $N_{i-1,1}$ and $N_{i,1}$ (look at fig.2.15). From eq.(2.19), his derivative is also a combination of $N_{i-1,1}$ and $N_{i,1}$. It's easy to ascertain, with the aid of the aforementioned equation, that the right and left limits for $u \rightarrow u_i$ are different, and thus the function is not differentiable for $u = u_i$. Hence, a p -order basis function is $p - k$ times differentiable at $u = u_i$, with k being the knot multiplicity.

Another effect of multiple knots is to reduce the extension of the interval on which a basis function is nonzero. The implication will be more clear in the next section, when dealing with curves. On the geometric side, the basis functions involved with multiple knots will have a smaller support, look more distorted, like leaning toward the interested knots, and look less smooth.

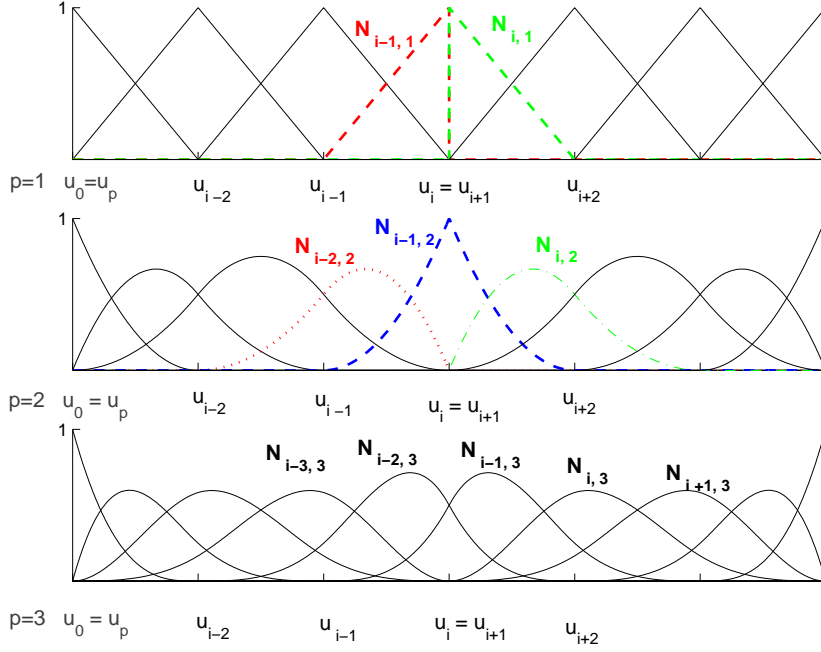


Figure 2.15: B-spline basis functions on a knot vector with a knot of multiplicity 2

2.3.3 B-Spline Curves

A p -th degree B-spline curve is defined by

$$\mathbf{C}(u) = \sum_{i=0}^n N_{i,p}(u) \mathbf{P}_i \quad u \in [a, b] \quad (2.22)$$

where \mathbf{P}_i are the n control points and $N_{i,p}$ are the previously defined p th-degree basis functions, defined on the nonperiodic knot vector:

$$U = \{\underbrace{a, \dots, a}_{p+1}, u_{p+1}, \dots, u_{m-p-1}, \underbrace{b, \dots, b}_{p+1}\}$$

Unless otherwise stated, the knot vector amplitude is normalized to the unity, i.e. $a = 0$ and $b = 1$. Much of the B-spline curve properties arises from the basis function properties:

P.3.1 If $n = p$ and $U = \{0, \dots, 0, 1, \dots, 1\}$, then $\mathbf{C}(u)$ is a Bézier curve.

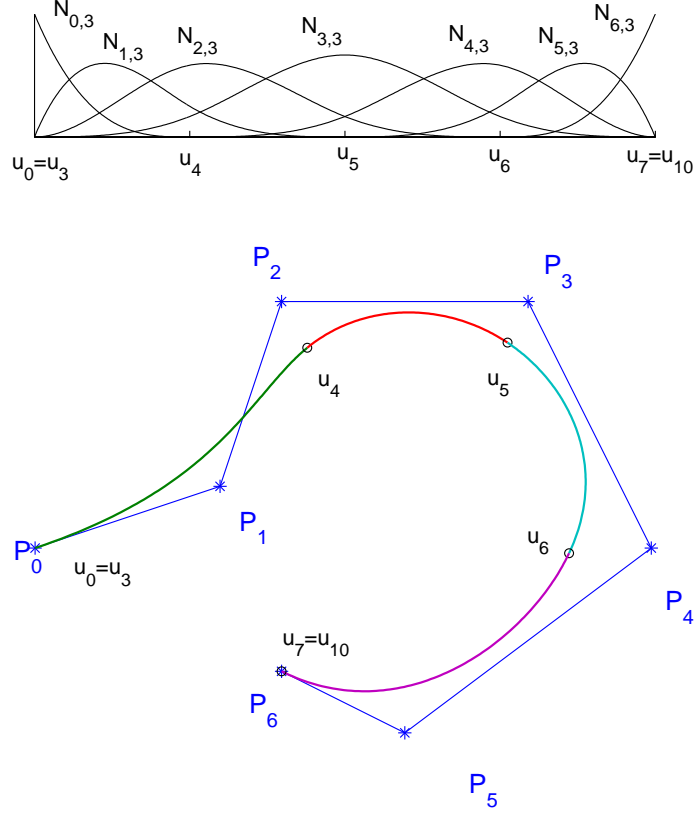


Figure 2.16: Cubic basis function over the knot vector $U = \{0,0,0,0,\frac{1}{4},\frac{1}{2},\frac{3}{4},1,1,1,1\}$, and associated cubic curve with control points P_i .

P.3.2 It yields the relation $m = n + p + 1$ which relates the knots and control point numbers and the degree.

P.3.3 The curve interpolates the control polygon endpoints, that is $\mathbf{C}(0) = \mathbf{P}_0$ and $\mathbf{C}(1) = \mathbf{P}_n$

P.3.4 Any affine transformation is applied to the curve by applying it to the control points (it follows from the partition of unit property of the basis functions).

P.3.5 Strong convex hull property: the curve is contained in the convex hull of its control polygon. If $u \in [u_i, u_{i+1})$, with $p \leq i < m - p - 1$ then $\mathbf{C}(u)$ is in the convex hull of the control points $\mathbf{P}_{i-p}, \dots, \mathbf{P}_i$, due to properties from P.2.2 to

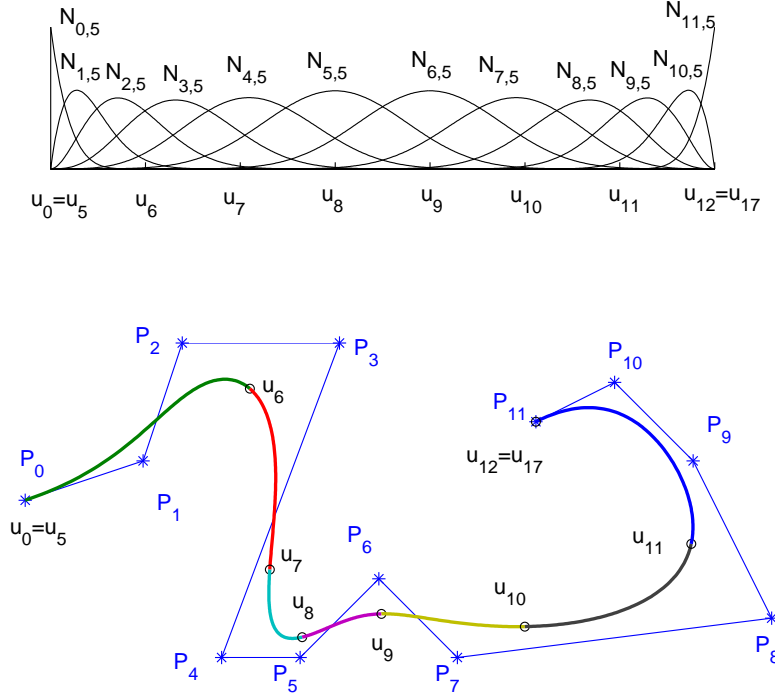


Figure 2.17: Fifth degree basis function over the knot vector $U = \{0, 0, 0, 0, 0, 0, \frac{1}{7}, \frac{2}{7}, \frac{3}{7}, \frac{4}{7}, \frac{5}{7}, \frac{6}{7}, 1, 1, 1, 1, 1\}$, and associated fifth degree B-spline curve with control points P_i .

P.2.4 of the basis functions. This property has practical application in many contests, like form manipulation, intersection etc.

P.3.6 Local modification scheme: moving a generic control point P_i changes the curve $C(u)$ only in the interval $[u_i, u_{i+p+1})$ (fig.2.18). This is a direct consequence of the local support of the basis function $N_{i,p}(u)$.

P.3.7 The control polygon represent a piecewise linear approximation to the curve; as a general rule, the lower the degree the closer the curve follows its control polygon.

P.3.8 The basis functions act likes switches in the movement along the curve from

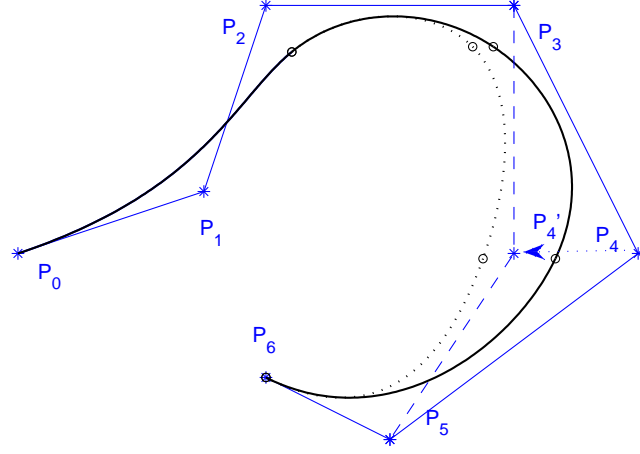


Figure 2.18: Cubic curve on $U = \{0,0,0,0,\frac{1}{4},\frac{1}{2},\frac{3}{4},1,1,1,1\}$. Moving \mathbf{P}_i changes the curve in the interval $[u_i, u_{i+p+1}]$.

$u = 0$ to $u = 1$; as u moves past a knot u_i , $N_{i-p,p}$ switches off (and so does the associated control point \mathbf{P}_{i-p}) and $N_{i+1,p}$ switches on.

P.3.9 Variation diminishing property: no plane has more intersections with the curve than with his control polygon.

P.3.10 The continuity and differentiability of $\mathbf{C}(u)$ follow from that of the $N_{i,p}$ (since $\mathbf{C}(u)$ is just a linear combination of the $N_{i,p}$). Thus, $\mathbf{C}(u)$ is infinitely differentiable in the interior of knots interval, and is at least $p - k$ times continuously differentiable at a knot of multiplicity k (fig.2.19). Anyway, sometimes even discontinuous functions can be combined in such way that the result is continuous, therefore a proper control point configuration can lead to continuity order higher than the one following from the basis functions, as shown in fig.2.20.

P.3.11 It is possible and sometimes useful to use multiple control points.

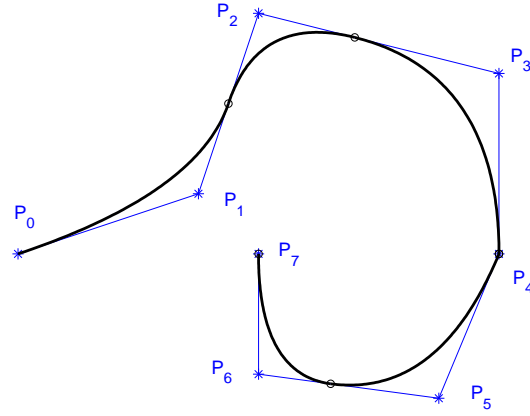


Figure 2.19: Quadratic curve on $U = \{0, 0, 0, \frac{1}{6}, \frac{1}{3}, \frac{7}{12}, \frac{7}{12}, \frac{5}{6}, 1, 1, 1\}$. Notice the cusp at $u = u_5 = u_6$

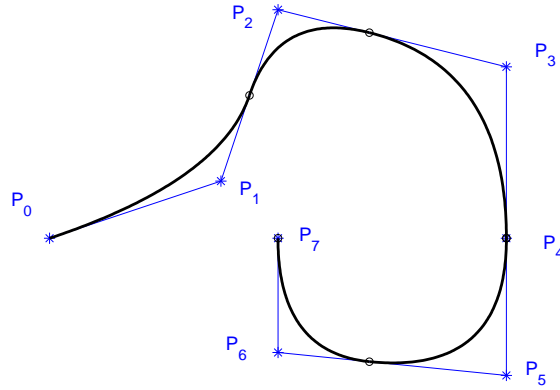


Figure 2.20: Cubic curve on $U = \{0, 0, 0, \frac{1}{6}, \frac{1}{3}, \frac{7}{12}, \frac{7}{12}, \frac{5}{6}, 1, 1, 1\}$. Even if the knot u_5 has multiplicity two, there aren't any cusps at $u = u_5 = u_6$

Derivatives of a B-spline curve

From eq.(2.22) it follows

$$\mathbf{C}^{(k)}(u) = \sum_{i=0}^n N_{i,p}^{(k)}(u) \mathbf{P}_i \quad (2.23)$$

Manipulating the last equation, with the aid of eq.(2.19), yields to a recursive scheme, particularly suited for computational purposes:

$$\mathbf{C}^{(k)}(u) = \sum_{i=0}^{n-k} N_{i,p-k}(u) \mathbf{P}_i^{(k)}$$

(2.24)

with $\mathbf{P}_i^{(k)} = \begin{cases} \mathbf{P}_i & k = 0 \\ \frac{p-k+1}{u_{i+p+1} - u_{i+k}} (\mathbf{P}_{i+1}^{(k-1)} - \mathbf{P}_i^{(k-1)}) & k > 0 \end{cases}$

The first order derivative is:

$$\mathbf{C}'(u) = \sum_{i=0}^{n-1} N_{i+1,p-1}(u) \mathbf{Q}_i \quad \text{where} \quad \mathbf{Q}_i = p \frac{\mathbf{P}_{i+1} - \mathbf{P}_i}{u_{i+p+1} - u_{i+1}} \quad (2.25)$$

Let U' be the knot vector obtained from U dropping the first and last knot

$$U' = \{\underbrace{0, \dots, 0}_p, u_{p+1}, \dots, u_{m-p-1}, \underbrace{1, \dots, 1}_p\}$$

The function $N_{i+1,p-1}$ computed on U is equal to $N_{i,p-1}$ computed on U' , thus:

$$\mathbf{C}'(u) = \sum_{i=0}^{n-1} N_{i,p-1}(u) \mathbf{Q}_i \quad (2.26)$$

Hence, \mathbf{C}' is a $(p-1)$ th-degree B-spline curve.

The endpoints first derivatives of a B-spline curve are given by

$$\begin{aligned} \mathbf{C}'(0) &= \mathbf{Q}_0 = \frac{p}{u_{p+1}} (\mathbf{P}_1 - \mathbf{P}_0) \\ \mathbf{C}'(1) &= \mathbf{Q}_{n-1} = \frac{p}{1 - u_{m-p-1}} (\mathbf{P}_n - \mathbf{P}_{n-1}) \end{aligned} \quad (2.27)$$

Equation (2.27) has a noticeable geometric interpretation: the derivative at the endpoints is the vector joining the two end control points, in the increasing sense.

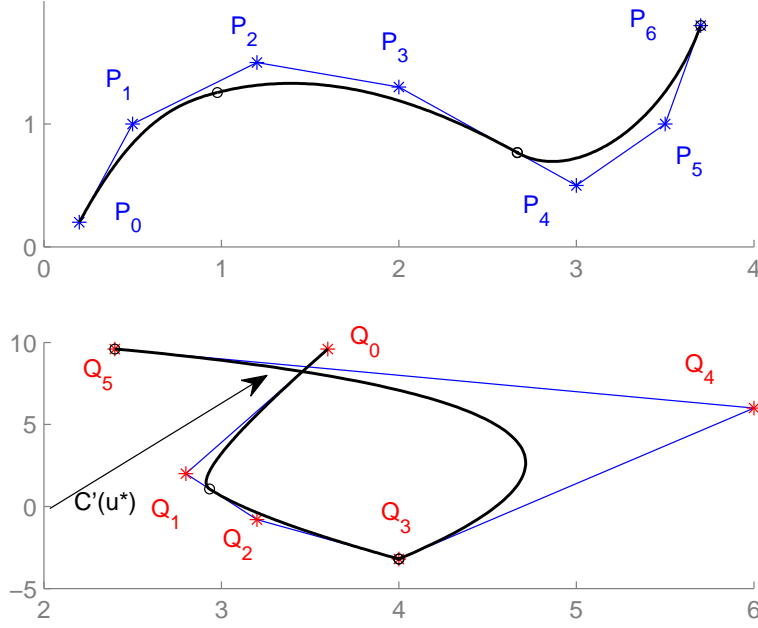


Figure 2.21: Cubic curve on $U = \{0,0,0,0,\frac{1}{4},\frac{3}{4},\frac{3}{4},1,1,1,1\}$. First derivative is a quadratic B-spline on knot vector $U = \{0,0,0,\frac{1}{4},\frac{3}{4},\frac{3}{4},1,1,1\}$ with control points Q_i defined as in eq.(2.25).

The second derivative at endpoints :

$$\mathbf{C}^{(2)}(0) = \frac{p(p-1)}{u_{p+1}} \left(\frac{\mathbf{P}_0}{u_{p+1}} - \frac{(u_{p+1} + u_{p+2})\mathbf{P}_1}{u_{p+1}u_{p+2}} + \frac{\mathbf{P}_2}{u_{p+2}} \right) \quad (2.28)$$

$$\begin{aligned} \mathbf{C}^{(2)}(1) &= \frac{p(p-1)}{1 - u_{m-p-1}} \cdot \\ &\left(\frac{\mathbf{P}_n}{1 - u_{m-p-1}} - \frac{(2 - u_{m-p-1} - u_{m-p-2})\mathbf{P}_{n-1}}{(1 - u_{m-p-1})(1 - u_{m-p-2})} + \frac{\mathbf{P}_{n-2}}{1 - u_{m-p-2}} \right) \end{aligned} \quad (2.29)$$

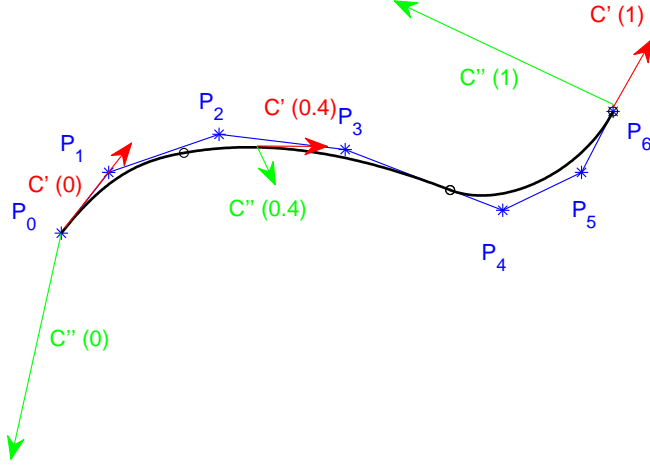


Figure 2.22: A cubic curve on $U = \{0,0,0,0,\frac{1}{4},\frac{3}{4},1,1,1,1\}$ and his first and second derivatives at endpoints, and at $u = 0.4$ (the first derivatives are scaled of a ten factor, the second of a twenty factor).

2.3.4 B-Spline Surfaces

Given a bidirectional net of control points, two knot vectors and the associated univariate B-spline functions, a B-spline surface is defined by:

$$\mathbf{S}(u,v) = \sum_{i=0}^n \sum_{j=0}^m N_{i,p}(u) N_{j,q}(v) \mathbf{P}_{i,j} \quad (2.30)$$

with

$$U = \underbrace{\{0, \dots, 0\}}_{p+1}, u_{p+1}, \dots, u_{r-p-1}, \underbrace{\{1, \dots, 1\}}_{p+1}$$

$$V = \underbrace{\{0, \dots, 0\}}_{q+1}, v_{q+1}, \dots, v_{s-q-1}, \underbrace{\{1, \dots, 1\}}_{q+1}$$

U and V are the knot vectors, and have respectively $r+1$ and $s+1$ knots. Extending property P3.3 of B-Spline curves (section 2.3.3) yields to: $r = n + p + 1$ and $s =$

$m + q + 1$. Using the matrix form yields to:

$$\mathbf{S}(u,v) = [N_{k,p}(u)]^T [\mathbf{P}_{k,l}] [N_{l,q}(v)] \quad (2.31)$$

Tensor product basis functions

The tensor product (or bivariate) basis functions properties arises from the corresponding properties of the univariate basis functions (listed in section 2.3.2):

P.4.1 Nonnegativity: $N_{i,p}(u) N_{j,q}(v) \geq 0$ for all i, j, p, q, u, v .

P.4.2 Partition of unity: $\sum_{i=0}^n \sum_{j=0}^m N_{i,p}(u) N_{j,q}(v) = 1$ for all $(u,v) \in [0,1] \times [0,1]$.

P.4.3 The functions degenerate to products of Bernstein polynomials of the same degree, that is, $N_{i,p}(u) N_{j,q}(v) = B_{i,p}(u) B_{j,q}(v)$ for all i, j , if $U = \underbrace{\{0, \dots, 0\}}_{p+1}, \underbrace{\{1, \dots, 1\}}_{p+1}$ and $V = \underbrace{\{0, \dots, 0\}}_{q+1}, \underbrace{\{1, \dots, 1\}}_{q+1}$.

P.4.4 Local support: $N_{i,p}(u) N_{j,q}(v) = 0$ if (u,v) is outside $[u_i, u_{i+p+1}) \times [v_j, v_{j+q+1})$.

P.4.5 In any domain $[u_{i_0}, u_{i_0+1}) \times [v_{j_0}, v_{j_0+1})$ at most $(p+1)(q+1)$ basis functions are nonzero, in particular the $N_{i,p} N_{j,q}$ for $i_0 - p \leq i \leq i_0$ and $j_0 - q \leq j \leq j_0$.

P.4.6 If $p, q > 0$ then $N_{i,p} N_{j,q}$ attains exactly one maximum.

P.4.7 In the interior region of every domain defined by u and v knot lines, all the partial derivatives of $N_{i,p} N_{j,q}$ exist; at a $u(v)$ knot it is $p - k(q - k)$ times differentiable with respect to $u(v)$, where k is the knot multiplicity.

B-spline surfaces

Again, the B-spline surfaces properties arises from the basis function properties:

P.5.1 Any surface defined over the knot vectors depicted in basis functions property P.4.3 is a Bézier surface.

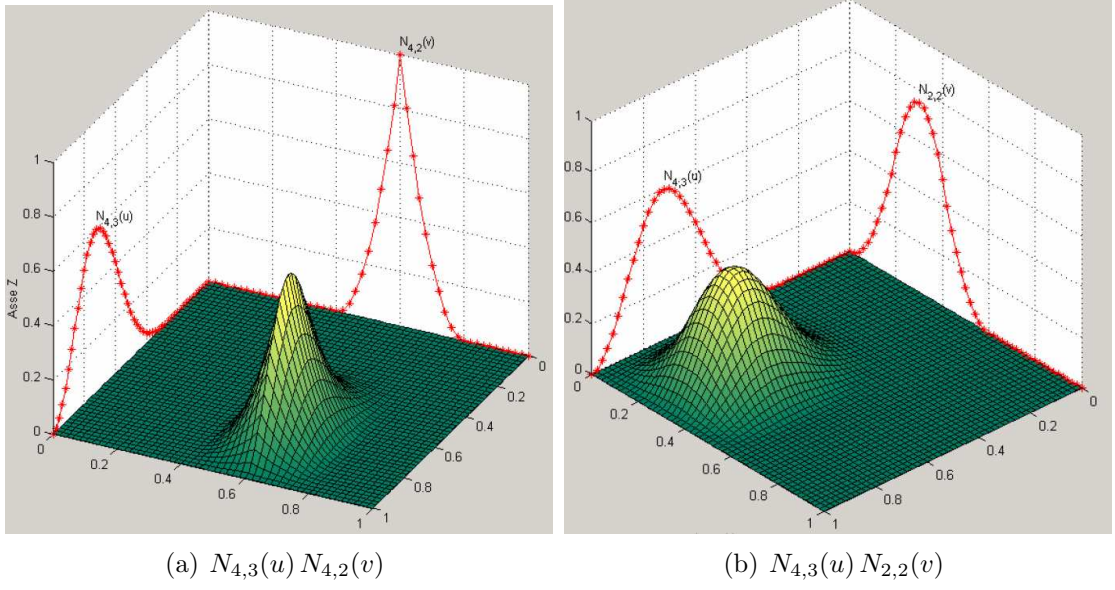


Figure 2.23: Cubic \times quadratic basis functions. $U = \{0,0,0,0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1,1,1,1\}$,
 $V = \{0,0,0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1,1,1\}$

P.5.2 Due to property P.4.2, the surface interpolates the four corner control points:

$$\mathbf{S}(0,0) = \mathbf{P}_{0,0}, \mathbf{S}(1,0) = \mathbf{P}_{n,0}, \mathbf{S}(0,1) = \mathbf{P}_{0,m}, \mathbf{S}(1,1) = \mathbf{P}_{n,m}$$

P.5.3 Affine invariance: an affine transformation is applied to the surface by applying it to the control points; this follows from P.4.2.

P.5.4 Strong convex hull property: if $(u,v) \in [u_{i_0}, u_{i_0+1}] \times [v_{j_0}, v_{j_0+1}]$, then $\mathbf{S}(u,v)$ is in the convex hull of the control points $\mathbf{P}_{i,j}$, $i_0 - p \leq i \leq i_0$ and $j_0 - q \leq j \leq j_0$; this follow from P.4.1, P.4.2, P.4.5.

P.5.5 Triangulation of the control net forms a piecewise planar approximation to the surface. As for the curves, the lower the degree the better the approximation (see P.3.7).

P.5.6 Local modification scheme: movement of the control point $\mathbf{P}_{i,j}$ affects the surface only in the rectangle $[u_i, u_{i+p+1}] \times [v_j, v_{j+q+1}]$; this is a consequence of property local support of the basis functions (P.4.4).

P.5.7 Continuity and differentiability: follows from that of the basis functions. That is, $\mathbf{S}(u,v)$ is $p - k$ ($q - k$) times differentiable in the u (v) direction at a u (v) knot of multiplicity k . Again, as for the curves, it is possible to position the control points in such a way to overcome the effect of discontinuities of the basis functions. It is also possible to use multiple coincident control points.

Unlike univariate case, there is no known variation diminishing property for B-spline surfaces.

Fixed $u = u_0$ the respective v -isoparametric curve on $\mathbf{S}(u,v)$ has the following expression:

$$\begin{aligned} \mathbf{C}_{u_0}(v) = \mathbf{S}(u_0, v) &= \sum_{j=0}^m N_{j,q}(v) \left(\sum_{i=0}^n N_{i,p}(u_0) \mathbf{P}_{i,j} \right) = \sum_{j=0}^m N_{j,q}(v) \mathbf{Q}_j(u_0) \\ \text{where } \mathbf{Q}_j(u_0) &= \sum_{i=0}^n N_{i,p}(u_0) \mathbf{P}_{i,j} \end{aligned} \quad (2.32)$$

In fig.2.24 a B-spline surface with the control polygon is shown.

Derivatives of a B-spline surface

The derivatives are obtained computing derivatives of the basis functions.

$$\frac{\partial^{k+1}}{\partial^k u \partial^l v} \mathbf{S}(u,v) = \sum_{i=0}^n \sum_{j=0}^m N_{i,p}^{(k)} N_{j,q}^{(l)} \mathbf{P}_{i,j} \quad (2.33)$$

Formally differentiating $\mathbf{S}(u,v)$ and using eq.(2.26) yields to:

$$\mathbf{S}_u(u,v) = \sum_{i=0}^{n-1} \sum_{j=0}^m N_{i,p-1}(u) N_{j,q}(v) \mathbf{P}_{i,j}^{(1,0)} \quad (2.34)$$

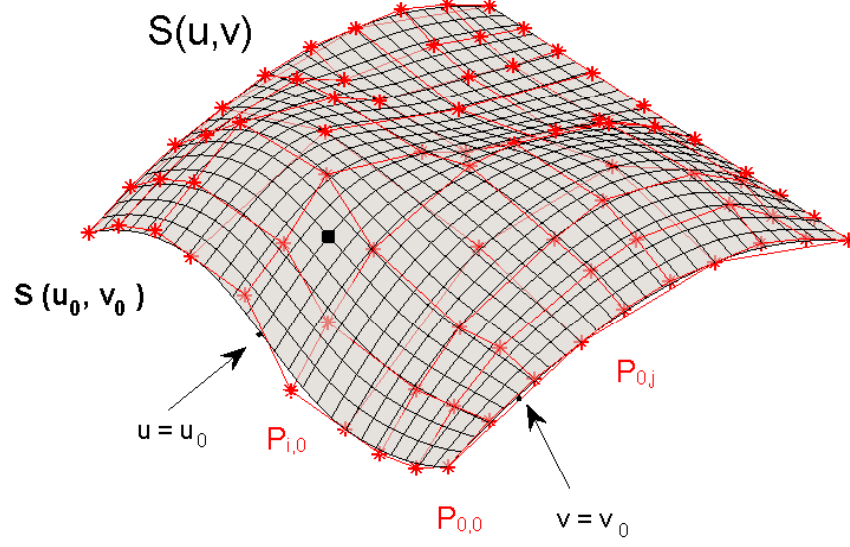


Figure 2.24: A B-spline surface and its control net (in red). Note the isoparametric curves on the surface.

where

$$\mathbf{P}_{i,j}^{(1,0)} = p \frac{\mathbf{P}_{i+1,j} - \mathbf{P}_{i,j}}{u_{i+p+1} - u_{i+1}}$$

$$U^{(1)} = \{\underbrace{0, \dots, 0}_p, u_{p+1}, \dots, u_{r-p-1}, \underbrace{1, \dots, 1}_p\}$$

$$V^{(0)} = V$$

In a symmetrical way:

$$S_v(u,v) = \sum_{i=0}^{n-1} \sum_{j=0}^m N_{i,p}(u) N_{j,q-1}(v) \mathbf{P}_{i,j}^{(0,1)} \quad (2.35)$$

where

$$\mathbf{P}_{i,j}^{(0,1)} = q \frac{\mathbf{P}_{i,j+1} - \mathbf{P}_{i,j}}{u_{j+q+1} - u_{j+1}}$$

$$U^{(0)} = U$$

$$V^{(1)} = \{\underbrace{0, \dots, 0}_q, v_{q+1}, \dots, u_{s-q-1}, \underbrace{1, \dots, 1}_q\}$$

Applying first eq.(2.34), then eq.(2.35) yields

$$\mathbf{S}_{uv}(u, v) = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} N_{i,p-1}(u) N_{j,q-1}(v) \mathbf{P}_{i,j}^{(1,1)} \quad (2.36)$$

where

$$\mathbf{P}_{i,j}^{(1,1)} = q \frac{\mathbf{P}_{i,j+1}^{(1,0)} - \mathbf{P}_{i,j}^{(1,0)}}{v_{j+q+1} - v_{j+1}}$$

The general formula is given from

$$\frac{\partial^{k+1}}{\partial^k u \partial^l v} \mathbf{S}(u, v) = \sum_{i=0}^{n-k} \sum_{j=0}^{m-l} N_{i,p-k}(u) N_{j,q-l}(v) \mathbf{P}_{i,j}^{(k,l)} \quad (2.37)$$

where

$$\mathbf{P}_{i,j}^{(k,l)} = (q-l+1) \frac{\mathbf{P}_{i,j+1}^{(k,l-1)} - \mathbf{P}_{i,j}^{(k,l-1)}}{v_{j+q+1} - v_{j+l}}$$

It would be useful to derive formulas for corner derivatives; using equations from (2.34) to (2.37):

$$\begin{aligned}
 \mathbf{S}_u(0,0) &= \mathbf{P}_{0,0}^{(1,0)} = \frac{p}{u_{p+1}}(\mathbf{P}_{1,0} - \mathbf{P}_{0,0}) \\
 \mathbf{S}_v(0,0) &= \mathbf{P}_{0,0}^{(0,1)} = \frac{q}{v_{q+1}}(\mathbf{P}_{0,1} - \mathbf{P}_{0,0}) \\
 \mathbf{S}_{uv}(0,0) &= \mathbf{P}_{0,0}^{(1,1)} = \frac{q}{v_{q+1}}(\mathbf{P}_{0,1}^{(1,0)} - \mathbf{P}_{0,0}^{(1,0)}) \\
 &= \frac{p q}{u_{p+1} v_{q+1}}(\mathbf{P}_{1,1} - \mathbf{P}_{0,1} - \mathbf{P}_{1,0} + \mathbf{P}_{0,0})
 \end{aligned} \tag{2.38}$$

Now let $u_0 = 0$ and $v_0 = 0$. Recalling basis function properties is easy to state that the isocurves $\mathbf{C}_{u_0}(v)$ and $\mathbf{C}_{v_0}(u)$ are given by:

$$\mathbf{C}_{u_0}(v) = \sum_{j=0}^m N_{j,q}(v) \mathbf{P}_{0,j} \quad \mathbf{C}_{v_0}(u) = \sum_{i=0}^n N_{i,p}(u) \mathbf{P}_{i,0} \tag{2.39}$$

From eq.(2.27) it follows that:

$$\mathbf{S}_u(0,0) = \mathbf{C}'_{u_0}(0) \quad \mathbf{S}_v(0,0) = \mathbf{C}'_{v_0}(0) \tag{2.40}$$

2.4 Rational B-Splines

Although B-spline introduction give a versatile and powerful tool to model complex shapes, they can't represent precisely important geometric shapes as conics. In fact, B-splines basis are basically non-rational polynomials, thus are not capable of representing the over mentioned class of shapes. It is known from classical mathematics that all the conic curves can be represented using rational functions, that is functions that are defined as the ratio of two polynomials. Therefore, to use such class of functions as basis functions yields to Nurbs, Non uniform rational B-splines, a generalization of B-splines with the same advantages but with a bigger design flexibility.

2.4.1 Nurbs Curves

A p th-degree NURBS curve is defined as:

$$\mathbf{C}(u) = \frac{\sum_{i=0}^n N_{i,p}(u) w_i \mathbf{P}_i}{\sum_{i=0}^n N_{i,p}(u) w_i} \quad u \in [a, b] \quad (2.41)$$

where $N_{i,p}$ are the B-spline basis functions defined over the same non periodic knot vector (see section 2.3.3), w_i are the *weights*. Assume, unless otherwise stated that $a = 0$ and $b = 1$, and $w_i > 0 \quad \forall i$. Defining the *rational basis functions* $R_{i,p}$ as

$$R_{i,p} = \frac{N_{i,p}(u) w_i}{\sum_{j=0}^n N_{j,p}(u)} \quad (2.42)$$

allows to rewrite eq.(2.41) in the form

$$\mathbf{C}(u) = \sum_{i=0}^n R_{i,p}(u) \mathbf{P}_i \quad (2.43)$$

The properties of the NURBS basis functions follows the properties of B-spline basis functions and eq.(2.42). The $R_{i,p}$ have the same properties of $N_{i,p}$, with only a one more consideration:

- if $w = \bar{w}$ ($\bar{w} \neq 0$) for all i , then $N_{i,p} = R_{i,p}$ for all i .

NURBS curve properties arise from rational basis functions and are, a part from the properties related with weights w_i , coincident with the B-spline curve properties (2.3.3). The weights add a further flexibility since:

- if weight w_i is changed, it affects only the portion of the curve on the interval $u \in [u_i, u_{i+p+1})$. Qualitatively an increment of w_i pulls the portion of curve $\mathbf{C}(u)$ with $u \in [u_i, u_{i+p+1})$ toward \mathbf{P}_i ; the opposite does a decrement of w_i . Varying w_i the movement of $\mathbf{C}(u)$ for fixed u , is along a straight line passing for \mathbf{P}_i .

Figure 2.25 illustrate such a behavior. Thereby, both control point movements and weight modification give a local shape control, providing great flexibility in interactive shape design.

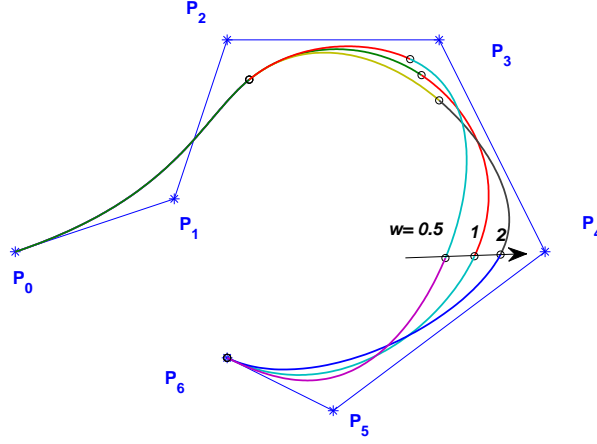


Figure 2.25: Effect of weight modification (w_4) on a NURBS curve.

Rational curves with coordinate functions in the form of ratio of two polynomials, with the same denominator for each of the coordinate, have an elegant geometric interpretation. Defining a perspective map, H , with center at the origin, which maps from a four-dimensional space to a three-dimensional Euclidean space

$$\mathbf{P} = H\{\mathbf{P}^w\} = H\{(X, Y, Z, W)\} = \begin{cases} \left(\frac{X}{W}, \frac{Y}{W}, \frac{Z}{W} \right) & \text{if } W \neq 0 \\ \text{direction } (X, Y, Z) & \text{if } W = 0 \end{cases} \quad (2.44)$$

For a given set of control points, $\{\mathbf{P}_i\}$, and weights $\{w_i\}$, let's construct the four dimensional weighted control points $\mathbf{P}_i^w = (w_i x_i, w_i y_i, w_i z_i, w_i)$ and define the nonrational B-spline curve in four dimensional space as

$$\mathbf{C}^w(u) = \sum_{i=0}^n N_{i,p}(u) \mathbf{P}_i^w \quad (2.45)$$

Applying the perspective map, H , to $\mathbf{C}^w(u)$ yields the following rational B-spline curve:

$$\begin{aligned}\mathbf{C}(u) &= H\{\mathbf{C}^w(u)\} = H\left\{\sum_{i=0}^n N_{i,p}(u) \mathbf{P}_i^w\right\} \\ &= \frac{\sum_{i=0}^n N_{i,p}(u) w_i \mathbf{P}_i}{\sum_{i=0}^n N_{i,p}(u) w_i} = \sum_{i=0}^n R_{i,p}(u) \mathbf{P}_i\end{aligned}\tag{2.46}$$

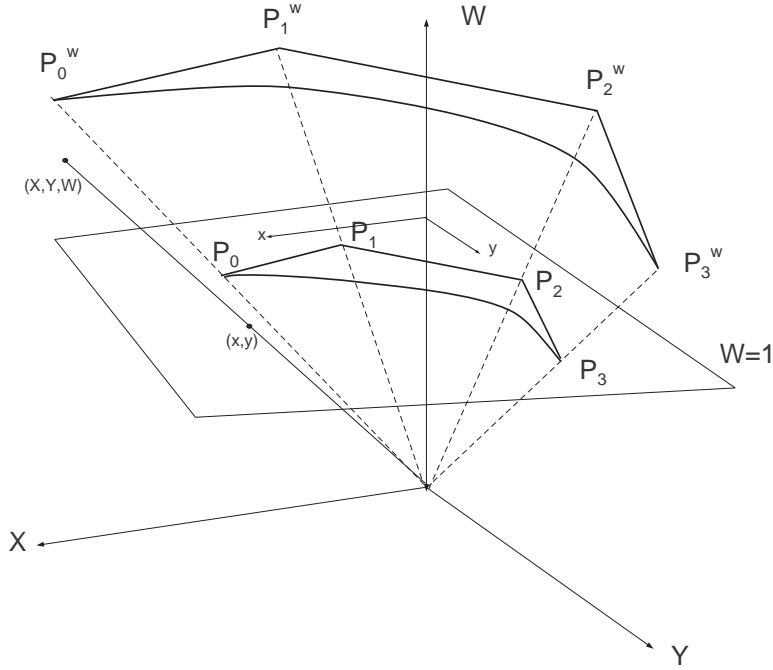


Figure 2.26: A geometric construction of a rational B-spline curve.

This is an important point since leads to efficient processing and data storage: the NURBS curves can be efficiently handled as B-splines in a four-dimensional space. In some situation of free form shape design, like interpolation of set of points with fair geometric shapes, the mapping leads to a linear problem instead of a non linear one [10].

Derivatives of NURBS curves

Derivatives of rational functions are complicated, involving denominators to high powers. The easiest way to compute derivatives is relying on the four dimensional B-spline representation, for which applies the previous formulas derived for three-dimensional space (section 2.3.3). For a detailed analysis refer to [8]. Evaluation of first order derivative at endpoints yields to:

$$\mathbf{C}'(0) = \frac{p}{u_{p+1}} \frac{w_1}{w_0} (\mathbf{P}_1 - \mathbf{P}_0) \quad (2.47)$$

and

$$\mathbf{C}'(1) = \frac{p}{1 - u_{m-p-1}} \frac{w_n - 1}{w_n} (\mathbf{P}_n - \mathbf{P}_{n-1}) \quad (2.48)$$

2.4.2 Nurbs Surfaces

The generalization from univariate to bivariate NURBS basis functions can be deduced from the same B-spline generalization, depicted in section 2.3.4. A NURBS surface of degree p, q in the u and v directions is a bivariate vector-valued piecewise rational function described by

$$\mathbf{S}(u, v) = \frac{\sum_{i=0}^n \sum_{j=0}^m N_{i,p}(u) N_{j,q}(v) w_{i,j} \mathbf{P}_{i,j}}{\sum_{i=0}^n \sum_{j=0}^m N_{i,p}(u) N_{j,q}(v) w_{i,j}} \quad 0 \leq u, v \leq 1 \quad (2.49)$$

where the terminology is the same as for NURBS curve and B-spline surfaces (see sections 2.4.1 and 2.3.4). Introducing the bivariate rational basis functions

$$R_{i,j}(u, v) = \frac{N_{i,p}(u) N_{j,q}(v) w_{i,j}}{\sum_{k=0}^n \sum_{l=0}^m N_{k,p}(u) N_{l,q}(v) w_{k,l}} \quad (2.50)$$

the surface can be written as

$$\mathbf{S}(u, v) = \sum_{i=0}^n \sum_{j=0}^m R_{i,j}(u, v) \mathbf{P}_{i,j} \quad (2.51)$$

The bivariate rational basis functions properties are the same as the B-spline one's. Note that

- if all weights $w_{i,j} = \bar{w}$ with $\bar{w} \neq 0$ then $R_{i,j}(u,v) = N_{i,p}(u) N_{j,q}(v)$ for all i, j .

The same properties of B-spline surfaces are carried here, (with the proper generalization from non rational to rational polynomials). Added property is the local shape modification through the weights movements. Increasing a weight $w_{i,j}$ pulls the points of the surface $\mathbf{S}(u,v)$ with $(u,v) \in [u_i, u_{i+p+1}) \times [v_j, v_{j+q+1})$ toward $\mathbf{P}_{i,j}$, the opposite does a weight decrement. As for the curves, the movements of $\mathbf{S}(u,v)$ is along a straight line passing for $\mathbf{P}_{i,j}$.

The homogeneous coordinates representation is convenient also for NURBS surfaces:

$$\mathbf{S}^w(u,v) = \sum_{i=0}^n \sum_{j=0}^m N_{i,p}(u) N_{j,q}(v) \mathbf{P}_{i,j}^w \quad (2.52)$$

where $\mathbf{P}_{i,j}^w = (w_{i,j} x_{i,j}, w_{i,j} y_{i,j}, w_{i,j} z_{i,j})$. Thus $\mathbf{S}(u,v) = H\{\mathbf{S}^w(u,v)\}$.

Isoparametric on NURBS surfaces are easily defined with homogeneous coordinates.

Fixed $u = u_0$:

$$\mathbf{C}_{u_0}^w(v) = \mathbf{S}^w(u_0, v) = \sum_{i=0}^2 \sum_{j=0}^n N_{i,p}(u_0) N_{j,q}(v) \mathbf{P}_{i,j}^w \quad (2.53)$$

$$(2.54)$$

2.5 Fundamental Geometric Algorithms

In order to implement and manipulate NURBS, a group of fundamental algorithms is necessary. In this section only a brief discussion on some of the most important processes is given. For deeper analysis and mathematical details refer to [8]. It is worth a note that some of this algorithms are conceptually extendible to other parametric forms, and are not just NURBS concerned.

2.5.1 Knot Insertion

Let $\mathbf{C}(u)$ be a B-spline⁴ defined on the generic knot vector $U = \{u_0, \dots, u_n\}$, and with control points \mathbf{P}_i . Now consider \bar{U} obtained from U adding a knot $\bar{u} \in [u_0, u_n]$. This algorithm determines a new set of control point \mathbf{Q}_i such that :

$$\mathbf{C}(u) = \sum_{i=0}^n N_{i,p} \mathbf{P}_i = \sum_{i=0}^{n+1} \bar{N}_{i,p} \mathbf{Q}_i \quad (2.55)$$

Actually the results show that only p new control points must be computed. It is also possible to insert knot multiple times, or many knots at time (such an algorithm is called *knot refinement*). Due to tensor product surface characteristics, the algorithm is easily extended to surfaces.

Knot insertion is one of the most important of all the B-spline algorithms, being useful to:

- valuating points and derivatives on curves and surfaces,
- subdividing curves and surfaces,
- adding control points in order to increase flexibility in shape control (interactive design).

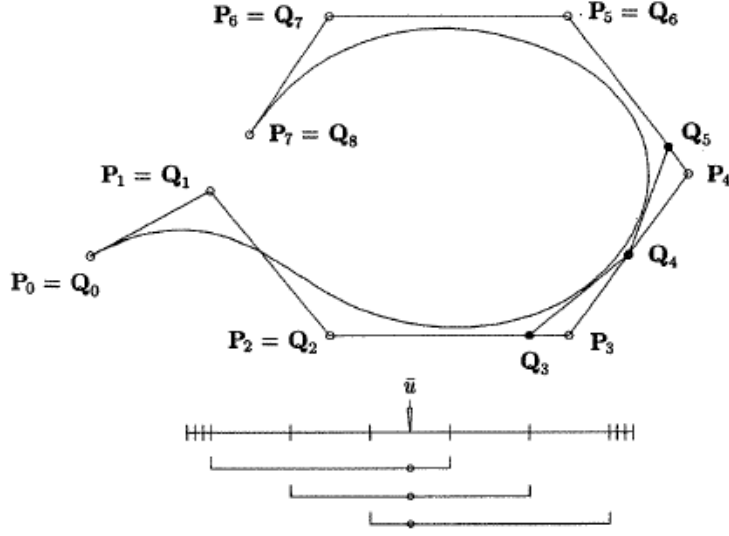
Figure 2.27 shows an example of knot insertion on a B-spline curve.

2.5.2 Knot Removal

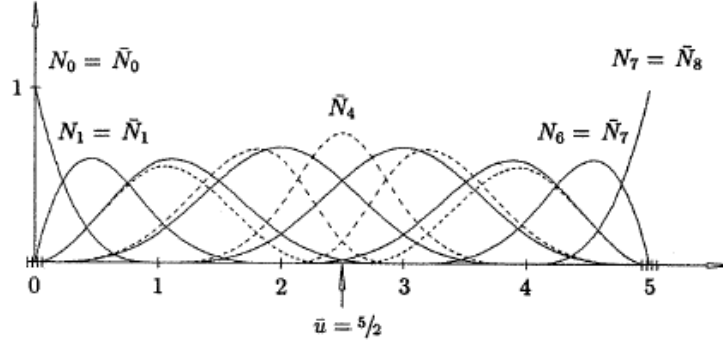
Knot removal is the reverse process of knot insertion. However, it is more complicated since it's not always possible to remove a knot; thus a knot removal algorithm must first determine if the knot is removable and how many times, then eventually compute the new control points, \mathbf{Q}_i . Knot removal utility carries out in these situations:

- converting a spline curve or surface presented in power basis form to B-spline form;

⁴generalization to NURBS is easily obtained expressing the control points in homogeneous coordinates



(a) Control polygon after inserting $\bar{u} = \frac{5}{2}$ into the knot vector U



(b) Original (solid) and the new (dashed) basis functions before and after knot insertion.

Figure 2.27: Knot insertion into a cubic curve defined over the knot vector $U = \{0,0,0,0,1,2,3,4,5,5,5,5\}$ (drawn from [8]).

- when interactively shaping B-spline knots are sometimes added to increase flexibility; then, after manipulation, a knot removal can be invoked in order to obtain the most compact representation of the curve or surface.

Fig.2.28 summarizes the steps of knot removal from a B-spline curve .

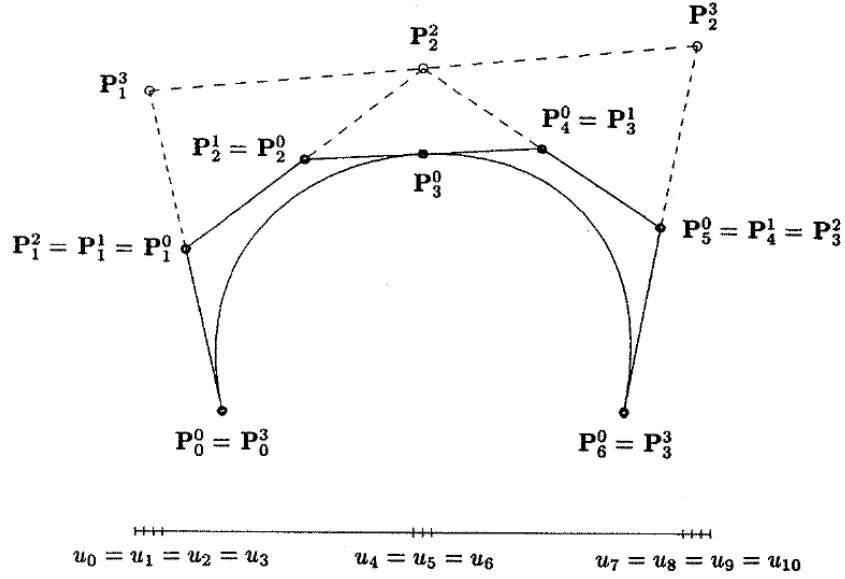


Figure 2.28: Knot removal from a cubic curve with triple knot (from [8]).

2.5.3 Degree Elevation

Given a B-spline curve (surface), degree elevation is the process of obtaining an higher degree curve (surface) identical to the original one, both geometrically and parametrically. Thus, referring to the following mathematical problem, involving a single degree elevation for a curve

$$\mathbf{C}_p(u) = \sum_{i=0}^n N_{i,p} \mathbf{P}_i = \mathbf{C}_{p+1}(u) = \sum_{i=0}^{\bar{n}} \bar{N}_{i,p+1} \mathbf{Q}_i \quad (2.56)$$

degree elevation computes the unknown \mathbf{Q}_i and \bar{U} . Figure 2.29 summarizes degree elevation of a third degree B-spline curve.

The two typical situations in which degree elevation is involved are:

- construction of surfaces from a set of curves. Using tensor product surfaces requires that these curves have a common degree (as will be seen in subsequent chapters).

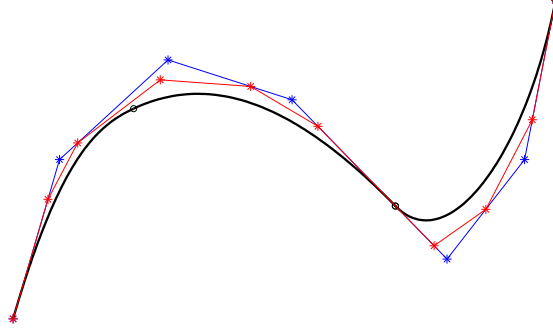


Figure 2.29: Degree elevation on a third degree B-spline curve, $U = \{0,0,0,0,1,2,3,4,4,4,4\}$. The new control polygon is the one in red.

- combining two NURBS curves with a common endpoint in a unique NURBS curve. The first step requires to elevate the curves to a common degree.

2.5.4 Degree Reduction

As for knot refinement, also degree reduction is not always possible. However, due to floating point round off error, $\mathbf{C}(u)$ and $\bar{\mathbf{C}}(u)$ are not expected to coincide precisely even in an ideal situation, thus a maximum allowable error (TOL) is defined, and a curve or surface is declared to be degree reducible if

$$\max |\mathbf{C}(u) - \bar{\mathbf{C}}(u)| \leq \text{TOL} \quad (2.57)$$

2.5.5 Other Advanced Geometric Algorithms

Many other fundamental algorithms are involved in NURBS manipulation. Algorithms like *point inversion* and *projection*, *surface tangent vector inversion*, *curves and surfaces transformations and projections*, and others, are of main importance. For a complete treatise refer to chapter 6 of [8].

Another class of algorithms used in ASD is the surface intersection algorithms. These are typically used in calculating intersections between fillets and bodies, or tfillet and wings (features encountered in chapter 1). For an exhaustive analysis see [3].

3

Free Form Surface Design

3.1 Introduction

Traditionally NURBS shape modification and manipulation was achieved with in an indirect way, that is, controlling the degree of freedoms (control points for B-splines, control points and weights for NURBS) till the desired shape is achieved. This technique is very easy but it has several drawbacks [10]. One of the them is that the control points or weights are not directly related to the modified shape of curves or surfaces. Therefore, interactive design using this scheme is often cumbersome. Sometimes a large number of control points must be manipulated in order to modify even a small piece of a curve segment. It is also not clear which degree of freedom should be manipulated and how it should be manipulated. On the other hand, the direct manipulation method provides the designer a higher level interface and shape design is more intuitive. In this scheme, arbitrary constraints on a curve or a surface can be set. For example, a point on a curve can be selected and moved to a desired positions. The new DOFs, which satisfy the specified constraints, are automatically computed.

It's plain that ASD relies on a direct approach, since the control points are arranged in such way to fit the prescribed data points and derivatives vectors in a direct mathematical way, and not with a trial and error approach.

3.2 Fitting

Fitting consists in the construction of curves and surfaces which fit a rather arbitrary set of geometric data, such as points and derivative vectors. There can be distinguished two types of situations, *interpolation* and *approximation*.

The intent of interpolation is to construct a form (NURBS in this instance) which satisfies the given data precisely. For example the curve passes through the given points and assume the given derivatives at the prescribed points.

In approximation, the forms couldn't satisfy the given data precisely, but only approximately. Usually the maximum bound on the deviation from the given data is specified (fig.3.1). From a practical point of view, approximation is well suited

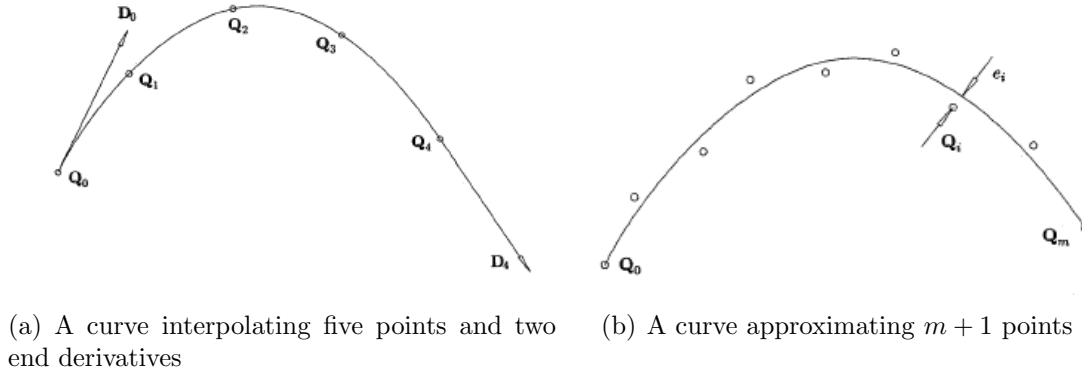


Figure 3.1: Difference between interpolation and approximation (from [8]).

in all the situations where the fitting data are affected from errors or noise. Classical examples are scanning and acquisition devices which introduce measurement noise, or surface to surface intersection algorithms (marching methods) which leads to computational noise. In such of situation, it is important to capture the *shape* of the data, not to wiggle its way through every point.

On the opposite scenario, when the NURBS must satisfy exactly given data, interpolation methods should be used. The interpolation algorithms are simpler and faster than the approximation algorithms.

In ASD only interpolation techniques are implemented, assuming that all the input data are unaffected from errors. However, it should be noticed that surface or

curve intersection algorithms are used in the modeling process; this data are possibly affected from numerical noise, and interpolating them could lead to unpleasant results.

Literally hundreds of paper have been written on fitting topic; many of the reported techniques are heuristic. The problem arises because fitting process never determines a unique solution. In fact there are infinite NURBS curves (surfaces) that interpolate or approximate a set of data. As a result the interpolating curve (surface) obtained could be far from the expected shape.

Very little has been published on setting the weights in the fitting process. Most often, all weights are simply set to 1, so the interpolating NURBS degenerate to interpolating B-spline. In ASD only B-spline interpolation problems are handled.

Most fitting methods are further classified into global and local.

3.2.1 Global Interpolation

With a global algorithm a set of equations or an optimization problem is set up and solved. If the given data consist only of points and derivatives, if the degree of the NURBS, the knots and weights have been somehow preselected, and the only unknown are the control points, then the system is linear, hence easy to solve. If given data consist on curvature and knots or weights are system unknowns, then the resulting problem would be nonlinear.

Since desired level of continuity at each point is easy to set, due to NURBS characteristics (summarized in section 2.4), results are usually pleasant looking shapes. One of the drawbacks is the inability to cope with straight segments: if three or more points of the data set are collinear it would be desirable, most of times, an interpolating straight segment (see fig.3.2). On the contrary global interpolation leads to a wavy shape if only positional constraints are specified. Finally, a perturbation of any one input data item can change the shape of the entire curve or surface; however, the magnitude of the change decreases with increasing distance from the affected data item.

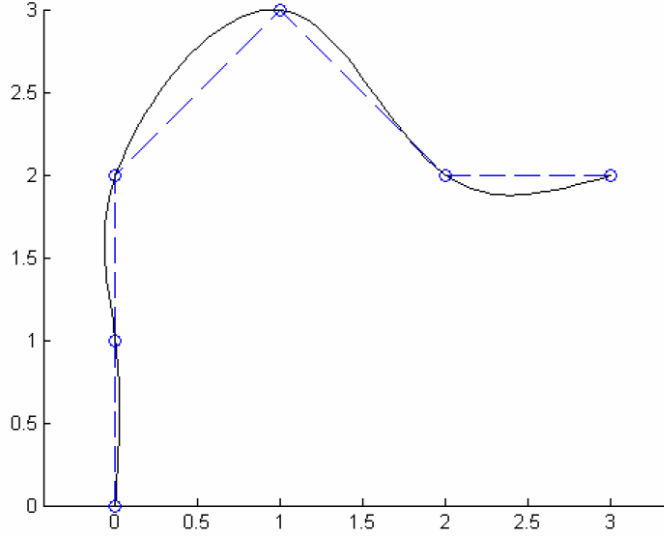


Figure 3.2: Global interpolation with three collinear points

Global curve interpolation

Suppose a given set of points $\{\mathbf{Q}_k\}$, $k = 0, \dots, n$ would be interpolated with a p th degree B-spline. First step is to assign a parameter value, \bar{u}_k , to each \mathbf{Q}_k , and to select an appropriate knot vector $U = \{u_0, \dots, u_m\}$. The corresponding $(n + 1) \times (n + 1)$ linear system is

$$\mathbf{Q}_k = \mathbf{C}(\bar{u}_k) = \sum_{i=0}^n N_{i,p}(\bar{u}_k) \mathbf{P}_i \quad (3.1)$$

Here the $n + 1$ control points \mathbf{P}_i are the unknowns. In matrix form:

$$[Q] = [N] \cdot [P] \quad (3.2)$$

If also derivatives appear as input data, then both the control point (unknowns) and knot vector number should be raised. Expressing the derivatives in terms of basis function derivatives and control points (see eq.(2.23)) the linear system is easily set.

Usually the fitting data consist in interpolation points, first and second order derivatives. Greater order derivatives are seldom used. That's related with the geometric relationship between derivatives and curve shape: first order derivative is locally tangent to the curve and his magnitude depends on the parameterization,

second order derivative is related with the curvature (see section 3.4 for more details on connection between derivatives and geometry). Thus,

$$\begin{aligned}\mathbf{Q}_k &= \mathbf{C}(\bar{u}_k) = \sum_{i=0}^n N_{i,p}(\bar{u}_k) \mathbf{P}_i \\ \mathbf{Q}'_l &= \mathbf{C}'(\bar{u}_l) = \sum_{i=0}^n N'_{i,p}(\bar{u}_l) \mathbf{P}_i \\ \mathbf{Q}''_m &= \mathbf{C}''(\bar{u}_m) = \sum_{i=0}^n N''_{i,p}(\bar{u}_m) \mathbf{P}_i\end{aligned}$$

and the linear system is easily obtained. Worth a note that the selection of the knot vector and the parameter values should be carefully undertaken, in order to avoid a singular matrix. See section 3.3 for explanation about how to set these parameters. In fig.3.3(a) is shown a curve interpolating 5 points. If an horizontal derivative at

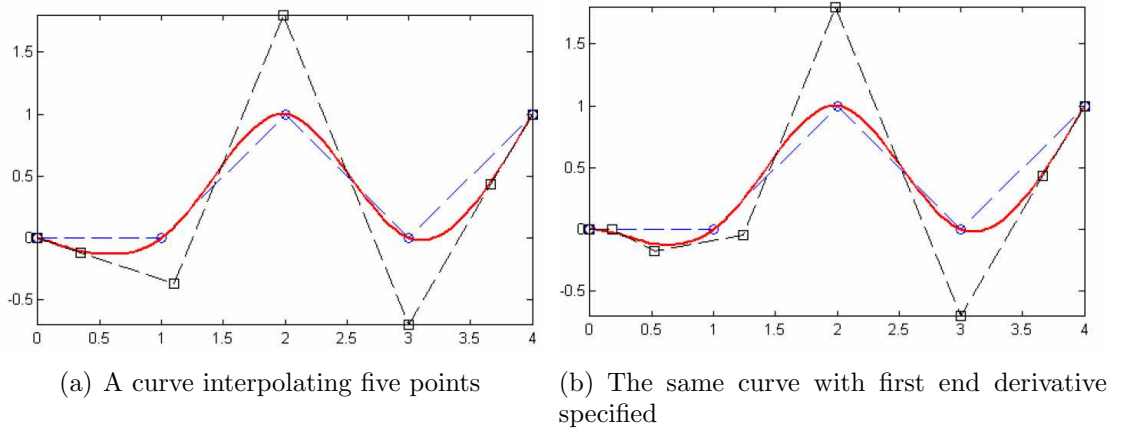


Figure 3.3: Curve interpolation, and curve interpolation with derivative specified.

the first end is added as fitting data, then the curve is modified as shown in the fig.3.3(b). Note the added control point (which implies also one more knot).

Global surface interpolation

Let $\{\mathbf{Q}_{k,l}\}$, with $k = 0, \dots, n$ and $l = 0, \dots, m$ be a set of $(n+1) \times (m+1)$ data points. The interpolating $(p \times q)$ th degree B-spline surface would be of the form

$$\mathbf{Q}_{k,l} = \mathbf{S}(\bar{u}_k, \bar{v}_l) = \sum_{i=0}^n \sum_{j=0}^m N_{i,p}(\bar{u}_k) N_{j,q}(\bar{v}_l) \mathbf{P}_{i,j} \quad (3.3)$$

Suppose the parameters (\bar{u}_k, \bar{v}_l) and the knot vector U and V are computed in a reasonable way. Then eq.(3.3) represents a set of $(n+1) \times (m+1)$ linear equations in the unknowns $\mathbf{P}_{i,j}$. However, since $\mathbf{S}(u,v)$ is a tensor product surface, the $\mathbf{P}_{i,j}$ can be obtained more simply and efficiently as a sequence of curve interpolations. For fixed l eq.(3.3) can be rearranged as

$$\mathbf{Q}_{k,l} = \sum_{i=0}^n N_{i,p}(\bar{u}_k) \left(\sum_{j=0}^m N_{j,q}(\bar{v}_l) \mathbf{P}_{i,j} \right) = \sum_{i=0}^n N_{i,p}(\bar{u}_k) \mathbf{R}_{i,l} \quad (3.4)$$

where

$$\mathbf{R}_{i,l} = \sum_{j=0}^m N_{j,q}(\bar{v}_l) \mathbf{P}_{i,j} \quad (3.5)$$

Eq.(3.4) is just a curve interpolation through the points $\mathbf{Q}_{k,l}$ $k = 0, \dots, n$. The $\mathbf{R}_{i,l}$ are the control points of the isoparametric curve on $\mathbf{S}(u,v)$ at fixed \bar{v}_l . Then, fixing i and letting l vary, eq.(3.5) represent curve interpolation through the points $\mathbf{R}_{i,0}, \dots, \mathbf{R}_{i,m}$, with $\mathbf{P}_{i,0}, \dots, \mathbf{P}_{i,m}$ as the computed control points. Thus, the algorithm to obtain all the $\mathbf{P}_{i,j}$ consist in the following steps:

- $(m+1)$ curve interpolations through the points $\mathbf{Q}_{i,0}, \dots, \mathbf{Q}_{i,l}$ (for $l = 0, \dots, m$) using U and the parameters \bar{u}_k ; this yields the $\mathbf{R}_{i,l}$;
- $(n+1)$ curve interpolations through $\mathbf{R}_{i,0}, \dots, \mathbf{R}_{i,m}$ (for $i = 0, \dots, n$) using V and the parameters \bar{v}_l ; this yields the $\mathbf{P}_{i,j}$.

Obviously the algorithm is symmetric, the same holds inverting the interpolation sequence. An example of surface interpolation is depicted in fig.3.4. Derivative constraints can also be incorporated, conceptually with the same approach as for the

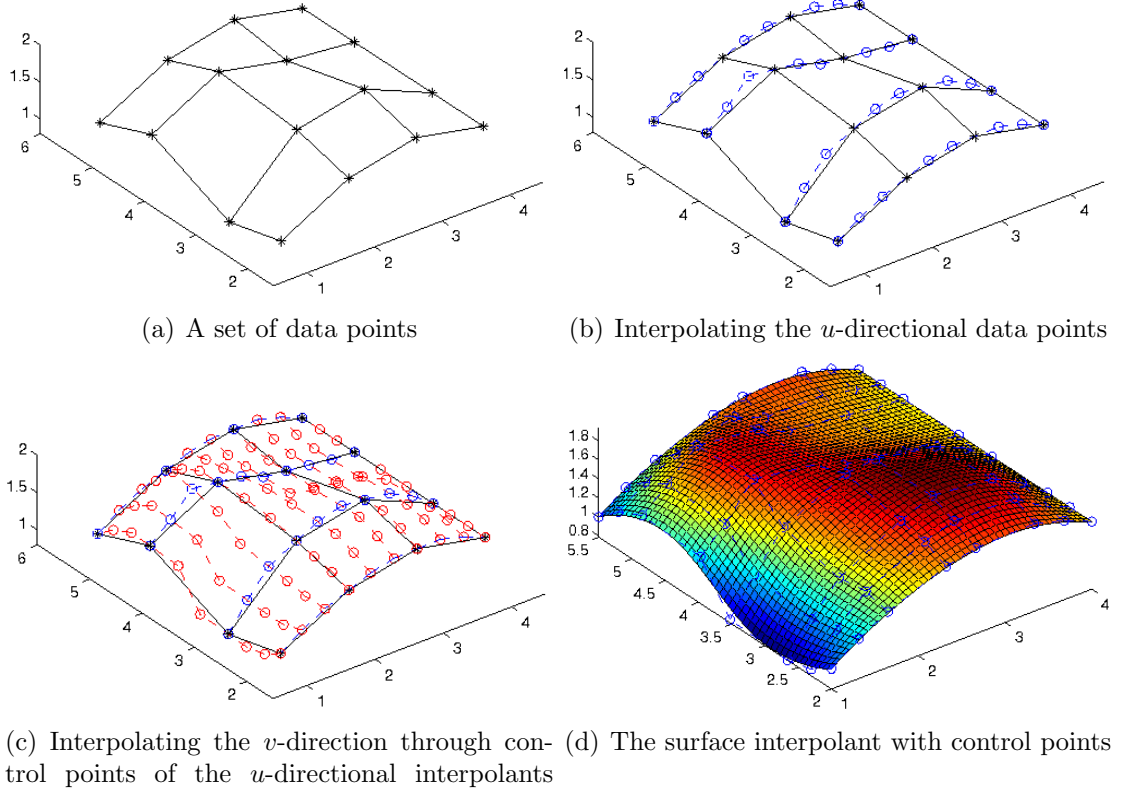


Figure 3.4: Surface interpolations through subsequent curve interpolations.

curves. However difficulties arises if the number of data constraints is not the same in every row or column; with clever use of curve knot insertion and surface knot removal the algorithm can be extended to general situations [8].

3.2.2 Local Interpolation

The local scheme interpolation is usually more geometric in nature. The typical construction proceeds piecewise, between every data constraint, leading to local properties: a perturbation in a data item affects the curve or surface only locally. These algorithms are usually computationally less expensive than global methods. Another benefit is the capability of dealing with cusps, straight lines segments and other local anomalies in an efficient way. On the other hand, achieving desired level of

continuity at segment boundaries is cumbersome, and local methods often result in multiple interior knots.

Local curve interpolation

Let $\{Q_k\}$, $k = 0, \dots, n$ be a set of point to be interpolated. Local curve interpolation consist in constructing n polynomial or rational curve segments, $C_i(u)$, $i = 0, \dots, n-1$ such that the Q_i and Q_{i+1} are the endpoints of $C_i(u)$. Neighboring segments are joined with prescribed level of continuity, and the construction proceeds segment-wise. Any equation which arises are locally to only a few neighboring segments.

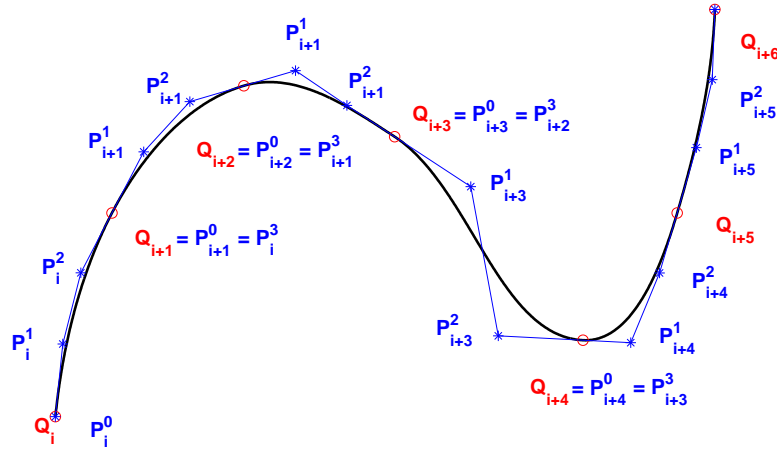


Figure 3.5: Local interpolation: curve segments endpoints are coincident with the data points Q (in red). The internal control points of every curve segment should somehow be computed.

In the framework of NURBS the segments are polynomial or rational Bézier curves, then corresponding NURBS is obtained selecting a suitable knot vector. Notice that to obtain the Bézier segments, $C_i(u)$, the inner Bézier control points have to be computed. If the fitting data comprises also derivatives, they should be used to compute the inner control points. If don't, or the unknown are more than the constraint equations, then some geometric consideration (as have been done in order to solve

the problem. Usually, local algorithms rely on local tangent estimator algorithms [11; 12] to overcome these difficulties.

An example of a local curve interpolation algorithm, on which relies ASD, is provided in section 4.1.1.

Local surface interpolation

Generalizing to the bivariate case, a surface is obtained through definition of $n \cdot m$ Bézier patches. It holds what stated for the univariate situation, even if the added geometric constraints necessary for closing the problem are a little more tricky. An example of a local surface interpolation algorithm, on which relies ASD, is provided in section 4.1.2.

3.2.3 Transfinite interpolation

In transfinite interpolation a curve network is interpolated through a surface. This method has a different flavor than the *set of points* interpolation, and is due mainly to Coon and Gordon with respectively the Coons patches and Gordon surfaces. Since this methods are not directly implemented in ASD, they won't be described here (see [7; 8] for a treatise). However, the *lofting* capability, which is a particular form of transfinite interpolation, has been added to the geometric modeler.

Lofting (or Skinning)

Let $\{\mathbf{C}_k(u)\}$, $k = 0, \dots, K$ be a set of curves, called *sectional curves*. Skinning is a process of blending these section curves together to form a surface. The blend direction is usually called the *longitudinal direction*.

Lofting dates back many decades, before computers, and the earliest computerized system to incorporate lofting was CONSURF. That system was based on rational cubic curves.

Based on B-splines, the skinning is defined as follows. Let

$$\mathbf{C}_k^w(u) = \sum_{i=0}^n N_{i,p}(u) \mathbf{P}_{i,k}^w \quad k = 0, \dots, K \quad (3.6)$$

be the rational or nonrational section curves, all defined on the same knot vector, U , and with the same degree p (if necessary the curves can be brought to a common p and U by means of the algorithms of section 2.5). Then, for the v direction a degree q is chosen, and parameters $\{\bar{v}_k\}$, $k = 0, \dots, K$ and a knot vector V are computed. These are then used to do $n+1$ interpolations across the control points of the section curves, yielding the control points $\mathbf{Q}_{i,j}^w$ of the skinned surface. Therefore, $\mathbf{Q}_{i,j}^w$ is the j th control point of the interpolating curve through $\mathbf{P}_{i,0}^w, \dots, \mathbf{P}_{i,K}^w$. The process of skinning is reported in fig.3.6.

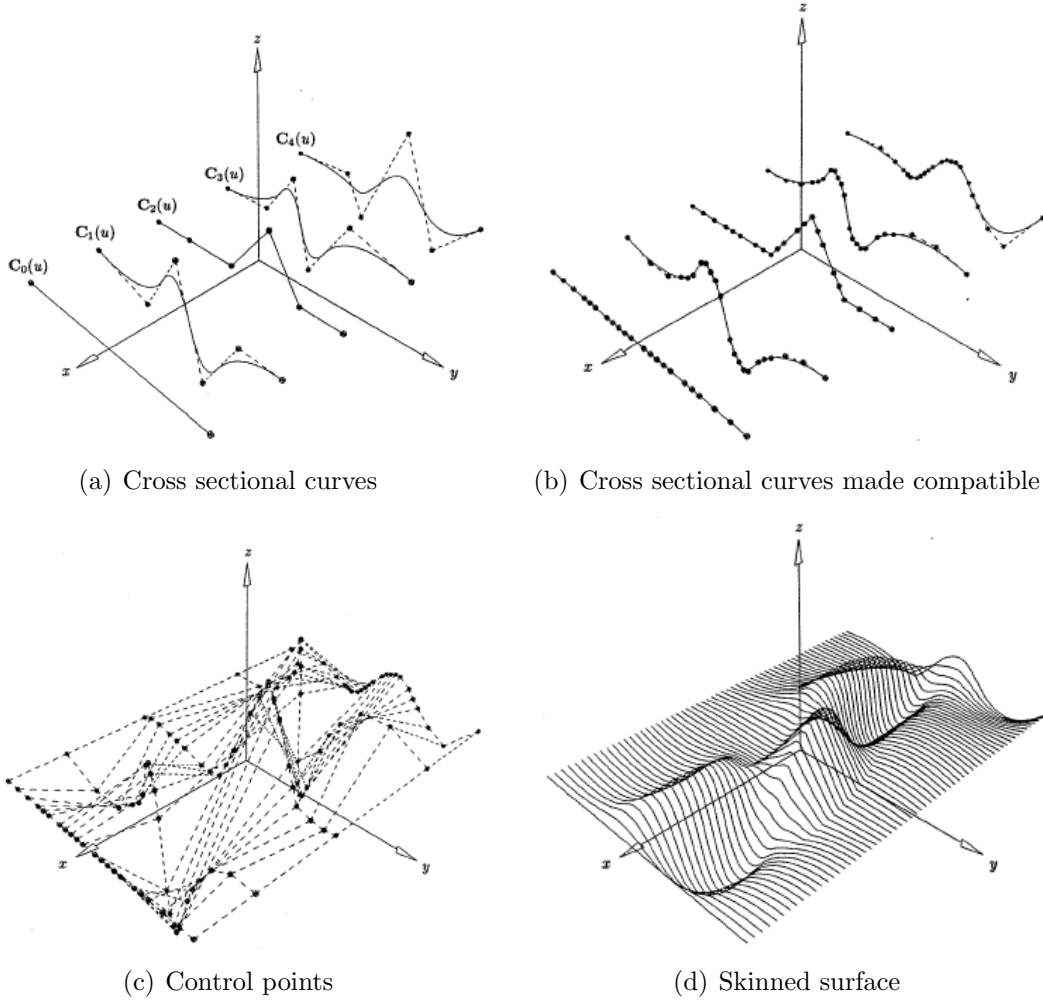


Figure 3.6: The process of surface skinning [8].

The process of transforming the cross sectional curves to a compatible form raises the number of control points in the u direction, and thus the overall control point number. This is the price to pay for flexibility and quality.

In fact, during B-spline(NURBS) surface interpolation the determination of one knot vector for column points, and one for the row points. This can lead to a unpleasant surface, regardless the choice of the knot vector and knot parameters. Taking advantage of the surface interpolation as subsequent curve interpolations, all curves should fit to a unique knot vector, which even if reasonable in mean terms, could be inappropriate for the interpolation of some particular curves. That won't happen to lofted surfaces, since the curves are already defined.

3.3 Parameterization

Till now the set of knot parameters and the knot vector has been assumed as known. In this section it will be shown how properly determine both of them. This fixing is a keypoint, since the quality of any interpolant curve depends strongly on the selection on the parameters \bar{u}_k , and knot vector choice. There are several parameterization techniques in the literature, three of the most relevant are reported in the following.

3.3.1 Uniform Parameterization

Known also as equally space, this kind of parameterization states that:

$$\Delta \bar{u}_i = \bar{u}_{i+1} - \bar{u}_i = \text{constant} \quad (3.7)$$

It is the most primitive one, and performs poorly, especially when the data points are scattered in a very unevenly fashion. The reason for the overall poor performance can be blamed on the fact that it *ignores* the geometry of the data points.

As an heuristic explanation, the parameter u can be interpreted as time, and $\mathbf{C}(u)$ as the trajectory. If the distance between two successive points is very large, the speed would be high in order to cover the segment in the fixed time. If the next two data points are close to each other, there will be an overshooting because the

speed can not be changed abruptly (it is assumed at least a C^1 trajectory, thus a continuous velocity).

3.3.2 Chord Length Parameterization

This method, known also as chordal parameterization, selects the parametric values according to the distances

$$\| \mathbf{P}_{i+1} - \mathbf{P}_i \| \propto \bar{u}_{i+1} - \bar{u}_i \quad (3.8)$$

or, equivalently

$$\frac{\Delta \bar{u}_i}{\Delta \bar{u}_{i+1}} = \frac{\| \mathbf{P}_{i+1} - \mathbf{P}_i \|}{\| \mathbf{P}_{i+2} - \mathbf{P}_{i+1} \|} \quad (3.9)$$

The idea here is that the distance approximate the arc distances between two subsequent points.

3.3.3 Centripetal Parameterization

$$\| \mathbf{P}_{i+1} - \mathbf{P}_i \|^{\frac{1}{2}} \propto \bar{u}_{i+1} - \bar{u}_i \quad (3.10)$$

or, equivalently

$$\frac{\Delta \bar{u}_i}{\Delta \bar{u}_{i+1}} = \left(\frac{\| \mathbf{P}_{i+1} - \mathbf{P}_i \|}{\| \mathbf{P}_{i+2} - \mathbf{P}_{i+1} \|} \right)^{\frac{1}{2}} \quad (3.11)$$

3.3.4 Performance of the Different Parameterizations

Not reported above, even if one of the most famous parameterization, is the Foley parameterization; anyway in literature a multitude of method for choosing parameters exists, each with his advantages and disadvantages. ([7; 13]).

In fig.(3.7) are depicted results with different parameterizations. The knot vector is distributed uniformly.

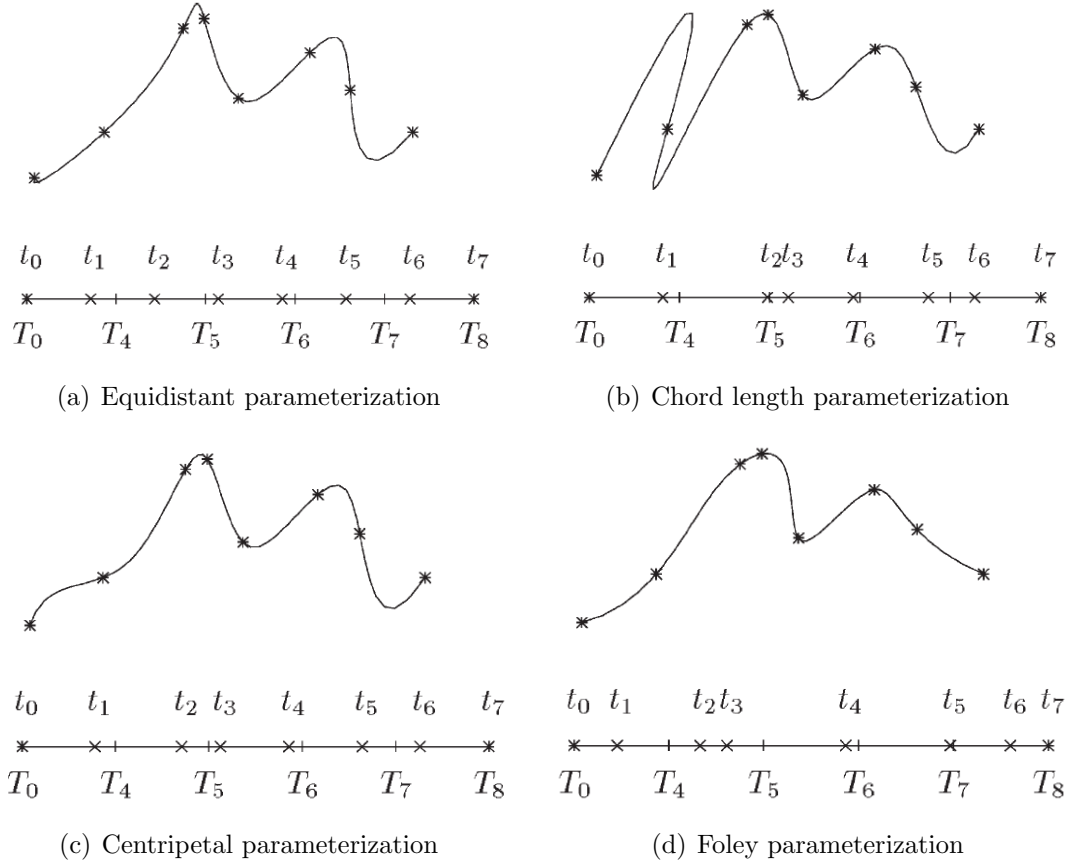


Figure 3.7: Various parameterizations with uniform knot vector. \times indicates the parameter values, $+$ the knot values.

3.3.5 Knot Vector Selection

The knot vector selection is as important as the selection of the parameter values; experimentations indicate that the knot vector selection similar to parameter distribution provide the more natural looking curves. Results of similar parameterization and knot vector is shown in fig.3.8

A comparison between figures 3.7 and 3.8 shows the improvements in shape, indicating that knot vector selection is no less important than parameterization.

Further, such a choice has the advantage of speed up the computation of the matrix $[N]$ in eq.(3.2) since there are $p - 1$ nonzero terms $N_{i,p}(\bar{u}_k)$ in most of its rows instead of p nonzero terms, as in the conventional knot selection.

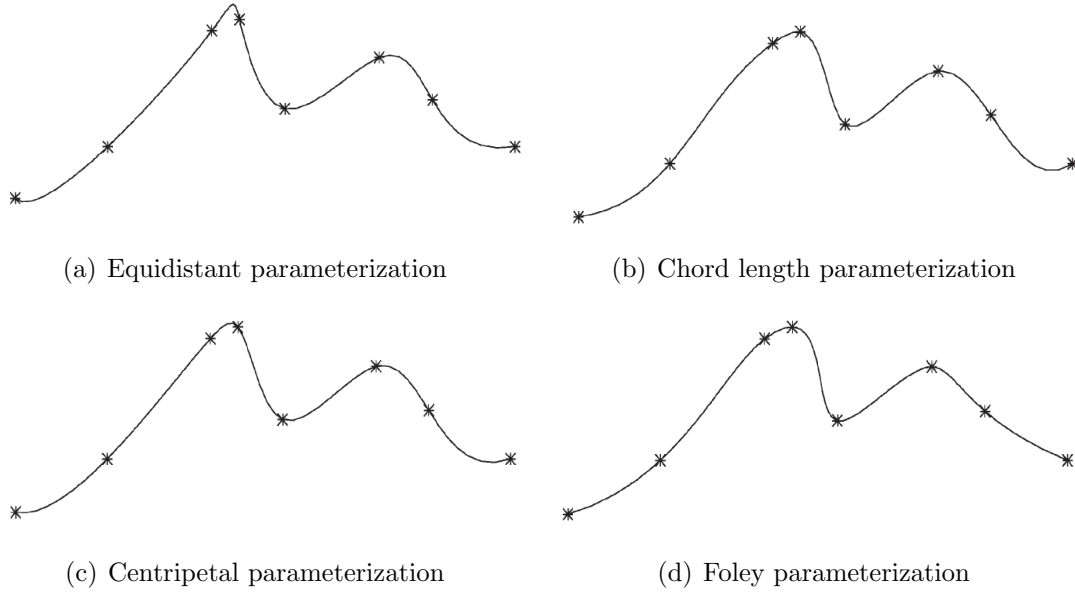


Figure 3.8: The effect of choosing knot values similar to parameter values.

3.3.6 Parameterization and Knot Vector Selection for Surfaces

For a tensor product surface the problems arises from topology. In fact, a unique knot vector and parameterization for each of the two directions could be defined. Thus, in order to satisfy in mean terms the geometric distribution of the points, the usually choose is to perform an average of all the row/column related parameters.

$$\bar{u}_k = \frac{1}{m+1} \sum_{l=0}^m \bar{u}_k^l \quad k = 0, \dots, n \quad (3.12)$$

where the notation is the same of eq.(3.3), and for each fixed l , \bar{u}_k^l is computed with one of the aforementioned methods. Clearly this construction can't be so flexible to model in a nice looking way all the rows or columns of data points. In fact, is nowhere guaranteed that \bar{u}_k and \bar{u}_k^l are even very different, thus the knot vector would be inappropriate for description of some rows/columns of points, leading to far from optimal shapes. This states especially for very unevenly distributed points.

3.4 Notion of Continuity

For parametric forms the notion of continuity assumes different meaning, depending on the sphere of analysis. Before going further on the topic, is important to define the arc-length parametrization. Given a curve $\mathbf{C}(u)$ with $u \in [u_a, u_b]$, the arc-length s as a function of u , is defined as

$$s(u) = \int_{u_a}^u \|\mathbf{C}'(u)\| \, du \quad (3.13)$$

which is the length of \mathbf{C} from $\mathbf{C}(u_a)$ to $\mathbf{C}(u_b)$. The curve $\mathbf{C}(s) = \mathbf{C}(u^{-1}(s))$ is the corresponding arc-length parameterized curve. Although the arc-length is conceptually important, it is used primarily for theoretical purposes.

A detailed survey on continuity is reported in [14].

3.4.1 Continuity of Curves

Let $r(u)$ and $s(v)$ be two segments of two parametric curves, joining at their endpoints so that $\mathbf{r}(u_0) = \mathbf{s}(v_0)$.

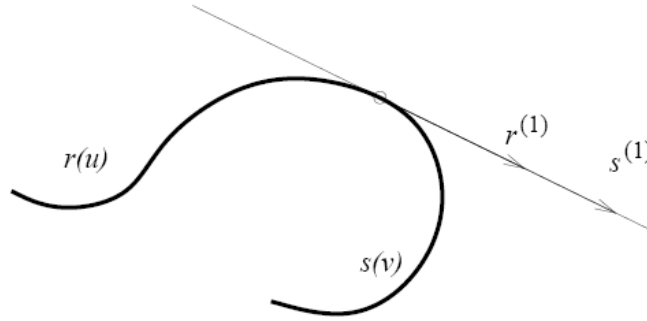


Figure 3.9: Two curve segments joining with derivative of same direction but different magnitude.

Parametric continuity

Parametric continuity is the classical notion of continuity in analysis: if the n th order derivatives of a function exist and are continuous, then the function is n th

order parametric continuous. In this context, the

- two curves are n th order parametric continuous at u_0, v_0 if and only if $\mathbf{r}^{(n)}(u_0) = \mathbf{s}^{(n)}(v_0)$.

Anyway, two curve segment need not have the same derivative vector at joining point (C^1) in order to have the same tangent line. In an similar way, they don't need to be C^2 to be curvature continuous. The crucial observation is that derivatives depend also on parametrization of curves, while tangent and curvature are intrinsic properties of the geometric shape, so they are not dependent from the parameterization.

Geometric continuity

A more intrinsic notion of continuity should avoid to depend from parameterization. This leads to a definition of continuity called geometric, or visual continuity.

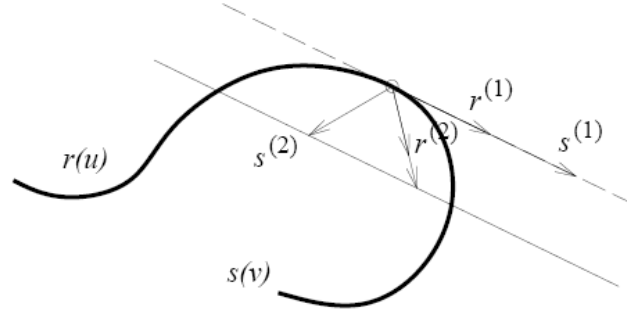


Figure 3.10: Relation between derivative vectors for geometric continuity.

- Curves $\mathbf{r}(u)$ and $\mathbf{s}(v)$ are n th order geometric continuous at u_0, v_0 , if and only if there exists a reparameterization $u = u(\tilde{u})$ such that $\tilde{\mathbf{r}}(\tilde{u}) = \mathbf{r}(u(\tilde{u}))$ and $\mathbf{s}(v)$ are C^n at $\mathbf{s}(v_0)$.

Geometric continuity is denoted by G^n ; the term geometric continuity was first used by Barsky, the term visual continuity by Farin.

The chain and product rule of differentiation show that $\tilde{\mathbf{r}}^{(k)}(\tilde{u})$ can be written in terms of

$$\frac{d^i \mathbf{r}(u)}{du^i} \quad \text{and} \quad \frac{du(\tilde{u})}{d\tilde{u}^i} \quad \text{with} \quad i = 1, \dots, k$$

For example:

$$\frac{d^2 \tilde{\mathbf{r}}}{d\tilde{u}^2} = \frac{d^2 \mathbf{r}(u)}{du^2} \left(\frac{du}{d\tilde{u}} \right)^2 + \frac{d\mathbf{r}}{du} \frac{d^2 u}{d\tilde{u}^2}$$

Letting β_i indicate $u^{(i)}(\tilde{u}_0)$, then the following equations hold:

$$\mathbf{s}^{(0)}(v_0) = \mathbf{r}^{(0)}(u_0) \quad (3.14a)$$

$$\mathbf{s}^{(1)}(v_0) = \beta_1 \mathbf{r}^{(1)}(u_0) \quad (3.14b)$$

$$\mathbf{s}^{(2)}(v_0) = \beta_1^2 \mathbf{r}^{(2)}(u_0) + \beta_2 \mathbf{r}^{(1)}(u_0) \quad (3.14c)$$

$$\mathbf{s}^{(3)}(v_0) = \beta_1^3 \mathbf{r}^{(3)}(u_0) + 3\beta_1\beta_2 \mathbf{r}^{(2)}(u_0) + \beta_3 \mathbf{r}^{(1)}(u_0) \quad (3.14d)$$

...

Equation (3.14a) amounts to positional continuity, eq.(3.14b) means that the derivative vectors differ only a scalar factor, eq.(3.14c) prescribes a dependency as depicted in fig.3.10.

The parameters β_i are called *shape handles* because they can be used to model shape of the curve. In particular, β_1 is called *bias* and β_2 *tension*, due to their specific shape changing effect. One particular spline that satisfies the β -constraints is the β -spline. More on such topics can be found in [15; 16; 17].

It can be shown [16] that another equivalent formulation can be provided by the following.

- Curves $\mathbf{r}(u)$ and $\mathbf{s}(v)$ are n th order geometric continuous at u_0 and v_0 , if and only if the corresponding arc length parameterized curves $\tilde{\mathbf{r}}(t)$ and $\tilde{\mathbf{s}}(w)$ are C^n at $\tilde{s}(w(v_0))$.

Differential geometry approach

From a purely geometrical point of view, the direction of the tangent line is determined by the tangent vector, the normalized derivative vector which has unit length. Two curves need not have the same derivative vector in order to have the same tangent vector. There is a particular parameterization that gives a unit length derivative vector at every point, so that the derivative and tangent vector are equal: the arc-length parameterization.

Let $\mathbf{s}(w)$ be an arc-length parameterized curve. The tangent vector \mathbf{t} is then

$$\mathbf{t}_1(w) = \mathbf{s}^{(1)}(w) \quad \|\mathbf{t}_1\| = 1 \quad (3.15)$$

The normal curvature vector is defined as

$$\mathbf{t}_2(w) = \frac{\mathbf{t}_1^{(1)}(w)}{\kappa_1(w)} \quad (3.16)$$

where $\kappa_1(w)$, called *curvature*, is a scalar such that $\|\mathbf{t}_2(w)\| = 1$. The binormal curvature vector, normal to \mathbf{t}_1 and \mathbf{t}_2 , is defined as:

$$\mathbf{t}_3(w) = \frac{\mathbf{t}_2^{(1)}(w) + \kappa_1(w)\mathbf{t}_1(w)}{\kappa_2(w)}$$

where $\kappa_2(w)$, called *torsion* is a scalar such that $\|\mathbf{t}_3(w)\| = 1$. Planar curves have zero torsion. Curvature and the opposite of torsion have an intuitive geometrical meaning, being the angular velocities of $\mathbf{t}_1(w)$ and $\mathbf{t}_2(w)$ respectively. The plane spanned by the tangent and the normal curvature vector is called the osculating plane.

The notions of tangent, normal curvature and binormal curvature can be generalized to the so called generalized curvatures:

$$\begin{aligned} \mathbf{t}_1(w) &= \mathbf{s}^{(1)}(w) \\ \kappa_0 &= 0 \\ \mathbf{t}_{i+1}(w) &= \frac{\mathbf{t}_i^{(1)}(w) + \kappa_{i-1}(w)\mathbf{t}_{i-1}(w)}{\kappa_i(w)} \end{aligned} \quad (3.17)$$

where $\kappa_i(w)$ is such that $\|\mathbf{t}_{i+1}(w)\| = 1$.

In \mathbb{R}^d , the *Frenet frame* is defined as the first d curvature vectors $(\mathbf{t}_1(w), \dots, \mathbf{t}_d(w))$. Thus, the following definition of *Frenet frame continuity*:

- Two curves $r(u)$ and $s(v)$ are n th order ($n > 0$) Frenet frame continuous (F^n) at u_0 and v_0 if and only if the first n curvature vectors and scalar curvatures coincide.

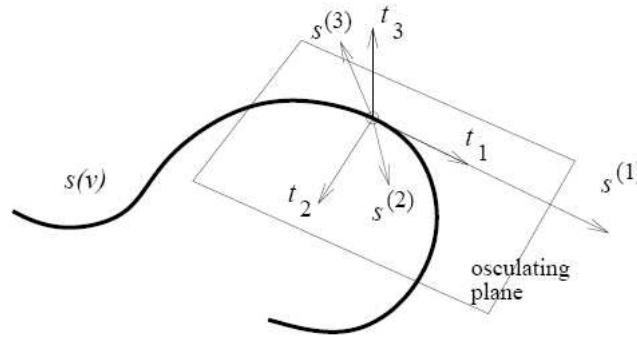


Figure 3.11: Frenet frame.

For $n = 1$ and $n = 2$, F^n and G^n continuity agree. Thus, G^1 condition requires the same tangent vectors, G^2 requires also the same normal curvature vectors and the same curvature.

3.4.2 Continuity of Surfaces

The generalization of the continuity notion at surfaces is a little more tricky, thus only a brief discussion is submitted. For more details refer to [7; 14; 17; 18]. The practical situation of interest is the continuity of two parametric surfaces $\mathbf{r}(t,u)$ and $\mathbf{s}(v,w)$, in particular along a common curve or edge. To avoid potential problems

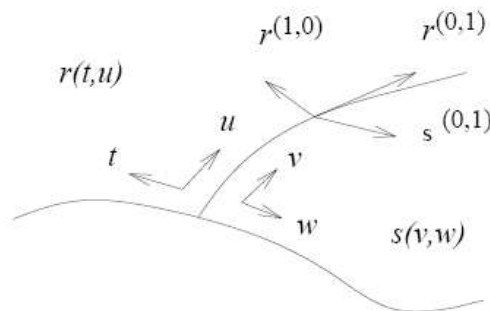


Figure 3.12: Two surfaces joining at the common edge, and their first partial derivatives.

with the parameterization assume a *regular* parameterization, that is first derivatives exist and are linearly independent.

Parametric continuity

- Two surfaces $\mathbf{r}(t,u)$ and $\mathbf{s}(v,w)$ are C^n -continuous at (t_0, u_0) and (u_0, w_0) , if and only if $\mathbf{r}^{(i,j)}(t_0, u_0) = \mathbf{s}^{(i,j)}(v_0, w_0)$, $i + j = 0, \dots, n$.

Note that the two surfaces need not have the same first order partial derivatives in order to have the same tangent plane [18]. Moreover, on closed surfaces singularities occur where the derivative of the surface is not defined [14].

Geometric continuity

- Two surfaces $\mathbf{r}(t,u)$ and $\mathbf{s}(v,w)$ are G^n -continuous at (t_0, u_0) and (u_0, w_0) if and only if there exist a reparameterization $t = t(\tilde{t}, \tilde{u})$ and $u = u(\tilde{t}, \tilde{u})$ such that $\tilde{\mathbf{r}}(\tilde{t}, \tilde{u}) = \mathbf{r}(t(\tilde{t}, \tilde{u}), u(\tilde{t}, \tilde{u}))$ and $\mathbf{s}(v,w)$ are C^n -continuous at $\mathbf{r}(t_0, u_0)$ and $\mathbf{s}(u_0, w_0)$.

Applying the bivariate chain and product rule, and following the steps as for the above seen univariate situation, leads to a set of equation analogous to eq.(3.14) [14; 17].

Differential geometry approach

The tangent plane at $\mathbf{s}(v_0, w_0)$ is spanned by the vectors $\mathbf{s}^{(1,0)}(v_0, w_0)$ and $\mathbf{s}^{(0,1)}(v_0, w_0)$ [14; 18]. Equivalently, the tangent plane is normal to the surface normal vector

$$\mathbf{n}(v_0, w_0) = \frac{\mathbf{s}^{(1,0)}(v_0, w_0) \times \mathbf{s}^{(0,1)}(v_0, w_0)}{\|\mathbf{s}^{(1,0)}(v_0, w_0) \times \mathbf{s}^{(0,1)}(v_0, w_0)\|} \quad (3.18)$$

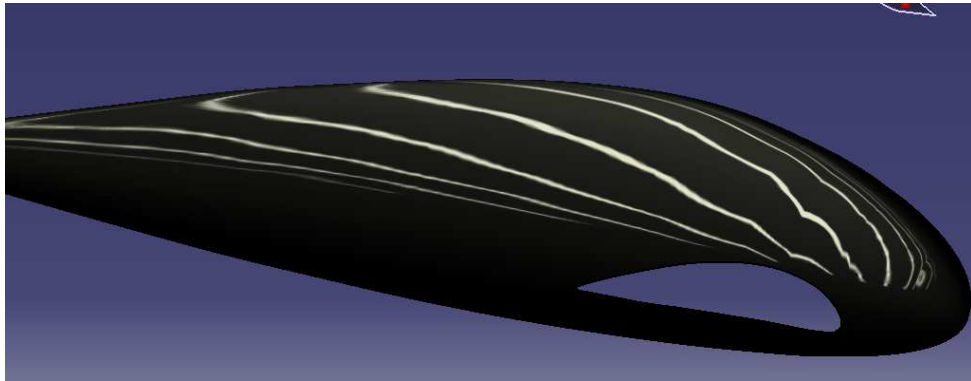
Thus, the tangent planes at $\mathbf{r}(t_0, u_0)$ and $\mathbf{s}(v_0, w_0)$ coincide if and only if $\mathbf{r}^{(1,0)}(t_0, u_0)$, $\mathbf{r}^{(0,1)}(t_0, u_0)$, $\mathbf{s}^{(1,0)}(u_0, v_0)$, $\mathbf{s}^{(0,1)}(u_0, v_0)$ are coplanar, and geometric continuity is achieved at the common point.

Note that for continuity along a common curve, the continuity at every point should be achieved.

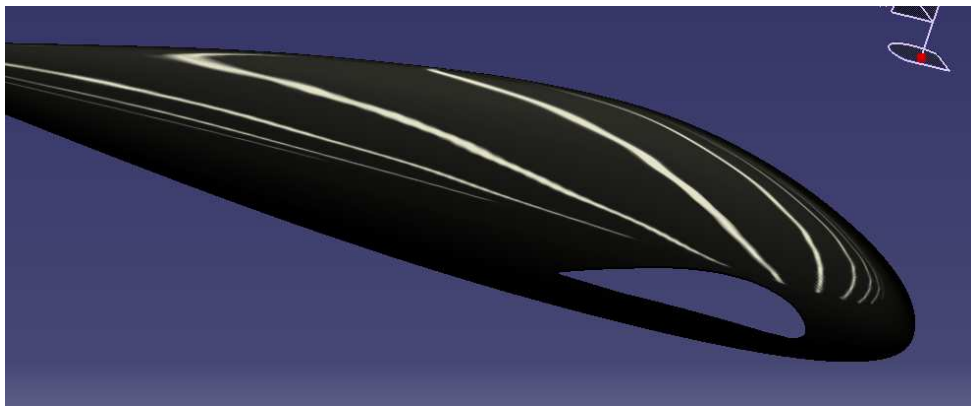
The second order continuity is based on curvature. The definition is a little bit more involving, thus it will be omitted. As before, [7; 14; 17; 18] are valuable sources of detailed information about the topic.

3.5 Visual Aspects of Continuity

Although tangent and tangent plane discontinuities are easy to detect, human eyes is also capable of detecting curvature discontinuities, due to light reflection discontinuities. An example is provided in figure 3.13, where a light reflection analysis on two surfaces is depicted. Note how the first surface, which is not curvature continuous, shows discontinuities in light ray reflection.



(a) G1 surface



(b) G2 surface

Figure 3.13: Differences in light reflection between G^1 and G^2 surfaces.

The care directed to fair shapes by the automotive design, or design process in general, led to surface classification in relation of their geometric properties. Class A surfaces is a term used to describe a set of freeform surfaces of high quality, that is, strictly speaking, the surfaces have curvature and tangency alignment to near perfect

aesthetical reflection quality. However, many technicians interpret class A surfaces to be G^2 (or even G^3), both in their internal domain than at the common edges with other surfaces.

3.6 Implementation of G^2 or C^2 Continuity with Local and Global Algorithms

To meet the requirements mentioned in section 1.5 on surface continuity, at least G^2 continuous algorithms should be implemented. In this section the problems which arise adopting both a global and local approach will be discussed.

As first step just rearrange derivatives equation for a parametric curve in order to split geometric and parametric characteristics. For a generic curve with a generic parameterization it holds (using eq.(3.14, 3.15, 3.16)):

$$\begin{aligned} \mathbf{C}'(u) &= \frac{d\mathbf{C}}{ds} \frac{ds}{du} = \frac{ds}{du} \mathbf{t} = \alpha \mathbf{t} \\ \mathbf{C}''(u) &= \frac{d^2s}{du^2} \mathbf{t} + \alpha \frac{d\mathbf{t}}{du} = \gamma \mathbf{t} + \alpha^2 \frac{d\mathbf{t}}{du} = \gamma \mathbf{t} + \alpha^2 \mathbf{k} \end{aligned} \quad (3.19)$$

where s is the arc-length, \mathbf{t} the tangent versor, \mathbf{k} the curvature vector.

3.6.1 Limitations of the Local Approach for G^2 or C^2 Continuity

In ASD all the interpolation algorithms are local and features (bi)cubic B-spline. However, it can be easily shown that third degree is not enough flexible. Assume a Bézier curve as the i th segment of the interpolant curve. Then, at the edge points it holds:

$$\begin{aligned} \mathbf{C}'_i(0) &= 3(\mathbf{P}_1^i - \mathbf{P}_0^i) & \mathbf{C}''_i(0) &= 6(\mathbf{P}_2^i - 2\mathbf{P}_1^i + \mathbf{P}_0^i) \\ \mathbf{C}'_i(1) &= 3(\mathbf{P}_3^i - \mathbf{P}_2^i) & \mathbf{C}''_i(1) &= 6(\mathbf{P}_3^i - 2\mathbf{P}_2^i + \mathbf{P}_1^i) \end{aligned} \quad (3.20)$$

where, with the aid of fig.3.5, the notation is straightforward. To ensure G^2 continuity, according to what gained in section 3.4.1, the tangent and curvature vector of

the adjacent Bézier segments should coincide at the common point, thus, according to eq.(3.19)

$$\begin{aligned} \mathbf{C}'_i(0) &= \alpha_i \mathbf{t}_i & \mathbf{C}''_i(0) &= \gamma_i \mathbf{t}_i + \alpha_i^2 \mathbf{k}_i \\ \mathbf{C}'_i(1) &= \beta_i \mathbf{t}_{i+1} & \mathbf{C}''_i(1) &= \delta_i \mathbf{t}_{i+1} + \beta_i^2 \mathbf{k}_{i+1} \end{aligned} \quad (3.21)$$

Assume that the tangent and curvature vectors are known through an estimation based on the data point distribution. Combining equations (3.20, 3.21), a system of twelve scalar equations on the unknown $\mathbf{P}_1^i, \mathbf{P}_2^i, \alpha_i, \beta_i, \gamma_i, \delta_i$ is finally obtained. The degree of freedom are not enough, and the system is generally not solvable.

It should be noted that, without estimating the tangent and curvature vector at the data points, the continuity conditions are imposed only at the point in common with the precedent or the subsequent Bézier segment, depending from the direction of building. In this way, there are two degree of freedom (the *bias* and *tension* of section 3.4.1), for each segment.

However, is not possible to force straight segments, not even impose a particular tangent or curvature vector at any point. The reduced flexibility is not satisfactory.

With higher degree curves, the problem of flexibility is overcome. But, two classes of problems arises:

- no universal method to select the degree of freedom of each segment in a reasonable and well performing way exists;
- unlike tangent, no curvature estimator performs well universally.

An example of how to avoid the first class of shortcoming is represented by a fifth-degree piecewise Bézier, with C^2 imposition at the edge points. Anyway, to complete the building a first and second order derivative estimation should be done for every point.

The generalization to surfaces adds more difficulties. At the end, the local approach is distinctly not well suited for handling G^2 continuity.

3.6.2 Limitations of the Global Approach for C^2 continuity

With a global approach is more practical to work with C^2 continuity, since the continuity requirement is immediately controlled by the degree of the B-spline and the

number of multiple knots (see section 2.3.3). Most of the problems are related with the extra flexibility needed for straight segment imposition and derivative constraints. For a curve, knot insertion will be useful to account for this extra constraints, however for tensor product surfaces the process will be more tricky due to the common knot vector.

Should also pointed out, that with a global approach the constraints on tangent and curvature vector are more easy formulated taking into account the parameterization. In fact,

$$\mathbf{C}'(\bar{u}_k) = \sum_{i=0}^n N'_{i,p}(\bar{u}_k) \mathbf{P}_i = \alpha_k \mathbf{t}_k \quad (3.22)$$

with α_k fixed in a proper way, is simpler than

$$\left(\sum_{i=0}^n N'_{i,p}(\bar{u}_k) \mathbf{P}_i \right) \times \mathbf{t}_k = 0 \quad (3.23)$$

because, in the resolution of the final linear system $[Q] = [N] \cdot [P]$, the latter couples the three spatial components, returning a more complex and time consuming system to solve (see [19; 20; 21] for a deeper discussion about constraints).

For curvature constraints:

$$\mathbf{C}''(\bar{u}_k) = \sum_{i=0}^n N'_{i,p}(\bar{u}_k) \mathbf{P}_i = \gamma_k \mathbf{t}_k + \alpha^2 \mathbf{k}_k \quad (3.24)$$

with α_k and γ_k fixed in a proper way, leads to a linear system, where imposing just the \mathbf{k}_k and \mathbf{t}_k leads to a non linear constraint.

The method chosen to fix the parameter dependent terms can not behave properly in all the situations. This is even more problematic when working with surfaces, due to the tensor product surface limitations.

What needed is then a frame capable of:

- handling all the applicable constraints, avoiding the involvement of sequences of knot insertion;
- setting the unconstrained degree of freedom, i.e. control points, when these are

more than the constraints; if possible these degree of freedom should be handled in order to override as much as possible the singular behavior it may arise from presetting the knot vector, the parameterization and the parameterization dependent terms of the simple constraint formulation;

- generating fair interpolated curves and surfaces.

Indeed, the *ratio* underlying the positioning of the extra degree of freedom should be robust, in order to have a unique flexible algorithm which give raise always to likely and fair shapes.

3.7 Variational Analysis and Modeling of Free Forms

Generally speaking, interpolation data set admits infinite solutions: there exist an infinite number of B-splines that interpolate the data. Likewise, in situations with more control points than the data to fit, there exist an infinite number of solutions.

The question arises of which one of these infinite existing curves/surfaces should be chosen. As a parameter for selecting the solution, fairness criteria could be adopted.

3.7.1 Fairness of Curves and Surfaces

The inherent subjectivity to assessing the appearance of a curve or surface makes the definition of pleasing appearance and fairness not straightforward. This difficulty is compounded by the application specificity character, which leads to different definitions.

Much of the work on curve and surface design has been to develop methods that behave naturally in response to user specification. The *draftsman's spline* was used to draw fair curves. Mathematical modeling of such spline suggested that the obtained shape was the one minimizing his strain energy, and thus, the squared curvature. That is, the functional

$$\Phi = \int k^2 \, ds \tag{3.25}$$

where k is the curvature and s the arc-length, is taken as measure of fairness and thus minimized.

For surfaces, fairing were again related to minimization of the strain energy, that is, minimizing the area integral of the sum of the principal curvatures squared.

$$\Phi = \int k_1^2 + k_2^2 \, dA \quad (3.26)$$

Some authors observed that if the length of the curve is not restricted in any way, the bend energy decreases by introducing large loops. Therefore a *stretch energy* term was also added [21; 22].

The earliest discussion about fairness was submitted from Birkhoff [23]. In analyzing various art form, when talking about vases, he describes the *Requirements for Regularity of Contour*. In particular he concluded that:

- the curvature should vary gradually, and should not oscillate more than once on any arc of the contour not containing a point of inflection
- the maximum rate of change of curvature should be as small as possible; this eliminates both unnecessarily large curvatures and rapid changes in curvature along the contour

Draw on Birkoff's work Moreton defined the minimum variation curve (MVC) [24; 25], curves which minimizes the variation of curvature:

$$\Phi = \int \left(\frac{d\mathbf{k}}{ds} \right)^2 ds \quad (3.27)$$

Moreton generalized his univariate approach by defining the minimum variation surfaces (MVS), surfaces minimizing the arc integral of the sum of the squared magnitudes of the derivatives of the normal curvatures taken in the principal direction.

$$\Phi = \int \left(\frac{d\mathbf{k}_n}{d\mathbf{e}_1} \right)^2 + \left(\frac{d\mathbf{k}_n}{d\mathbf{e}_2} \right)^2 dA \quad (3.28)$$

where, the normal curvature k_n , is the curvature of the curve projected onto the plane containing the curves tangent \mathbf{t} and the surface normal \mathbf{n} .

Such schemes can give rise to very fair surfaces, but the associated non linear optimization problem prevents them for being used for interactive surface design. A linearization of the above fairing functionals yields to:

$$\Phi = \int \|\mathbf{C}''(u)\|^2 \, du \quad \text{or} \quad \Phi = \int \|\mathbf{C}'''(u)\|^2 \, du \quad (3.29)$$

which gives a linear system to solve. The same approximation is done for surfaces:

$$\Phi = \int \left(\frac{\partial^i \mathbf{S}(u,v)}{\partial u^i} \right)^2 + \left(\frac{\partial^i \mathbf{S}(u,v)}{\partial v^i} \right)^2 \, du \, dv \quad (3.30)$$

with i begin the degree of derivative used for optimization. The approximation is parameterization dependent, and will be worse when the absolute value of the derivative of the curve is fluctuating more [22]. Among the literature, the following papers could be consulted as example of variational modeling: [10; 21; 22; 26; 27; 28].

3.7.2 Constrained Optimization

Curves

The problem then becomes the optimization of the functional chosen as fairness indicator, with the constraints given from the data to fit and the imposed derivatives.

For a curve, the first step consist in expressing the functional in terms of the unknown, which are the control points

$$\Phi = \frac{1}{2} \int_{\mathbf{C}} (\mathbf{C}^{(d)}(u))^2 \, du = \frac{1}{2} \int_{\mathbf{C}} \left(\sum_{i=0}^n N_{i,p}^{(d)}(u) \mathbf{P}_i \right)^2 \, du \quad (3.31)$$

where a functional of the d -th order derivative is chosen as example. In matrix form

$$\Phi = \frac{1}{2} \int_{\mathbf{C}} ([N^d] \, [P])^2 \, du = \frac{1}{2} \int_{\mathbf{C}} [P]^T [N^d]^T [N^d] [P] \, du \quad (3.32)$$

The control points are not parameter dependent, thus

$$\Phi = \frac{1}{2} [P]^T \left(\int_{\mathbf{C}} [N^d]^T [N^d] \, du \right) [P] \quad (3.33)$$

Just observe the similarity with the variational principle in finite elements method. The matrix K , defined with

$$[K] = \int_{\mathbf{C}} [N^d]^T [N^d] \, du \quad (3.34)$$

is called the stiffness matrix.

Thus the unconstrained variational problem takes the following form:

$$\text{Minimize} \quad \Phi = \frac{1}{2} [P]^T [K] [P] \quad (3.35)$$

To take into account the constraints, two main approaches can be undertaken: the Lagrange multiplier method, or the penalty method [29; 30; 31]. Using Lagrange multiplier all the specified constraints can be satisfied exactly. However, its drawback is that the total number of equations to be solved will increase. On the contrary, the penalty method does not possess this drawback, but it can only ensure the constraints to be satisfied approximately. Actually many other considerations about advantages and disadvantages of these two method could be done, look at [27; 28; 31].

In the present work a Lagrange multiplier approach has been adopted, due to its simple implementation. The constraints can be expressed as

$$\mathbf{g}(\mathbf{P}) = 0 \rightarrow \begin{cases} \sum_{i=0}^n N_{i,p}(\bar{u}_k) \mathbf{P}_i - Q_k = 0 \\ \sum_{i=0}^n N'_{i,p}(\bar{u}_l) \mathbf{P}_i - Q'_l = 0 \\ \sum_{i=0}^n N''_{i,p}(\bar{u}_m) \mathbf{P}_i - Q''_m = 0 \end{cases} \quad (3.36)$$

or in the matrix form

$$[g] = [N_c] [P] - [Q] = 0 \quad (3.37)$$

The constrained variational problem can be reduced to an unconstrained variational problem with the following modified functional:

$$\tilde{\Phi} = \underbrace{\frac{1}{2} [P]^T [K] [P]}_{\Phi} + [\lambda]^T [g] \quad (3.38)$$

where $[\lambda]$ is the *Lagrange multiplier* column vector, and has the same dimension as the number of constraint equations. Differentiating eq.(3.38) in order to find the stationary points:

$$\delta \tilde{\Phi} = [\delta P]^T [K] [P] + \delta ([\lambda]^T [g]) = 0 \quad (3.39)$$

which gives, with the aid of equations (3.37, 3.38)

$$\begin{cases} \frac{\partial \tilde{\Phi}}{\partial \mathbf{P}} = 0 \\ \frac{\partial \tilde{\Phi}}{\partial \boldsymbol{\lambda}} = 0 \end{cases} \longrightarrow \begin{cases} [K][P] + [\lambda]^T [N_c] = 0 \\ [N_c] [P] - [Q] = 0 \end{cases} \quad (3.40)$$

The above is a linear system of $n + n_c$ equations, where n_c are the number of the constraints, which can be represented in this form:

$$\begin{bmatrix} [K] & [N_c]^T \\ [N_c] & 0 \end{bmatrix} \begin{bmatrix} P \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ Q \end{bmatrix} \quad (3.41)$$

The linear system is defined as long the matrices $[K]$ and $[N_c]$ are computed (see appendix A for details). Since the B-spline basis $N_{i,p}$ are piecewise polynomial of order p , the Gaussian quadrature can be used to evaluate exactly the integral inside the stiffness matrix (eq.(3.34)). The computation of the constraints matrix $[N_c]$ is easily achieved since it is nothing more than evaluation of the B-spline basis functions

and their derivatives at the knot parameters associated with the constraints.

It is worth a note that, in the above matrix form, $[P]$ is intended as a column vector, even if its three spatial components would have required $[P]$ to be a matrix. The generalization is conceptually easy to do, but the notation would have been labored. Moreover, due to the constraints which set directly the derivatives, the development and resolution of the equations could be done separately for the three coordinates.

A final note should be devoted to similar algorithms developed for general case of NURBS. Thanks to the perspective mapping from the 3D Cartesian coordinate space to the 4D homogeneous coordinate space, the variational approach could be extended to NURBS. However, the weights at the control points should be pre-selected somehow [10].

Surfaces

The same process can be generalized to surfaces . However, taking advance of the tensor product properties, a scheme interpolating curves along one parameter, and then interpolating the control points in the other direction would be more efficient.

3.7.3 Conclusions

The basis to build an algorithm which meet the requirements of section 3.6.2 have been roughed in. In the next chapter the new interpolating algorithm will be analyzed in detail, both from the theoretical and capability point of view.

4

Surface Modeling in ASD

In this section will be first shown in details the main old geometric algorithms implement in ASD. After discussing their advantages and drawbacks, the new algorithms are presented.

4.1 The old Algorithms featured in ASD

The geometric modeling in ASD is achieved through local, G^1 algorithms. Their behavior is excellent in terms of speed of computation, numerical stability and robustness. The collinear points are managed as well. However, the generated shapes can seldom lead to non-optimum fairness, due also to the merely first order geometric continuity achieved across the interpolating points.

4.1.1 ASD Curve Algorithm - `local_interp_crv`

Let \mathbf{Q}_i be a set of $n + 1$ points to be interpolated. The local cubic G^1 curve interpolating algorithm is built joining cubic Bézier segments. The four control points of the i th Bézier curve are denoted as \mathbf{P}_j^i . The ends control points of each Bézier segment are coincident with the two data points to interpolate, that is:

$$\mathbf{P}_0^i = \mathbf{Q}_i \quad \mathbf{P}_3^i = \mathbf{Q}_{i+1} \quad \forall i \leq N - 1 \quad (4.1)$$

Let \mathbf{t}_i be the tangent vectors at each point: if they are not part of the interpolation

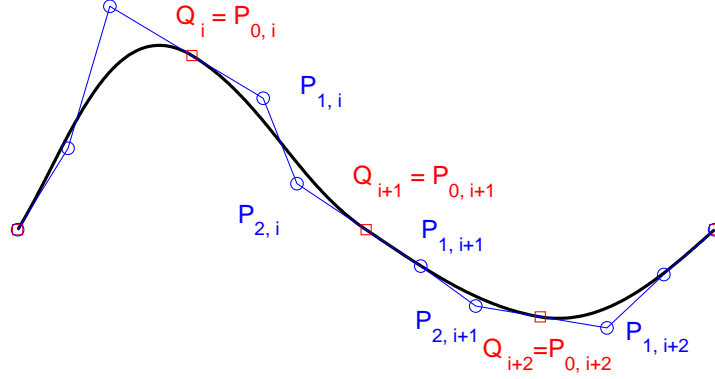


Figure 4.1: Piecewise Bézier curve

problem then an evaluation method should be adopted. Denote the line segments with $\mathbf{q}_i = \mathbf{Q}_i - \mathbf{Q}_{i-1}$. One of the most interesting method was created by Akima [11]. He sets:

$$\mathbf{t}_i = \frac{\mathbf{C}'_i}{|\mathbf{C}'_i|} \quad \mathbf{C}'_i = (1 - \alpha_i) \mathbf{q}_i + \alpha_i \mathbf{q}_{i+1} \quad (4.2)$$

$$\alpha_i = \frac{|\mathbf{q}_{i-1} \times \mathbf{q}_i|}{|\mathbf{q}_{i-1} \times \mathbf{q}_i| + |\mathbf{q}_{i+1} \times \mathbf{q}_{i+2}|}$$

The advantage is that three collinear points, $\mathbf{Q}_{k-1}, \mathbf{Q}_k, \mathbf{Q}_{k+1}$ yield a \mathbf{t}_i which is parallel to the line segment. At the same time, if the points $\mathbf{Q}_{k-2}, \mathbf{Q}_{k-1}, \mathbf{Q}_k$ and $\mathbf{Q}_k, \mathbf{Q}_{k+1}, \mathbf{Q}_{k+2}$ are collinear, the denominator of eq.(4.2) vanishes. This implies either a corner at \mathbf{Q}_i or a straight line segment from \mathbf{Q}_{i-2} to \mathbf{Q}_{i+2} . In such situations α_i can be defined in a number of ways:

- $\alpha_i = 1$, which implies $\mathbf{C}'_i = \mathbf{q}_{i+1}$; that is a corner at \mathbf{Q}_i ;
- $\alpha_i = \frac{1}{2}$, which implies $\mathbf{C}'_i = \frac{1}{2}(\mathbf{q}_i + \mathbf{q}_{i+1})$; this smoothes out the corner.

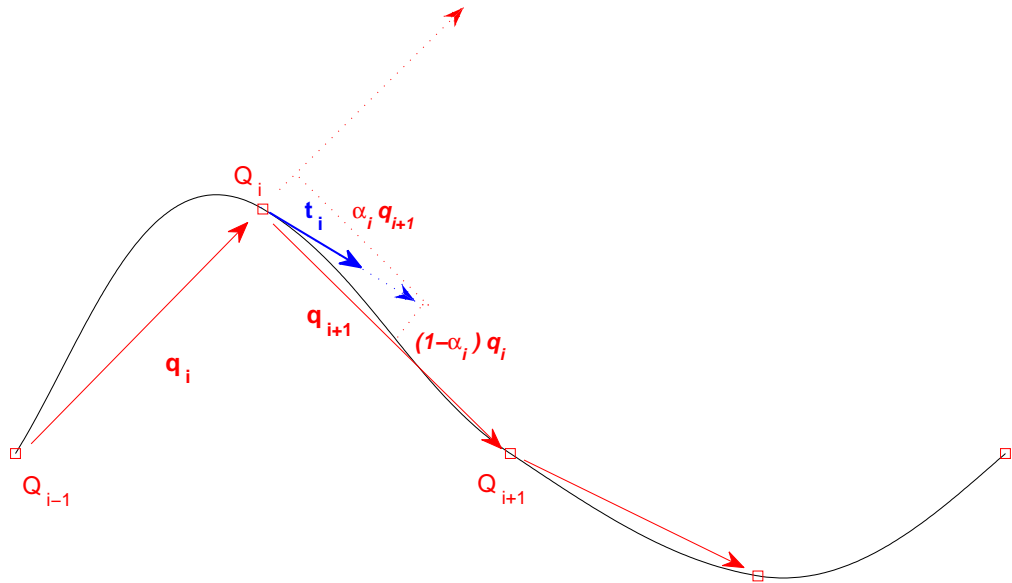


Figure 4.2: Setting local tangent with Akima's method

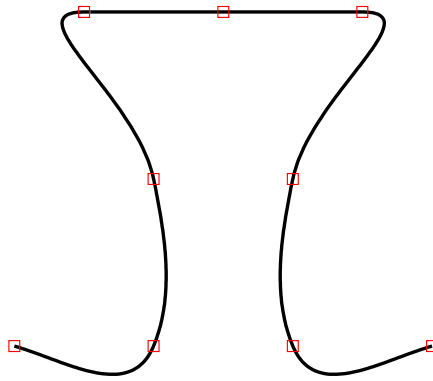


Figure 4.3: Straight line segment with three collinear points

When calling the algorithm, a flag would specify the required behavior. The end conditions should be treated in a different manner, if they are not specified of course.

A method which gives good results is:

$$\begin{aligned} \mathbf{q}_0 &= 2\mathbf{q}_1 - \mathbf{q}_2 & \mathbf{q}_{-1} &= 2\mathbf{q}_0 - \mathbf{q}_1 \\ \mathbf{q}_{n+1} &= 2\mathbf{q}_n - \mathbf{q}_{n-1} & \mathbf{q}_{n+2} &= 2\mathbf{q}_{n+1} - \mathbf{q}_n \end{aligned}$$

If the desired curve should be a closed curve with at least G^1 continuity, then:

$$\begin{aligned} \mathbf{q}_0 &= \mathbf{q}_n - \mathbf{q}_n - 1 & \mathbf{q}_{-1} &= \mathbf{q}_{n-1} - \mathbf{q}_{n-2} \\ \mathbf{q}_{n+1} &= \mathbf{q}_1 - \mathbf{q}_0 & \mathbf{q}_{n+2} &= \mathbf{q}_2 - \mathbf{q}_1 \end{aligned}$$

that is, the tangents are the same at the (coincident) end points. If the curve is a closed one, a flag will tell the algorithm which interpolation method to adopt. In

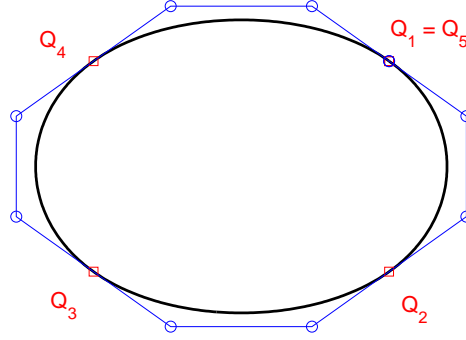


Figure 4.4: Close curve with G^1 continuity

fact, airplane surfaces consist of both closed curve with edges (most of the airfoils) and smooth closed curve (sections of the fuselage).

From eq.(2.8) for the internal control points \mathbf{P}_1^i and \mathbf{P}_2^i

$$\begin{aligned} \mathbf{P}_1^i &= \mathbf{P}_0^i + \frac{1}{3} \beta_0^i \mathbf{t}_i & \mathbf{P}_2^i &= \mathbf{P}_3^i - \frac{1}{3} \beta_3^i \mathbf{t}_{i+1} \\ \text{with } \beta_0^i &= \|\mathbf{C}'_i\| & \beta_3^i &= \|\mathbf{C}'_{i+1}\| \end{aligned} \tag{4.3}$$

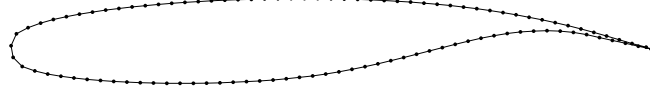


Figure 4.5: Supercritical airfoil NASASC2 interpolated with ASD algorithm.

The two unknowns are related with the internal parameterization. A good choice is to set, for each segment, equal derivatives magnitude at the endpoints and the middle point, that is (omitting the segment index i):

$$\beta = \|\mathbf{C}'(0)\| = \|\mathbf{C}'(\frac{1}{2})\| = \|\mathbf{C}'(1)\| \quad (4.4)$$

Using de Casteljau algorithm (section 2.2.5) to evaluate the middle point derivative yields to

$$\beta = \|\mathbf{C}(\frac{1}{2})\| = \frac{3}{4} \|\mathbf{P}_3 + \mathbf{P}_2 - \mathbf{P}_1 - \mathbf{P}_0\| \quad (4.5)$$

Finally, using equation (4.3) leads to the following equation:

$$16\beta^2 = \beta^2 \|\mathbf{t}_0 + \mathbf{t}_3\|^2 - 12\beta (\mathbf{P}_3 - \mathbf{P}_0) \cdot (\mathbf{t}_0 + \mathbf{t}_3) + 36 \|\mathbf{P}_3 - \mathbf{P}_0\|^2 \quad (4.6)$$

The equation admits two real solutions, only one of them is positive. Given β it's easy to compute the internal control points. Applying the scheme for every segment leads to the complete definition of the control points of all the Bézier curves.

Now the piecewise Bézier curve should be expressed in B-spline form. It is possible to obtain a C^1 continuous cubic and to achieve a good approximation to uniform parameterization setting

$$\bar{u}_{i+1} = \bar{u}_i + 3 \|\mathbf{P}_1^i - \mathbf{P}_0^i\| \quad (4.7)$$

This choice equals the speed (respect the B-spline parameter u) at every segment. Finally, the B-spline curve is defined from the control points:

$$\mathbf{Q}_0, \mathbf{P}_1^0, \mathbf{P}_2^0, \mathbf{P}_1^1, \mathbf{P}_2^1, \mathbf{P}_3^1, \dots, \mathbf{P}_1^{n-1}, \mathbf{P}_2^{n-1}, \mathbf{Q}_n \quad (4.8)$$

and knot vector:

$$U = \{0, 0, 0, 0, \frac{\bar{u}_1}{\bar{u}_n}, \frac{\bar{u}_1}{\bar{u}_n}, \frac{\bar{u}_2}{\bar{u}_n}, \frac{\bar{u}_2}{\bar{u}_n}, \dots, \frac{\bar{u}_{n-1}}{\bar{u}_{n-1}}, \frac{\bar{u}_{n-1}}{\bar{u}_{n-1}}, 1, 1, 1, 1\} \quad (4.9)$$

4.1.2 ASD Bicubic Surface Algorithm - local_interp_sfc

This scheme leads to C^1 , bicubic B-spline surfaces. Let

$$\{\mathbf{Q}_{k,l}\}, \quad k = 0, \dots, n \quad \text{and} \quad l = 0, \dots, m \quad (4.10)$$

be a set of points, and let $\{(\bar{u}_k, \bar{v}_l)\}$ be the corresponding parameter pairs, computed by chord length averaging (section 3.3.6). The surface is obtained by joining $n \cdot m$ bicubic Bézier patches, $\{\mathbf{B}_{k,l}(u,v)\}$, where $\mathbf{Q}_{k,l}, \mathbf{Q}_{k+1,l}, \mathbf{Q}_{k,l+1}, \mathbf{Q}_{k+1,l+1}$ are the corner points of the patch. Except for the surface boundaries, all rows and columns of control points containing the original $\{\mathbf{Q}_{k,l}\}$ are removed leaving $(2n+2)(2m+2)$ control points in the final B-spline surface. The knot vectors are:

$$\begin{aligned} U &= \{0, 0, 0, 0, \bar{u}_1, \bar{u}_1, \bar{u}_2, \bar{u}_2, \dots, \bar{u}_{n-1}, \bar{u}_{n-1}, \bar{u}_n, \bar{u}_n, \bar{u}_n\} \\ V &= \{0, 0, 0, 0, \bar{v}_1, \bar{v}_1, \bar{v}_2, \bar{v}_2, \dots, \bar{v}_{m-1}, \bar{v}_{m-1}, \bar{v}_m, \bar{v}_m, \bar{v}_m\} \end{aligned} \quad (4.11)$$

Every patch has 16 control points. The 12 boundary points are obtained looping through the $m+1$ rows and $n+1$ columns of data and using a cubic interpolation scheme. The scheme is slightly different from the previously detailed one, since all the rows (columns) must have the same parameterization in a tensor product surface. Thus, C^1 continuity at the segments endpoints can be still forced, but is not possible to force equal speed at the midpoint of the Bézier segments. In detail, consider the interpolation of the row $l = l_0$ (consisting of the points $\mathbf{Q}_{0,l_0}, \dots, \mathbf{Q}_{n,l_0}$), of total chord length r_{l_0} . The tangent estimation could be done again with Akima's method,

and denote with \mathbf{t}_{k,l_0}^u the unit tangent in the u direction at the point \mathbf{Q}_{k,l_0} .

The interior Bézier point on this row are then computed by

$$\begin{aligned} \mathbf{P}_{1,0}^{(k,l_0)} &= \mathbf{Q}_{k,l_0} + \beta \mathbf{t}_{k,l_0}^u & \mathbf{P}_{2,0}^{(k,l_0)} &= \mathbf{Q}_{k+1,l_0} - \beta \mathbf{t}_{k+1,l_0}^u \\ \beta &= \frac{r_{l_0}(\bar{u}_{k+1} - \bar{u}_k)}{3} \end{aligned} \quad (4.12)$$

For the computation of the four internal control points of each Bézier patch, some estimations for the mixed partial derivative, $\mathbf{D}_{k,l}^{uv}$, at each $\mathbf{Q}_{k,l}$ should be done. A good choice is the three point Bessel method. In the univariate situation it states:

$$\begin{aligned} \mathbf{d}_k &= \frac{\mathbf{q}_k}{\Delta \bar{u}_k} & \mathbf{D}_k &= (1 - a_k) \mathbf{d}_k + a_k \mathbf{d}_{k+1} \\ \text{with} \quad a_k &= \frac{\Delta \bar{u}_k}{\Delta \bar{u}_k + \Delta \bar{u}_{k+1}} \end{aligned} \quad (4.13)$$

Let r_l and s_k denote the total chord length of the l th row and k th column.

$$\mathbf{D}_{k,l}^u = r_l \mathbf{t}_{k,l}^u \quad \mathbf{D}_{k,l}^v = s_k \mathbf{t}_{k,l}^v \quad (4.14)$$

Then, set

$$\begin{aligned} \mathbf{d}_{k,l}^{uv} &= (1 - a_k) \frac{\mathbf{D}_{k,l}^v - \mathbf{D}_{k-1,l}^v}{\Delta \bar{u}_k} + a_k \frac{\mathbf{D}_{k+1,l}^v - \mathbf{D}_{k,l}^v}{\Delta \bar{u}_{k+1}} \\ \mathbf{d}_{k,l}^{uv} &= (1 - b_l) \frac{\mathbf{D}_{k,l}^u - \mathbf{D}_{k,l-1}^u}{\Delta \bar{v}_l} + b_l \frac{\mathbf{D}_{k,l+1}^u - \mathbf{D}_{k,l}^u}{\Delta \bar{v}_{l+1}} \end{aligned} \quad (4.15)$$

with

$$a_k = \frac{\Delta \bar{u}_k}{\Delta \bar{u}_k + \Delta \bar{u}_{k+1}} \quad b_l = \frac{\Delta \bar{v}_l}{\Delta \bar{v}_l + \Delta \bar{v}_{l+1}} \quad (4.16)$$

Finally

$$\mathbf{D}_{k,l}^{uv} = \frac{a_k \mathbf{d}_{k,l}^{uv} + b_l \mathbf{d}_{k,l}^{vu}}{a_k + b_l} \quad (4.17)$$

The four interior control points of the (k, l) th patch are now computed using eq.(4.17)

and eq.(2.38)

$$\begin{aligned}
\mathbf{P}_{1,1}^{k,l} &= \gamma \mathbf{D}_{k,l}^{uv} + \mathbf{P}_{0,1}^{k,l} + \mathbf{P}_{1,0}^{k,l} - \mathbf{P}_{0,0}^{k,l} \\
\mathbf{P}_{2,1}^{k,l} &= -\gamma \mathbf{D}_{k+1,l}^{uv} + \mathbf{P}_{3,1}^{k,l} - \mathbf{P}_{3,0}^{k,l} + \mathbf{P}_{2,0}^{k,l} \\
\mathbf{P}_{1,2}^{k,l} &= -\gamma \mathbf{D}_{k,l+1}^{uv} + \mathbf{P}_{1,3}^{k,l} - \mathbf{P}_{0,3}^{k,l} + \mathbf{P}_{0,2}^{k,l} \\
\mathbf{P}_{2,2}^{k,l} &= \gamma \mathbf{D}_{k+1,l+1}^{uv} + \mathbf{P}_{2,3}^{k,l} + \mathbf{P}_{3,2}^{k,l} - \mathbf{P}_{3,3}^{k,l}
\end{aligned} \tag{4.18}$$

with

$$\gamma = \frac{\Delta \bar{u}_{k+1} \Delta \bar{v}_{l+1}}{9}$$

4.2 The new Geometric Algorithms

The requirements for the new algorithms have been stated in the above chapter. A summary of the key points are here reported:

- C^2 continuity at each interpolation point (and off course in the interior domain);
- flexibility and the capability of handling straight segments (or flat patches);
- capability of handling up to second order derivative constraints;
- generation of fair curves (surfaces).

4.2.1 The new Curve Algorithm - `global_interp_crv`

The problem is better approached from the piecewise Bézier of the local algorithms. Is quite straightforward that a fifth degree Bézier, with its four inner control points, provide the necessary flexibility in order to match the first and second order derivative, imposed as constraints or as continuity condition with the adjacent Bézier segments.

Regardless of the degree, the first step to build B-spline joining together this Bézier segments is to define a knot vector and a knot parameter set, with one of the method talked about in section 3.3. Whatever the choice, knot parameter and knot

vector are chosen with the same distribution. The knot vector would be of this form:

$$U = \{\underbrace{0, \dots, 0}_{p+1}, \underbrace{\bar{u}_1, \dots, \bar{u}_1}_p, \dots, \underbrace{\bar{u}_k, \dots, \bar{u}_k}_p, \dots, \underbrace{\bar{u}_{N-1}, \dots, \bar{u}_{N-1}}_p, \underbrace{1, \dots, 1}_{p+1}\} \quad (4.19)$$

Due to B-spline properties, the multiplicity of the internal knot vector is related with the continuity. Therefore, adjusting the internal points of each Bézier segments to meet C^2 continuity, means that the B-spline could be refined till internal knots have multiplicity $p - 2$. Thus, the final knot vector would be of the form

$$U = \{\underbrace{0, \dots, 0}_{p+1}, \underbrace{\bar{u}_1, \bar{u}_1, \bar{u}_1}_{p-2}, \dots, \underbrace{\bar{u}_k, \bar{u}_k, \bar{u}_k}_{p-2}, \dots, \underbrace{\bar{u}_{N-1}, \bar{u}_{N-1}, \bar{u}_{N-1}}_{p-2}, \underbrace{1, \dots, 1}_{p+1}\}$$

The number of the control points is then immediately defined, due to B-spline properties, from

$$n = \underbrace{(N - 2) \cdot (p - 2) + 2 \cdot N \cdot (p + 1)}_m - p - 1 \quad (4.20)$$

where N is the number of point to be interpolated, n is the number of control points, m is the knot vector length. Assume a set of points to be interpolated

$$\{Q_k\} \quad k = 0, \dots, N$$

and d and dd constraints on first and second order derivatives:

$$\begin{aligned} \{Q'_k\} \quad k &= 0, \dots, K \\ \{Q''_{kk}\} \quad k &= 0, \dots, K \end{aligned}$$

The last step in defining the problem is the selection of the fairing functional, which would be of the form

$$\Phi = \frac{1}{2} \int_C \|\mathbf{C}^{(d)}\|^2 \, du \quad (4.21)$$

with d being the *optimized* order derivative. At that point, provided

- the degree p of the B-Spline,
- the distribution of the knot vector and the parameterization,
- the fitting data

the linear system is of section 3.7.2 is easily achieved in explicit terms.

It is worth a comment on the above points. Interpolating points are part of the input data. Also derivative constraints can be imposed at edges, in order to join the curve with the desired smoothness. But, if straight segments are wished when three or more points are collinear, first and second order derivative constraints have to be set also in the inner region of the curve. Derivative constraints at the inner region are also needed if a local flavor of the interpolating curve is required. For the aforementioned reasons fifth degree B-spline is the choice when seeking for flexibility, the third order not being able to satisfy second order derivative constraints (it leads to an over determined system).

The imposition of the derivatives, when actually only shape constrains are needed, leads to a *fair* choice of the parameterization parameters. For the first order derivative:

$$\mathbf{C}' = \alpha \mathbf{t} \quad (4.22)$$

a reasonable estimation of α would be [8]

$$\alpha = \sum_{k=0}^N \|\mathbf{Q}_{k+1} - \mathbf{Q}_k\| = c_{tot} \quad (4.23)$$

which performs well with chord length parameterization. For the second order derivative:

$$\mathbf{C}'' = \dot{\alpha} \mathbf{t} + \alpha^2 \mathbf{k} \quad (4.24)$$

a usually well behaving estimation of $\dot{\alpha}$ would be

$$\dot{\alpha} = 0 \quad (4.25)$$

Note that, in an interactive environment, α and $\dot{\alpha}$ can be used as additional shape controls.

A final note should be devoted to closed curve, with or without continuity. If smooth closure is needed, then the following condition, which can be easily integrated in the system, is added:

$$\begin{cases} \mathbf{C}'(0) = \mathbf{C}'(1) \\ \mathbf{C}''(0) = \mathbf{C}''(1) \end{cases} \implies \begin{cases} \sum_{i=0}^n N'_{i,p}(0) \mathbf{P}_i = \sum_{i=0}^n N'_{i,p}(1) \mathbf{P}_i \\ \sum_{i=0}^n N''_{i,p}(0) \mathbf{P}_i = \sum_{i=0}^n N''_{i,p}(1) \mathbf{P}_i \end{cases} \quad (4.26)$$

Finally, the algorithm offers the following capabilities:

- possibility of choosing between smooth or not smooth closed curve interpolation;
- possibility of interpolating the data points with local tangents estimated with Akima's method;
- choose if aligned points should be interpolated with straight segments, and eventually settings the tolerance for which points should be considered aligned;
- choose between three different parameterization;
- choose the derivative order of the shape optimization functional;
- choose between third or fifth degree B-spline.

The flexibility of the algorithm is shown in the figures 4.6 - 4.9.

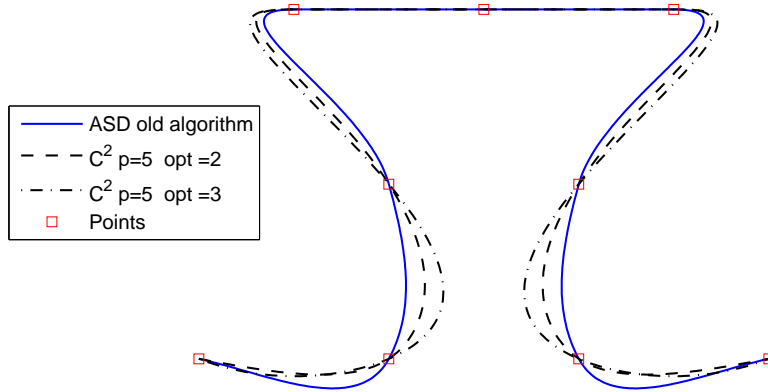


Figure 4.6: New algorithm behavior with three collinear points, for both fifth degree B-spline with fairing over second or third order derivative.

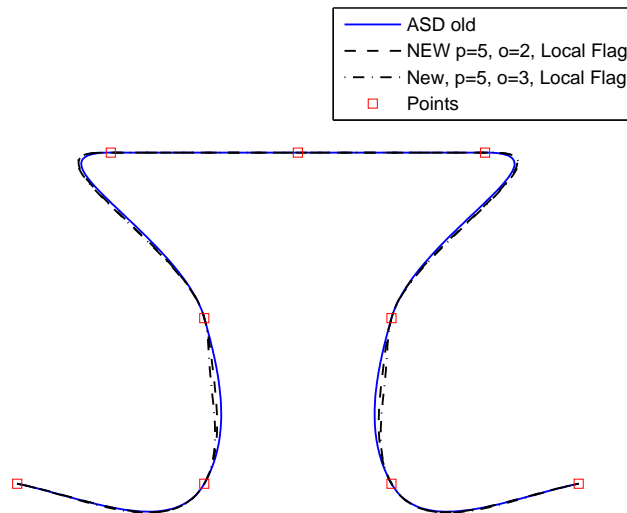


Figure 4.7: New algorithm behavior with three collinear points, for both fifth degree B-spline with fairing over second or third order derivative, and the same tangent vectors of the old local algorithm.

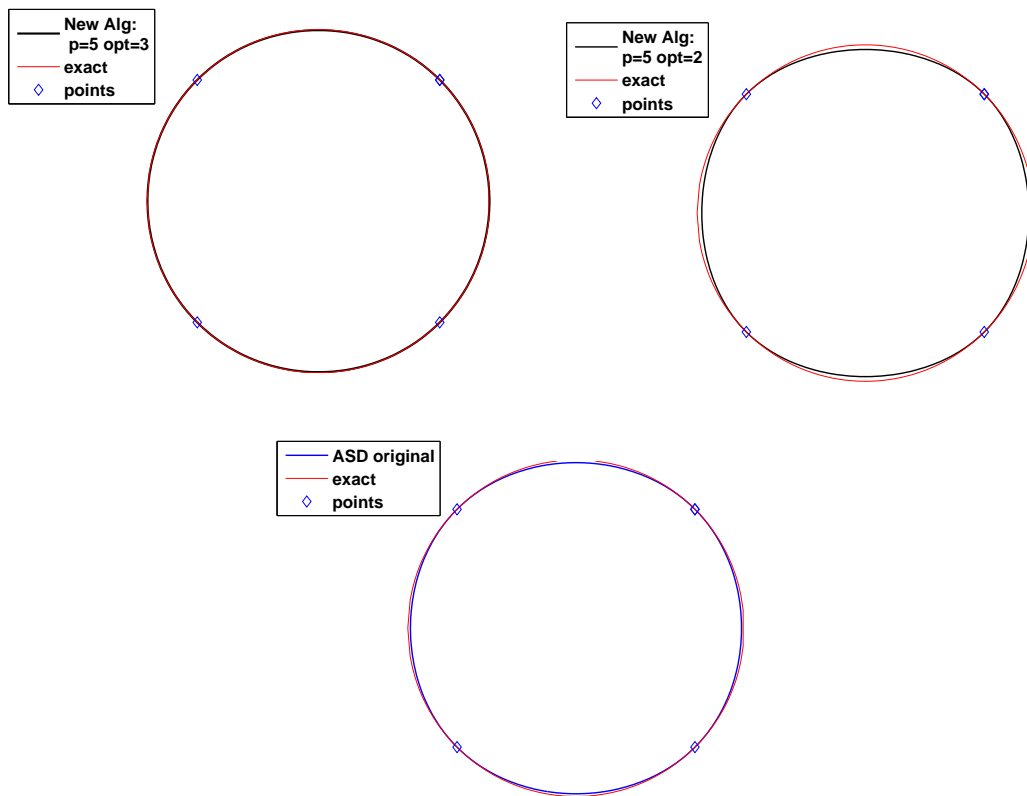


Figure 4.8: New algorithm behavior with 5 points with smooth flag. The third order derivative fairing based curve matches nearly exactly the circumference.

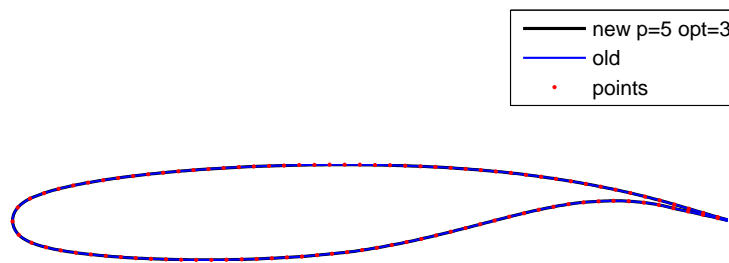


Figure 4.9: Airfoil interpolation with the new and old algorithms. Note that, if the points to interpolate are many and properly distributed, the two interpolant curves are nearly matching each other.

4.2.2 The new Surface Algorithm - `global_interp_sfc`

If $\mathbf{Q}_{k,l}$ is a $N1 \times N2$ matrix of points to be interpolated with a (p, q) th degree B-spline, then it holds :

$$\mathbf{Q}_{k,l} = \mathbf{S}(\bar{u}_k, \bar{v}_l) = \sum_{i=0}^{n_1} \sum_{j=0}^{n_2} N_{i,p}(\bar{u}_k) N_{j,q}(\bar{v}_l) \mathbf{P}_{i,j} \quad (4.27)$$

The product tensor surfaces enables to extend effortlessly the curves algorithm for surface interpolation. Thus, the same *frame* should be adapted to fit the bivariate situation. The two knot vectors are obtained through the usual average process:

$$\begin{aligned} \bar{u}_k &= \frac{1}{m_2 + 1} \sum_{l=0}^{m_2} \bar{u}_k^l & k &= 0, \dots, m_1 \\ \bar{v}_l &= \frac{1}{m_1 + 1} \sum_{k=0}^{m_1} \bar{v}_l^k & l &= 0, \dots, m_2 \end{aligned} \quad (4.28)$$

where the parameters \bar{u}_k^l and the \bar{v}_l^k are obtained with one of the methods adopted in the univariate problem. Just note that it still holds the eq.(4.20) applied to both the directions.

Bringing back the surface interpolation to a sequence of curve interpolation

$$\mathbf{Q}_{k,l} = \sum_{i=0}^{n_1} N_{i,p}(\bar{u}_k) \mathbf{R}_{i,l} \quad (4.29)$$

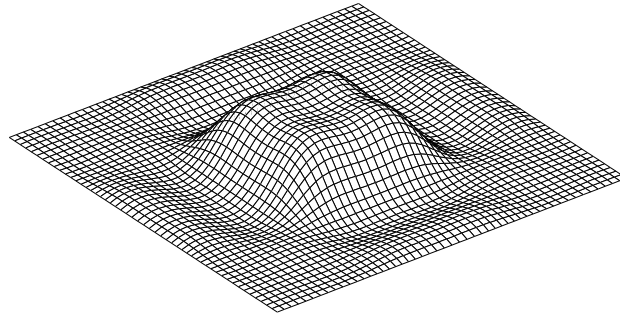
with

$$\mathbf{R}_{i,l} = \sum_{j=0}^{n_2} N_{j,q}(\bar{v}_l) \mathbf{P}_{i,j} \quad (4.30)$$

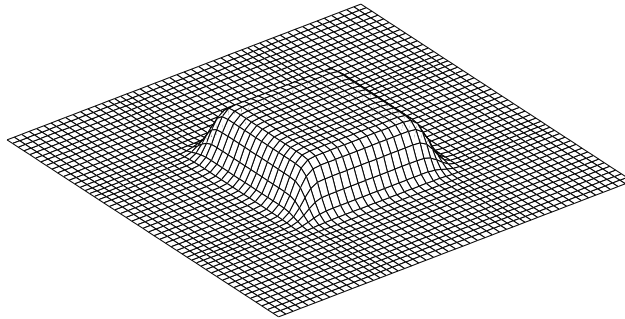
Thus, the process consists in fairing

- first the $n_1 + 1$ isoparametric curves,
- then the $n_2 + 1$ curves interpolating the isoparametric curve control points.

Capabilities of this algorithm are strictly related with those of the univariate one. However, due to the sequence of curve interpolation, some restrictions arise.



(a) Alignment Flag OFF



(b) Alignment Flag ON

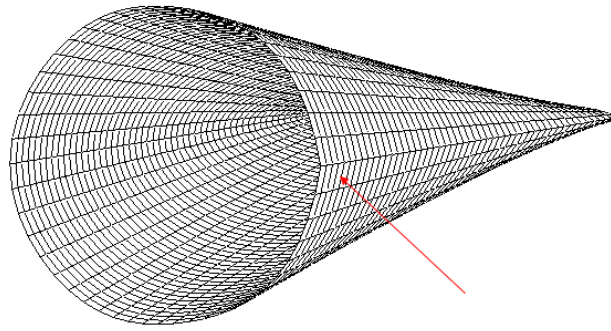
Figure 4.10: Behavior of the algorithm with and without the alignment flag option activated. Note how the interpolating algorithm is capable of detecting and drawing flat patches

As an example, the imposition of derivative constraints in both the directions is a little tricky. Besides this, the algorithm behaves in a satisfactory way in front of the typical aeronautical surfaces to be interpolated and, due to its flexibility, even to more generic shapes.

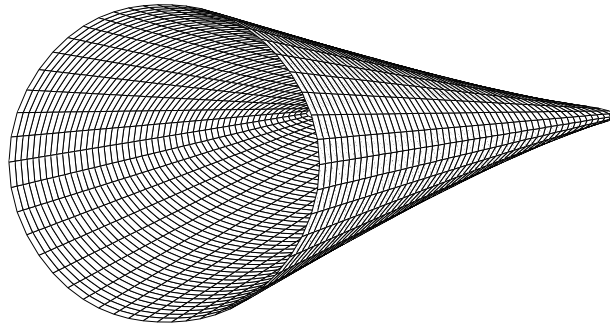
Some results and capabilities of the algorithm are shown in fig.4.10 and 4.11.

4.2.3 Lofting Algorithm

If the sections to be interpolated differ much in shape, and at the same time exact section reproduction is required, it may happen that the unique knot vector, obtained



(a) Corner Flag ON



(b) Corner Flag OFF

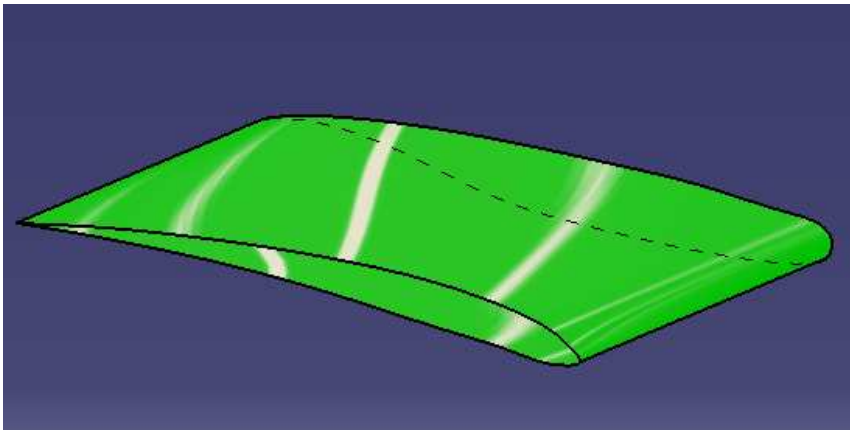
Figure 4.11: Behavior of the algorithm with and without the corner flag option activated. Note how the interpolating algorithm is capable of closing with smoothness a closed surface

as average of section knot vectors, represents for a some sectional curve a *unhappy* choice, leading to unsatisfactory results. Look back at section 3.3 to realize the consequence of a not proper knot vector and parameterization.

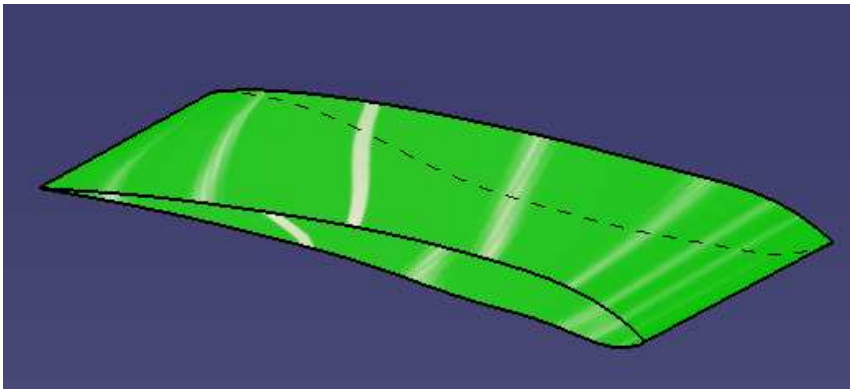
For aeronautical shapes, is particularly important to avoid not accurate shapes, especially when dealing with airfoils. An example is shown in fig. 4.12.

At the moment, lofting is implemented only for wing features, with the sectional curves being the airfoils (the spanwise direction interpolation is linear). The process consists in:

- definition of each sectional curve (airfoil) starting from its points interpolation through the above univariate algorithm ;



(a) Lofting



(b) Normal Interpolation

Figure 4.12: Wing with quite dissimilar airfoil shapes at tip and root. Each airfoil is interpolated with just twenty points.

- knot insertion when correspondent knots differ to a selected amount;
- linear interpolation of the sectional curves in the spanwise direction.

Lofting tend to be redundant if the sectional curves are defined through a high number of interpolation points.

4.3 The ASD Advanced NURBS GUI

The implementation of the new algorithm gives a consistent versatility of ASD in terms of shape generation. For each feature, the Advanced NURBS GUI gives advanced control of the geometric shape generation. Thus, the most appropriate interpolating algorithm could be used depending on the particular needs. In fig.4.13 is depicted a general example of the interface. Although similar, the GUI is specialized for every feature. For body, bulk, wound, wingbody features the properties refer

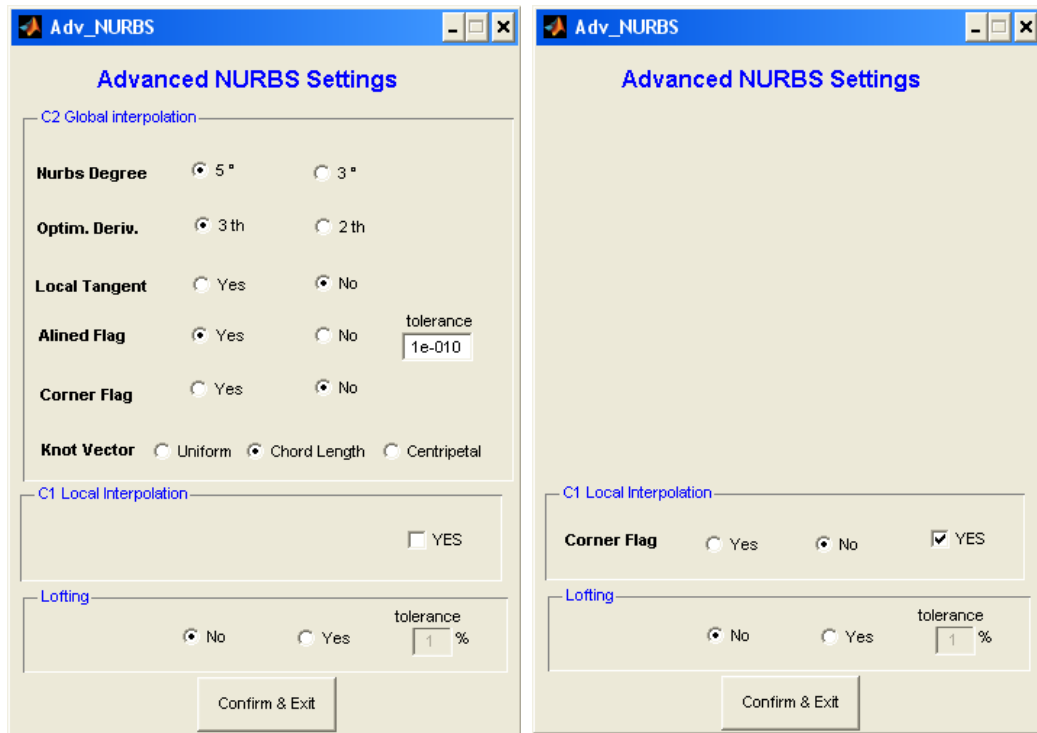


Figure 4.13: The Advanced NURBS GUI

to both directions, for wing the properties refer only to the direction running along chord, since the wings are interpolated in a linear way along spanwise direction.

4.3.1 C^1 Algorithm Parameter

It is possible to choose if to maintain the original C^1 local algorithm or to move to a C^2 global one. If the data set are very large and curvature continuity is not

important the former one is the choice to do. In fact, it leads to very fast geometric generation without problems of numerical instability. Anyway, the flexibility is reduced, the shape with few data set couldn't be the most fair and the use of other parameterization is not supported. In this interpolating scheme, handle of continuity of closed curves is also provided.

4.3.2 C^2 Algorithm Parameter

When high quality surfaces or greater flexibility is required, it is necessary to choose the global algorithm.

Degree selection

The global algorithm permits to choose between third or fifth degree B-spline. Anyway, only the fifth permits to handle the parameters, in third degree algorithm this is not yet supported. The use of third degree B-spline should be undertaken when the data set to interpolate is very unevenly spaced, and fifth degree could lead to unwanted bulges (in this situation the effects of linearization of the optimization functional come out).

Optimization derivative selection

It is possible to choose between second or third order derivative square minimization as fairness functional. For third degree B-spline only second order derivative based functional is supported. This options has relevance when only a few points to interpolate are provided and/or particular shapes, like circular section shapes, are desired as most closely as possible (see also fig.4.8 for a different performance of the fairness functional on a circumference, or fig.4.6 for another particular situation).

Local tangent selection

To force the global algorithm to interpolate the same data set as the local algorithm, is possible to impose at each point tangent vector estimated with Akima's method.

This will be useful to achieve C^2 maintaining the shapes obtained with the local method. Again this feature is addressed to quintic B-spline.

Aligned flag selection

When selected the interpolating algorithm draws straight segments between three or more aligned points. The tolerance field permits to set the gap within to consider the points to be aligned. This capability is not yet supported for third order algorithm.

Corner flag selection

When closed curve are encountered, a smooth (C^2) or not smooth closure can be imposed. Fuselage and wings represent classical examples of the two situations. For instance, a wing trailing edge is not desirable to be closed with a smooth curve.

Knot vector selection

The knot vector can be built using uniform, chord length and centripetal parameterization. For a few and unevenly spaced data points the results can be much different.

4.3.3 Lofting

The field tolerance sets the limits to which add or not the new knot. Lofting gives high quality surfaces at the price of high computational cost. It should be used only when necessary.

4.4 Some Results with the new Geometric Engine

In the following section some results obtained with the new geometric engine of ASD are shown.

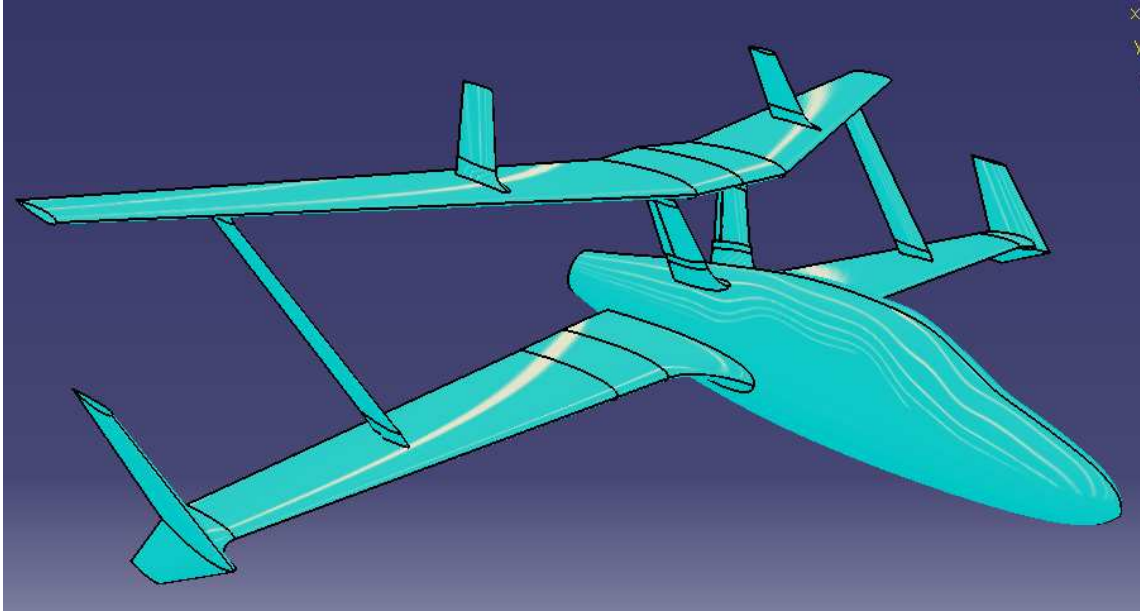


Figure 4.14: A complex configuration built to verify the new algorithm capabilities. The CA-TIA visualization option is the *Shading with edges* one. As shown, no curvature discontinuities is detected in the inner region of the surfaces.

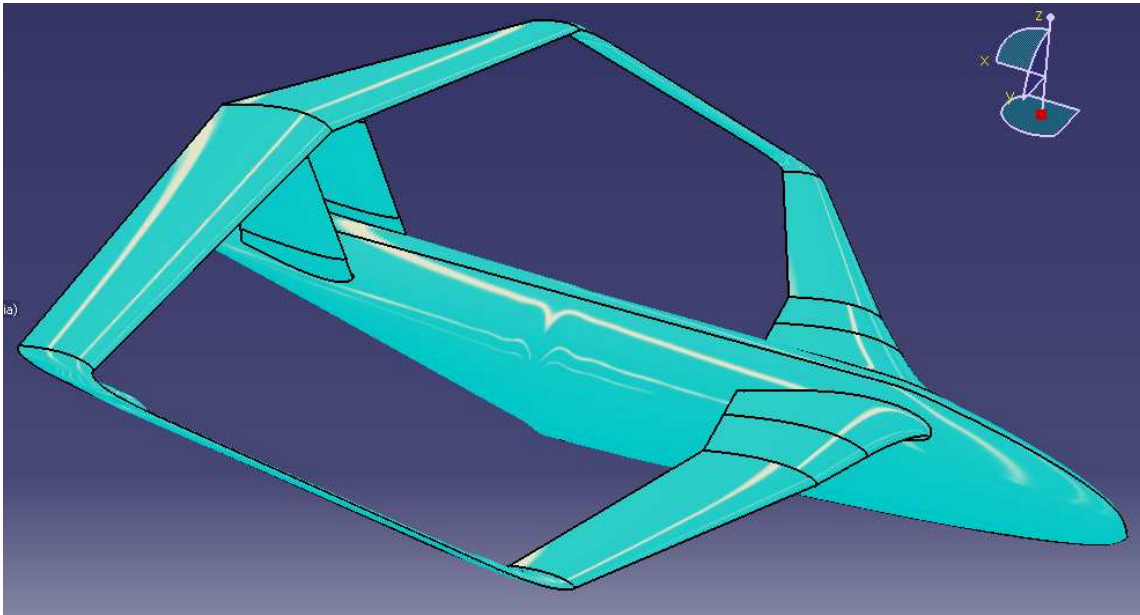


Figure 4.15: A PrandtlPlane Canadair configuration.

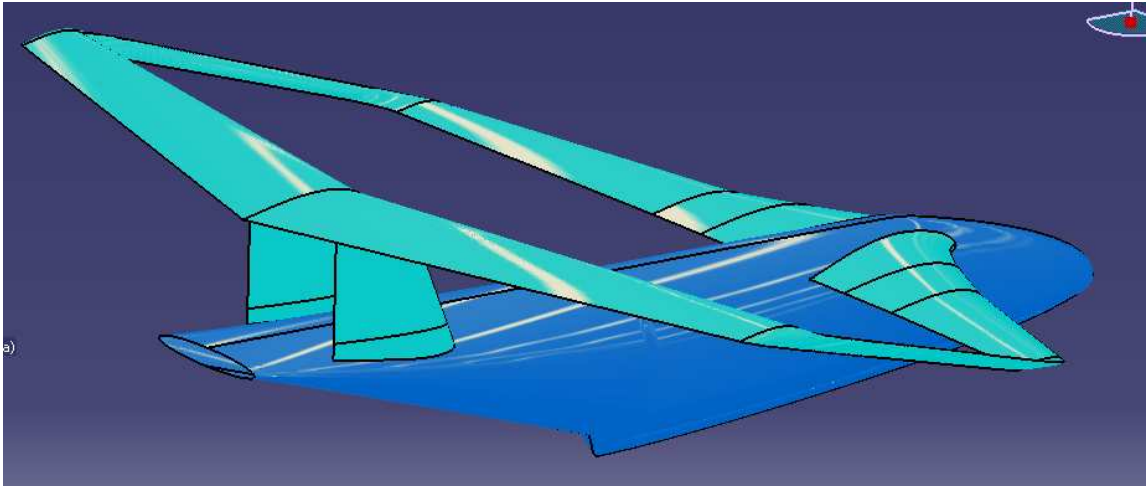


Figure 4.16: A PrandtlPlane Canadair configuration.

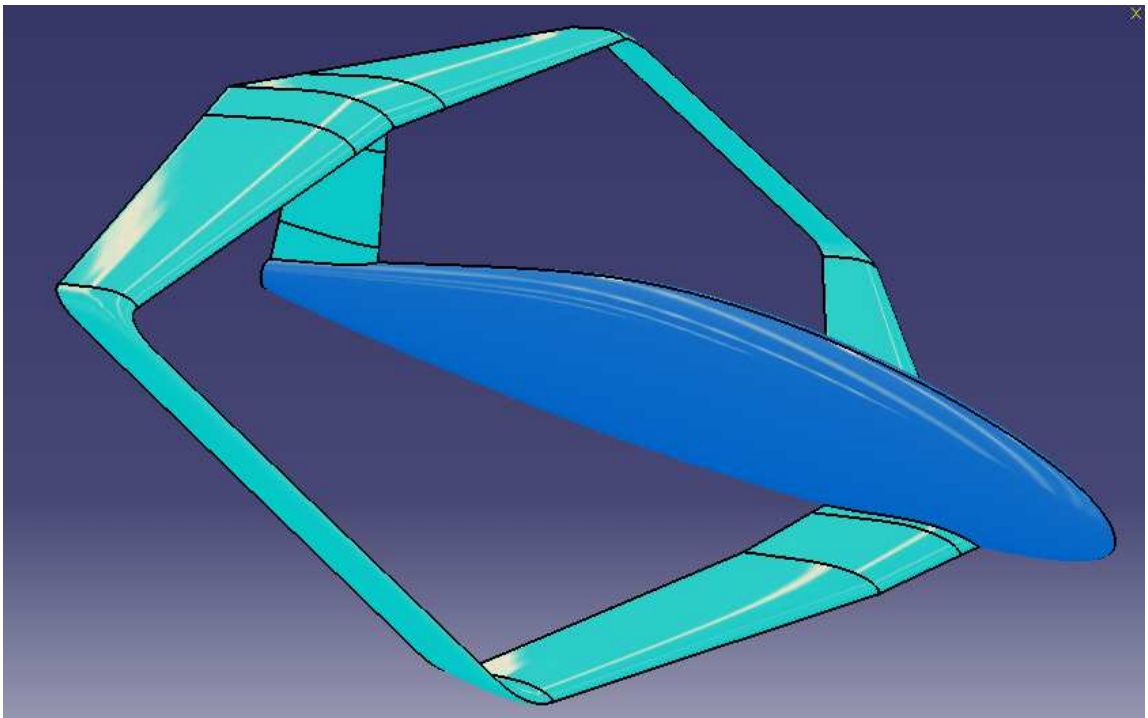


Figure 4.17: The PrandtlPlane ULV Tandem.

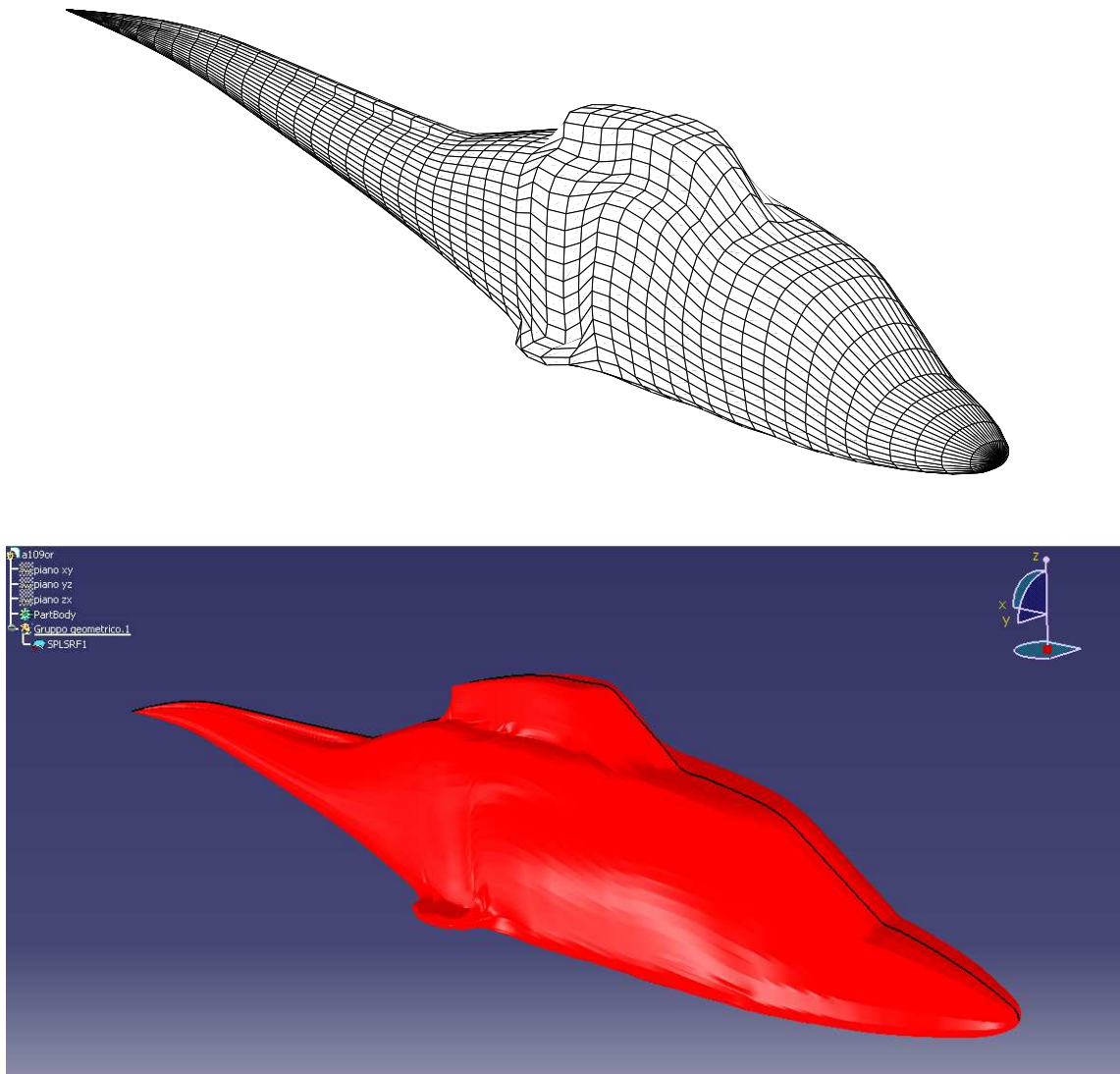


Figure 4.18: The body of an Helicopter similar to the Agusta A109 .

4.5 Future Improvements

Surely the improved flexibility raises ASD value in terms of geometric modeling. Improvements can still be made, for the C^2 algorithm, in the following area:

- on the efficiency side, the algorithms should be revised to avoid matrices close to singular for very large sets of interpolation points. In fact, as for FEM methods, the matrix are *stiff* if considerable large problems are handled. It may be considered to use penalty method approach instead of Lagrange multiplier one. An idea may be also the use of wavelets, as described in [32].
- It should be checked if some constrains tend to be linearly dependent, at least considering numerical roundoff;
- implementation of robust simultaneous both direction derivative constraints capability;

Talking about the geometric engine in general terms, it feels the need of an approximation fitting algorithm, in order to avoid bulges and waves especially at intersections, where, due also to numerical errors of the intersection algorithms, they most frequently arise. In this sense the flexibility of the fifth degree algorithm is counter-productive, and all the limitations of the linear fairing functional come out.

5

Structured Grid Generation Module

As shown in section 1.3, ASD features an internal powerful mesher, yet capable to generate only unstructured grid. On the other side, PanAir, such as many other programs, requires a structured quadrilateral mesh as input geometry, thus an internal structured grid generator should be implemented in order to have an unique integrated environment. Moreover, due to Pan Air grid requirements, particular caution should be reserved for network edges connected along common interface, as indicated in section 6.2.3.

5.1 Introduction

The partial differential equations that govern physics are not usually amenable to analytical solutions, except for very simple cases. Therefore, the domains of computation are split into smaller subdomains (made up of geometric primitives like hexahedra and tetrahedra in 3D and quadrilaterals and triangles in 2D) and the governing equations are then discretized and solved inside each of these subdomains. The subdomains are often called elements or cells, and the collection of all elements or cells is called a *mesh* or *grid*.

The accuracy of numerical computations is influenced to a large extent by the quality of the grid used, hence grid generation techniques should allow to control the

grid structure and the distribution of grid points. At the same time, since grid generation is an intermediate stage between the geometric definition and the mathematical solver, it should ideally be fast and automatic, requiring minimal user intervention.

Despite tremendous advances in grid generation in the last decades, it still represents one of the major bottleneck in terms of time and automation [33]. This holds especially for complex three-dimensional configuration and CFD analysis, where the requirements of deeply refined and smooth grid to avoid considerable numerical error can be met with much more effort.

However, surface meshes are easier to generate, and at the same time panel methods are less sensitive than CFD to grid quality, this being especially true for higher order panel method like PanAir. Moreover, the NURBS parametric description of the surfaces is advantageous for accurate grid generation. All these aspects enable to achieve an acceptable grid generation at almost interactive speed, this being of primary importance in an optimization optic. It is cumbersome to build a structured grid for complex geometric shapes, and the limitations of the structured grid topology arise. It is then necessary, for a successful meshing, to split the surface in more regions, and mesh every block. Fulfillment of requirements at the common interfaces of this blocks represent a further trouble.

5.1.1 Grid Connectivity-Based Classification

The most basic form of mesh classification is based upon the connectivity of the mesh: structured or unstructured.

Structured grid

A structured mesh is characterized by regular connectivity that can be expressed as a two or three dimensional array. This restricts the element choices to quadrilaterals in 2D or hexahedra in 3D. The regularity of the connectivity allows to conserve space since neighborhood relationships are defined by the storage arrangement. Additional classification can be made upon whether the mesh is conformal or not, that is, the intersection between any two elements is a sub-element of both (a face, an edge, a

node or nothing) and the maximal dimensional shared element must be only one and complete. An example of a structured surface mesh is shown in fig.5.1(b).

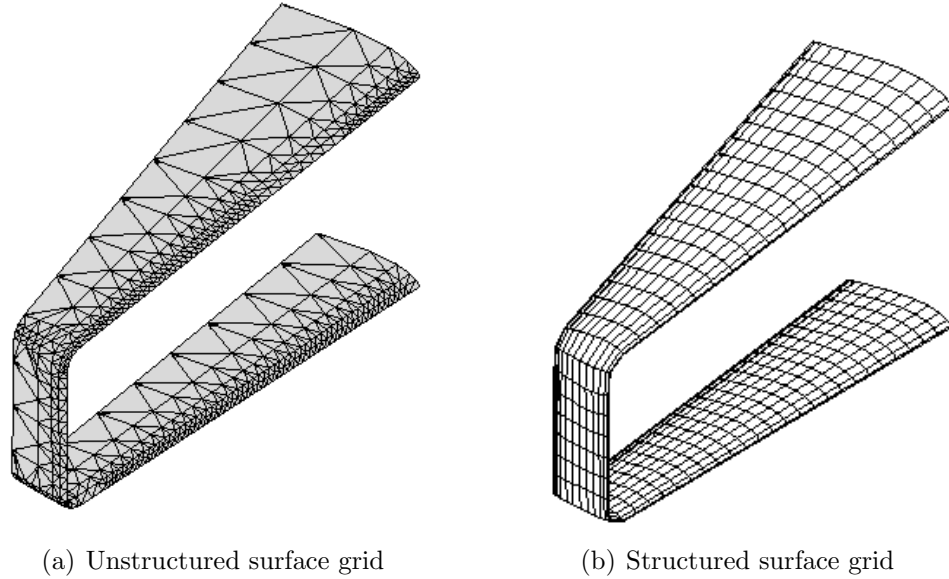


Figure 5.1: An example of two different class of surface grids.

Unstructured grid

An unstructured mesh is characterized by irregular connectivity that is not readily expressed as a two or three dimensional array in computer memory. This allows for any possible element that a solver might be able to use. Compared to structured meshes, the storage requirements for an unstructured mesh can be substantially larger since the neighborhood connectivity must be explicitly stored. An example of a unstructured surface mesh is shown in fig.5.1(a).

5.1.2 An Overview of Structured Mesh Generation

Nurbs are a very powerful representation for grid generation. First of all, modern CAD represent shapes with NURBS, thus it is an easy choice to import the surface with the same description in the grid generator module. Further, with NURBS it is

possible to maintain high level of accuracy, efficiency and numerical stability during the grid generation.

The grid generation process starts either with the specification of the boundary condition along the physical boundaries or with the distribution of grid points directly on the parametric boundaries. A point inversion algorithm give immediately the correspondence between physical boundaries points and parameter values. Then the grid is generated in the parametric space using different methods, and finally it is mapped back into physical space [34; 35]. This process is classified as algebraic

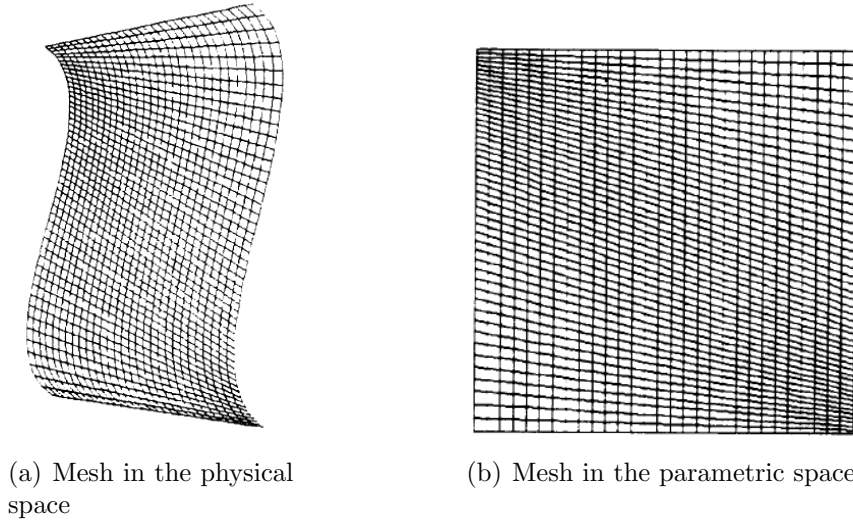


Figure 5.2: Correspondence between parametric and physical space mesh.

method; it is easy to implement, fast and efficient, but results may not be smooth, the main reason being a bad parameterization. In fact, parameterization changes the distribution of points, thus can affect the grid distribution, leading to highly skewed meshes. It is worth a note that slope discontinuities in the boundary is propagating in the inner region. The most successful smoothing schemes are based on elliptic system of partial differential equations that relate the physical and computational variables. This algorithms are time consuming, but capable of generating very smooth grids [36; 35].

To overcome topological limitations of the structured grid when handling extremely complex shapes, a *multi-block* approach can be adopted. The main drawback of

this approach is the large amount of time and effort required, since usually grid points must coincide at the common interfaces of the blocks [37].

By moving (or even adding) grid points it is possible to improve accuracy of the solution: *grid adaptation* is the technique to concentrate grid points in regions of high gradients allowing accurate results without the use of an excessively fine grid in the entire computational domain [38; 39].

5.2 The New Grid Generation Module

The new meshing module couples the ASD unstructured mesher with a new structured grid generator. After selection of the desired features to mesh, structured or unstructured mesh generator can be launched from the main GUI (fig.5.3). Obviously the selected features should contain generated surface data. In the following

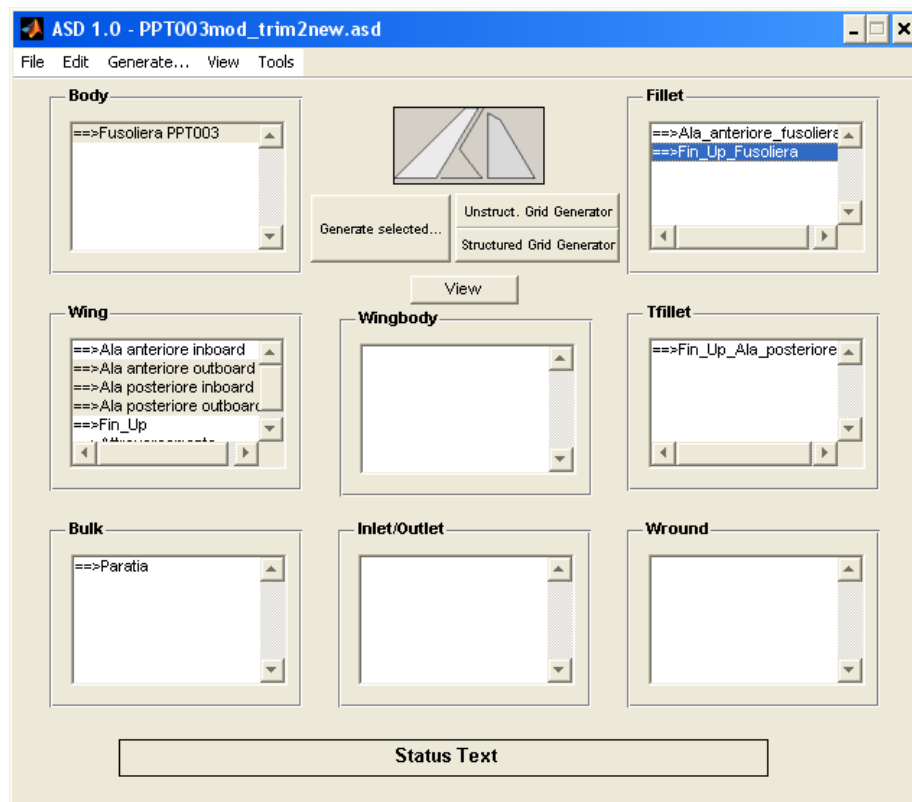


Figure 5.3: ASD main window

sections description of the structured grid generator is provided. The capabilities of the unstructured mesher have been already analyzed in section 1.3.

5.3 Requirements for the Structured Grid Generation

The grid generation capabilities are aimed to fulfil Pan Air input grid requirements, dealing with ASD surface generation particularities.

5.3.1 ASD and the Mesh

ASD represents every surface feature with NURBS, being thus naturally oriented to gridding. It is not possible, a part from simple configurations, to reduce complex geometry description to a unique NURBS due the insight topological limits of tensor product surfaces. Thus, a multi-block grid approach is needed. In practical terms, every feature generated in ASD and selected for gridding represents a separate block, and further, pierced fuselage and wing should be split in two or more simply connected region, each of this region representing a block. In fact, to obtain a proper mesh, every pierced surface should be split in simply connected sub-regions, as depicted in fig.5.4. The subdivision wouldn't represent a problem from a pure grid generation perspective, since every block is described with NURBS and thus easy to mesh; this however holds if no constraints, such as point matching condition, are imposed at the boundaries. But, even if each analysis program has different requirements for the input network, the blocks have almost always some kind of constraints on the grid points at the block boundary edges, even just for having a coherent geometric description. In the next section it will be shown how for complete aircraft configurations the required geometric flexibility will affect the process of structured grid generation, in term of price and grid quality control.

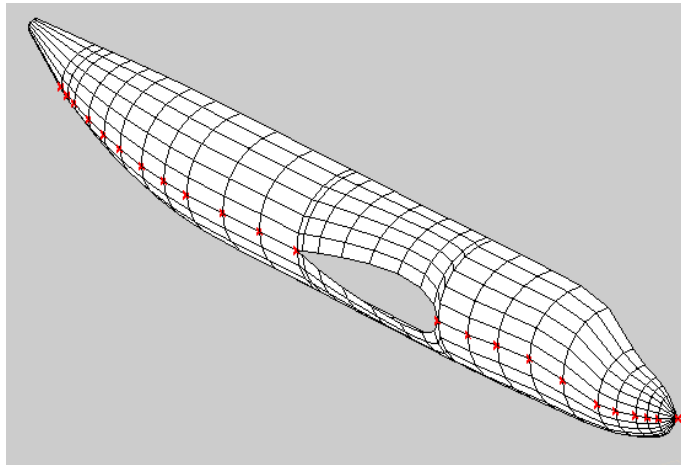


Figure 5.4: Structured mesh for a not simply connected surface: the network is split in two sub networks representing simply connected regions. The common edge points are marked in red.

5.3.2 Pan Air and the Mesh

Pan Air requirements in terms of mesh will be described in section 6.2.3. Briefly, the network edges connected along a common interface, called *abutment*, should satisfy particular conditions. In fact, the panel edge points should match along the common edge, or should fall in the straight line between the other network points. Figure 5.5 clarify abutment rules. Final result is an interface without gaps between the panel edges of the networks.

Pan Air has some internal geometric functions that look for connected networks and move the point to fill the gaps. However, it would be better if the grid generation

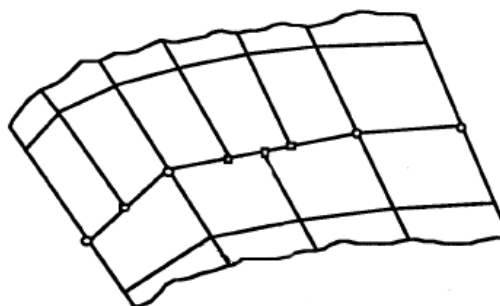


Figure 5.5: Correct distribution of points at the common edge of two networks.

process immediately takes care of this requirement. Relying on Pan Air internal functions won't be a smart strategy: just consider that these functions are not always able to recognize the proper facing networks, due also to numerical errors in the input geometry, and disasters could easily occur. A robust strategy will rely on ASD features definition, which Pan Air couldn't access, to detect the connected networks. Once mesh on the common boundary are correctly defined, the grid generation could proceed with the internal domain meshing.

Such a process leads to a generality loss since meshing of connected networks is no longer independent each from the other. However, it should be stated that many other powerful codes require the same, or at least similar, network specifications.

5.4 Logics and Mesh Organization

The grid generation procedure should reach a compromise between flexibility and simplicity. Thus, a clever underlying logic should be adopted.

5.4.1 Subdivision in Logical Subsets

The first step is a subdivision of the configuration in *logical subsets*. This process keeps to the following key points:

- a generic wing or wingbody feature, not yet owing other defined logical subsets, is considered as a starting point;
- then starting from one side of the feature, the adjacent feature is added as element of the subset and removed from the list of available features; exception are fillet features, which will be added later. If the feature has free boundaries, or the next neighbor feature is a body, the process has to be interrupted, and if not yet done, restarted from the opposite side of the starting feature; whatever the situation, every feature added to a subset should be removed from the list of available feature.

- Tfillet are then added to the subset containing the piercing wing, only if the pierced wing is not a member of the same subset. This is needed to avoid self-intersections inside a logical subset (fig.5.6).

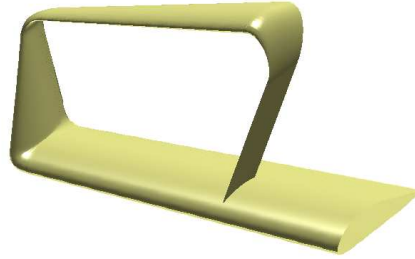


Figure 5.6: Wing pierced from tfillet of the same logical subset.

The final picture shows one or more logical subsets, connected each other only through a body or tfillet feature. An example of results of this process on a complex configuration is depicted in fig. 5.7

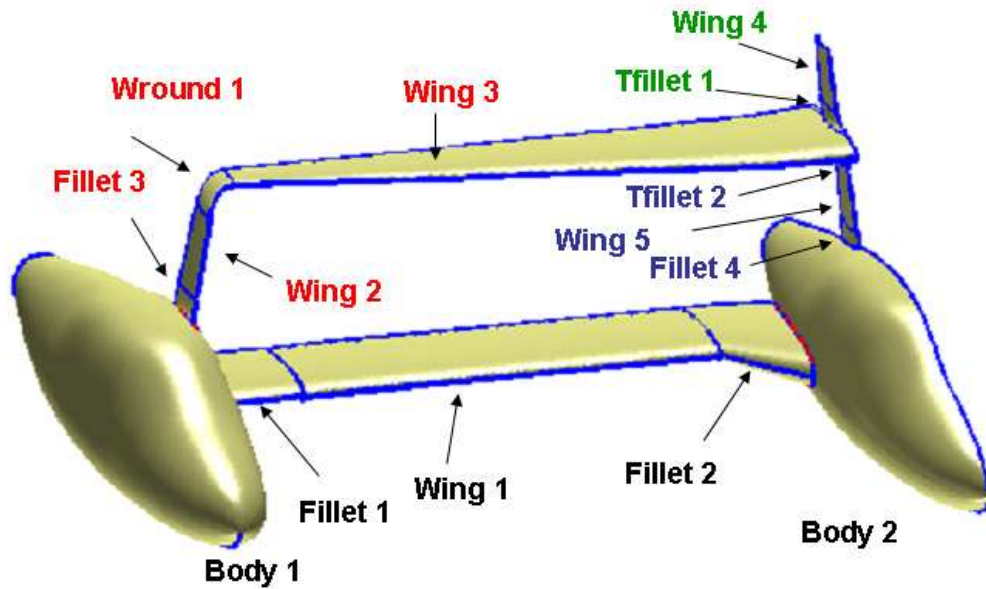


Figure 5.7: Subdivision of a complex configuration in logical subsets. The color of the name of the features identifies the different subsets elements.

5.4.2 Meshing the Logical Subsets

Next step provides a mean of selecting the features that dictate the mesh. First of all, a common nominal chordwise grid distribution is defined by the user. The choice is between a uniform and cosinusoidal chordwise distribution. The cosinusoidal should be the preferred one, since the magnitude of the gradients of flow properties are especially preeminent at leading and trailing edges, thus a finer grid is necessary there.

Next, for each logical subset a nominal chordwise number of panels is defined. This could be done in two ways: defining the number of panels in regard to the overall largest (in terms of chord) wing, or directly setting it for each subset. The logics of the first choice aims to bound the coarsest (in chordwise direction) mesh region to the prescribed level.

Also a common mean *aspect-ratio* for wing grid is defined. This parameter fixes the spanwise grid distribution in such a way that, for every panel row, the mean spanwise dimension is the product of the aspect ratio with the mean chordwise dimension (in the parameter space). In fig.5.8 it is shown how these parameters influence the grid of a logical subset composed of two wings and one wround features.

The way these grid parameters are carried throughout all the mesh is summarized in the following.

- Inside every logical subset a hierarchy between wings is defined, starting from pierced wings with larger to smaller surface plant, ending with not pierced wings, again from larger to smaller.
- Starting from the dominant wing, the mesh is developed from the boundaries to the inner surface. If the boundary vertices aren't already fixed, then the wished (cosinusoidal or uniform) chordwise distribution can be easily imposed, and then, according to the aspect-ratio parameter, the same should be done for the spanwise distribution. If for one boundary edges the vertices are already fixed (induced from an adjacent dominant wing), then, on the opposite edge, the wished chordwise vertex distribution should be built. Once defined the

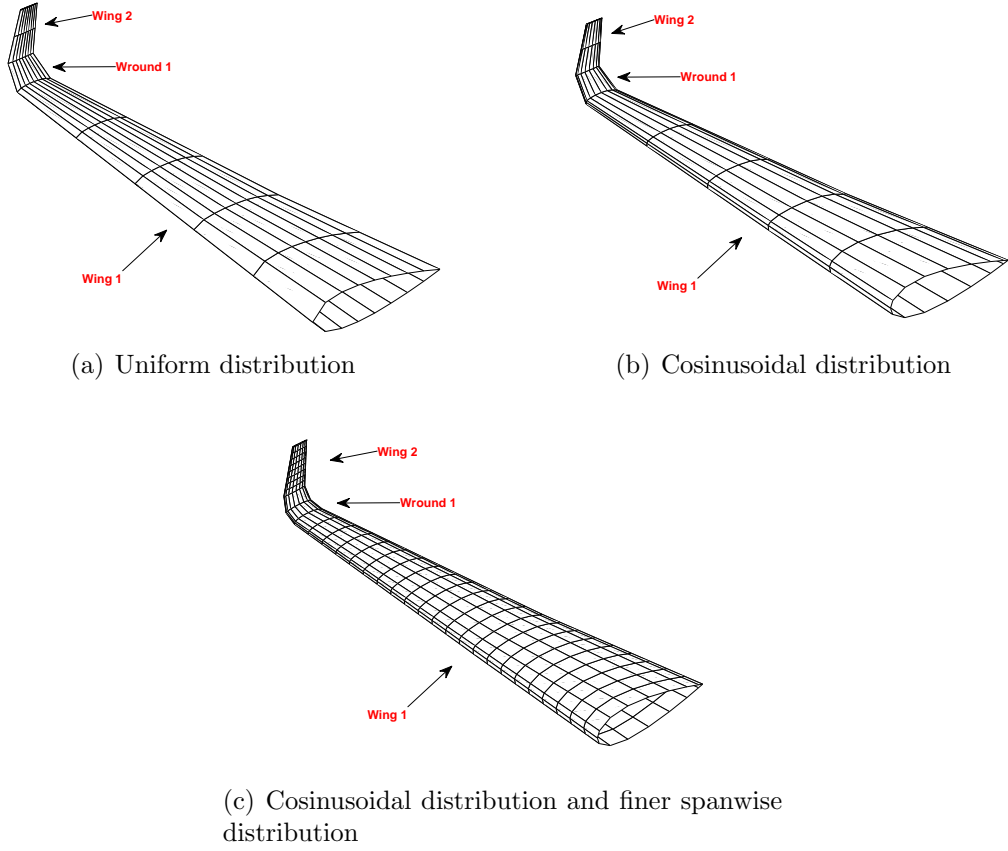
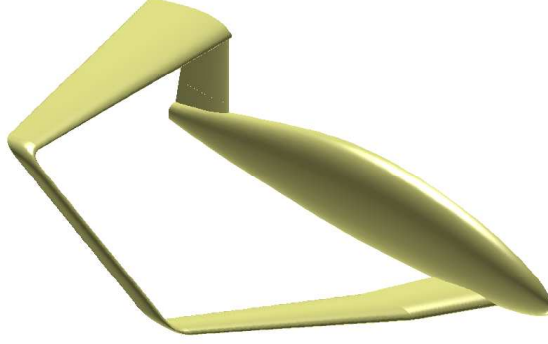


Figure 5.8: Uniform and cosinusoidal chordwise distributions for a logical subset composed of two wings and one wround feature.

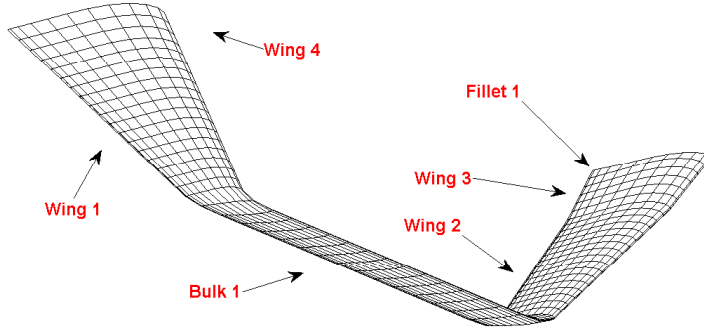
boundary grid points, the mesh is naturally built by a linear combination of the corresponding points on the boundaries.

- The grid generation on the remaining features of the subset is easily obtained in the same way. In fact, bulk and wround features connect wings, thus the grid at common edges with wings are imposed from the latter. Fillet and tfillet features have the grid at the common edge with the wing imposed from the latter, and the opposite edge free (due to the rules of definition of the subsets).

It should be pointed that the uniform or cosinusoidal distribution are easily obtained in the parameter space, and due to the proper parameterization of the representing NURBS, a reasonable consistent distribution in the real space is expected.



(a) Complete configuration to mesh



(b) One logical subset and its component features.

Figure 5.9: Process of meshing the logical subset of the complete aircraft surface.

An elucidating example is shown in fig.5.9. Here, the largest is wing1, but dominant is wing4, since it is the unique pierced wing of the subset. Thus, it is first meshed according to the parameters selected (number of subdivisions in the chordwise and spanwise directions, chordwise distribution). Second in the hierarchy comes wing1, which has a common edge with the dominant wing4. Thus, the grid point of wing1 at the common edge are imposed. To do this, a point inversion algorithm is necessary to evaluate the parameters corresponding to those points in the NURBS describing wing1. Selecting independently the opposite edge mesh points, the grid is finally generated with a linear interpolation. The process continues for all the wings, following the hierarchy. Once meshed all the wings, the bulk1 common edges with

wing1 and wing2 are fixed, and the spanwise distribution of points is determined with the intent of maintaining the same panel medium aspect ratio as that imposed for the wings.

A note ends this process; common edges of different NURBS are not always coincident along the interface. Referring to fig.5.10, the two wing surfaces are build through interpolation of a set of data, usually coincident at the common edge. But, due to the tensor product surface characteristics, the knot vector is obtained through an average process of the section points distribution. Thus, if different airfoils are used at tip and root of the same wing, it could happen that chordwise direction knot vectors of wing A and wing B are different. As a results, it is possible that the surface edges at interface doesn't completely match, leaving some small gaps. In such a situation, when trying to impose the points of wing A to wing B along the common edge, the point inversion algorithm gives back parameter corresponding to the closest points. Usually, these gaps are very small but, to avoid problems with the PanAir internal functions, it is better to fix this issue moving the respective points to be coincident.

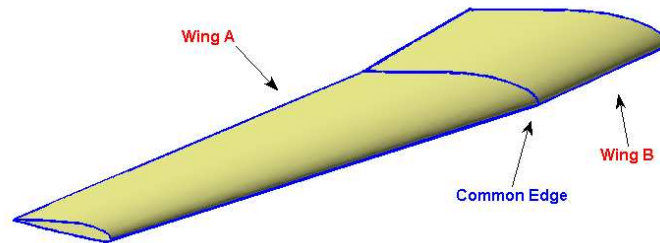


Figure 5.10: Structured mesh for a not simply connected surface: the network is split in two sub networks representing simply connected regions. The common edge points are marked in red.

5.4.3 Connections between Logical Subset Grids

Even if the grid is well developed inside every logical subset, their interfaces should still undergone some fixing process. As stated, the two possible scenario of subset connections are through fillet-body or tfillet-wing features.

With reference to fig.5.11, the intersection with tfillet and wing features is addressed in the following way:

- the grid points induced at the common edge from the tfillet and from the wing grids are compared.
- If some points are closer than a specified tolerance, they are modified to coincide.
- The not matching points belonging the tfillet grid are added to the wing grid, and used to define new corresponding points on the opposite edge, with the intent to add *induced* lines grid on the wing (in dashed red line). This new points should be adjusted in order to let them be aligned with two adjacent points of the wing3 grid.

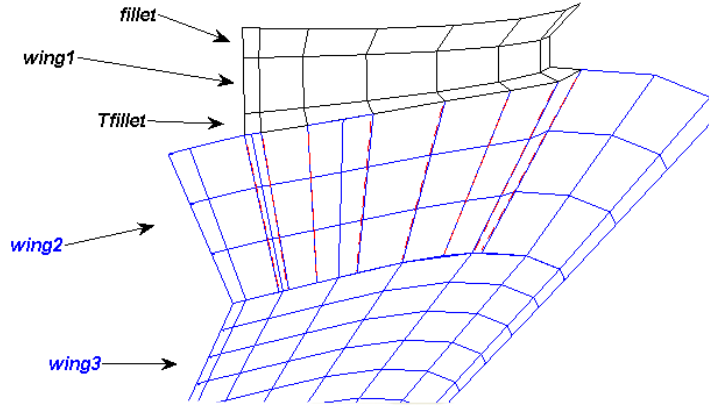


Figure 5.11: Process of connecting two logical subset (characterized by blue and black grid color) through a tfillet-wing connection. The red dashed lines represent the mesh lines induced from the tfillet.

The grid generation of body features is a little more cumbersome, since more fillet features induce grid points on the boundary edges at the same time. The process follows these steps:

- The body feature is subdivided in simply connected regions (as shown in fig.5.12)

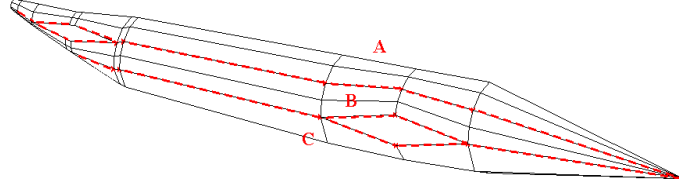


Figure 5.12: Subdivision in simply connected regions

- The natural mesh of the body feature is controlled through two main parameters. A first *deviation from straightness* parameter defines when to draw a *transversal* line mesh. Starting from the origin, the top line curve segment is compared with the length of the straight line connecting both ends. When the difference is greater than the parameter, grid points on the opposite sides of the surfaces are drawn, in order to define a line mesh. Such control helps to have a mesh that reproduces the original shape of the body (fig.5.13). The other

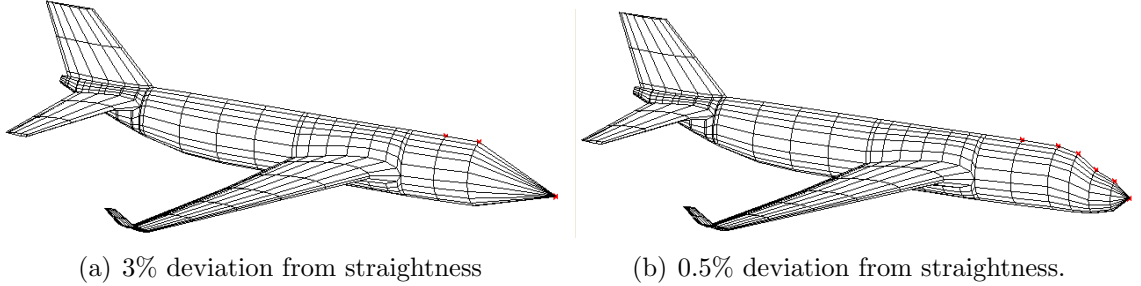


Figure 5.13: Different body mesh obtained with two different values of *deviation from straightness* parameter.

parameter is the manual frame specification, which allows the user to add one or more line mesh at the desired location, specified by cartesian coordinate x .

- The fillet and body natural mesh points at the interfaces are compared. If they are not to close, for each mesh point of the fillet is sketched a line mesh in the body, as depicted in fig.5.14. Of course, at the interfaces the points should be adjusted in order to follow the usual requirements of no gaps.

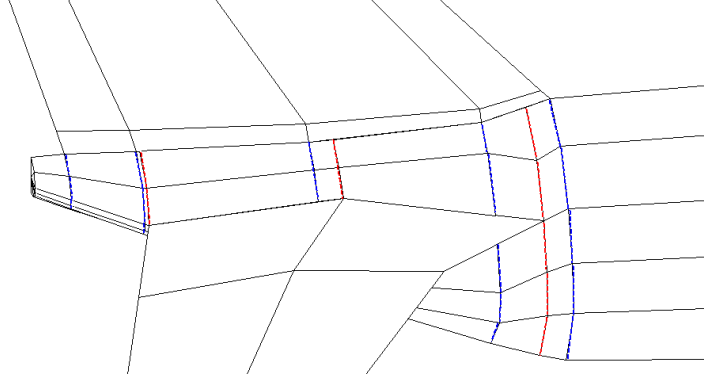
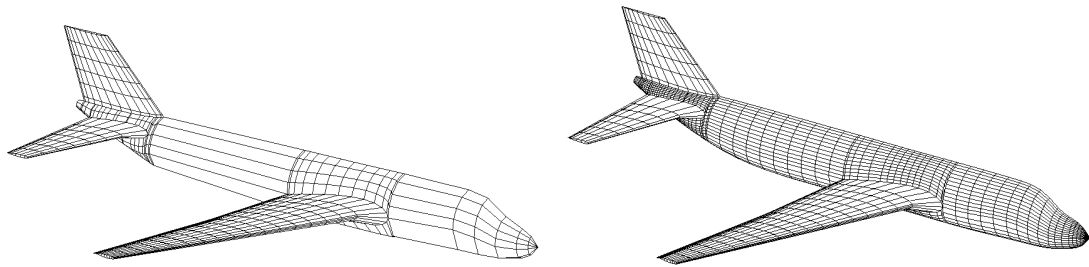


Figure 5.14: Mesh lines (in red and blue) induced from the fillet grids in the body

- This process is not optimal since induced grid lines may fall close, and thus aspect ratio of some panels could have values far from the unity: it is necessary to control this grid quality parameter. This is done by adding further horizontal or vertical mesh lines until the required maximum aspect ratio is obtained. An example of grid obtained with different values of this parameter is shown in fig.5.15. It is worth a note that it is not always possible to fulfil this requirement, thus the number of iterations (lines added) must be limited to a value. In fact, adding a line could help in some regions, and worsen in other areas.



(a) Grid with maximum body panel aspect ratio of 20 (b) Grid with maximum body panel aspect ratio of 5.

Figure 5.15: Different body mesh obtained with two different values of the body panel maximum aspect ratio parameter.

5.5 The Structure Grid Generation Interface

The main interface is depicted in fig.5.16

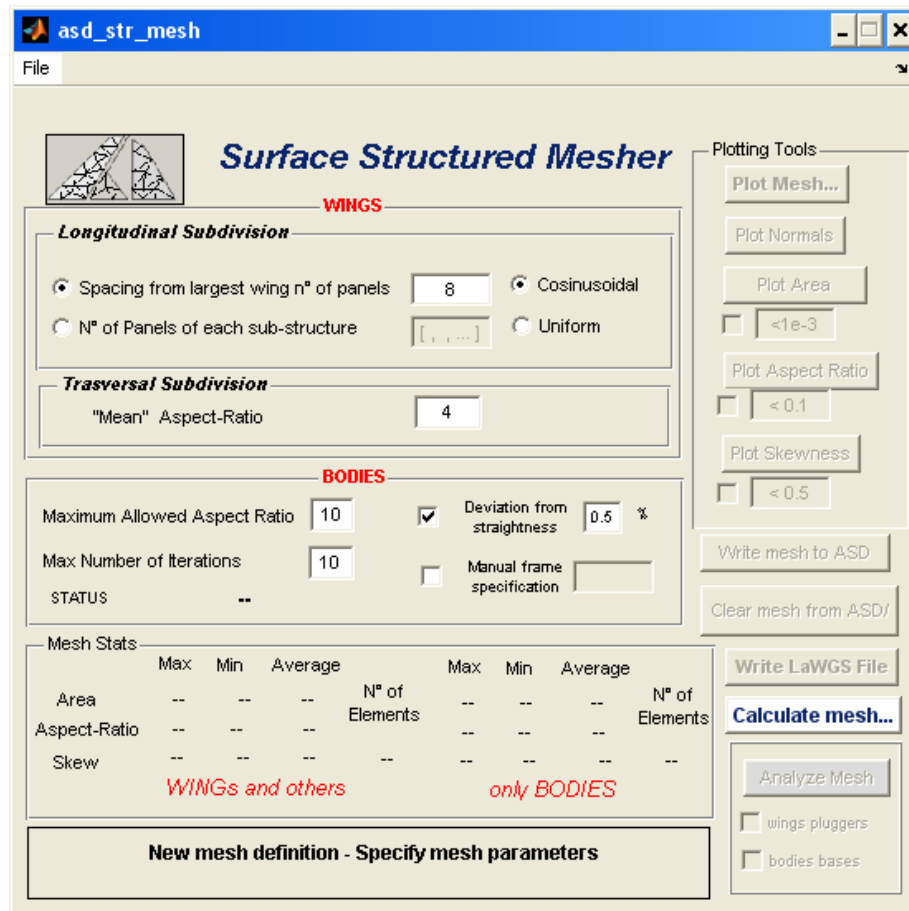


Figure 5.16: ASD main window

5.5.1 Wing and Body Feature Parameters

Mesh parameters are those described in the previous sections, both for wing and body features. To aid the user to define the chordwise number of panels for each logical subset, when selected, a view of the subsets is given.

5.5.2 Mesh Generation

When the user has defined the wished parameters, mesh is ready to be generated. The status bar gives information about the generation status. Once generated, the grid could be stored in the ASD feature fields, or could be written in an external file according to LaWGS format ([40]). The mesh is also ready to be plotted or analyzed.

5.5.3 Mesh Analysis and Stats

Mesh analysis is an instrument to check quantitatively grid quality. The results are given in the *mesh stats panel*, separately for body and not body features. The statistics include the maximum, minimum and average value of the mesh geometric parameters aspect ratio, skew, area. Also the number of panels is supplied within the statistics. The statistics could or not account for wing plugger and body bases, panels needed for closing wing and bodies edges respectively, in order to bound the internal and external region.

5.5.4 Plotting Tools

The *plotting panel* gives the opportunity of a graphic visualization of the mesh quality parameters. A filter is also applicable, in order to detect the critical area in terms of grid quality.

The tools provide also the capacity of plotting the outgoing normal vector for each panel, following the LaWGS definition ([40]). This is more a tool for developers, and should be used to check that the grid is properly described.

5.5.5 Grid Storing

If grid are stored in ASD, on each feature the field *structured mesh* is added. This field reports the grid coordinate in a three dimensional array and in the Pan Air input format. Thus, when saving an ASD session, also the grid is saved. In such situations, loading the structured grid module won't delete the mesh, until the *Clear mesh from ASD* button is triggered.

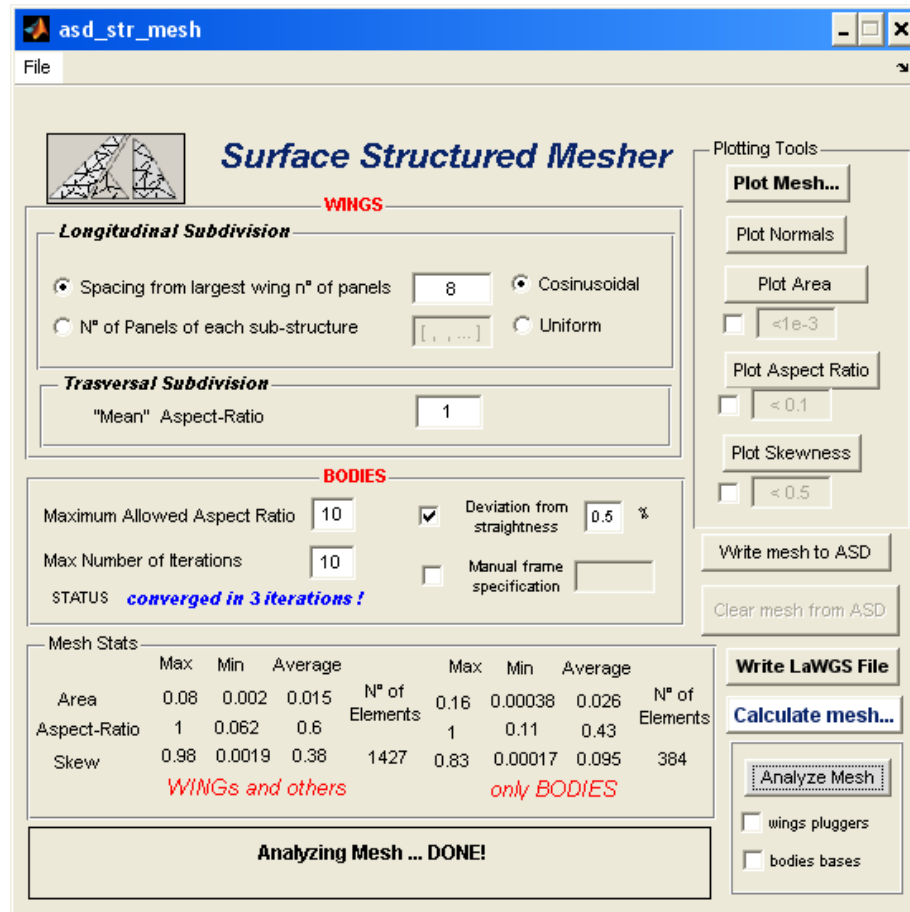


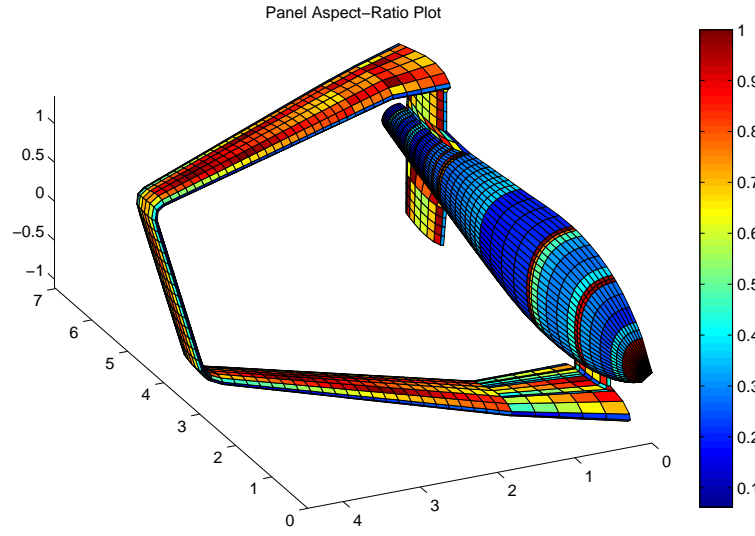
Figure 5.17: Grid Generation GUI: note the *Mesh stats* panel, reporting grid statistics

5.5.6 Pan Air Preprocessor Launcher

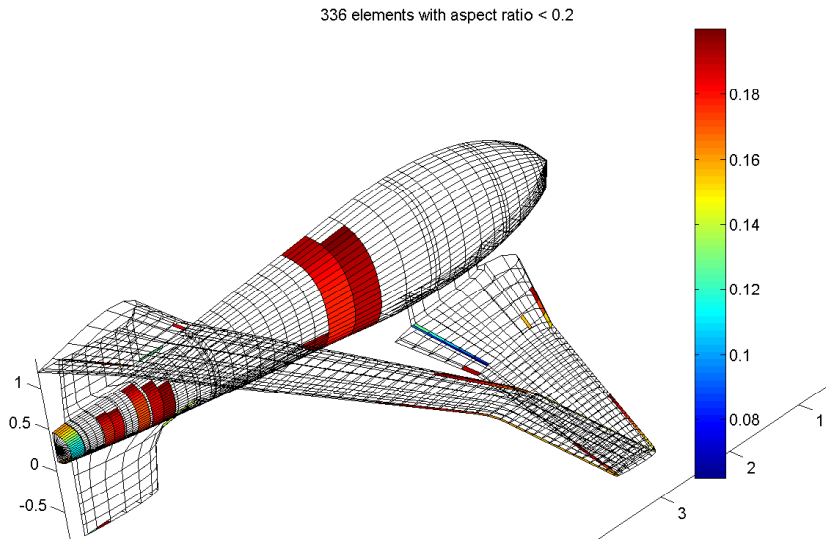
Once the mesh is saved, the Pan Air preprocessor could be launched from the menu.

5.6 Examples of Structured Grid Generated

In this section some examples of structured grid generated on conventional and unconventional aircraft configurations are reported.



(a) Aspect-ratio plot of a grid



(b) Panels with aspect-ratio smaller than 0.2

Figure 5.18: Plot of panels aspect-ratio. Note how this feature could be used to see the overall mesh parameter, or to look for mesh panels satisfying a user defined filter.

5.7 Limitations and Future Improvements

The structured grid generator is very easy to use, and the mesh are created in a few seconds. This is very important in an optimization optic. The main shortcomings

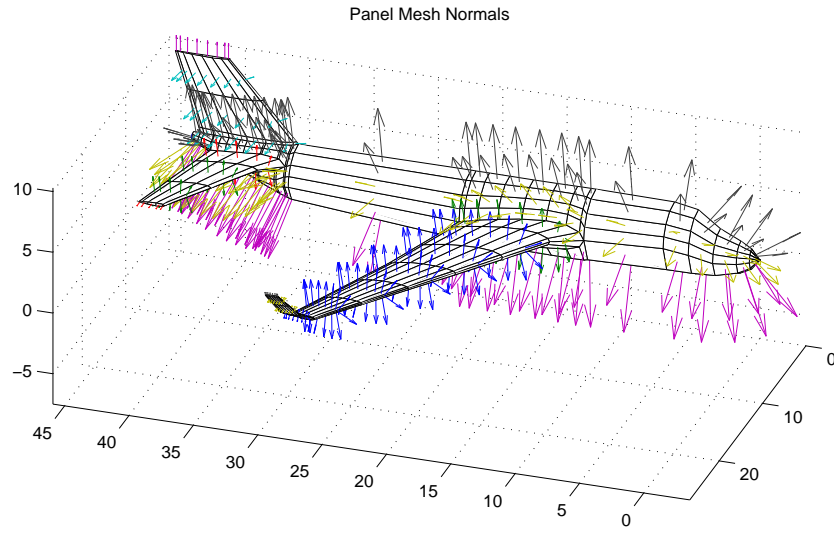


Figure 5.19: Plot of the outgoing normal vectors of the panels.

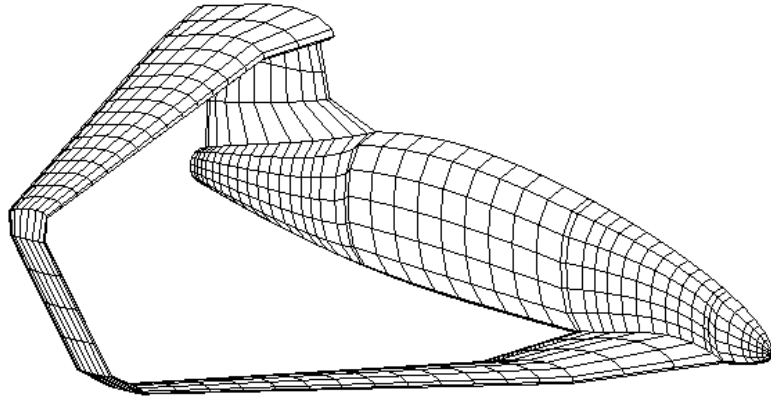


Figure 5.20: Grid of a PrandtlPlane configuration.

are about the integration of the subset meshes. In fact, some regions present a very enriched grid in order to avoid low mesh quality. An example is the aft region, where fin and tail meshes induce a grid on the body. In this region may be not of interest to have a so refined grid. It should be implemented another way to mesh this region in a more efficient way. Is important to underline that many of this problems arise

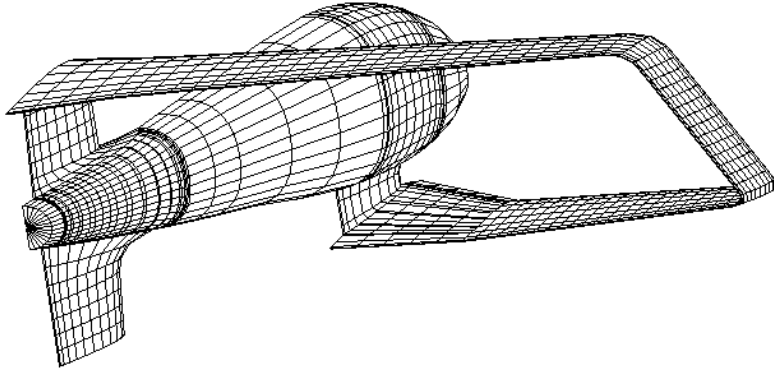


Figure 5.21: Grid of a PrandtlPlane configuration.

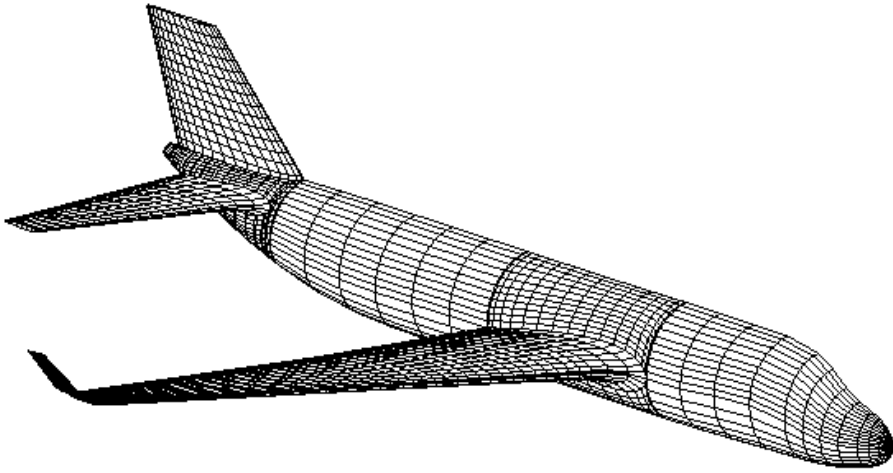


Figure 5.22: Grid of a conventional aircraft configuration.

from the topological limitation of quadrilateral structured grid.

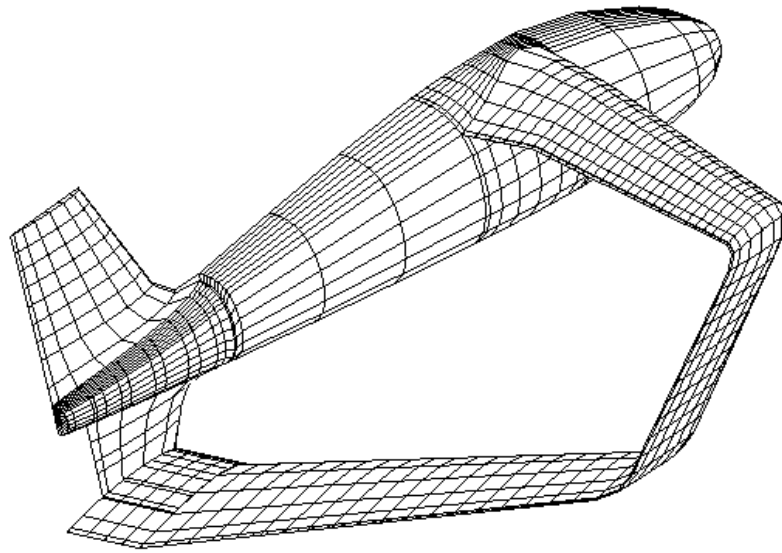


Figure 5.23: Grid of a PrandtlPlane configuration

6

Panel Method: Pan Air

Panel methods are numerical schemes which can be used for solving linear, inviscid, irrotational flow at subsonic or supersonic free-stream Mach numbers, represented by a linear partial differential equation (the Prandtl-Glauert equation). The differential equation, through a standard mathematical process, could be expressed as an integral equation over a boundary domain. This domain is then approximated with a set of panels on which unknown *singularity strengths* are defined. Imposing boundary conditions at a discrete set of points, such as panel centers, yields to a system of linear equations relating the unknown singularity strengths. The equations are then solved to obtain the singularity strengths, which, one known, provide complete information about the flow.

6.1 Panel Methods

6.1.1 The Prandtl-Glauert equation

Navier-Stokes equations

The basic equations describing the flow of a viscous compressible, heat-conducting fluid are the Navier-Stokes equations¹. These are [41; 42]:

(a) The continuity equation

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{V}) = \frac{\partial \rho}{\partial t} + \sum_{i=1}^3 \frac{\partial (\rho V_i)}{\partial x_i} = 0 \quad (6.1)$$

where $\nabla = \left(\frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2}, \frac{\partial}{\partial x_3} \right)$ is the gradient operator with respect to the location vector $\mathbf{x} = (x_1, x_2, x_3)$. Be aware that hereinafter location vector may be also expressed in the alternative notation $\mathbf{x} = (x, y, z)$ for practical purposes. In addition t is time, $\rho(\mathbf{x}, t)$ is the density, and $\mathbf{V}(\mathbf{x}, t)$ is the total velocity.

(b) The momentum equation

$$\frac{\partial}{\partial t} (\rho V_j) + \sum_{i=1}^3 \frac{\partial}{\partial x_i} (\rho V_i V_j) = -\frac{\partial p}{\partial x_j} + \sum_{i=1}^3 \frac{\partial \tau_{ji}}{\partial x_i} + \rho f_j \quad j = 1, 2, 3 \quad (6.2)$$

where τ_{ij} is the deviatoric portion of the *stress tensor* which vanishes for a frictionless fluid, $\mathbf{f}(\mathbf{x}, t)$ is an external force per unit mass exerted on the fluid, and $p(\mathbf{x}, t)$ is the pressure.

¹To be rigorous, the term *Navier Stokes equation* refers strictly to the momentum equation, being then generalized to indicate the whole set of equations

(c) The energy equation

$$\begin{aligned} \frac{\partial}{\partial t} \left(\rho e + \frac{1}{2} \rho |\mathbf{V}|^2 \right) + \sum_{i=1}^3 \frac{\partial}{\partial x_i} \left[\left(\rho e + \frac{1}{2} \rho |\mathbf{V}|^2 \right) V_i \right] = \\ \rho \sum_{i=1}^3 f_i V_i - \sum_{i=1}^3 \frac{\partial}{\partial x_i} (p V_i) + \sum_{i,m=1}^3 \frac{\partial}{\partial x_i} \left(\tau_{im} V_m + k \frac{\partial T}{\partial x_i} \right) \end{aligned} \quad (6.3)$$

where $e(\mathbf{x},t)$ is the *internal energy* of the fluid, k is the heat conductivity coefficient for the fluid, and $T(\mathbf{x},t)$ is the temperature.

(d) The equation of state

$$f(\rho, p, T) = 0 \quad (6.4)$$

where the function f depends on the type of fluid; for a perfect gas, eq.(6.4) can be written as

$$p = \rho R T \quad (6.5)$$

where R is the gas constant.

Euler's equation

The Navier-Stokes equations can be simplified by neglecting the viscosity effects, which is equivalent to setting the deviatoric stress tensor $\tau_{ij} = 0$. Combining the momentum and continuity equations leads to

$$\rho \frac{dV_j}{dt} = - \frac{\partial p}{\partial x_j} + \rho f_j \quad (6.6)$$

where the usual convective derivative operator is defined,

$$\frac{d}{dt} = \frac{\partial}{\partial t} + \sum_i V_i \frac{\partial}{\partial x_i} \quad (6.7)$$

Eq.(6.6) is called Euler's equation. The derivation of a full system of equations is obtained as follows. By means of the continuity equation, the energy equation can

be split in two parts [41], the first expressing the conservation of mechanical energy

$$\rho \frac{d}{dt} \left(\frac{1}{2} |\mathbf{V}|^2 \right) = -\mathbf{V} \cdot \nabla p + \rho \mathbf{V} \cdot \mathbf{f} \quad (6.8)$$

and the other the conservation of thermal energy

$$q = \frac{1}{\rho} \nabla \cdot (k \nabla T) = \frac{de}{dt} + p \frac{d}{dt} \left(\frac{1}{\rho} \right) \quad (6.9)$$

The steady non-linear potential flow

With further assumptions the Euler's equation can be reduced to a single equation, decoupled from the momentum equation (which transforms into the Bernoulli equation). First, assume an *isoentropic flow* so that no heat is added to the fluid

$$q = 0 \quad (6.10)$$

Then, assuming irrotationality, or $\nabla \times \mathbf{V} = 0$, implies the existence of a *potential* function $\Phi(\mathbf{x}, t)$ such that

$$\nabla \Phi = \mathbf{V} \quad (6.11)$$

If a freestream potential Φ_∞ whose gradient is the uniform velocity \mathbf{V}_∞ exist, it can be written

$$\phi = \Phi - \Phi_\infty \quad (6.12)$$

and

$$\mathbf{V} = \nabla \Phi = \nabla \Phi_\infty + \nabla \phi = \mathbf{V}_\infty + \nabla \phi \quad (6.13)$$

The quantity ϕ is called the perturbation potential, and $\mathbf{v} = \nabla \phi$ is the perturbation velocity. Assume now, without loss of generality, that the freestream is aligned in the x direction. Thus, the velocity can be expressed as

$$\begin{array}{ll} x) & U = U_\infty + u \\ y) & V = v \\ z) & W = w \end{array}$$

where u , v , w are the perturbation velocity components. Finally, refer to the so called *small perturbation*² *assumption*

$$|\mathbf{v}| \ll a_\infty \quad (6.14)$$

everywhere, where a_∞ is the freestream speed of sound.

Based on all the previous assumptions it can be obtained the unsteady potential equation (refer to [41; 42] for details). With the further assumption of steady flow all the time derivatives can be eliminated, obtaining [43] (denoting differentiation by subscript)

$$\begin{aligned} (1 - M_\infty^2) \phi_{xx} + \phi_{yy} + \phi_{zz} = M_\infty^2 \left[\frac{1}{2}(\gamma - 1) (2u + |\mathbf{v}|^2) \nabla^2 \phi \right. \\ \left. + (2u + u^2) \phi_{xx} + v^2 \phi_{yy} + 2vw \phi_{yz} + w^2 \phi_{zz} + 2(1 + u) (v \phi_{xy} + w \phi_{xz}) \right] \end{aligned} \quad (6.15)$$

where $\gamma = \frac{C_p}{C_v}$ is the ratio of the specific heats.

The Prandtl-Glauert equation

The linearization of equation (6.15) leads to the Prandtl-Glauert equation (see appendix A of [42]):

$$(1 - M_\infty^2) \phi_{xx} + \phi_{yy} + \phi_{zz} = 0 \quad (6.16)$$

The linearization is well advised if:

$$M_\infty^2 |\mathbf{v}|^2 \ll |1 - M_\infty^2| \quad (6.17)$$

and

$$M_\infty^2 |\mathbf{v}|^2 \ll 1 \quad (6.18)$$

²other small perturbation assumptions exist in the literature

Like as eq.(6.14) also eq.(6.17) and eq.(6.18) are called small perturbation assumptions. The consequences of the above simplifications are discussed in section 6.1.3.

6.1.2 Panel Method Theory

This section will outline the process by which the Prandtl-Glauert equation is converted to an integral equation, and the way in which a general panel method solves that integral equation.

Coordinate scaling

Eq. (6.16) can be further simplified by performing a scaling of the coordinate system. Defining the compressibility scale factor β by

$$a = \frac{(1 - M_\infty^2)}{|1 - M_\infty^2|} \quad \beta = \sqrt{|(1 - M_\infty^2)|} \quad (6.19)$$

the required scaled coordinate are given by

$$\begin{cases} \tilde{x} = x \\ \tilde{y} = \beta y \\ \tilde{z} = \beta z \end{cases} \quad (6.20)$$

Thus, the transformed Prandtl-Glauert equation is

$$a \phi_{\tilde{x}\tilde{x}} + \phi_{\tilde{y}\tilde{y}} + \phi_{\tilde{z}\tilde{z}} = 0 \quad (6.21)$$

where $a = 1$ for subsonic flows (Laplace's elliptic equation) and $a = -1$ for supersonic flows (hyperbolic equation). Both equations occurs also in other branches of physics.

Boundary conditions

The physical description of a real flow at a surface is given by the no-slip condition

$$\mathbf{V} = 0 \quad (6.22)$$

However, for inviscid flow, the tangential component of the velocity cannot be prescribed (unless the pressure is known) thus the above is replaced with the impermeability boundary condition, representing the inability of fluid to pass through solid surfaces

$$\mathbf{V} \cdot \mathbf{n} = 0 \quad (6.23)$$

or

$$\nabla\phi \cdot \mathbf{n} = -\mathbf{V}_\infty \cdot \mathbf{n} \quad (6.24)$$

There is also an alternative formulation of the above boundary condition; it can be shown ([44]) that neglecting terms of the same order as those neglected in reducing equation (6.15) to the Prandtl-Glauert equation, it holds:

$$\rho\mathbf{V} = \rho_\infty\mathbf{W} \simeq \rho_\infty(\tilde{\nabla}\phi + \mathbf{V}_\infty) \quad (6.25)$$

where $\mathbf{W} = \frac{\rho}{\rho_\infty}\mathbf{V}$ is called the *mass flux*, $\tilde{\nabla}\phi + \mathbf{V}_\infty$ is called the total linearized mass flux, and $\tilde{\nabla}\phi$ (also denoted as \mathbf{w}) is called the linearized perturbation mass flux. Thus the boundary condition becomes:

$$\tilde{\nabla}\phi \cdot \mathbf{n} = -\mathbf{V}_\infty \cdot \mathbf{n} \quad (6.26)$$

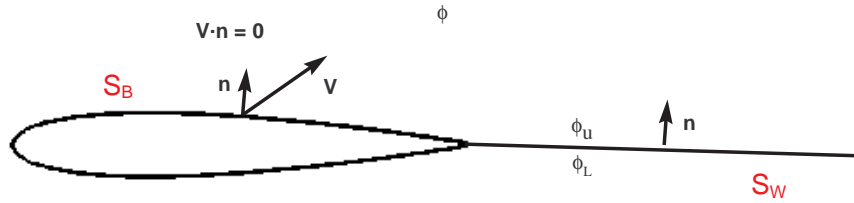


Figure 6.1: Body S_B and wake S_W surfaces.

Panel method may impose different boundary conditions with the aim of modeling the particular physical problem. Wakes, separation area, like body bases, or flow through surfaces, like fan faces, are also modeled. Note that, in the latter situation, the surface where to impose the boundary conditions doesn't represent a physical surface.

Integral formulation for non lifting body

Assume a fluid region denoted by \mathcal{V} , internally bounded from the body surface \mathcal{S}_B , and externally bounded from the surface \mathcal{S}_R , and remember that the fluid is not viscous and initially irrotational. With the aid of the Green's third identity and the Gauss theorem, and moving the external contour surface \mathcal{S}_R to the infinity, the Laplace equation admits the following integral representation [45; 46; 47]:

$$E(\mathbf{P})\phi(\mathbf{P}) = -\frac{1}{4\pi} \int_{\mathcal{S}_B} \left(\frac{\mathbf{n} \cdot \nabla \phi(\mathbf{Q})}{R} - \phi(\mathbf{Q}) \mathbf{n} \cdot \nabla \frac{1}{R} \right) dS \quad (6.27)$$

where \mathbf{P} and \mathbf{Q} are points such that $\mathbf{P} \in \mathcal{V}$ and $\mathbf{Q} \in \mathcal{S}_B$, $R = \|\mathbf{P} - \mathbf{Q}\|$, \mathbf{n} is the unit normal to the surface, assumed positive if pointing toward the region \mathcal{V} , and

$$E(\mathbf{P}) = \begin{cases} 1 & \text{if } \mathbf{P} \in \mathcal{V} \\ \frac{1}{2} & \text{if } \mathbf{P} \in \mathcal{S}_B \end{cases} \quad (6.28)$$

Once determined the values of $\nabla \phi$ and ϕ on the boundary \mathcal{S}_B , the perturbation potential can be evaluated for all the points inside \mathcal{V} . Eq.(6.27) is the fundamental integral representation formula which a panel method uses to obtain a solution to the potential flow problem. When combined with appropriate boundary conditions it can be manipulated to yield an integral equation on the singularity surface \mathcal{S}_B . A panel method then obtains an approximate solution of this integral equation by means of the numerical method of collocation.

Two function defined on \mathcal{S}_B are generally introduced because of their importance in the manipulation of eq.(6.27). The first is the *source strength* defined by

$$\sigma(\mathbf{Q}) = \mathbf{n} \cdot \nabla \phi(\mathbf{Q}) \quad (6.29)$$

and the second is the *doublet strength*, defined by

$$\mu(\mathbf{Q}) = \phi(\mathbf{Q}) \quad (6.30)$$

These quantities are called *singularity strengths* since they measure the singular behavior of ϕ on \mathcal{S}_B ; using these quantities eq.(6.27) for points in the inner region ($E = 1$), becomes:

$$\phi(\mathbf{P}) = -\frac{1}{4\pi} \int_{\mathcal{S}_B} \left[\frac{\sigma}{R} - \mu \mathbf{n} \cdot \nabla \frac{1}{R} \right] d\mathcal{S} \quad (6.31)$$

The generalization to arbitrary Mach number is easily achieved, as follows. For subsonic flows is enough to use the scaled coordinates to obtain a formally identical solution, being immediately achieved the expression of the above equations in terms of original coordinates. In fact, defined the compressible gradient operator as $\tilde{\nabla} = (a\beta^2 \frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z})$, eq.(6.31) turns into

$$\phi(\mathbf{P}) = -\frac{1}{4\pi} \int_{\mathcal{S}_B} \left[\frac{\sigma}{R} - \mu \mathbf{n} \cdot \tilde{\nabla} \frac{1}{R} \right] d\mathcal{S} \quad (6.32)$$

where now every term is expressed in the original coordinate system, and where

$$R = \sqrt{(x_P - x_Q)^2 + a\beta^2(y_P - y_Q)^2 + a\beta^2(z_P - z_Q)^2}$$

$$\sigma(\mathbf{Q}) = \mathbf{n} \cdot \tilde{\nabla} \phi(\mathbf{Q})$$

For supersonic flows ($M_\infty > 1$) the above equations should be adapted substituting 4π with 2π , and extending the integral to just the portion of \mathcal{S}_B that lies in the upstream Mach cone emanating from the influenced point \mathbf{P} (\mathcal{D}_P) [46; 47]

$$\phi(\mathbf{P}) = -\frac{1}{2\pi} \int_{\mathcal{S}_B \cap \mathcal{D}_P} \left[\frac{\sigma}{R} - \mu \mathbf{n} \cdot \tilde{\nabla} \frac{1}{R} \right] d\mathcal{S} \quad (6.33)$$

Once ϕ is determined on \mathcal{S}_B , is possible to evaluate the pressure everywhere in the field by means of Bernoulli equation.

Integral formulation for lifting body

Wake should be taken into account. There are a few aspects related with wakes.

First, if no wake would exist, the singularity surface (\mathcal{S}_B) is closed and bounded, thus, for D'Alembert's paradox [48; 49] the overall load on \mathcal{S}_B is null (this is true

only for steady problems).

Another notice is about the pertinence of considering the flow irrotational in all the volume \mathcal{V} . In fact, the irrotationality condition holds true only in the inner region of \mathcal{V} , while for the material points which have been get in contact with the body surface Kelvin's theorem is not valid [48; 49]; the region made up by these fluid particles is called *potential wake*, \mathcal{S}_W , and the potential is not defined on it. The vorticity generated through the dynamic interaction between fluid and body surface is thus convected through the trailing edge in the field, and forms the wake surface. This vorticity, besides providing a more close modeling of the physics, the more closer as well as Reynolds number increases [50], doesn't contradict the hypothesis of potential fluid: the vorticity can be confined to a layer with zero thickness considered as part of the boundary.

The process of obtaining the integral formulation still holds if the inner boundary is a surface surrounding the body and the wake, and approaching the union of them, or briefly, $\mathcal{S}_{\mathcal{BW}} = \mathcal{S}_B + \mathcal{S}_W$ [48]. The boundary conditions to be applied on the wake are

$$\Delta(\nabla\phi \cdot \mathbf{n}) = 0 \quad (6.34)$$

and

$$\frac{d\Delta\phi}{dt} = 0 \quad (6.35)$$

where Δ stays for the difference between the absolute value of a generic quantity on the two faces of \mathcal{S}_W , and $\frac{d}{dt}$ is the material derivative. Equation (6.35) states that the jump of the potential function is constant following the wake material point.

The potential equation reduces to:

$$E(\mathbf{P})\phi(\mathbf{P}) = -\frac{1}{4\pi} \int_{\mathcal{S}_B} \left(\frac{\mathbf{n} \cdot \nabla\phi(\mathbf{Q})}{R} - \phi(\mathbf{Q}) \mathbf{n} \cdot \nabla \frac{1}{R} \right) d\mathcal{S} - \frac{1}{4\pi} \int_{\mathcal{S}_W} \Delta\phi \mathbf{n} \cdot \nabla \frac{1}{R} d\mathcal{S} \quad (6.36)$$

and the normal surface vector on the wake is defined from side L to side U in agreement with $\Delta\phi = \phi_U - \phi_L$.

The condition to impose at the trailing edge, where body (wing) and wake connect

each other, is

$$(\phi_U - \phi_L) \Big|_{T.E.} = \Delta\phi \Big|_W \quad (6.37)$$

where $(\phi_U - \phi_L)$ in the first member is referred to the trailing edge of the body surface. The physical motivation beyond the last mathematical condition relies on the Joukowsky condition, which stated that no concentrated vortex exists at the trailing edge. More details are reported in [48; 50; 49]

Discretization

The first part of discretization process consists of the development of a finite dimensional representation of σ and μ . The singularity surface \mathcal{S}_B and \mathcal{S}_W are approximated by a collection of N and M panels respectively. Next a collection of N points on the body panels are chosen, such as panel centers, edges or corners³. The values of σ and μ at these points are identified as unknowns and are called the singularity parameters, and are χ_i, ϕ_i for the body panels, $\Delta\phi_k$ for the wake panels. In Pan Air, approximate distributions $\sigma(\mathbf{Q})$ and $\mu(\mathbf{Q})$ are then developed applying a combination of linear least squares fitting techniques and polynomial interpolation processes to extend the discrete values of the singularity parameters to all the points on the surface. The following representation for σ and μ is then obtained:

$$\sigma(\mathbf{Q}) = \sum_{i=1}^N \chi_i s_i(\mathbf{Q}) \quad \mu(\mathbf{Q}) = \sum_{i=1}^N \phi_i m_i(\mathbf{Q}) \quad (6.38)$$

for the body, and

$$\mu(\mathbf{Q}) = \sum_{k=1}^M \Delta\phi_k m_k(\mathbf{Q}) \quad (6.39)$$

for the wake, where the functions s_i and m_i are called the source and doublet basis functions. Next, using eq.(6.31) with the specified functional form of σ and μ , it is

³it should be pointed that the control points are close but not exactly lying on the panel edges and corners, in order to avoid singular behaviors.

possible to express the perturbation potential in terms of singularity parameters

$$E(\mathbf{P}) \phi(\mathbf{P}) = -\frac{1}{4\pi} \left(\sum_{i=1}^N \left(\chi_i \int_{\mathcal{S}_B} \left(\frac{s_i}{R} \right) dS - \phi_i \int_{\mathcal{S}_B} m_i \mathbf{n} \cdot \nabla \frac{1}{R} dS \right) \right) + \frac{1}{4\pi} \left(\sum_{k=1}^M \left(\Delta\phi_k \int_{\mathcal{S}_W} m_k \mathbf{n} \cdot \nabla \frac{1}{R} dS \right) \right) \quad (6.40)$$

Finally,

- applying eqs.(6.35) and (6.37) in order to express the $\Delta\phi_k$ in terms of the ϕ_i ,
- recalling the source expression (eq.(6.29)),
- imposing the N linear boundary conditions, $\nabla\phi \cdot \mathbf{n} = -\mathbf{V}_\infty \cdot \mathbf{n}$ on the points of the body panels

yields a linear system of N unknowns in N equations. In matrix form:

$$[\text{AIC}]\{\phi\} = \{b\} \quad (6.41)$$

where $[\text{AIC}]$ is called the matrix of aerodynamic influence coefficients, $\{\phi\}$ is the vector of the unknown perturbation potential at the control points, and the elements of $\{b\}$ are known from boundary conditions. Solving the linear system makes it possible to compute from the eq.(6.31) the potential and subsequently the velocity field for each point of the region \mathcal{V} .

6.1.3 Limits of Application of Panel Method

All the simplifications done to obtain the Prandtl-Glauert equation, restrict the field of application, since the underlying equation is not longer able to model the main aspects of the physical problem. Thus Prandtl-Glauert (eq.(6.16)) is not able to model flows where viscous, heat and rotational behavior are not negligible. Finally, all the small perturbation assumptions have restricted further the modeling capability. Equations (6.17) and (6.18) should be carefully considered. Clearly *transonic*

flow ($M_\infty \simeq 1$) and *hypersonic* flow ($M_\infty \gg 1$) are not consistent with the aforementioned assumptions, thus Prandtl-Glauert equation can not described such flows. But, another restriction is set from the magnitude of the perturbation quantity \mathbf{v} : for high angles of attack, or thick configuration, the perturbation velocity tend to be large, and thus eq.(6.17),(6.18) hold for a narrow range of Mach number.

As known, flows around aerodynamic (at low angles of attack) bodies present only a limited region where viscosity and rotational effects are not negligible. Thus, for a correct modeling is necessary to assume that those regions have negligible effects on the overall potential flow. Of course in region of separated flow, as the aft fuselage region, results given from a panel method are not reliable; however, this wouldn't mean that the overall prediction, like configuration lift and induced drag are not realistic. Another problem is related with wake positioning. In fact, wakes are generally inserted in a roughly streamwise direction emanating from the trailing edge of all *lifting* surfaces such as wings, fins, However, several major exceptions exist. One is the case in which wake passes near other lifting surfaces (like tail of the airplane): the wake influence on this surfaces is significant. Another is the case of the *leading edge vortex*, a phenomenon that occurs at the leading edge of a highly swept wing at large angles of attack. The wake tends to roll up, and its exact location is important in determining the aerodynamic behavior of the configuration. However, some panel codes are able to iteratively determine wake position and shape, improving thus results of simulations at expense of computational time ([49]).

Other panel method codes are capable of resolving unsteady potential flows. The flexibility of these codes can be successfully used to solve complex problems like helicopter rotor flux ([49], [51]).

Finally, codes like *Tran Air* are capable of simulating transonic flows through numerical resolution of the non-linear potential flow equation ([52]).

6.2 Pan Air

Pan Air (acronym of Panel Aerodynamics) is generally considered to be the first actual surface panel code with reliable numerics, even for supersonic flows. Relative

insensitivity and stability of computed results to paneling was a key for its success. In addition, the boundary condition flexibility allowed users to experiment with various types of modeling, leading to a wide variety of applications never entirely envisioned by the developers. Thus, within the limitations of linear potential flow theory, it could be used as an *analytical wind tunnel* for the analysis of completely arbitrary configurations. The code was first developed in the early 1970s at NASA and Boeing (where it was known as A502 code). Later, many features were added. The code was successfully used in the project of big transport airplanes (Boeing 737, 707, 747) and military fighters. Sometimes it was used in conjunction with A598 (a Boeing code for boundary layer analysis), even in fields like yacht design.

Today, A502 is still used to provide quick estimates for preliminary design studies. A relative new feature takes advantage of available linear sensitivities to predict a large number of perturbation to stability and control characteristics and stability derivatives, including control surface sensitivities. A typical application may involve many subcases submitted in a single run, with solutions available in an hour or so. Within the limitations of the code, all major stability and control derivatives can be generated in a single run (at a single Mach). The method is typically used to calculate increments between similar configurations. As an example, the code was recently used to calculate stability and control increments between a known baseline and a new configuration. A total of 2400 characteristics were computed for eight configurations by one engineer in a two-day period.

6.2.1 Pan Air Capabilities

Pan Air can handle the simple configurations considered in preliminary design, and at the same time serve as an *analytical wind tunnel* for the analysis of flow about detailed, complex configurations. Capabilities of Pan Air version 3.0 (which is the version integrated with the code subject of the thesis) include:

- the ability to handle, within limitations of linear potential flow theory, completely arbitrary configurations, using either exact or linearized boundary conditions;

- the ability to handle asymmetric configurations as well as those with one or two planes of symmetry;
- the ability to handle symmetric configurations in either symmetric or asymmetric flow;
- the ability to superimpose an incremental velocity on the freestream, either locally or globally, in order to simulate effects such as a rotational motion, differing angles of attack for different portions of a configuration, or a propeller slipstream;
- the ability to calculate pressures, forces and moments using a variety of pressure formulas (such as isoentropic, linear . . .), including the forces and moments due to momentum flux through the surface;
- the ability to calculate leading edge and side edge thrust forces and moments for thin configurations;
- the ability to perform non-iterative design of a configuration, a process in which a desired pressure or tangential velocity distribution is specified. The program then determines the *residual* normal flow through the surface required to obtain the desired pressure distribution;
- the ability to calculate streamlines and to evaluate flow properties at user specified off body points.

6.2.2 Pan Air Technology

Reliable supersonic analysis needs careful modeling, since numerical stability problems arises if doublet distribution is not continuous at panel interfaces. In fact, a spurious line vortex of strength $\Delta\mu$ is produced from doublet value jump at edges. Thus a velocity field is introduced from discretization. In subsonic flow these spurious velocities decay rapidly with distance from the edge and usually do not cause serious problems. In supersonic flows these velocities persist, their effect propagating down to Mach cones. Consequently, erroneous incremental flows continue to exist at

control points, thereby introducing errors in the aerodynamic influence coefficients matrix. These errors are frequently serious enough to produce a totally incorrect solution for the flow. As a first requirement, panels should then meet exactly each other at edges. Then, the doublet basis functions couldn't obviously be piecewise constant on each panel. Pan Air employs a linear source variation and quadratic doublet variation. Detailed explanation of why piecewise linear for source and piecewise quadratic for doublets is presented in [42].

For more details about Pan Air refer to [53],[42],[54],[55].

6.2.3 Pan Air Geometry Input

Network description should be in LaWGS format. For details refer to [40]. Briefly, assume that a network is composed of $N \times M$ grid points, where N and M represent respectively columns and rows. If an order is established for the columns (or rows) description (from first to last or vice versa), then the order of description of the rows (columns) should follow a right hand convention coherent with the outward pointing normal to the surface, that is the order of representation defines which side of the surface has to be considered external (and thus facing the flow). This is a key point in order to set boundary condition consistently with the real flow. Worth a note that, since the code accepts only structured grid, for each network the number of elements for each rows and columns is constant; if more flexibility is needed for a satisfying geometric reproduction, then the surface must be split in more networks.

Particular attention may be devoted to network edges connected along a common interface. Such connection are called *abutment*. As stated, edges abutments ensures continuity of the doublet strengths across network edges. The program requires that abutting network edges must have exact panel edge points which match along the network edge, or panel edge points which are on the straight line between the exact points. Thus, the interfaces of two or more doublet network surfaces must match, i.e. have no gaps between adjacent networks. To meet this requirements one or a combination of the following may be ensured:

- Input geometry has exact matching of every panel edge point along abutting network edges.

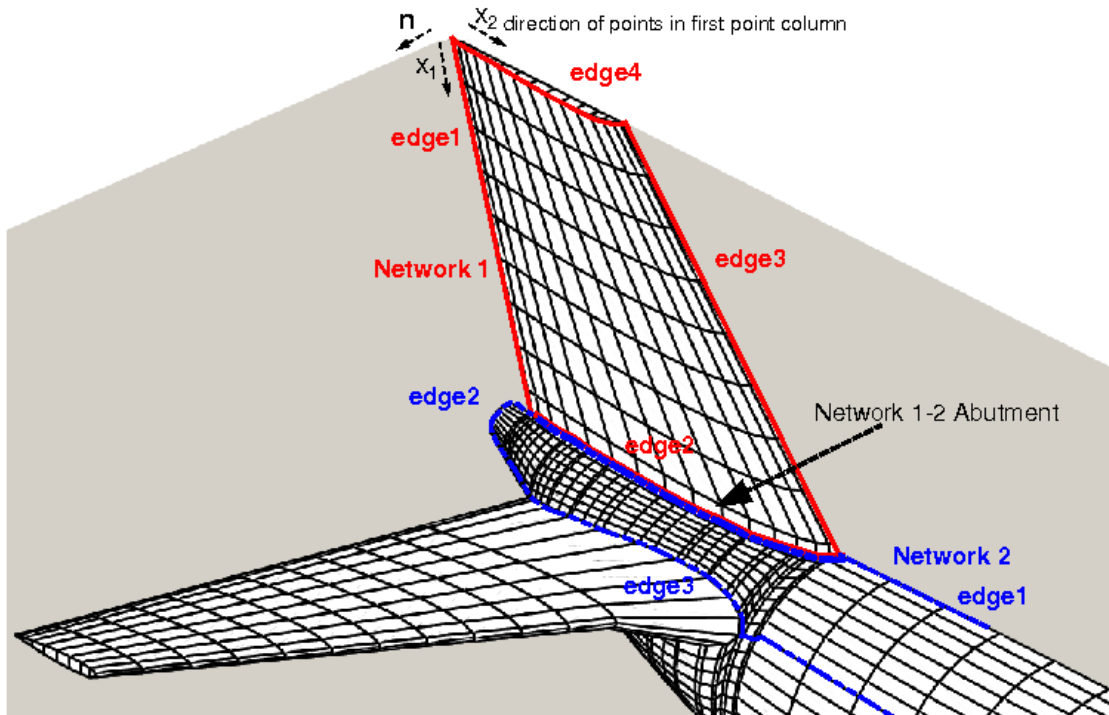


Figure 6.2: Abutment between network 1 and 2. Note the panels with matching and not matching points at abutment. Note also the order of description of rows and columns, consistent with a correct outward surface normal

- Input geometry nearly matches for every panel edge point along abutting network edges, and the liberalized abutment capability (an internal function of Pan Air) makes the abutment identical for points within a single tolerance. Small adjustments are made to the network edge points to make them abut without any gaps and to help them eliminate the small round-off error in the input network geometry.
- Input geometry contains some mismatched points along abutting network edges. These edges must be identified by the user for special treatment. The partial (full) network edge abutment (another internal function of Pan Air) has the capability to form a new common edge from matching points along a network. All non matching points are projected onto the new network edge.

Note that the latter two situations modify the original input geometry along abutting

network edges. Attention should be paid to set the tolerance values, since extraneous abutment could be created by the internal Pan Air functions assisting abutments.

6.2.4 An overview of configurations analyzed with Pan Air

In the following section some application of Pan Air are reported. Fig. from 6.3 to 6.5 shows Pan Air application for big transport aircrafts. One of the first important employ of Pan Air goes back at the precertification flight testing of the then new 737-300 [56]. The aircraft was not demonstrating the preflight wind tunnel based prediction of take-off lift/drag ratio. A fix was needed quickly to meet certification and delivery schedules. Specialized flight testing was undertaken to find the cause and to fix the performance shortfall. A Pan Air study was immediately undertaken to enhance understanding and provide guidance to the flight program. Eighteen complete configuration analysis were carried out over a period of three months (see fig.6.4). These included different flap settings, wind tunnel and flight wing twist, flow through and powered nacelle simulations, free air and wind tunnel walls, ground effect, seal and slotted flaps, and other geometric variations. These solutions explained and clarified the limitations of previous low speed wind tunnel test techniques and provided guidance in recovering the performance shortfall through *tuning* of the flap settings during the flight testing. The aircraft was certified and delivered on schedule. A comparison of the computation L/D predictions with flight is shown in fig.6.4(d).

A502 studies have been used to support other flight programs on a time critical basis. In particular, the code was used to support engine-airframe installation studies in the early 1980s, to evaluate wind tunnel tare and interference effects, and to provide Mach blockage corrections for testing large models. In addition, the code was used for the design of the wingtip pod for the Navy E6-A, a version of the Boeing 707. No wind tunnel testing was done before flight. The FAA has accepted A502 analysis for certification of certain aircraft features that were shown to have minimal change from previous accepted standards. Finally, A502 was used to develop a skin waviness criteria and measurement technique that led to the virtual elimination of failed altimeter split testing during the first flight of every B747-400 aircraft coming

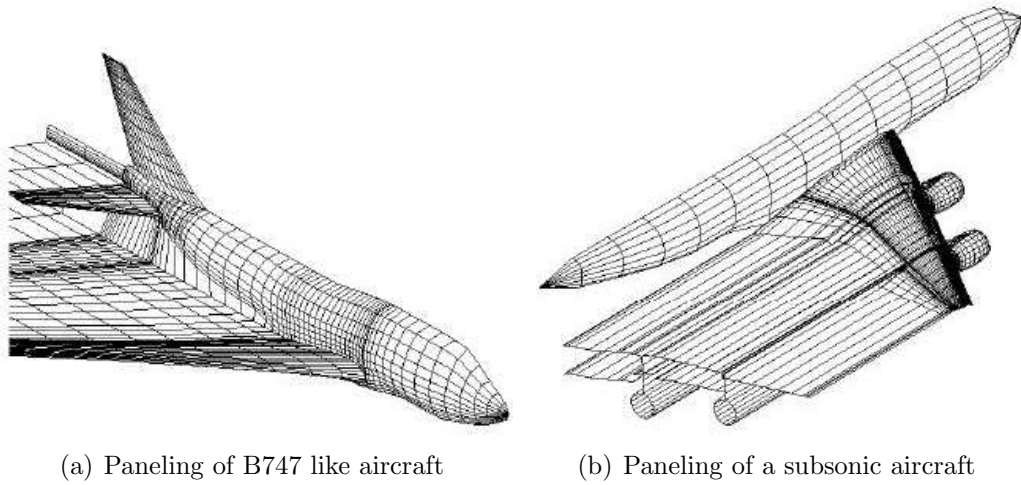


Figure 6.3: Paneling of two big transport aircraft. Note the particular wakes arrangements for both the configurations.

off the production line. Initially, one of every three aircraft was failing this test, requiring several days down time to fix the problem. The A502-based procedure could identify excessive skin waviness before first flight and led to manufacturing improvements to eliminate the root cause of the problem.

One of the most impressive early uses of the precursor code of Pan Air occurs in the initial design phase of the B747 Space Shuttle Carrier Aircraft. The purpose of the initial design phase was to define the modifications needed to accomplish the following missions: to ferry the Space Shuttle Orbiter; to air-launch the Orbiter; and to ferry the external fuel tank. To keep the cost of the program to a minimum, CFD was extensively used to investigate the Orbiter attitude during the ferry mission, the Orbiter trajectory and attitude during the launch test, and the external tank location and attitude during the ferry mission. At the conclusion of the design phase, the final configurations selected were tested in the wind tunnel to verify predictions. A typical example of a paneling scheme of the B747 with the Space Shuttle Orbiter is depicted in fig. 6.5(a). In this example, the Orbiter incidence angle was 8° with respect to the B747 reference plane. The predicted lift coefficient, C_L , as a function of wing angle of attack for this configuration is shown in fig. 6.5(c). The agreement between the analyses and wind tunnel data shown is excellent.

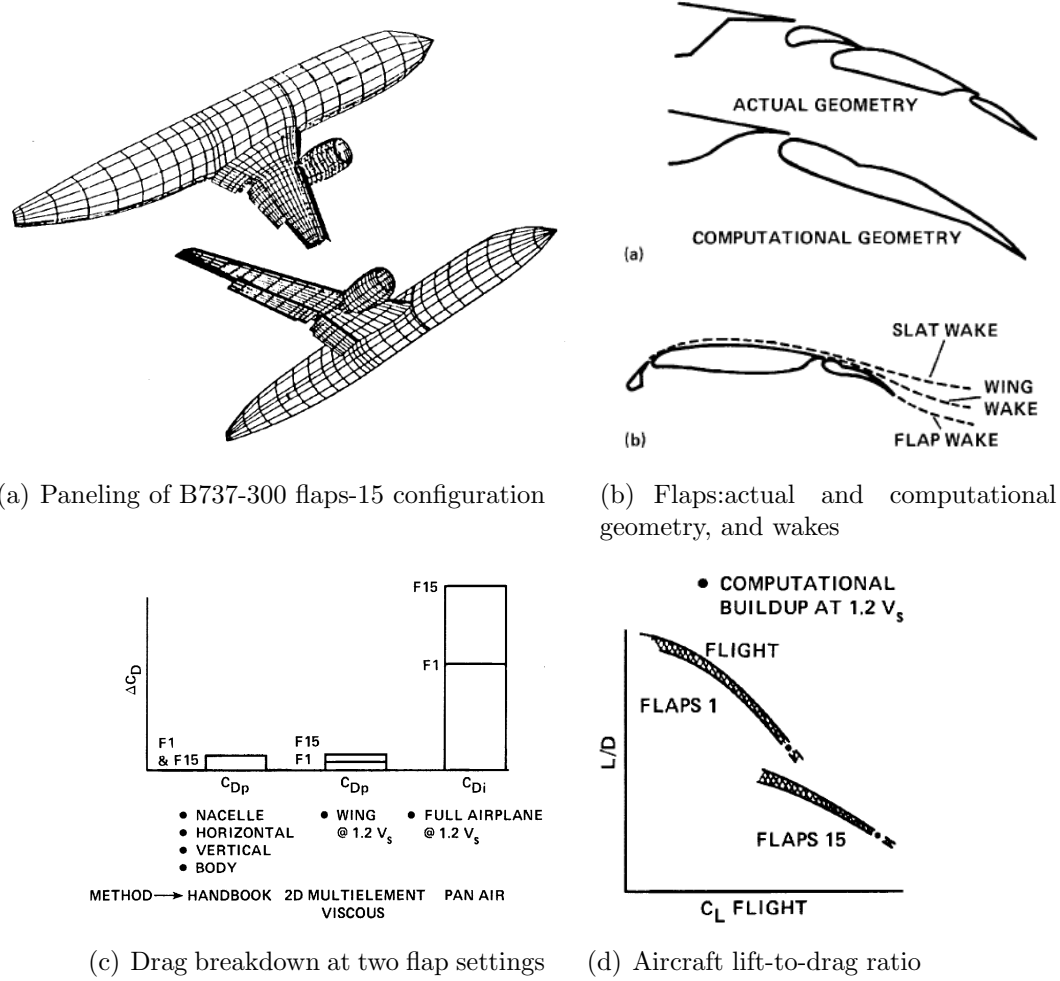
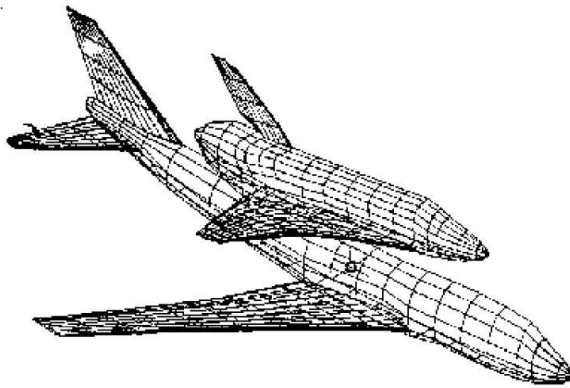


Figure 6.4: Pan Air study of the high-lift system of the Boeing 737-300.

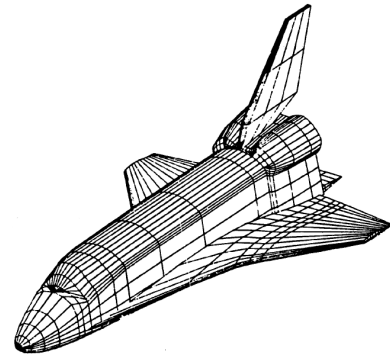
However, one of the most important features at time of Pan Air, was the reliable supersonic analysis (at least from a numerical point of view). Thus, many military aircraft configurations were designed with Pan Air aid. In fig. 6.6 is depicted the paneling scheme of a F15, a supersonic fighter.

As another example, in [57] development of an SR-71 aerospike rocket flight test configuration were sustained with both Pan Air and Tran Air analyses, supported with wind tunnel. In fig. 6.7 a pressure maps from Pan Air simulation is shown.

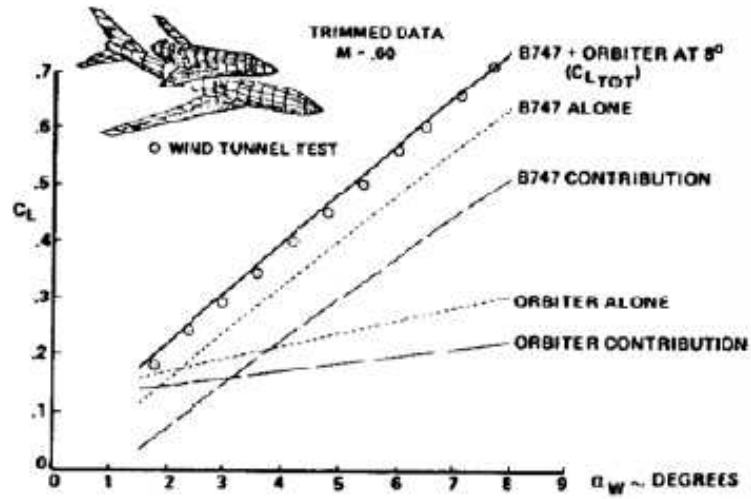
Pan Air use was not limited to aeronautic field. In fact, sailing both and yachts



(a) Paneling of B747 and Space Shuttle Orbiter



(b) Paneling of Orbiter



(c) B747-Orbiter lift coefficient

Figure 6.5: Boeing 747 with Space Shuttle Orbiter.

were also planned with the aid of the code. A paneling of a sailing both is shown in fig.6.8.

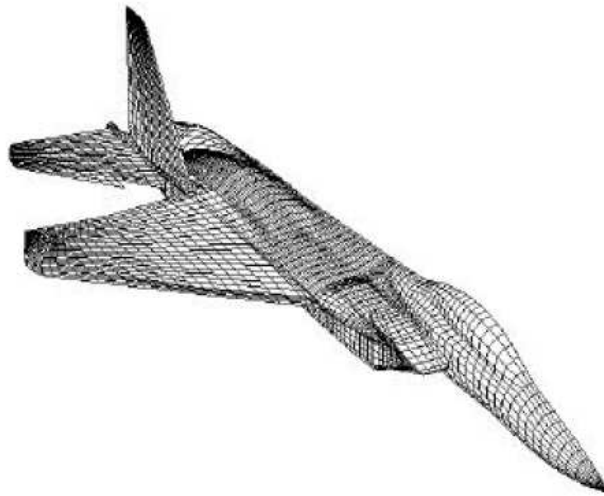


Figure 6.6: Paneling of a F15 fighter.

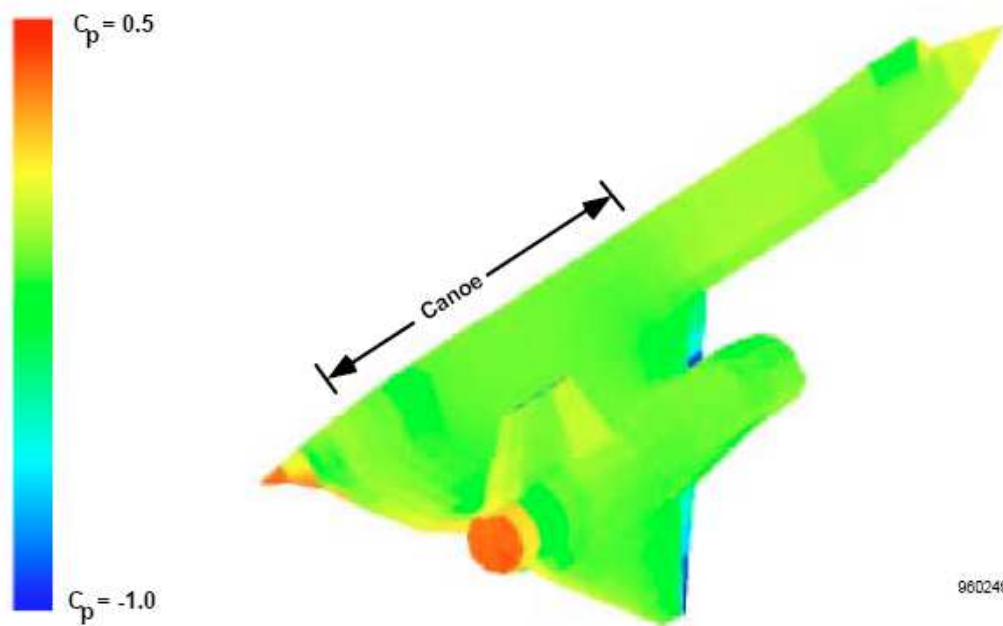


Figure 6.7: Pan Air surface pressure maps at Mach 0.6 and 4° angle of attack for a SR-71 aircraft.

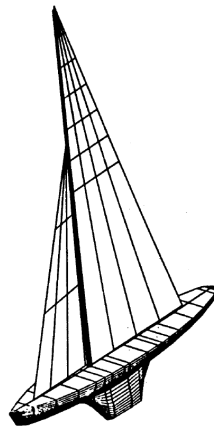


Figure 6.8: Sailing both paneling

Pan Air Pre/Post Processor Module

In this chapter the PanAir preprocessor and postprocessor GUI module is briefly presented. The preprocessor allows to easily set the flows properties and whatever necessary to begin an aerodynamic simulation, and eventually to start it. After Pan Air has completed the task, the results output file can be reviewed with a user friendly postprocessor.

7.1 Pan Air Preprocessor

After a structured mesh has been associated with a configuration, it is possible, from the structured grid generation module, to launch the *PanAir input generator*. The preprocessor enables to set most of the parameters defining a flow simulation on the input network. The main window is shown in fig. 7.1.

In the *Title an modality run* panel, the name to assess to the file is chosen, as well as the run modality. PanAir is capable of three main modalities of execution: **datacheck**, which validates inputs for a solution run, **solution**, which solves all the step defining and inverting the AIC¹ in order to calculate the solution, and **restart**, to use when additional solution run or flow property evaluation are needed on the same basic data, thus reusing the previous calculated and inverted AIC or the SIN

¹Aerodynamic influence coefficients matrix

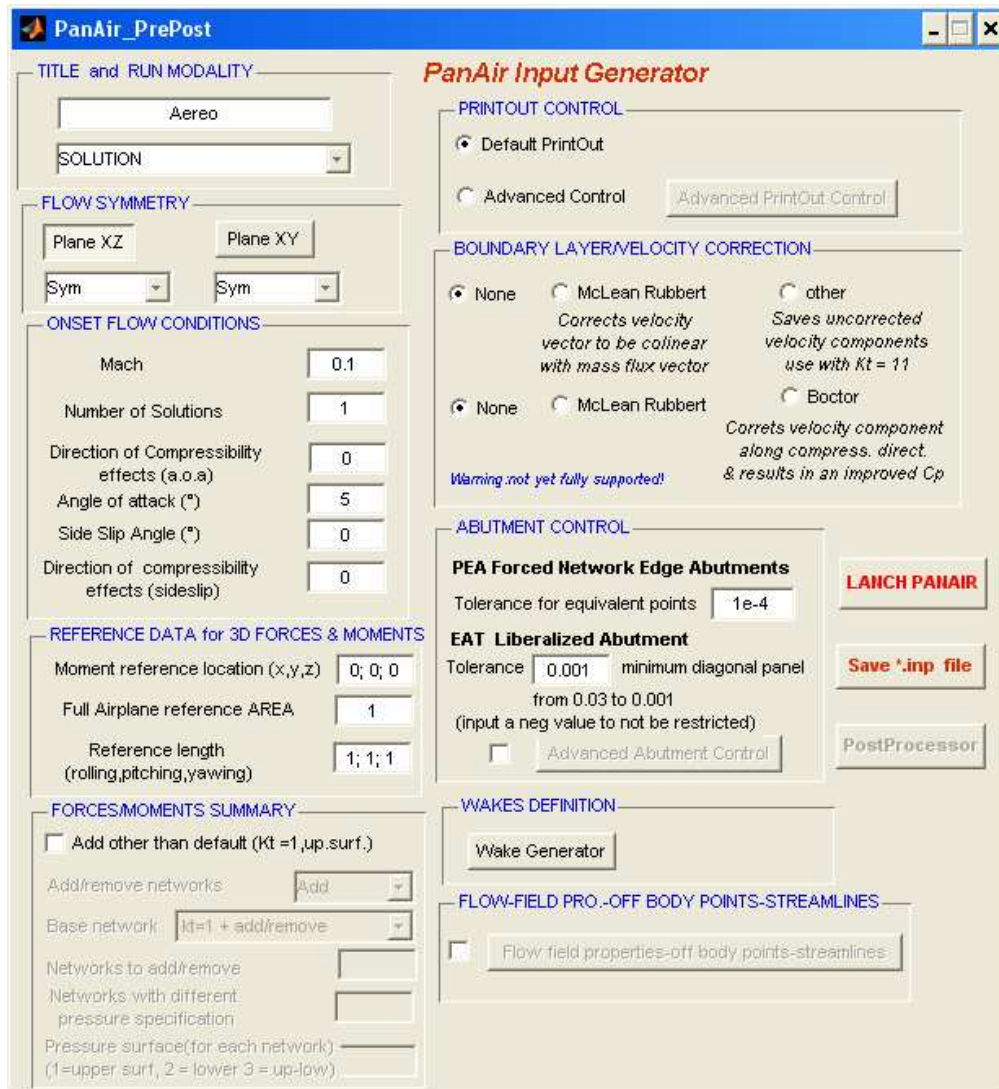


Figure 7.1: The Pan Air input generator GUI

(singularity strengths) matrix. All these options have more peculiarities, explained in depth in [53; 54].

Flow symmetry conditions are assessed on the next panel. It is possible to submit simulation considering symmetric and/or antisymmetric configurations respect both planes XY and ZX . In such a situation, the input network could be only a part of the whole. This approach represents a considerable savings over an analysis of the complete configuration. Remember that flow should be symmetric, thus the yaw

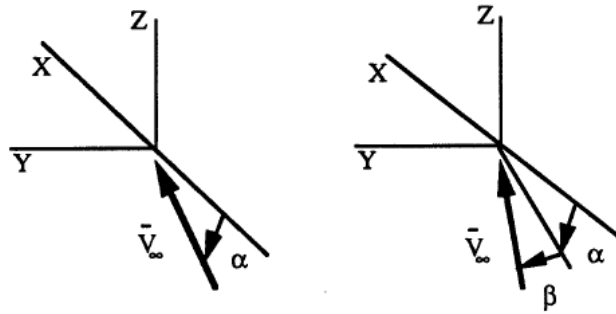


Figure 7.2: Angle of attack α and yaw angle β .

angle and/or angle of attack are forced to consistent value when needed.

The next panel enables to define onset flow parameters. Just one Mach number can be specified for each simulation, as well as one direction of compressibility, defined by the angles α and β depicted in fig.7.2. This direction should be coincident with the onset flow one, but if different solutions are required within the same run, a mean value could be selected. However, usually 5° degree of deviation for subsonic, 2° for supersonic don't produce significant errors. Up to four simultaneous number of solutions are possible for every run, for each of the solution being definable the free stream direction from the angle of attack and sideslip.

In the *reference data* box, the user can set:

- the full airplane reference area, used to normalize the forces and moments,
- the point with respect to which the moments are calculated,
- the reference length for the three components, used to normalize the moments.

The *Forces/Moments summary* panel allows one to compute the overall forces and moments acting on a configuration, defined by the user, consisting of just some between all the networks of the configuration. More details on this capability are explained in [53].

In the *Printout control* panel (fig.7.3) it is possible to define some parameters that determine program (PanAir) outputs concerned with the formulation of the boundary value problem, the resulting surface flow properties, and the resulting

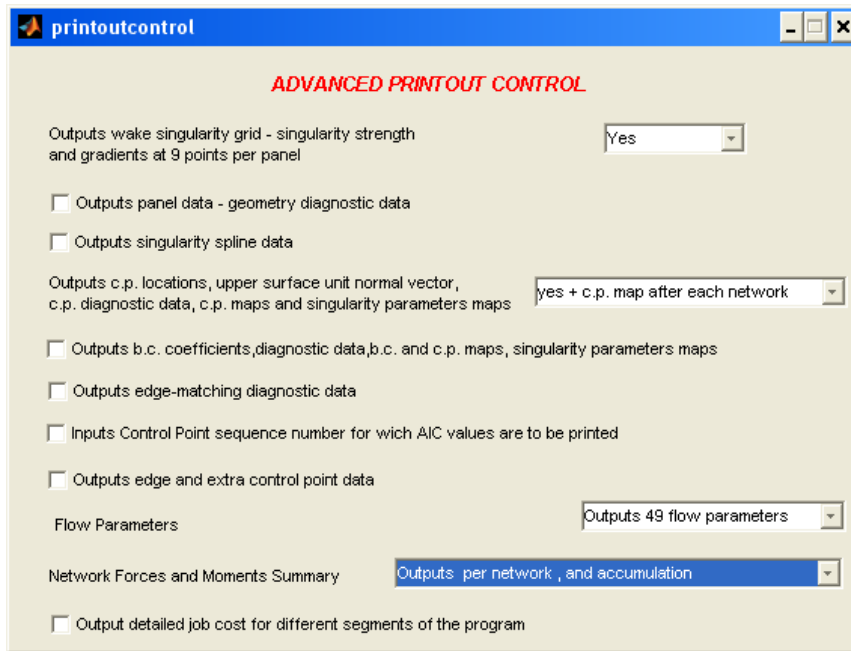


Figure 7.3: Advanced printout control.

surface forces and moments. The default printout is usually enough, since many of the advanced option are for diagnostic and development purposes, and not much of interest for the usual analysis. Again, refer to [53; 54] for a detailed explanation.

The boundary layer and velocity correction options are handled in the following panel. Both of them improve the flow property for flow slower than the onset flow (where, with reference to the small perturbation approximation of 6.1.1, the u magnitude is not longer less than the U_∞ , and the perturbations are not longer small). The biggest corrections are made near the stagnation point. The correction changes the surface velocity, and thus pressures and Mach number. These modified values replace the uncorrected ones in the printout, however, the program uses uncorrected velocity to calculate all forces and moments. The boundary layer correction formula is used with boundary layer analysis; the results from the correction are stored on an output file, available for a boundary layer analysis with the A598 code. The velocity correction are instead used for inlet flows.

The *Abutment Control* panels allows to set the basic or advanced parameter which control the behavior of the Pan Air internal functions employed to maintain

the continuity of doublet across network edges (more is written in section 6.2.3). Briefly, in the basic control modality the tolerances for the PEA (forced partial or full edge abutment) and EAT (liberalized abutments) functions are set, thus two points closer than the specified given tolerances are considered to be coincident for the given function. For a deeper control, the advanced modality is available.

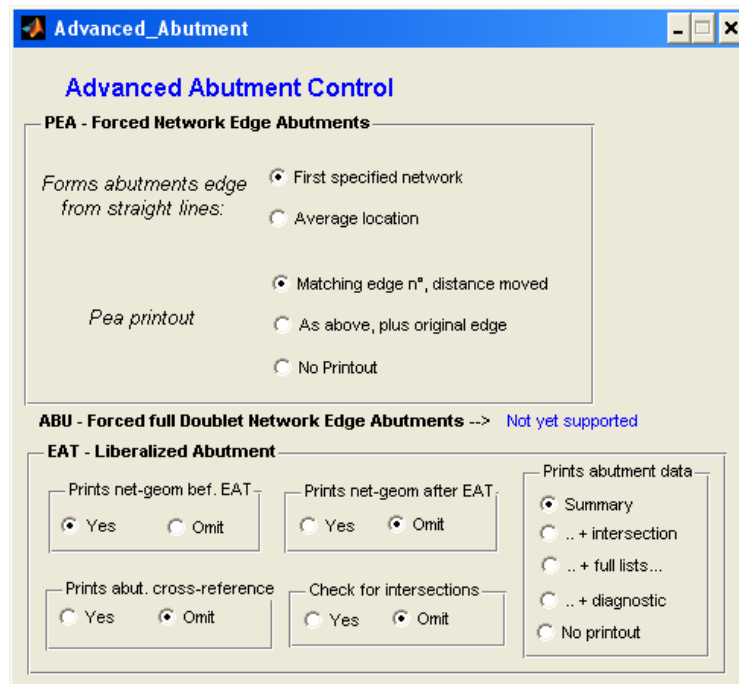


Figure 7.4: Advanced abutment control.

The *PEA* panel enables to control how to move points which doesn't satisfy the requirements, and how to write the modifications in the printout. The *EAT* panel features similar capabilities, both in moving the points that in the printout control. More about abutment handling can be found in [53; 54]. It should be stressed that abutment control process is of primarily importance, leading a not correct definition to unreliable results. Even if the structured grid generation module is built in such a way to automatically move the abutting edge points to fulfill PanAir requirements, the user is warmly suggested to checkout the abutment printout in the output file in order to avoid unreliable results.

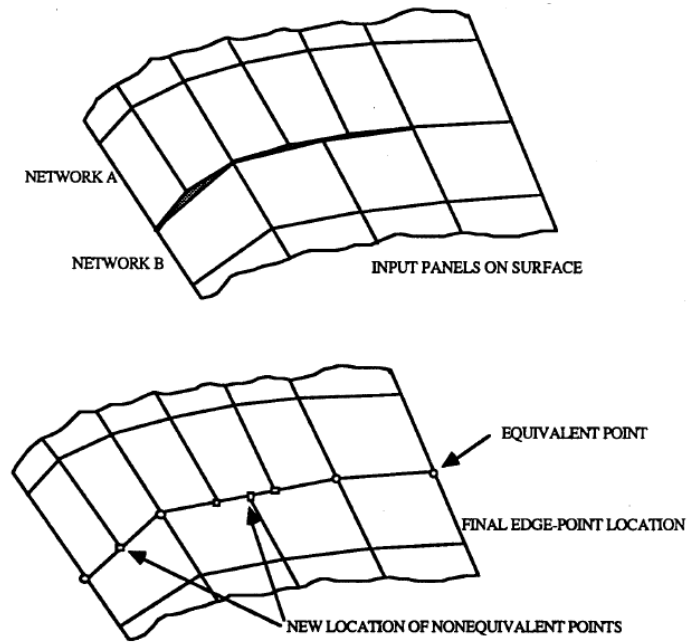


Figure 7.5: PEA processing for simple abutment of two network edges (from [53]).

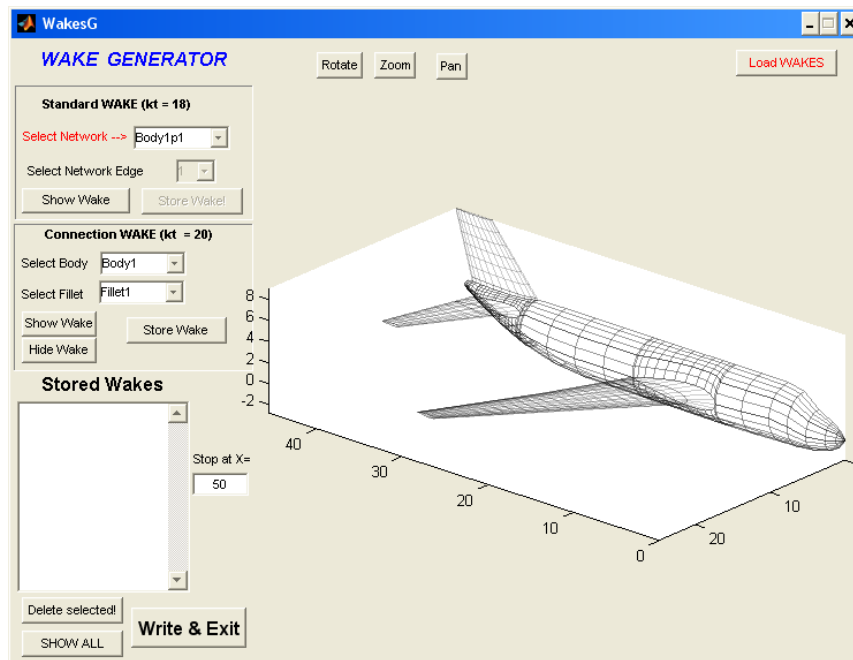


Figure 7.6: The wake generator GUI.

The *wake generator* GUI (fig.7.6) provide a mean to define and place the wakes. The interface enables to select each network (the selected is represented in red), to choose an edge of the network (represented in blue), and to place a wake starting from this edge downstream, as depicted in fig.7.7. At now, the placed wakes are parallel

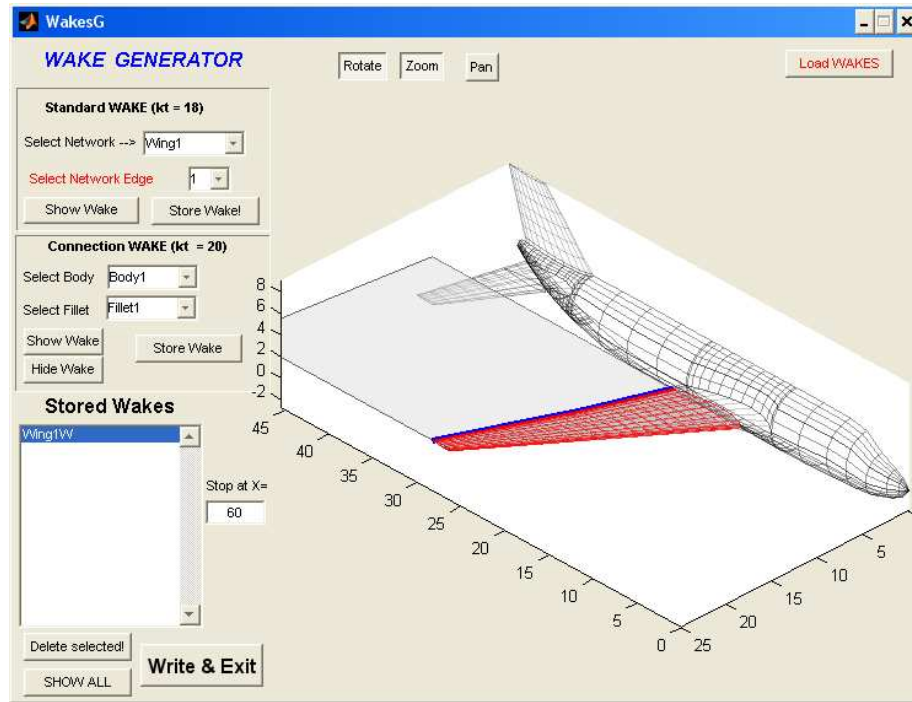


Figure 7.7: Placing a wake behind a wing in the wing generator GUI

to the XY plane. If the wake satisfies user requirements it can be stored. Note that, the wake generator distinguishes automatically between the different kind of wakes. For example, on a wake shedding from a bodybase, different boundary conditions are imposed than on a wake shedding from a sharp trailing edge (a wing). However, for further flexibility, the user should manually edit the *.inp* file. The connection wakes are needed to avoid zero doublet strength, and thus lift dropping to a null value, at the unabutted edges. This is the situation of root section of the wing. If the doublet strength is zero there, then, due to Kutta condition, it is zero at the trailing edge and the wing strip has null lift. Of course this is not acceptable, since the lift distribution doesn't fall to zero in the fuselage section. Thus, a wake with constant doublet strength is placed between this inner region of the wake and the

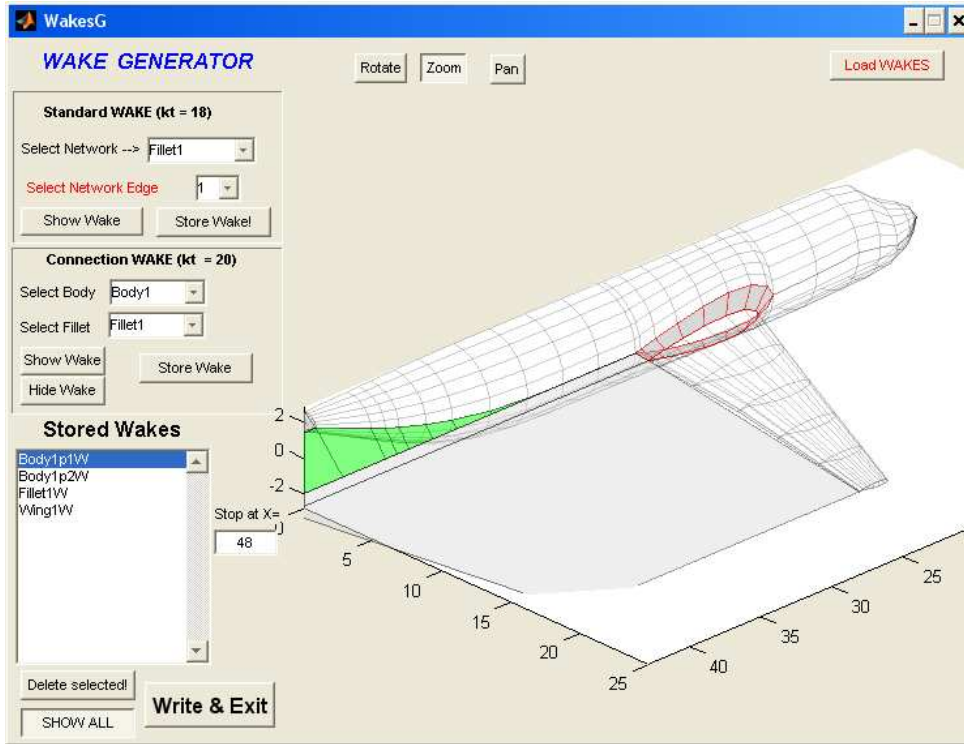


Figure 7.8: The connection wake between wing wake and fuselage.

fuselage. This procedure requires great knowledge of the physical problem, and is discussed in [42; 54]. If more flexibility is needed, it is possible to input the wake networks by a *.dat* file. This capability are still under development.

As last option, with the aid of the *flow field properties* panel, the user can define points and grid of points where flow properties are evaluated, and trace streamlines. The flow properties can be computed from all the surfaces or from a selected group of surface networks. The point locations can be entered directly, or as a network, defining the origin, three points defining thus the three directions, the distances and the number of points for each direction. Figure 7.10 clarifies the process. For the determination of streamlines, the starting points should be first defined, followed by the step size for spatial integration. The maximum allowed travel (stopping criteria) stops the integration when one of the three component reaches the specified value. Many other parameters can be set; for more details refer to [53]. However, in the *Array streams* panel, it is possible to set an array of starting points by setting the

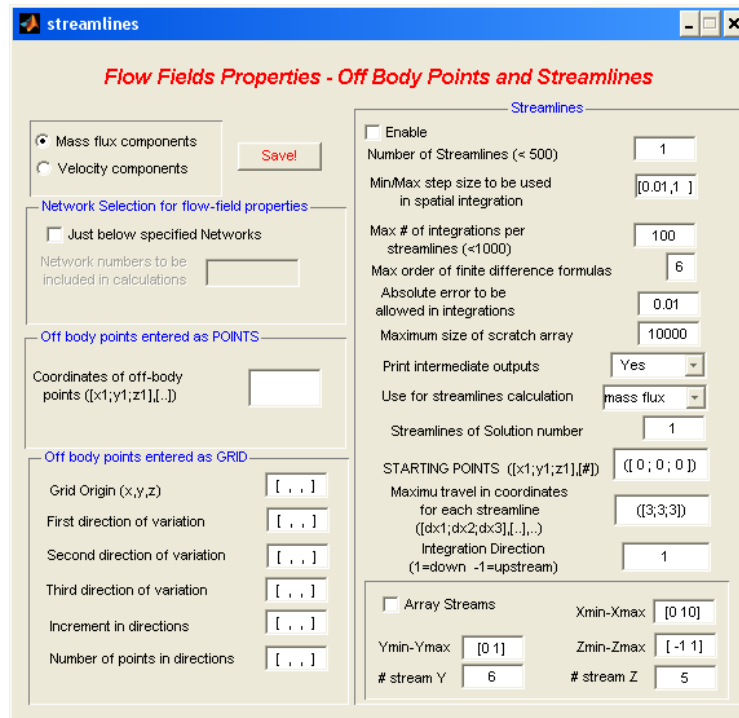


Figure 7.9: The flow field properties GUI.

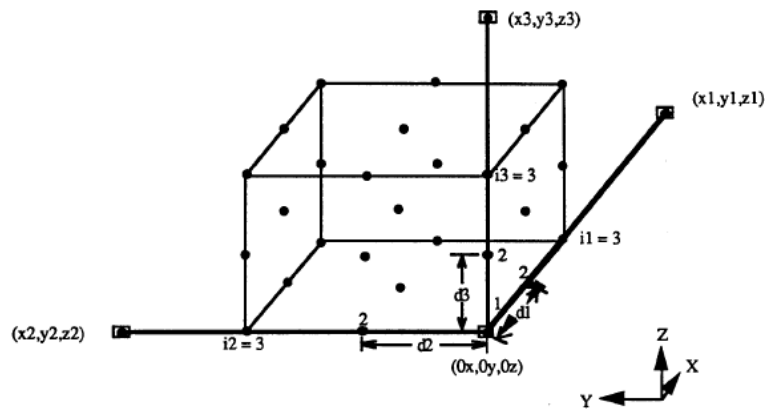


Figure 7.10: Definition of a grid of points where to evaluate flow properties.

two edge lengths and the number of points for each edge. This will speed up the process of defining streamlines.

At the moment, the preprocessor is not able to control all the PanAir capabilities. Important features like the *sectional properties*, where sectional forces, moments

and pressures are calculated along a specific plane, are not yet implemented in the preprocessor.

7.2 Pan Air Postprocessor

There are two ways to launch the postprocessor. The first is to click the pushbutton in the *PanAir input generator* window, the other is to use the menu in the ASD main window. In the first modality, the postprocessor will display results of the simulation run on the configuration displayed in the preprocessor, whereas in the latter modality the user should specify which PanAir output file is the one to analyze. Remember that results of PanAir runs are stored in these *.out* files. The file selection takes place through the *Load .out file* pushbutton, in the postprocessor main window. When the

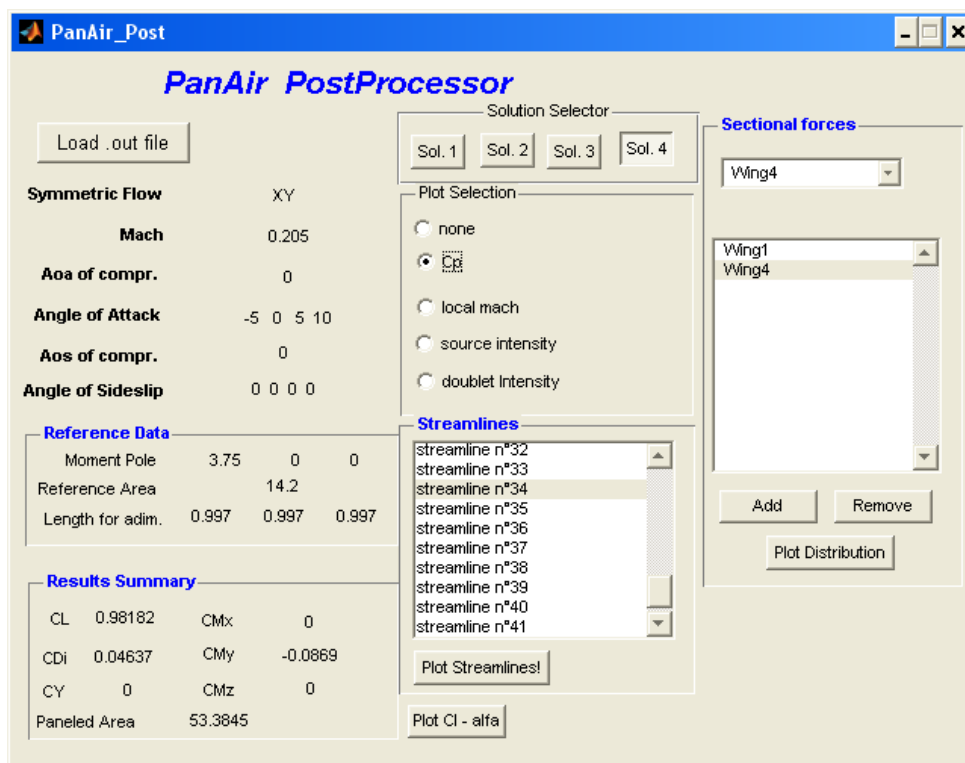


Figure 7.11: Main PostProcessor GUI with shown statistics and results.

file is selected, a window with the configuration networks is automatically plotted

(fig.7.12). In this same window it is possible to plot, for the solution submitted and one at the time, the flow or singularity characteristics for points on the surface. The values of the solution are shown by means of coloration of the surface, and a color bar gives the relation between colors and solution values. Both the selected flow characteristic and the onset flow angles (angle of attack and yaw angle) are recalled directly in the window. To change between solutions and flow characteristics it

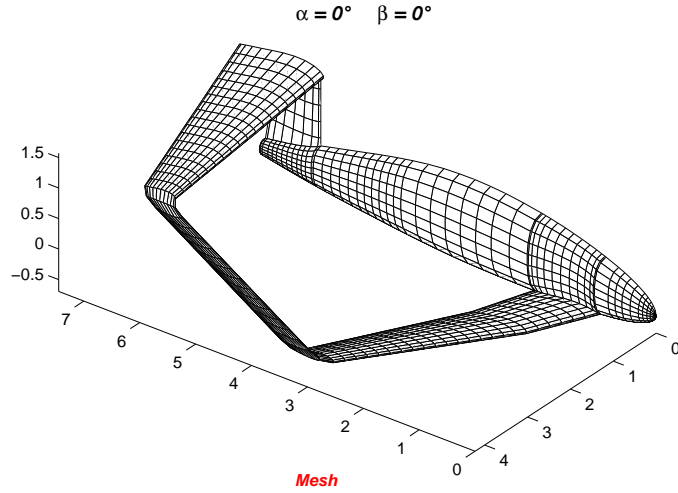


Figure 7.12: Configuration mesh.

suffices to push the proper buttons on the main postprocessor GUI. An example of results for pressure coefficient and local mach number is depicted in fig.7.13.

Summary of the most important onset flow conditions are reported, for each solution run. For the selected solution, down in the *reference data* panel, the reference point location, lengths and surface are shown, whereas in the *result summary* panel the most significant simulation results on the whole configuration and the total paneled area value are reported .

It may be useful to have a visualization of the forces compared to the angle of attack; pushing the proper button displays such a graph for the lift (C_L), induced drag (C_{Di}), and moment (C_{My} , moment in respect of axis y) coefficients, as depicted in fig.7.14.

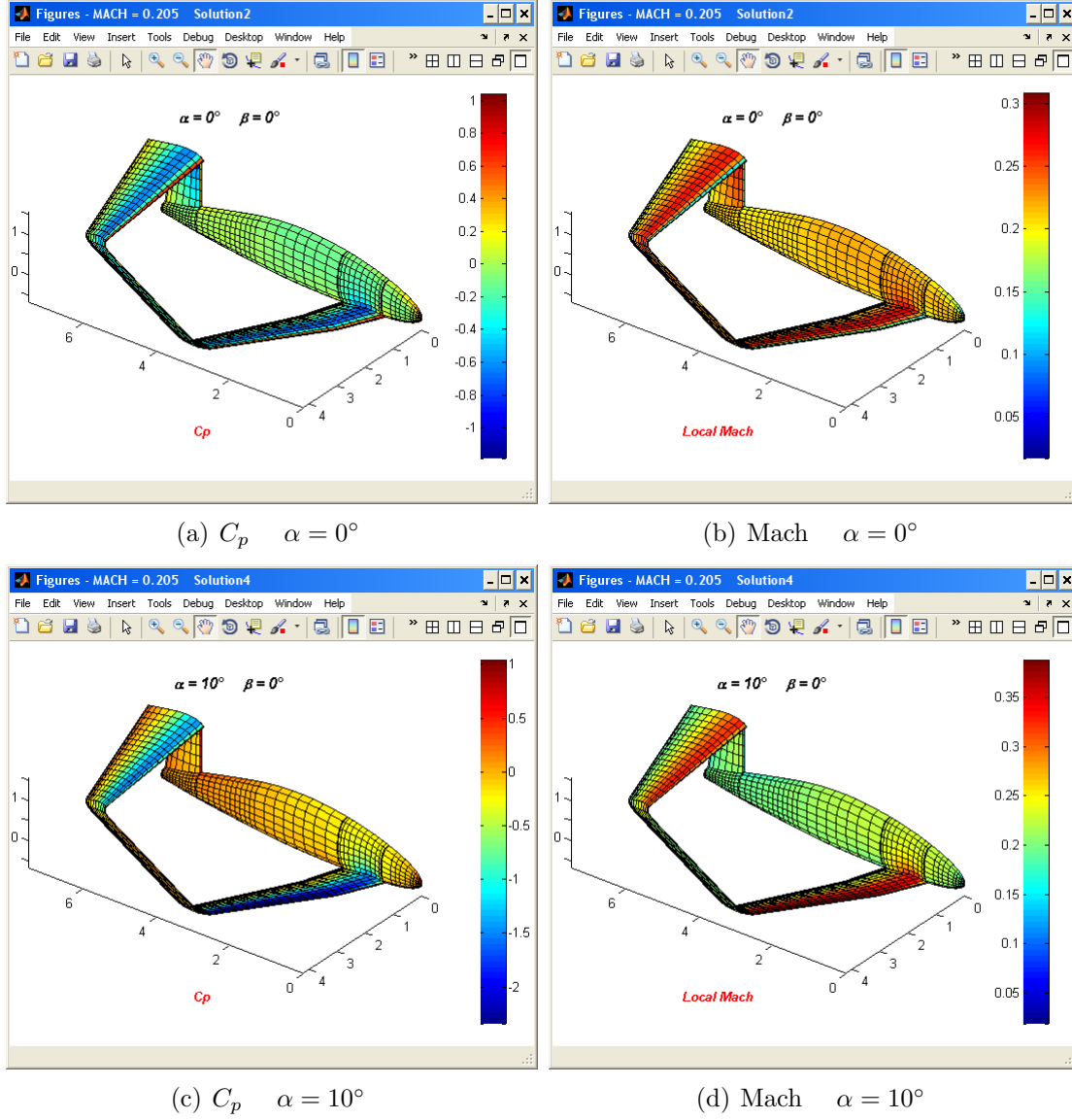


Figure 7.13: Coefficient pressure (C_p) and local mach number for two different solutions, one at angle of attack of 0° , the other at 10° .

The postprocessor enables also the visualization of the streamlines. In the proper listbox, all the streamlines defined at the preprocessing stage are listed. It is possible then to select more of the elements of the list and plot them. Of course the displayed streamlines refer to the selected solution. An example of streamline visualization is depicted in fig.7.15

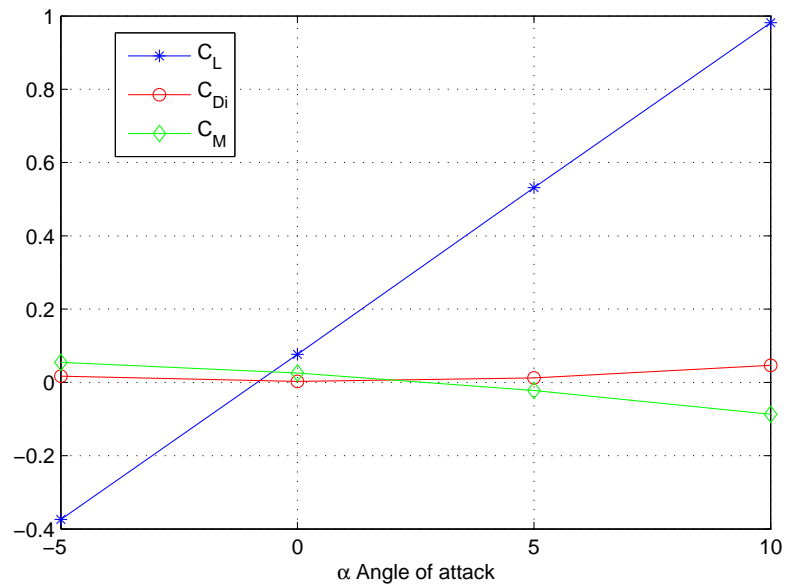


Figure 7.14: Plot of C_L , C_{Di} , C_{M_y} versus α .

$$\alpha = 10^\circ \quad \beta = 0^\circ$$

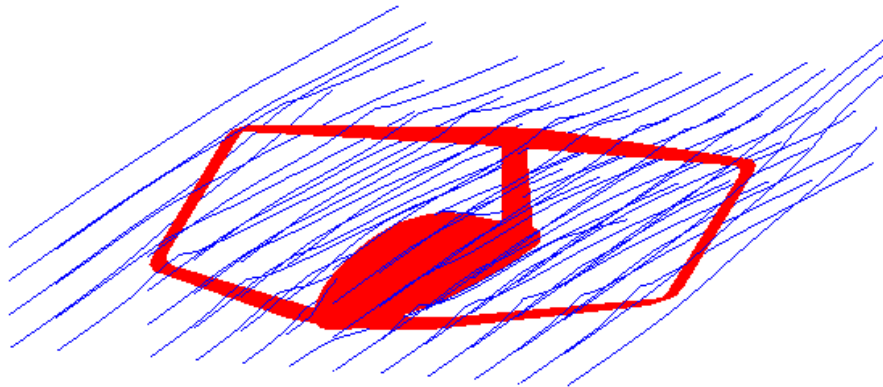


Figure 7.15: Streamlines.

Another useful tool implemented in the postprocessor is the ability to plot the two dimensional lift coefficient along the y direction for every selected network. In

the *sectional forces* panel just select the networks of interest, and add them to the listbox. Then, selecting the plot option, the lift coefficient versus the y-direction coordinate is plotted. An example is shown in fig.7.16 for the ULM PrandtlPlane configuration *Tandem* of fig.7.12. The wings and the bulk have been selected, and the plot shows the different behavior of upper and lower wings.

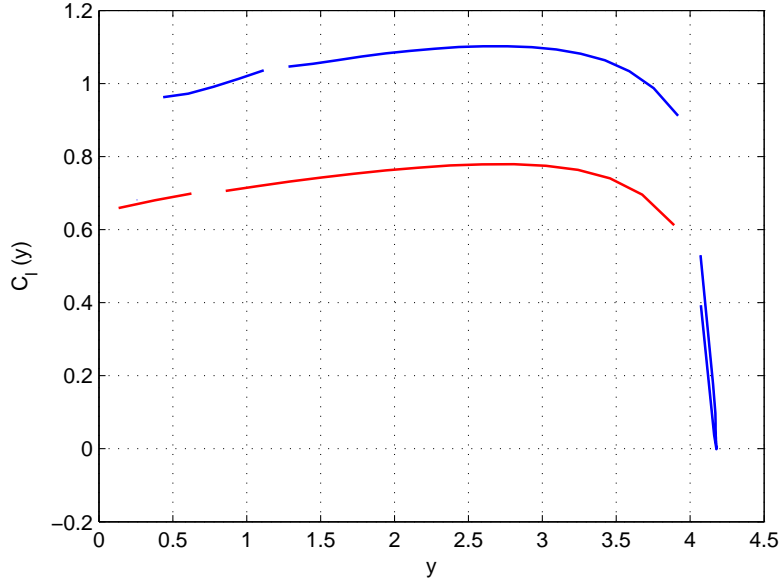


Figure 7.16: $C_l - y$ for the wings and bulk of the *Tandem* configuration at $\alpha = 10^\circ$.

At the moment some other features are implemented but not yet completely working. For example, the visualization of the pressure coefficient along the airfoil is an important tool to work with. An example of results obtained with this tool are presented in the next section.

To close the section also some graphics obtained from a traditional configuration analysis are shown.

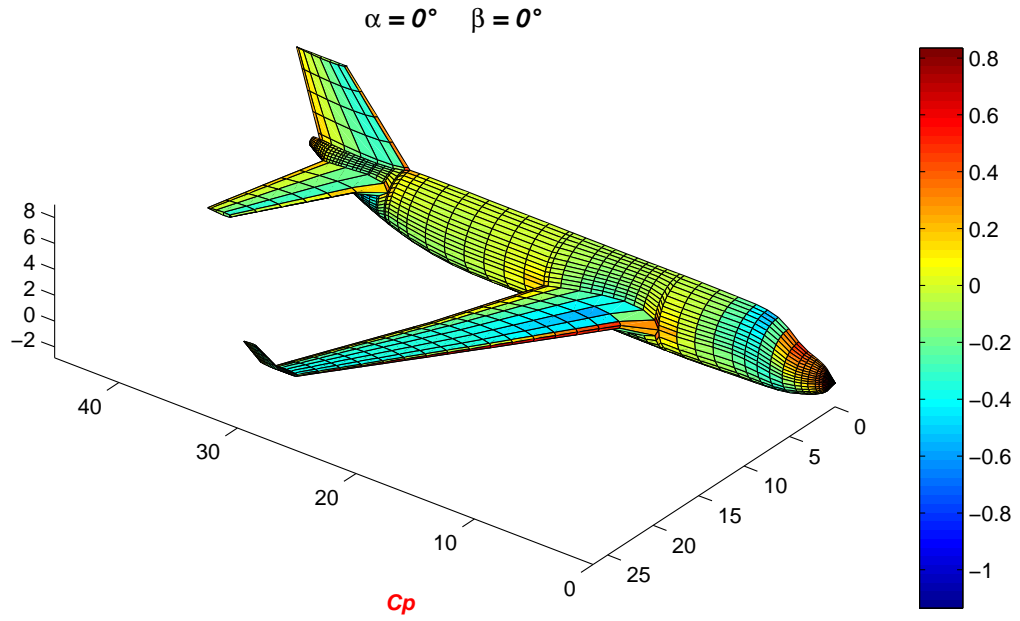


Figure 7.17: Pressure coefficient over a traditional configuration aircraft (A310 like) for zero angle of attack.

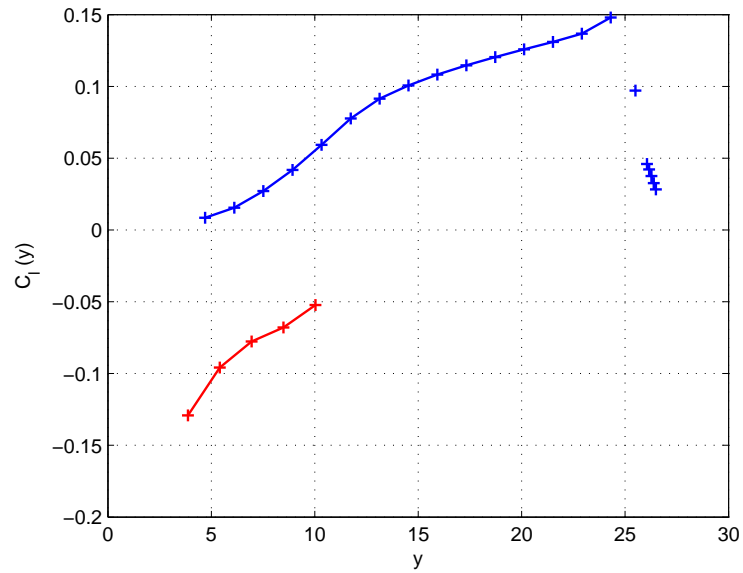


Figure 7.18: $C_l - y$ for the main wing and winglet (blue), tail (red) at $\alpha = 10^\circ$.

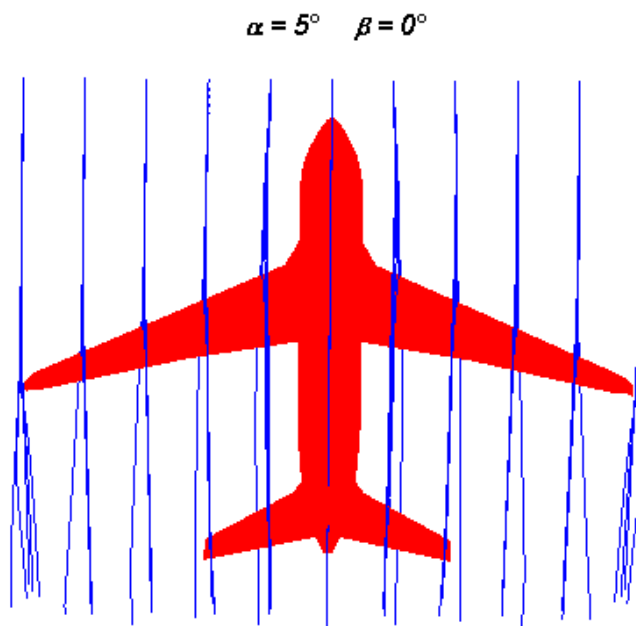
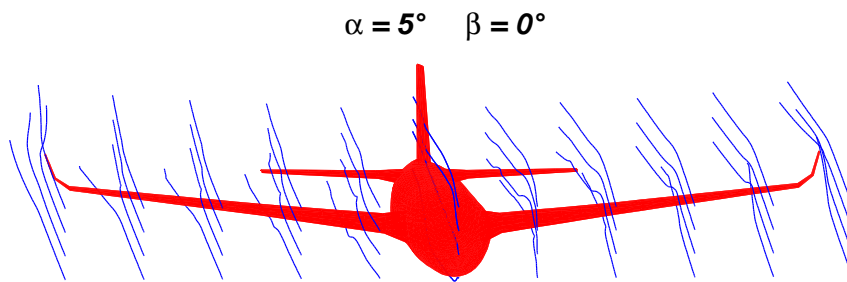


Figure 7.19: Streamlines at $\alpha = 5^\circ$.

7.3 A simple testcase

As a simple testcase, comparison between experimental results reported in [58] and results obtained with PanAir is carried out. In the aforementioned report, experimental results for a wing are given. The wing geometry is depicted in fig.7.20. The airfoil are the *NACA 65-210* with 10% thickness. The wing presents a 2° washout at the tip, with a linear spanwise distribution. The Mach number is 0.17. The first

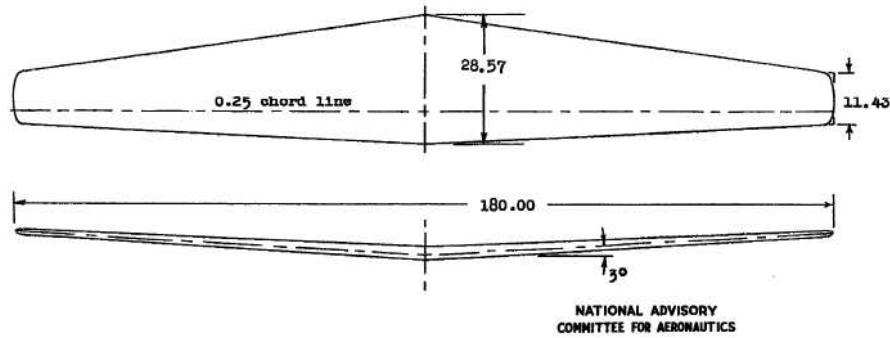


Figure 7.20: Wing tested in the report [58].

modeling step is fast achieved with ASD. The wing is depicted in fig.7.21. Very

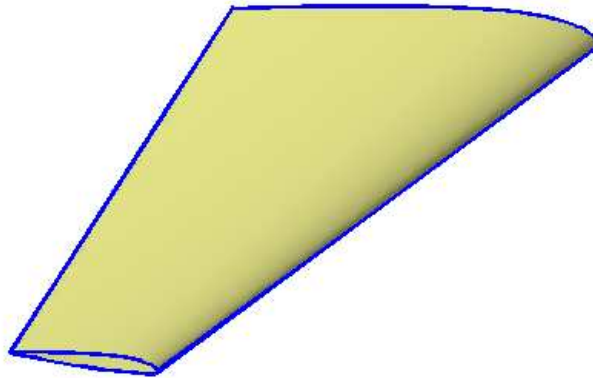


Figure 7.21: Geometrical definition of the wing with ASD.

fast and efficient is also the process of grid generation. A quite fine grid has been generated, as shown in fig.7.22. And finally, with aid of the preprocessor, input con-

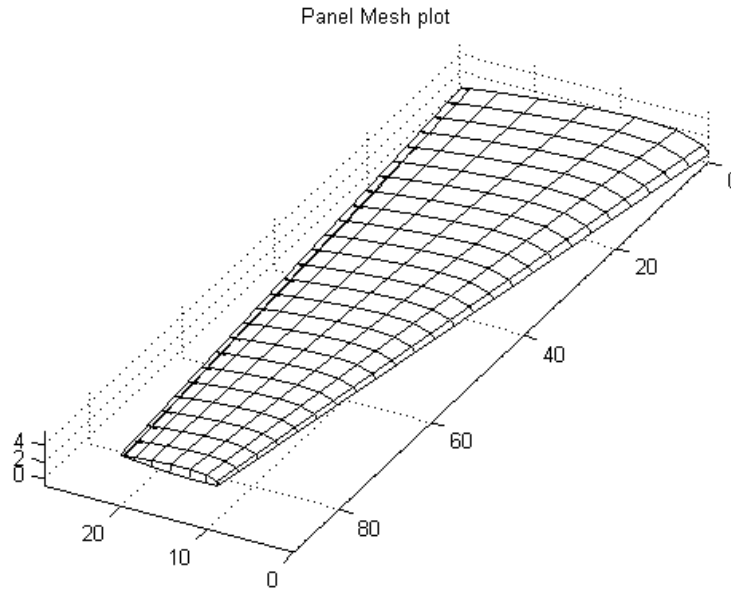
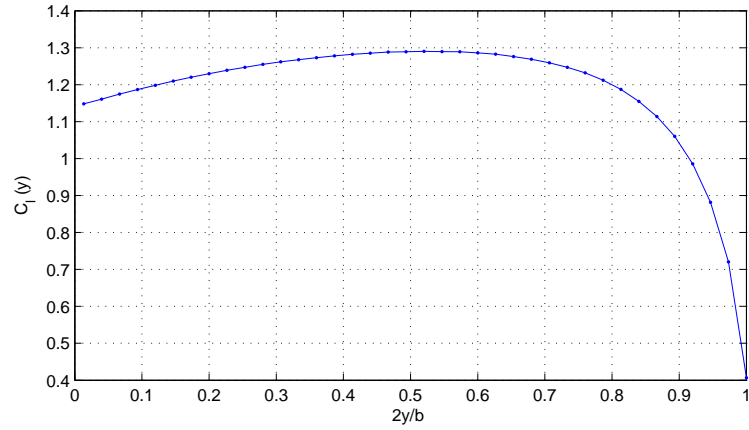


Figure 7.22: The meshed wing obtained with the structured grid generation module.

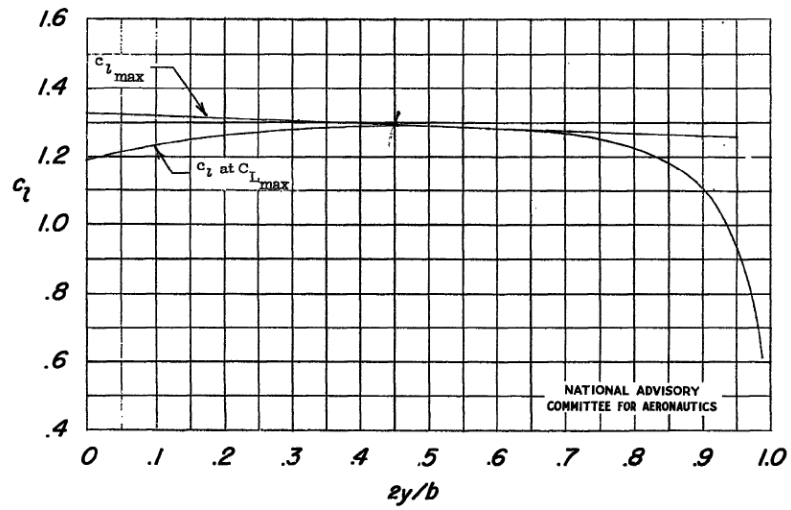
figuration file has been created for PanAir, and simulation has run. Once Pan Air has completed his task, through the postprocessor, the results have been immediately compared with the paper's data. The lift coefficient versus wingspan shows a very good agreement with the experimental results, even at configuration of max C_L for the wing. The value of C_L is also predicted in a very accurate way.

As an additional mean to check Pan Air simulation results, a plotting of the pressure coefficient along the chord at different spanwise sections is displayed, as shown in fig.7.24.

As a final validation problem, grid sensitivity and converge of the solution are investigated. For $\alpha = 2^\circ$ the sensitivity of the solution from the grid is depicted in fig.7.25. It easy to ascertain that the solution reliance on both chordwise and spanwise grid refinement is limited to satisfactory results; in detail, the solution vary of a 0.2% value when the grid is three times finer on both directions.



(a) Results obtained with PanAir



(b) Experimental results [58]

Figure 7.23: Comparison between numerical and experimental results, for $M = 0.17$ and $\alpha = 12.5$. In spite of the $C_{L_{\max}}$ configuration, the agreement is very good.

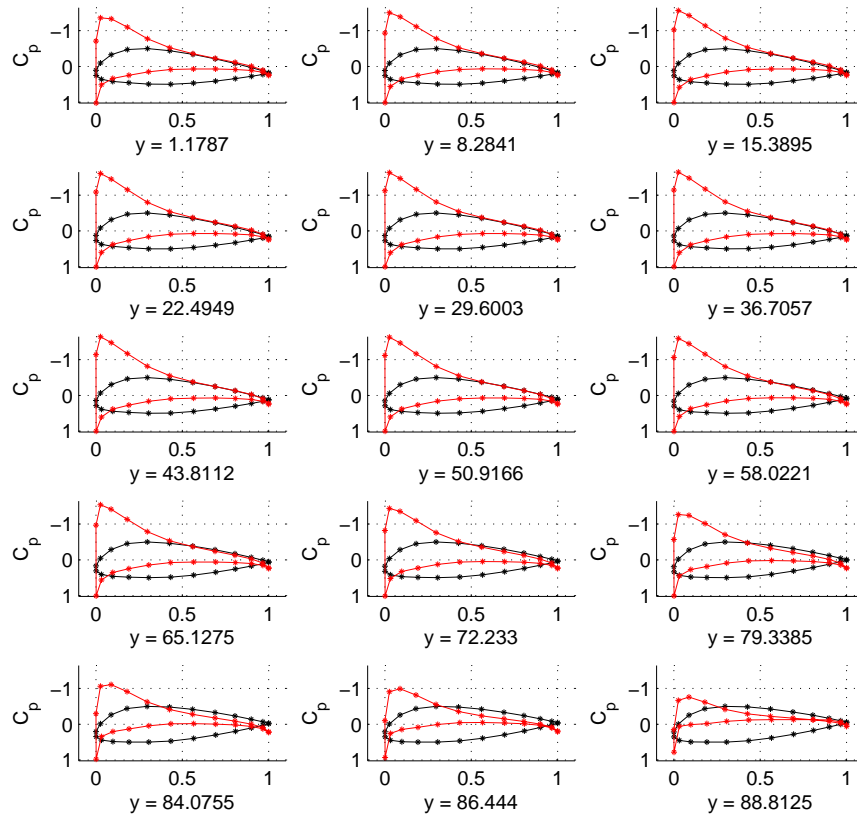


Figure 7.24: Pressure coefficient (in red) versus chord at different spanwise sections, for $\alpha = 7^\circ$. The airfoil sections are drawn in black.

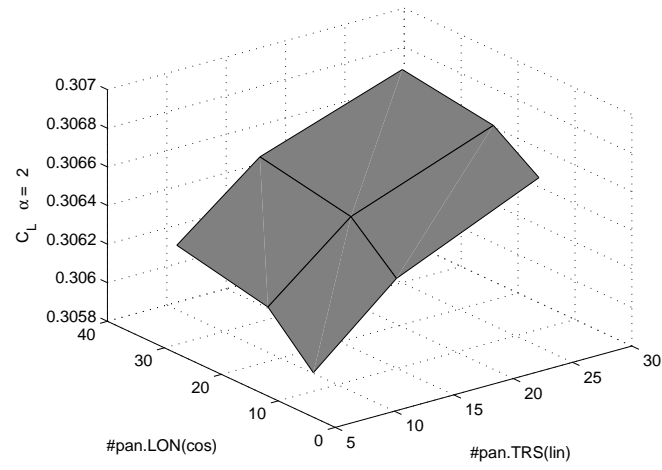


Figure 7.25: Grid sensitivity. The grid is refined in both chordwise and spanwise directions. Chordwise the grid has a cosinusoidal distribution, spanwise the distribution is linear.

Conclusions

This code was developed with the main objective to help in the preliminary design of PrandtlPlane configurations. The preliminary aerodynamic properties estimation represent the basis from where to start further investigations and modifications. The main problems to deal with are the aerodynamic optimization, in order to increase as much as possible the efficiency, stability and trim, where the aerodynamic derivatives should be estimated to check if the examined configuration is stable and to set the trim conditions, and dynamics, which studies the aircraft behavior in response of flight commands. The PrandtlPlane innovative configuration carries some peculiarities which lead to unpredictable behavior if relying only to classical results. As an example, small modifications to the wings can change the stability to a great extent.

During the research activity on PrandtlPlane the persistent lack of:

- a tool where geometric definition and modifications can be easily submitted;
- a tool which enables a fast grid generation, avoiding the bothersome and time consuming process of exporting the geometry, importing it in the meshing program, generating the grid and make it be compatible with the format required from the solver program;
- a tool where the aerodynamic flow characteristics are simply set, and run solutions are immediately displayed, avoiding thus long network import and

standardization processes, and cumbersome flow characteristics definitions and results visualization (for solver without pre and postprocessing).

- a tool which enables the exportation of configuration geometry and grid to the international most recognized standards.

made feel the strong need of a code which settles the above flaws. The geometric flexibility needed for the geometric design of the unconventional configurations guarantees that also conventional configuration can be sketched and analyzed.

Results at each stage are depicted in the corresponding chapters. However, it should be stressed that the overall process, consisting in a new geometric shape definition, grid generation and PanAir analysis, is usually completed in not more than tens of minutes.

The flexibility of the geometric engine, the NURBS lean to grid generation, the fast, easy and user friendly GUI, have appealed the interest of some universities and consulting societies.

The future improvements run in different ways.

Future geometric improvements

First, a more robust and faster geometric shape generation algorithm should be implemented, especially for high quality surfaces. This task must deal with the numerical implementation in order to avoid close to singular matrices that could be generated when interpolating a big amount of points. A pre conditioner or the use of variational multiresolution curves and surfaces wavelets may be an optimal choice [32].

Always on the geometrical side, both direction derivative constraints should be implemented for the C^2 interpolating algorithm.

Further, it feels the needs for a smoothing routine or an approximation algorithm, in order to avoid unwanted bulging of the surfaces due to numerical errors introduced mainly from the intersection algorithms.

Future grid generation improvements

Surely one of the first needs is the generalization of the structured grid generator to flapped configurations.

The structured grid generation undergoes smoothness and efficiency limitations, in particular in the body regions intersected from wings. Here, to fulfill both PanAir network requirements and mesh quality restrictions, a finer grid is usually built leading to inefficient increased total number of panels. The logics should be revised, and a more advanced grid technique should be adopted. At the same time, an elliptical grid generation or smoothing technique may be a solution if the grid is built for more severe solvers.

Future pre-postprocessor improvements

The preprocessor needs basically a more advanced wake interface. In fact, when the shape is known from experience or other means, it would be useful to draw the wake in an interactive environment providing thus more flexibility. The postprocessor needs also a sectional force and moments GUI.

Future aerodynamic solver improvements

Current trends seem to prefer low order panel method than higher order like Pan Air. Beyond this trend, a non stationary panel method would surely extend the field of application of the aerodynamic analysis. Integration of codes like TranAir, capable of Transonic analysis, and the Boeing code A598, which features a viscous panel analysis, would be easy since they requires input similar to PanAir, thus little work has to be done to integrate them in the platform. It should however kept in mind that more accurate analysis cost more, thus at a preliminary design stage they may be not convenient.

Addition of other modules

To improve platform capabilities, a flight mechanics module could be added. In this section, the results from the aerodynamic analysis would be used to evaluate

stability, and trim.

Surely, interfacing the code with a simple or an advanced external optimizer, like *modeFrontier*, would really improve the value of the present code. Future efforts should be addressed in this way.

List of figures

1.1	The ASD main window.	7
1.2	The body feature window.	8
1.3	Parameters involved in the definition of the body feature	9
1.4	Preview of the skeleton and resulting interpolated surface for a body feature. . .	9
1.5	The wing feature window.	10
1.6	Wing components	10
1.7	The Airfoil Viewer window	11
1.8	The Preview wing feature	13
1.9	The <i>Add/Modify Mobile Surface</i> window	14
1.10	Mobile surface deflection	14
1.11	The bulk feature window	15
1.12	A bulk example	15
1.13	The bulk control parameters	16
1.14	The Preview Bulk figure	17
1.15	The wingbody feature window	17
1.16	WingBody creation, type 1 (a) and (b), type 2 (c) and (d).	19
1.17	WingBody creation, type 3 (a) and (b), type 2 on leading edge and type 3 on trailing edge (c) and (d).	20
1.18	The inlet/outlet feature window	21
1.19	The inlet/outlet skeleton and center line	21
1.20	Effect of the interpolated sections in the geometry of the Inlet/Outlet.	22

LIST OF FIGURES

1.21	Inlet/Outlet Section Scale Factors.	23
1.22	The inlet/outlet preview	24
1.23	The fillet feature window	24
1.24	Smooth fillet derivatives on wing and body	25
1.25	Creation of fillet with Basic controls	26
1.26	Creation of fillet with Advanced controls	26
1.27	Example of smooth and linear fillet on the same configuration.	27
1.28	The Tfillet feature window	28
1.29	Smooth fillet derivatives on wing TFillet	28
1.30	Auxiliary airfoil definition and surface intersection	29
1.31	Smooth TFillet on T-tail configuration	29
1.32	The Wround feature window	30
1.33	The wround feature window	31
1.34	The <i>Surface Viewer</i> window	31
1.35	Surface Viewing: Render	32
1.36	Surface Viewing: Render Shaded	32
1.37	Surface Viewing: Three-Plane View	33
1.38	ASD Surface MESHER, the interface window	34
1.39	Mesh parameters: max chord distance	34
1.40	Mesh parameters: collapse ratio	35
1.41	Non conform mesh and conform mesh	36
1.42	Tri-mesh operation: closure of a mesh.	37
1.43	Feature operations: full Delaunay flip.	38
1.44	Airfoil Manager GUI	39
1.45	NACA Airfoil Generator GUI	40
1.46	Section Sketcher GUI	41
1.47	Flap Sketcher GUI	42
1.48	Flap Sketcher: <i>C</i> type line	43
1.49	Flap Sketcher: <i>S</i> type line	44
1.50	Flap Sketcher: <i>O</i> type line	44

1.51	A surface generated with ASD and imported with CATIA. Note the patch subdivision.	46
1.52	A surface generated with ASD and imported with CATIA. Note the patch subdivision.	47
1.53	Main decomposition of ASD code	48
1.54	Decomposition of the ASD geometrical engine	49
1.55	Decomposition of ASD mesher	50
2.1	Two examples of cubic Bézier curves.	58
2.2	The Bernstein polynomials	59
2.3	The recursive definition of the Bernstein polynomial, $B_{1,3}$	60
2.4	Evaluation of a point at $u = \frac{1}{3}$ with repeated linear interpolation, i.e. deCasteljau algorithm	60
2.5	Derivatives: (a) the derivatives of the cubic Bernstein polynomials; (b) the derivative $B'_{2,3}$ in terms of $B_{1,2}$ and $B_{2,2}$	61
2.6	A tensor product surface showing isoparametric curves (from [8])	63
2.7	The Bézier tensor product basis function $B_{0,2}(u) B_{1,3}(v)$ (from [8])	64
2.8	Bézier surface: note that the \mathbf{Q}_j (eq.(2.13)) don't lie on the surface.	65
2.9	A piecewise cubic polynomial curve with three segments represented in Bézier form	67
2.10	Non zero basis functions for different degrees and knot vectors	69
2.11	The recursive definition of B-spline basis: $N_{i,3}$ obtained as linear interpolation of $N_{i,2}$ and $N_{i+1,2}$	70
2.12	Dependencies between the basis functions	70
2.13	Cubic basis functions and corresponding derivatives	72
2.14	The recursive definition of B-spline derivatives: $N'_{i,3}$ as combination of $N_{i,2}$ and $N_{i+1,2}$	73
2.15	B-spline basis functions on a knot vector with a knot of multiplicity 2	74
2.16	Cubic basis function over the knot vector $U = \{0,0,0,0,\frac{1}{4},\frac{1}{2},\frac{3}{4},1,1,1,1\}$, and associated cubic curve with control points P_i	75
2.17	Fifth degree basis function over the knot vector $U = \{0,0,0,0,0,0,\frac{1}{7},\frac{2}{7},\frac{3}{7},\frac{4}{7},\frac{5}{7},\frac{6}{7},1,1,1,1,1\}$, and associated fifth degree B-spline curve with control points P_i	76

LIST OF FIGURES

2.18	Cubic curve on $U = \{0,0,0,0,\frac{1}{4},\frac{1}{2},\frac{3}{4},1,1,1,1\}$. Moving \mathbf{P}_i changes the curve in the interval $[u_i, u_{i+p+1}]$	77
2.19	Quadratic curve on $U = \{0,0,0,\frac{1}{6},\frac{1}{3},\frac{7}{12},\frac{7}{12},\frac{5}{6},1,1,1\}$. Notice the cusp at $u = u_5 = u_6$	78
2.20	Cubic curve on $U = \{0,0,0,\frac{1}{6},\frac{1}{3},\frac{7}{12},\frac{7}{12},\frac{5}{6},1,1,1\}$. Even if the knot u_5 has multiplicity two, there aren't any cusps at $u = u_5 = u_6$	78
2.21	Cubic curve on $U = \{0,0,0,0,\frac{1}{4},\frac{3}{4},\frac{3}{4},1,1,1,1\}$. First derivative is a quadratic B-spline on knot vector $U = \{0,0,0,\frac{1}{4},\frac{3}{4},\frac{3}{4},1,1,1\}$ with control points Q_i defined as in eq.(2.25).	80
2.22	A cubic curve on $U = \{0,0,0,0,\frac{1}{4},\frac{3}{4},\frac{3}{4},1,1,1,1\}$ and his first and second derivatives at endpoints, and at $u = 0.4$ (the first derivatives are scaled of a ten factor, the second of a twenty factor).	81
2.23	Cubic x quadratic basis functions. $U = \{0,0,0,0,\frac{1}{4},\frac{1}{2},\frac{3}{4},1,1,1,1\}$, $V = \{0,0,0,\frac{1}{4},\frac{1}{2},\frac{3}{4},1,1,1\}$	83
2.24	A B-spline surface and its control net (in red). Note the isoparametric curves on the surface.	85
2.25	Effect of weight modification (w_4) on a NURBS curve.	89
2.26	A geometric construction of a rational B-spline curve.	90
2.27	Knot insertion into a cubic curve defined over the knot vector $U = \{0,0,0,0,1,2,3,4,5,5,5,5\}$ (drawn from [8]).	94
2.28	Knot removal from a cubic curve with triple knot (from [8]).	95
2.29	Degree elevation on a third degree B-spline curve, $U = \{0,0,0,0,1,2,3,4,4,4,4\}$. The new control polygon is the one in red.	96
3.1	Difference between interpolation and approximation (from [8]).	98
3.2	Global interpolation with three collinear points	100
3.3	Curve interpolation, and curve interpolation with derivative specified.	101
3.4	Surface interpolations through subsequent curve interpolations.	103
3.5	Local interpolation: curve segments endpoints are coincident with the data points \mathbf{Q} (in red). The internal control points of every curve segment should somehow be computed.	104
3.6	The process of surface skinning [8].	106
3.7	Various parameterizations with uniform knot vector. \mathbf{x} indicates the parameter values, $+$ the knot values.	109

3.8	The effect of choosing knot values similar to parameter values.	110
3.9	Two curve segments joining with derivative of same direction but different magnitude.	111
3.10	Relation between derivative vectors for geometric continuity.	112
3.11	Frenet frame.	115
3.12	Two surfaces joining at the common edge, and their first partial derivatives.	115
3.13	Differences in light reflection between G^1 and G^2 surfaces.	117
4.1	Piecewise Bézier curve	128
4.2	Setting local tangent with Akima's method	129
4.3	Straight line segment with three collinear points	129
4.4	Close curve with G^1 continuity	130
4.5	Supercritical airfoil NASASC2 interpolated with ASD algorithm.	131
4.6	New algorithm behavior with three collinear points, for both fifth degree B-spline with fairing over second or third order derivative.	138
4.7	New algorithm behavior with three collinear points, for both fifth degree B-spline with fairing over second or third order derivative, and the same tangent vectors of the old local algorithm.	138
4.8	New algorithm behavior with 5 points with smooth flag. The third order derivative fairing based curve matches nearly exactly the circumference.	139
4.9	Airfoil interpolation with the new and old algorithms. Note that, if the points to interpolate are many and properly distributed, the two interpolant curves are nearly matching each other.	140
4.10	Behavior of the algorithm with and without the alignment flag option activated. Note how the interpolating algorithm is capable of detecting and drawing flat patches	142
4.11	Behavior of the algorithm with and without the corner flag option activated. Note how the interpolating algorithm is capable of closing with smoothness a closed surface	143
4.12	Wing with quite dissimilar airfoil shapes at tip and root. Each airfoil is interpo- lated with just twenty points.	144
4.13	The Advanced NURBS GUI	145

LIST OF FIGURES

4.14	A complex configuration built to verify the new algorithm capabilities. The CA-TIA visualization option is the <i>Shading with edges</i> one. As shown, no curvature discontinuities is detected in the inner region of the surfaces.	148
4.15	A PrandtlPlane Canadair configuration.	148
4.16	A PrandtlPlane Canadair configuration.	149
4.17	The PrandtlPlane ULV Tandem.	149
4.18	The body of an Helicopter similar to the Agusta A109	150
5.1	An example of two different class of surface grids.	155
5.2	Correspondence between parametric and physical space mesh.	156
5.3	ASD main window	157
5.4	Structured mesh for a not simply connected surface: the network is split in two sub networks representing simply connected regions. The common edge points are marked in red.	159
5.5	Correct distribution of points at the common edge of two networks.	159
5.6	Wing pierced from tfillet of the same logical subset.	161
5.7	Subdivision of a complex configuration in logical subsets. The color of the name of the features identifies the different subsets elements.	161
5.8	Uniform and cosinusoidal chordwise distributions for a logical subset composed of two wings and one wround feature.	163
5.9	Process of meshing the logical subset of the complete aircraft surface.	164
5.10	Structured mesh for a not simply connected surface: the network is split in two sub networks representing simply connected regions. The common edge points are marked in red.	165
5.11	Process of connecting two logical subset (characterized by blue and black grid color) trough a tfillet-wing connection. The red dashed lines represent the mesh lines induced from the tfillet.	166
5.12	Subdivision in simply connected regions	167
5.13	Different body mesh obtained with two different values of <i>deviation from straightness</i> parameter.	167
5.14	Mesh lines (in red and blue) induced from the fillet grids in the body	168

5.15	Different body mesh obtained with two different values of the body panel maximum aspect ratio parameter.	168
5.16	ASD main window	169
5.17	Grid Generation GUI: note the <i>Mesh stats</i> panel, reporting grid statistics	171
5.18	Plot of panels aspect-ratio. Note how this feature could be used to see the overall mesh parameter, or to look for mesh panels satisfying a user defined filter.	172
5.19	Plot of the outgoing normal vectors of the panels.	173
5.20	Grid of a PrandtlPlane configuration.	173
5.21	Grid of a PrandtlPlane configuration.	174
5.22	Grid of a conventional aircraft configuration.	174
5.23	Grid of a PrandtlPlane configuration	175
6.1	Body \mathcal{S}_B and wake \mathcal{S}_W surfaces.	183
6.2	Abutment between network 1 and 2. Note the panels with matching and not matching points at abutment. Note also the order of description of rows and columns, consistent with a correct outward surface normal	193
6.3	Paneling of two big transport aircrafts. Note the particular wakes arrangements for both the configurations.	195
6.4	Pan Air study of the high-lift system of the Boeing 737-300.	196
6.5	Boeing 747 with Space Shuttle Orbiter.	197
6.6	Paneling of a F15 fighter.	198
6.7	Pan Air surface pressure maps at Mach 0.6 and 4° angle of attack for a SR-71 aircraft.	198
6.8	Sailing both paneling	199
7.1	The Pan Air input generator GUI	202
7.2	Angle of attack α and yaw angle β	203
7.3	Advanced printout control.	204
7.4	Advanced abutment control.	205
7.5	PEA processing for simple abutment of two network edges (from [53]).	206
7.6	The wake generator GUI.	206
7.7	Placing a wake behind a wing in the wing generator GUI	207

LIST OF FIGURES

7.8	The connection wake between wing wake and fuselage.	208
7.9	The flow field properties GUI.	209
7.10	Definition of a grid of points where to evaluate flow properties.	209
7.11	Main PostProcessor GUI with shown statistics and results.	210
7.12	Configuration mesh.	211
7.13	Coefficient pressure (C_p) and local mach number for two different solutions, one at angle of attack of 0° , the other at 10°	212
7.14	Plot of C_L , C_{Di} , C_{M_y} versus α	213
7.15	Streamlines.	213
7.16	$C_l - y$ for the wings and bulk of the <i>Tandem</i> configuration at $\alpha = 10^\circ$	214
7.17	Pressure coefficient over a traditional configuration aircraft (A310 like) for zero angle of attack.	215
7.18	$C_l - y$ for the main wing and winglet (blue), tail (red) at $\alpha = 10^\circ$	215
7.19	Streamlines at $\alpha = 5^\circ$	216
7.20	Wing tested in the report [58].	217
7.21	Geometrical definition of the wing with ASD.	217
7.22	The meshed wing obtained with the structured grid generation module.	218
7.23	Comparison between numerical and experimental results, for $M = 0.17$ and $\alpha =$ 12.5. In spite of the $C_{L_{\max}}$ configuration, the agreement is very good.	219
7.24	Pressure coefficient (in red) versus chord at different spanwise sections, for $\alpha = 7^\circ$. The airfoil sections are drawn in black.	220
7.25	Grid sensitivity. The grid is refined in both chordwise and spanwise directions. Chordwise the grid has a cosinusoidal distribution, spanwise the distribution is linear.	221

A

Evaluation of the *stiffness matrix*

The stiffness matrix (eq.(3.31)) must be evaluated in order to solve the linear system shown in equation (3.41). This matrix is defined as:

$$K_{ij} = \int_{u=0}^1 N_{i,p}^{(d)}(u) N_{j,p}^{(d)}(u) \, du \quad (\text{A.1})$$

Due to local support of the B-spline basis functions, at a given u , belonging the knot span $[u_j, u_{j+1}]$, at most $p + 1$ of the generic basis functions $N_{i,p}$ are nonzero, namely $N_{j-p,p}, \dots, N_{j,p}$, hence:

$$K_{ij} = \int_{u_j}^{u_{i+p+1}} N_{i,p}^{(d)}(u) N_{j,p}^{(d)}(u) \, du \quad (\text{A.2})$$

where $j > i$. If $j \geq i + p + 1$ then $K_{ij} = 0$.

Further, eq.(A.2) can be simplified by:

$$K_{ij} = \sum_{k=j}^{i+p} \int_{u_k}^{u_{k+1}} N_{i,p}^{(d)}(u) N_{j,p}^{(d)}(u) \, du \quad (\text{A.3})$$

B-spline basis derivatives $N_{i,p}^{(d)}$ are piecewise polynomial of order $p - d$, thus the terms inside the integral are polynomials of order $2(p - d)$. The Gaussian quadrature

can be used to evaluate exactly eq.(A.3). In fact, it states that:

$$\int_{-1}^1 f(t) \, dt \approx \sum_{i=1}^n w_i f(t_i) \quad (\text{A.4})$$

where t_i are designated evaluation points (called Gaussian points), and w_i are the prescribed weight of that point in the sum. The order of precision of the formula is of $2n - 1$, thus, if $f(t)$ is a polynomial of order p , at least $\frac{p+1}{2}$ points are needed for an exact evaluation of the integral (clearly n must be an integer number, so it must be rounded if p is even).

Then, the integral of eq.(A.3) can be expressed from:

$$\int_{u_k}^{u_{k+1}} N_{i,p}^{(d)}(u) N_{j,p}^{(d)}(u) \, du = \frac{1}{2} (u_{k+1} - u_k) \sum_{l=1}^n w_l \left(N_{i,p}^{(d)}(\bar{u}^l) N_{j,p}^{(d)}(\bar{u}^l) \right) \quad (\text{A.5})$$

where w_l are the weights, and \bar{u}^l are the points, corresponding to the Gaussian points, where to evaluate the basis function derivatives, and are obtained through the transformation needed for changing the integration limits from $[u_k \quad u_{k+1}]$ to $[-1 \quad 1]$:

$$\bar{u}^l = \frac{1}{2} ((1 - t_l)u_k + (1 + t_l)u_{k+1}) \quad (\text{A.6})$$

In this way, the problem of evaluating the stiffness matrix brings back to summation and multiplication of basis function derivatives evaluated at points set from the Gauss quadrature.

$$K_{ij} = \sum_{k=j}^{i+p} \sum_{l=1}^n \frac{1}{2} (u_{k+1} - u_k) w_l \left(N_{i,p}^{(d)}(\bar{u}^l) N_{j,p}^{(d)}(\bar{u}^l) \right) \quad (\text{A.7})$$

Bibliography

- [1] IDS Ingegneria dei sistemi. *ASD Aerodynamic Surface Design, User Manual*.
- [2] F. Petri. Sviluppo del codice a.s.d. per la generazione parametrica di superfici aerodinamiche mediante nurbs. Master's thesis, Università degli Studi di Pisa, Pisa, Italy, 2005.
- [3] A. Rimondi. Generazione di configurazioni aerodinamiche mediante NURBS. Master's thesis, Università degli Studi di Pisa, Pisa, Italy, 2003.
- [4] IDS Ingegneria dei sistemi. *Documento di Architettura: ASD Aerodynamic Surface Design*.
- [5] Ira H. ABBOTT and Albert E. von DOENHOFF. *Theory of wing sections. Including a summary of airfoil data*. Dover, New-York, 1959.
- [6] G. Farin, J. Hoschek, and M. Kim. *Handbook of Computer Aided Geometric Design*. Elsevier, 2002.
- [7] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design*. Academic Press Professional, Inc., San Diego, CA, USA, 1997.
- [8] Les Piegel and Wayne Tiller. *The NURBS book (2nd ed.)*. Springer-Verlag New York, Inc., New York, NY, USA, 1997.
- [9] E. Dimas and D. Briassoulis. 3D geometric modelling based on nurbs: a review. *Adv. Eng. Softw.*, 30(9-11):741–751, 1999.
- [10] M. Pourazady and X. Xu. Direct manipulation of B-Spline and NURBS curves. *Advances in engineering software*, (31):107–118, 2000.

- [11] Hiroshi Akima. A new method of interpolation and smooth curve fitting based on local procedures. *J. ACM*, 17(4):589–602, 1970.
- [12] G. Albrecht, J. P. Bécar, G. Farin, and D. Hansford. On the approximation order of tangent estimators. *Comput. Aided Geom. Des.*, 25(2):80–95, 2008.
- [13] C. Lim. A universal parameterization in B-spline curve and surface interpolation. *Computer Aided Geometric Design*, (16):407–422, 1999.
- [14] R. C. Veltkamp. Survey of continuities of curves and surfaces. *Computer Graphics forum*, 11:93–112, 1992.
- [15] Brian A. Barsky and Anthony D. DeRose. Geometric continuity of parametric curves. Technical report, Berkeley, CA, USA, 1984.
- [16] Brian A. Barsky and Anthony D. DeRose. Three characterizations of geometric continuity for parametric curves. Technical Report UCB/CSD-88-417, EECS Department, University of California, Berkeley, May 1988.
- [17] Anthony D. DeRose. *Geometric Continuity: A Parametrization Independent Measure of Continuity for Computer Aided Geometric Design*. PhD thesis, EECS Department, University of California, Berkeley, Aug 1985.
- [18] M.P. do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice-Hall, 1976.
- [19] Barry Fowler and Richard Bartels. Constraint-based curve manipulation. *IEEE Comput. Graph. Appl.*, 13(5):43–49, 1993.
- [20] Barry Fowler. Geometric manipulation of tensor product surfaces. In *SI3D '92: Proceedings of the 1992 symposium on Interactive 3D graphics*, pages 101–108, New York, NY, USA, 1992. ACM.
- [21] George Celniker and Will Welch. Linear constraints for deformable non-uniform b-spline surfaces. In *SI3D '92: Proceedings of the 1992 symposium on Interactive 3D graphics*, pages 165–170, New York, NY, USA, 1992. ACM.
- [22] Wieger Wesselink and Remco C. Veltkamp. Interactive design of constrained variational curves. *Comput. Aided Geom. Des.*, 12(5):533–546, 1995.
- [23] G.D. Birkhoff. *Aesthetic Measure*. Harvard Univ. Press, 1933.
- [24] Henry Packard Moreton. *Minimum curvature variation curves, networks, and surfaces for fair free-form shape design*. PhD thesis, Berkeley, CA, USA, 1992.

- [25] Henry P. Moreton and Carlo H. Séquin. Functional optimization for fair surface design. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 167–176, New York, NY, USA, 1992. ACM.
- [26] George Celniker and Dave Gossard. Deformable curve and surface finite-elements for free-form shape design. In *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pages 257–266, New York, NY, USA, 1991. ACM.
- [27] William Welch and Andrew Witkin. Variational surface modeling. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 157–166, New York, NY, USA, 1992. ACM.
- [28] P. A. Sherar and P. A. Sherar. *Academic Year 2003-2004*. PhD thesis, 2004.
- [29] K. Washizu. *Variational Methods in Elasticity and Plasticity*. Pergamon Press, Oxford (etc.), 1975.
- [30] O.C. Zienkiewicz and R.L. Taylor. *The Finite Element Method: The Basis, Volume 1*. Butterworth-Heinemann Ltd, 2000.
- [31] T. Belytschko, W.K. Liu, and B. Moran. *Nonlinear Finite Elements for Continua and Structures*. John Wiley & Sons.
- [32] Stefanie Hahmann and Gershon Elber. Constrained multiresolution geometric modeling. In *Advances in Multiresolution for Geometric Modeling*, 2004.
- [33] Joe F. Thompson. A reflection on grid generation in the 90s: Trends, needs and influences. In *5th International Conference on Numerical Grid Generation in Computational Field Simulations*, pages 1029–1110. Mississippi State University, 1996.
- [34] T. Williams, N. Nadenthiran, H. Thornburg, and B.K. Soni. Genie: A multi-block structured grid system. Technical Report 19960029271, Marshall Space Flight Center, Nasa, March 1996.
- [35] A. Khamayseh and B. Hamann. Elliptic grid generation using nurbs surfaces. *Comput. Aided Geom. Des.*, 13(4):369–386, 1996.
- [36] J.F. Thompson, Z.U.A. Warsi, and C.W. Mastin. *Numerical Grid Generation: Foundations and Applications*. Elsevier North-Holland, Inc., New York, NY,

- USA, 1985.
- [37] J.S. Mathur. Grid generation for aerospace applications. *Aeronautical Society of India*, 1999.
 - [38] J.C. Yang, B.K.Soni, R.P. Roger, and S.C. Chan. Structured adaptive grid generation using algebraic methods. Technical Report 19950016997, Marshall Space Flight Center, Nasa, 1993.
 - [39] J.C. Yang and B.K. Soni. Algebraic grid adaptation method using non-uniform rational B-spline surface modeling. Technical Report 19920015179, Marshall Space Flight Center, NASA, 1992.
 - [40] C.B. Craidon. A description of the Langley wireframe geometry standard (LaWGS) format. Technical Report TM 85767, NASA, February 1985.
 - [41] H.W. Liepmann and A. Roshko. *Elements of Gasdynamics*. John Wiley & Sons, 1962.
 - [42] A. Epton and A. Magnus. A computer program for predicting subsonic and supersonic linear potential flows about arbitrary configurations using a higher order panel method. Volume I - Theory Document. Technical Report CR 3251, NASA, 1992.
 - [43] M.T. Landahl. *Unsteady Transonic Flow*. Cambridge University Press, 1989.
 - [44] L. Ward. *Linearized Theory of Steady High-Speed Flow*. McGraw-Hill, 1955.
 - [45] L. Morino. Boundary integral equation in aerodynamics. *Appl. Mech. Rev.*, (46), 1993.
 - [46] R. Kress. *Linear Integral Equations*. Springer-Verlag, 1989.
 - [47] O.D. Kellogg. *Foundations of Potential Theory*. Dover Publications, Inc., 1929.
 - [48] G. Bernardini. *Problematiche Aerodinamiche Relative alla Progettazione di Configurazioni Innovative*. PhD thesis, Politecnico di Milano, Nov 1999.
 - [49] J. Katz and A. Plotkin. *Low Speed Aerodynamics*. Cambridge University Press, 1991.
 - [50] G. Buresti. Fluidodynamics course material. Università di Pisa.
 - [51] A.R. Dusto and M.A. Epton. An advanced panel method for analysis of arbitrary configurations in unsteady subsonic flow. Technical Report CR 152323, NASA, July 1980.

- [52] F.T. Johnson, S.S. Samant, M.B. Bieterman, R.G. Melvin, D.P. Young, J.E. Bussoletti, and C.L. Hilmes. TranAir: A full-potential, solution-adaptive, rectangular grid code for predicting subsonic, transonic, and supersonic flows about arbitrary configurations. Technical Report CR 4349, NASA, 1992.
- [53] G. Saaris. A502i user's manual - Pan Air technology program for solving problems of potential flow about arbitrary configurations. Technical Report D6-54703-TN, The Boeing Company, February 1992.
- [54] K.W. Sidwell, P.K. Baruah, J.E. Bussoletti, R.T. Medan, R.S. Conner, and D.J. Purdon. A computer program for predicting subsonic and supersonic linear potential flows about arbitrary configurations using a higher order panel method. Volume II - User Manual. Technical Report CR 3252, NASA, 1992.
- [55] R.T. Medan, A.E. Magnus, K.W. Sidwell, and M.A. Epton. A computer program for predicting subsonic and supersonic linear potential flows about arbitrary configurations using a higher order panel method. volume III - Case Manual. Technical Report CR 3253, NASA, 1992.
- [56] F.T. Johnson, E.N. Tinoco, and N.J. Yu. Thirty years of development and application of CFD at Boeing Commercial Airplanes, Seattle. In *16th AIAA Computational Fluid Dynamics Conference*. AIAA, June 2003.
- [57] T.R. Moes, B.R. Cobleigh, T.R. Connors, T.H. Cox, S.C. Smith, and N. Shirakata. Wind-tunnel development of an SR-71 aerospike rocket flighttest configuration. Technical Report TM 4749, NASA, June 1996.
- [58] J.C. Sivells. Experimental and calculated characteristics of three wings of naca 64-210 and 65-210 airfoil sections with and without 2° washout. Technical Report TN 1422, NASA, 1947.