STRATEGY DEVELOPMENT FOR OBJECT–ORIENTED MULTIBLOCK GRID GENERATION AND ADAPTATION TO COMPLEX GEOMETRIES

YANG XIA

A thesis submitted in partial fulfillment of the
requirements of the University of Greenwich
for the Degree of Doctor of Philosophy

This research programme was carried out
in collaboration with the
Department of Parallel Computing,
Center of Logistics and Expert Systems

May 2001

Acknowledgements

I particularly thank my supervisors, Professor Dr Jochem Häuser, head of the Department of High Performance Computing, CLE, Germany, Professor Dr Mark Cross, and Dr Mayur K. Patel, School of Computing and Mathematical Sciences, University of Greenwich, London, U.K., for their support of my research and scientific advice during the writing of the thesis. I am indebted to Professor Häuser for his special education in grid generation, and continuous advice, as well as insights and expertise for several years. I am also indebted to Dr Patel for the benefit of profound discussion on special issues in grid adaptation, as well as his insights and expertise during the writing of the thesis. My thanks also go to Professor Dr Karl Bruns, Dean at the Faculty of Transport and Traffic of the University of Applied Sciences, Braunschweig–Wolfenbüttel, Germany, for financial support during this work.

Special thanks are due to Dr Peter R. Eiseman, PDC New York, U.S.A., for the discussion on strategy development in grid generation, and reading part of manuscript. I deeply thank Dr Zheming Cheng, PDC New York, U.S.A., for numerous discussions on special issues in complex geometries.

I would like to express deep gratitude to Mr. Jean Muylaert, head of the Aerothermodynamics Section, ESA ESTEC, Noordwijk, the Netherlands, and his team. He gave me the opportunity to participate in various activities in his section during the past several years.

My friends, Professor Dr Franz P. Lang, Technical University of Braunschweig, Germany, Dr Astrid M. Stange, Boston Consulting, Düsseldorf, Germany, Professor Dr Dr h.c. Folker H. Wittmann and Dr Xinhua Zhang–Wittmann, ETH Zürich, Switzerland, deserve my thanks for their encouragement and moral support over many years.

I own thanks to the late Professor Klaus Simons, Technical University of Braunschweig, Germany, his former personal assistant, my friend Burkhard Rogler, GFI Seesen/Harz, Germany, and Professor Dr Horst Günter, Technical University of Braunschweig, Germany, for their generous personal help in the past.

Finally, my mother and my brother deserve my thanks for their encouragement while this thesis was written.

April, 2001

Dedicated to the memory of my father

Abstract

The first part of the thesis deals with the strategy development for multiblock structured grid generation to complex geometries. Based on the grid generation practices over years, a set of grid construction rules is developed to provide the CFD engineer an object–oriented method for grid design.

The essential core of the object–oriented method is to decompose a complex meshing task into a set of sub–tasks, which are treated individually at a lower level of both geometry and topology. The grid construction rules cover the questions of dealing with selection of meshing direction, generation of surface description and block topology building. To explain this grid design method, an example, dealing with a highly complex geometry, is demonstrated.

The second part of the thesis deals with the strategy development for multiblock structured grid adaptation. Since a grid can be adapted with or without a flow solution, the terminologies *passive* and *active grid adaptation* are introduced to describe a solution–dependent or a solution–independent grid adaptation.

Passive grid adaptation is performed by generating adequate block topologies, such that a local enrichment of grid points can be achieved. It consists of three concepts: *one–dimensional clustering of grid lines*, *block encapsulation*, and *smart block*. The method for solution–dependent grid adaptation is developed based on the idea of grid optimization. A grid is adapted by minimization of objective functions, in which relations among weight functions and grid line distribution are formulated. The measures for grid quality, such as smoothness, cell aspect ratio, and orthogonality, are formulated as control terms of the objective functions to improve grid quality. In addition, a concept of *smart cell* used for solution–dependent grid adaptation is proposed in this thesis.

Notation

$a_{i,j,k}$	Sum of quadratic weight functions
$b_{i,j,k}$	Sum of products of quadratic weight functions and parameter coordinates
$\mathbf{e}_i$	Covariant basis vectors
$\mathbf{e}^i$	Contravariant basis vectors
$f_{i,j,k}$	Objective function for grid adaptation with control terms
$f^a_{i,j,k}$	Objective function for grid adaptation
$f^l_{i,j,k}$	Control term for cell aspect ratio
$f^o_{i,j,k}$	Control term for cell orthogonality
$f^s_{i,j,k}$	Control term for cell smoothness
$\mathbf{g}^{(n)}$	Gradient of an objective function at n–th search step
g	Determinant of $\|g_{ij}\|$
g_{ij}	Covariant components of the metric tensor
g^{ij}	Contravariant components of the metric tensor
$\mathbf{l}_1, \mathbf{l}_2, \ldots, \mathbf{l}_n$	Cartesian basis vectors
$p_{i,j,k}$	Grid point in physical space
$\hat{p}_{i,j,k}$	Grid point in parametric space
$w_{i,j,k}$	Weight function
$x^1, x^2, \ldots, x^n$	Cartesian coordinates
$\mathbf{H}$	Hessian matrix
P, Q, R	Source terms of the Poisson equation
$\mathbf{S}^{(n)}$	Search direction in n–th search step
α_i	Leading coefficients of a weight function
$\alpha^{(n)}$	Search step in n–th search iteration
$\alpha^{(n)}_g$	Global search step in n–th search iteration
$\beta^{(n)}$	Correct factor of search direction in n–th search iteration
γ_i	Penalty factors of an objective function
$\delta_{ij},\ \delta^j_i$	Kronecker symbols
$\xi^1, \xi^2, \ldots, \xi^n$	General curvilinear coordinates
λ	Size proportion of smart cell to the initial cell
σ	Difference of flow variables over cell diagonal
$\phi_{i,j,k}$	Weight variable to be normalized
ϕ_{min}	Global minimum of weight variables
ϕ_{max}	Global maximum of weight variables
$\psi_{i,j,k}$	Weight variable normed
ω	Relaxation factor
Γ^k_{ij}	Christoffel symbol

Contents

Chapter 1

Motivation of the Thesis

Recent advances in computational fluid dynamics have provided new possibilities for flow simulation with both increasingly complex physics and geometries. Flow fields that include complicated physical phenomena such as shock waves, flow separation, and combustion or heat transfer processes, are simulated by numerically solving partial differential equations and visualizing their solutions using powerful computer facilities. Flow simulations are used at different stages of design and optimization processes in many fields. They provide engineers a visual understanding of flow features, so that their influence can be considered during the engineering design. Given ever increasing geometric complexity, it is becoming more complicated to simulate the overall flow behavior and flow characteristics. One of the major difficulties is generating a computational grid whose grid lines are aligned with the geometric shape as well as the prevailing flow direction, and whose grid density provides sufficient resolution to capture the flow features of interest.

Numerical grid generation is a preprocess of decomposing a given solution domain in the form of a mesh. Solution domains with complex geometries can either be meshed by unstructured grids or structured grids. Unstructured grids are formed with no relation to coordinate directions, while structured grids are formed by the intersections of curvilinear coordinate surfaces [91].

In general, an unstructured grid consists of tetrahedral cells in the most basic form in three–dimensional space, but may be made of mixed hexahedral or tetrahedral elements [85]. Unstructured grid generation requires a large memory demand for both cells and relations among them. It would be hard to conclude that purely unstructured meshes can provide better solution although sometimes they do [96], [106]. However, the time saving and the ease of grid generation

for complex geometries shows a high efficiency in flow simulation [2], [59], [75], [91], [100]. It is therefore widely used in different industrial fields due to these advantages.

Multiblock structured grids that are unstructured globally with respect to block connectivity, but structured locally within a block enable the user to divide a complete solution domain into a set of subdomains that will be meshed with a low degree of geometric complexity. Because of its uniform data structure of each single block, the solution process can be performed by computing a single structured block, and then updating the data on block boundaries [41].

Although there are advantages using structured grids in flow simulations, the grid generation itself is a more complicated process. It requires a comprehensive knowledge of both geometry and topology, as well as flow physics. Another complicated task is to adapt a multiblock grid, since grid points within the solution domain and on a fixed boundary have different geometric constraints. Grid adaptation has to be performed with respect to the geometric features on the fixed boundary, as well as the coupling on block to block interfaces. The motivation of the present thesis is to develop a general strategy for structured grid generation as well as grid adaptation for highly complex geometries.

1.1 Challenges in Structured Grid Generation

A general process of grid generation consists of geometry data preparation, block structure building, *block building* or *grid topology building*, and grid generation, as depicted in Fig. 1.1. In order to develop the new strategy for structured grid generation for highly complex geometries, the general challenges are briefly addressed below.

1.1.1 Surface Description

In general, geometric objects of high complexity are modelled by CAD tools. Shapes of these objects are described by a set of ruled or approximated surfaces. They can be represented by wireframe or solid models, as depicted in Fig. 1.2. A wireframe model is given by representing the boundaries of all surfaces, while a volume model is represented by a closed surface of the body as the result from

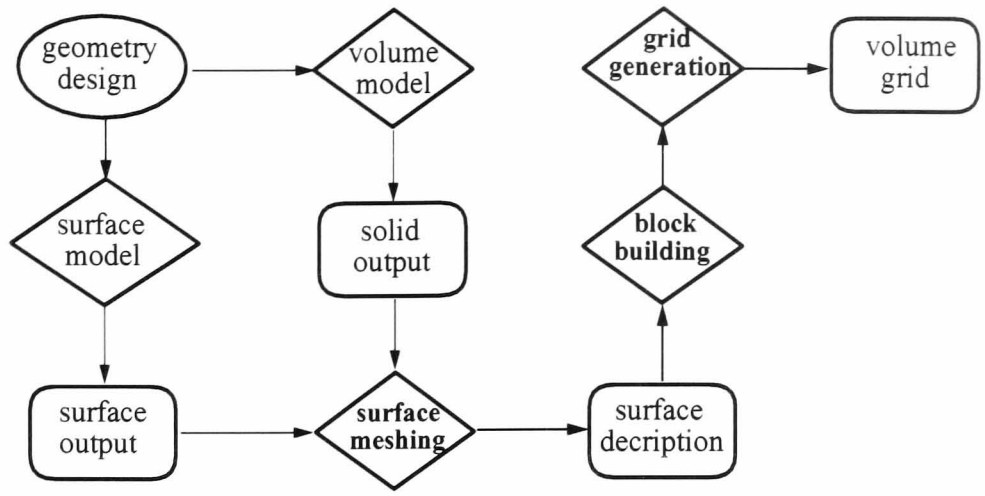


Figure 1.1: **The flow chart explains the process from CAD geometry design to grid generation. In this diagram, the human interaction, process and data are represented by symbols ellipsoid, parallelogram and rectangle with rounded off vertices, respectively.**

the union of all solids. Since CAD output does not give the complete surface description of a volume model, the boundary description of a meshing domain will be obtained by extracting the geometric data of the closed surface of a CAD volume model. A very general question dealing with modelling a meshing domain is

- *Can an arbitrary geometric configuration modelled be used for building a meshing domain?*

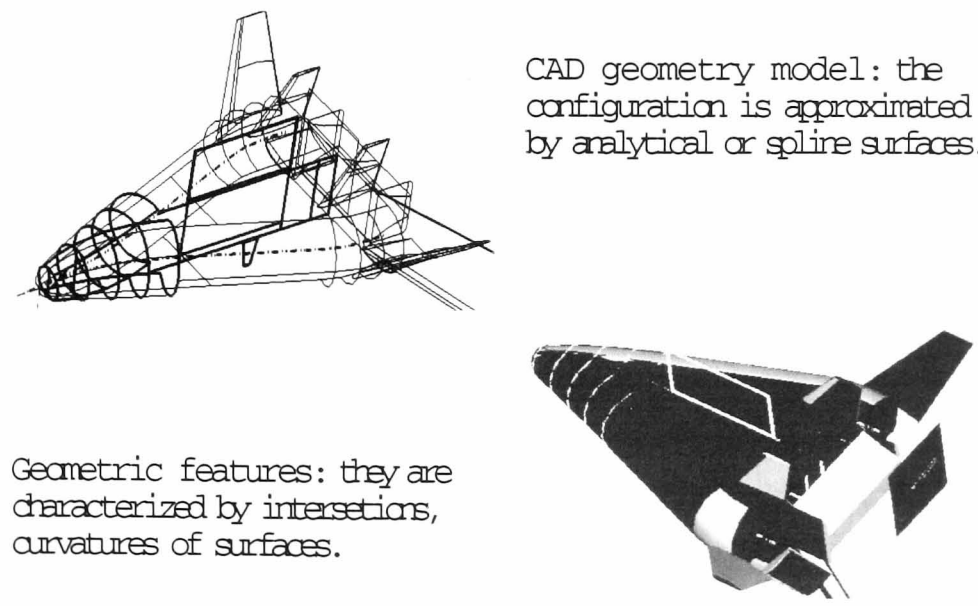


Figure 1.2: **A CAD model for a generic vehicle. The geometry is represented by a wireframe and a volume model. The shape features of geometry are described implicitly in the CAD output data.**

Shape features such as tangent planes, the principal curvatures, sharp edges, or intersection curves, are contained in the CAD geometric data. They characterize

the geometric complexity of a solid model. If a boundary description for the body, to be meshed, is given in the form of a set of points which represents a closed surface of the CAD volume model, in general, it may be difficult to generate blocks from complex geometries, e.g. in Fig. 1.3. It is convenient to decompose a complex surface geometry into parts. The question dealing with decomposing a surface geometry is

- *How can one divide a complex geometry into a set of parts?*

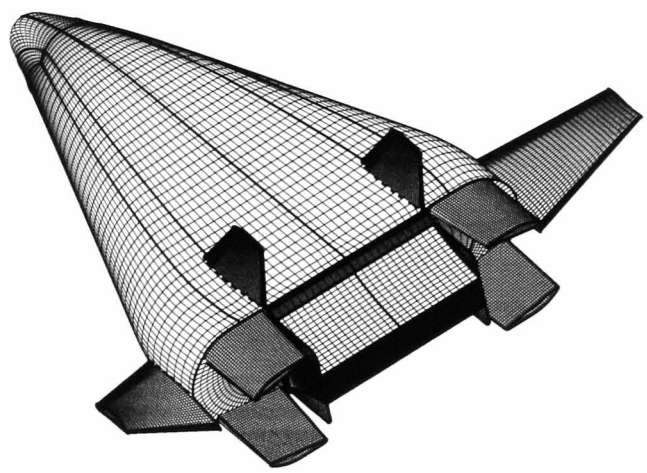


Figure 1.3: **The geometric configuration, modelled by the CAD tool, contains many shape features, such as sharp edges, intersection curves. In order to mesh them exactly, the complete surface is decomposed and meshed.**

1.1.2 Grid Topology Building

Grid topology, also known as *wireframe topology* or *block topology*, describes connectivity relations among blocks. A grid topology indicates the decomposition of a given solution domain, which can be represented by the topological elements vertices, edges, and faces. A block topology has an impact on mesh quality, e.g., on shapes of solid boundaries or curvature of grid lines. The influence of grid topology on mesh quality as well as on numerical accuracy of flow simulations has two main points [102].

▷ **Consistency of wireframe model with geometry**. A wireframe model should reflect geometric features. Certain geometric shapes have to be modelled by choosing adequate block topologies, shown in Fig. 1.4, since the consistency of a wireframe model with geometric shapes has an influence on grid quality.

▷ **Alignment of grid lines**. Blocks should be generated with respect to flow features. For instance, alignment of grid lines with flow directions and orthogonality of grid lines on body surfaces are dependent upon block contour to a large extent.

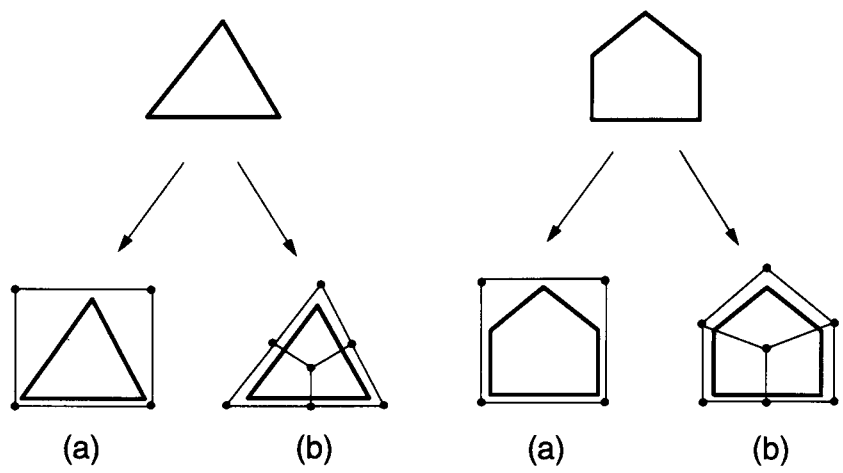


Figure 1.4: **Consistency of wireframe models with geometric shapes. In cases (a), the wireframe models do not reflect geometric shapes, grids generated from these block topologies may have the reduced quality.**

A rational block generation (domain decomposition) should consider geometric features, block arrangement, and grid line alignment together with the flow direction. Since there are no construction rules containing reduction of geometric complexity, generation of surface description and grid topology, it is difficult for many CFD engineers to generate grids exhibiting extremely high complexity in a short time period. Efficiency of grid generation depends substantially on their experience.

- *Is it possible to establish a set of grid construction rules, such that a multiblock structured grid will be efficiently generated?*

1.1.3 Grid Adaptation

Clustering of grid points is only necessary in regions where a high resolution is required. Grid density can be varied using different methods. A well known approach is the use of solution–adaptive grid generation. This prevents a local change in grid density from destroying the global grid structure. For instance, a Navier–Stokes computation requires a high density of grid lines near a solid wall. The grid line distribution for resolving the boundary layer is generated according to a prescribed stretching function. Since this flow feature is independent of

other flow phenomena, such as a bow shock, grid adaptation for different flow phenomena should be performed separately. The requirements for local grid density and separate treatment of flow phenomena leads to the consideration:

- ▷ **1. Heterogeneity of grid density**. A high grid density is often required in location, where a high gradient of flow variables can be computed. Although a global uniform aspect ratio of grid cells can reduce global numerical errors, it causes large memory demand and computing time. Heterogeneity of grid density requires a special grid generation method.
- ▷ **2. Re–usability of grids**. Local modification of grid topology, in cases of complex geometries, should not require a large effort. It is economical to generate a grid which is modular, i.e., it consists of a set of individual meshing objects that can be added to or removed from the existing grid structure according to the requirements of the flow simulation.

The questions arising in grid generation practices are

- *How does one generate a reusable grid topology that can be flexibly modified?*

and

- *How does one generate a grid whose grid density is globally heterogenous?*

1.2 Current Status of Structured Grid Generation

Industrial expectation for grid generation and CFD turn around time are that a CFD engineer generates a grid within one hour [47], and performs the simulation within one day for a special configuration, e.g., a turbine [48]. These challenges require not only more powerful tools for surface geometry modelling and grid topology generation, but also new techniques for grid generation as well as the need for efficient flow solvers [34], [90].

1.2.1 Overview of Grid Generation

The topics of grid generation encompass not only the pure mathematical aspects, but also various applications and grid design strategies. In the following, an

overview of these topics is presented.

Grid generation algorithms. Various algorithms are developed to generate multiblock blocks. Typically, they are based on the knowledge of the partial differential equations, interpolation strategies, and algebraic methods [15], [45], [46], [54], [84], [86]. Research and development of grid generation algorithms focus on the following aspects.

- ▷ **Grid quality control**. Many control functions with respect to smoothness, orthogonality, cell aspect ratio, and line distribution are researched. In addition, some algorithms are developed by using optimization functions, in which grid quality is formulated [50], [51].
- ▷ **Automatic topology generation**. Traditionally, a multiblock grid is generated by inputing geometric data, generating a block wireframe structure and the complete block connectivity. The new techniques provide the possibility to semi–automatically generate block topologies [29], [33], [68], [70], [83].

Tool development. The tool development is leaning towards user–friendly operation, geometric and graphic functionality and intelligent processing. It can be summarized as follows.

- ▷ **Graphics tools**. The goal is to provide the user with a visual impression of shape properties. The user "feels" and recognizes geometric problems intuitively. The grid generation process is performed interactively [52], [76], [81].
- ▷ **Automatic building of objects**. Some functionality of CAD tools such as translation, rotation, deformation or duplication of objects is integrated or implemented into the grid generation tools so that the user interaction becomes easier [77], [93].
- ▷ **Preprocessing tools**. Grid generation tools provide special functionalities for surface preparation, e.g., surface repair, modification, and modelling. Data preparation for surface description of grid generation is becoming less dependent on CAD meshing tools [17], [79].
- ▷ **Postprocessing tools** Tools are being added into grid generators, such as generation of interfaces of grid data to solvers (grid topology file) or conversion of grid data (block splitting or merging) are an important part of grid generation tools [29].

Application. Numerous commercial and research tools are developed, which are able to provide grids with "high quality". Grid generation is needed in different simulation fields.

- ▷ **High complexity**. Generating configurations with high geometric complexity, such as turbines, automobiles, aircraft, or spacecraft.

- ▷ **High accuracy**. Some structured grids show high accuracy of geometric micro–components modelled in a sophisticated way [37].

Strategy development. One of the missing steps in grid generation is a strategy for developing rules for grid construction.

- ▷ **Availability of objects**. Preparation of adequate geometric and topological information is available and contributes to finding an approach to assembling and disassembling complete objects.
- ▷ **Flexibility of topology building**. A grid topology can be flexibly modified. Adding or subtracting parts, a new grid topology will be obtained. This flexibility requires designing standard topological components whose interfaces will be connected easier to other components [82], [87].
- ▷ **Topology database**. Structured sets of topological elements are expected to support the user in properly deciding how to build a partial grid topology. Object–oriented grid generation. A topology database provides the user with a powerful workbench for grid generation [29], [39].

1.2.2 Overview of Grid Adaptation

There are two main schemes of grid adaptation, i.e., grid refinement and grid redistribution. In the present work, the latter scheme is considered. Grid adaptation using a redistribution scheme can be performed by different methods. Partial differential equation methods use control functions as a measure for grid point movement. Incorporating weight functions, the modified partial differential equations redistribute grid points in the solution domain. The robustness of this method provides good grid quality. Since all values at fixed boundaries are initial values for solving the equations, the movement of grid points at fixed boundaries requires special numerical treatment. Algebraic methods formulate a relationship between solution gradients or curvatures and grid geometry. An adaptation process can be quickly performed. Variational methods formulate weight functions and measures for grid quality in the form of functionals. Grid adaptation is performed by finding extreme values of the variational equations. Successful cases are found in [3], [4], [42], [43], [53], [57], [63], [71], [80], and [97].

For multiblock grids with complex geometry, the adaptation strategy and its implementation are still in research phrases. The key problems in multiblock adaptation are to couple grid points across block interfaces, and determine the restrictions of grid point movement on physical boundaries. The concept of *degrees of freedom* is used to qualify restrictions in grid point movement [13].

The main purpose of this approach is to prescribe boundary conditions for grid points at physical boundaries [14].

Remaining problems. Robust algorithms can be selected from successful implementation of single block adaptation. The key problem of their extension into multiblock cases is the reliable design of the adaptation procedure [9].

- ▷ **Different degrees of freedom**. For multiblock grids, a movement of grid points is subject to geometric constrains. Grid points on physical boundaries or within have different degrees of freedom. A classification of grid points according to their degree of freedom produces three groups. For every group a special algorithm should be employed to perform adaptive computations.
- ▷ **Block interface**. It deals with data exchange among grid points at both sides of a block interface (face connectivity). This has an important significance to ensure a C^1–continuity of grid lines across the interface.

1.3 Objective and Scope of the Thesis

Different degrees of difficulties in grid generation and grid adaptation are addressed in the literatures [90]. Most of the publications concentrate on mathematical methods in grid generation. At present, no systematic research for grid construction rules exists. Besides, multiblock structured grid adaptation using a redistribution scheme is considered as a difficult task in grid generation. The difficulties not only depend on the mathematical formulation, but also on the implementation of algorithms that are valid in three–dimensional multiblock adaptation. The objectives of the thesis are

- ▷ the development of an advanced method of grid generation of high grid complexity for the CFD engineer, establishment of a set of grid construction rules, and
- ▷ the extension of the adaptation algorithm from monoblock to multiblock grids.

1.3.1 Grid Construction Rules

Grid construction rules contain methods and techniques to divide a meshing task into three steps, namely, conceptual, internal, and external tasks. The conceptual task deals with building models that represent general shapes of abstract

objects described on a generic level. The internal task is the individual topological description of objects through local wireframe building, while the external task is the collection of specifications of relationships between topological objects.

The main purpose of establishing a set of construction rules based on experience is to propose an object–oriented method for multiblock grid generation. As analyzed above, the core of these rules is to divide a compound geometry into partial components, and to build local and global topologies. The construction rules are developed, based on the following considerations:

- ▷ **Simplification of geometry**. A complete configuration is decomposed into a set of components, defined as meshing objects. Their shapes comprise the lower level of geometric complexity.
- ▷ **Extraction of geometric features**. Through a decomposition of the structure, the geometric features can be precisely modelled by a fine block structure.
- ▷ **Object–oriented topology design**. Components are modelled individually at the lower level. A complete grid topology results in generating topological relations between objects followed by the assembly of all objects.

Fig. 1.5 depicts an generic aircraft in decomposed form. It consists of three main components, namely, fuselage, wing, and engine. At the local level, wireframe models are built for all components. Its complete grid topology is obtained by assembling all meshing objects.

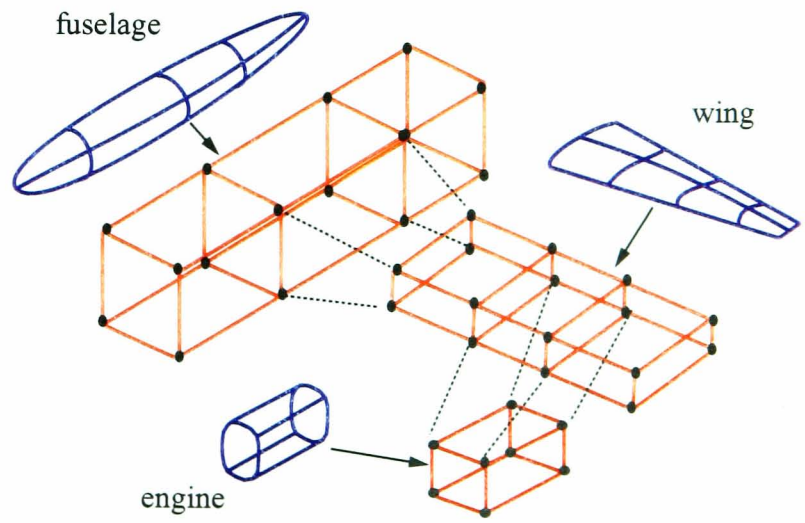


Figure 1.5: **An aircraft can be considered as a set of components. The components such as wing, fuselage and engine are built by their own wireframe topologies. Combining all components, an entire grid topology is obtained.**

1.3.2 Strategy for Grid Adaptation

Grid adaptation can be performed by generating a special block topology or by using a flow solution. These approaches are called *passive* and *active grid adaptation*, respectively.

Passive grid adaptation. A highly local grid density is generated without having an initial solution. This is realized by generating special block structures. It is a topology–oriented grid adaptation, and contains three concepts.

Concept 1: One–dimensional clustering of grid lines. A grid generation process is separated from a grid clustering process due to the following reasons. First, control of local grid density, such as grid line distribution in a boundary layer, requires an enormous computational intensity if done within the overall grid generation process. A special block structure, called *closed boundary layer topology* is employed to control grid line distribution within a set of closed blocks wrapping around the body surface. A high grid density is limited to one–dimension, namely, the fine distribution of the grid lines is generated in the direction off the fixed boundary. The grid density within the closed boundary layer will not be extended into the far field [38]. Second, the grid will be used not only for an Euler computation, but also for a Navier–Stokes computation. Thus, separation of grid generation and grid clustering can satisfy this requirement. Using an external cluster tool for enrichment, a Navier–Stokes grid is quickly generated based on the boundary layer topology, as depicted in Fig. 1.6.

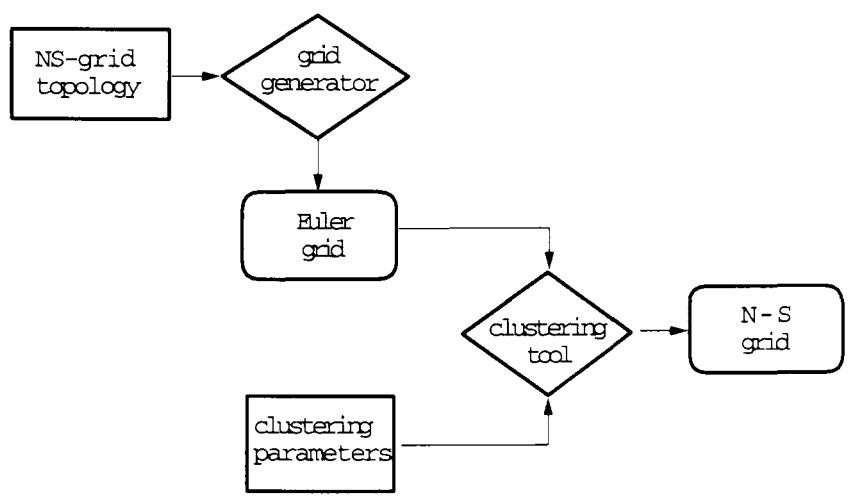


Figure 1.6: **One–dimensional boundary layer topology is used for generation of a N–S grid based on an existing Euler grid. In this diagram, input, process and output are represented by rectangle, parallelogram, and rectangle with rounded off vertices, respectively.**

Concept 2: Block encapsulation. A block encapsulation is a loop structure which ensures that grid lines remain within the local block structure. This block structure serves as a block refinement. Since the local block refinement is realized by introducing new blocks into an existing grid topology without changing the remainder of the grid topology, this method provides flexibility to improve the local block structure. Fig. 1.7 shows an example of a spatial block encapsulation of a grid for the generic space shuttle. A high grid density is necessary below the body flap for simulating flow re–attachment, while the grid density on the leeward side can be much lower.

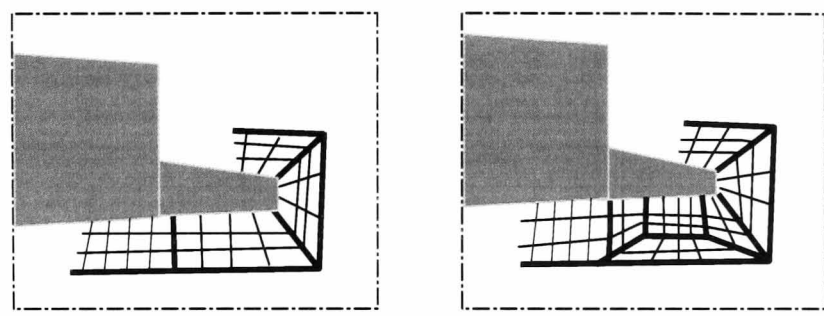


Figure 1.7: **Local block enrichment is generated by modifying the existing grid topology. The high density of grid points is reached by introducing spatial block encapsulation. This figure depicts a block encapsulation below a generic flap.**

Concept 3: Smart blocking. The targeted region is isolated by a set of blocks. The boundary of this region is fixed by a closed internal boundary. Cells within the targeted region can be much smaller than those outside. This block structure scales cell sizes in the solution domain. It is used to separate coarse and fine distribution of grid lines in a special region. In the example shown in Fig. 1.8 a grid density is concentrated near the region of the top of an antenna. Since this small component has to be modelled by a fine block structure, the smart blocking method is used to separate this region from its environment.

Active grid adaptation. Generation of solution–adaptive grids for flow computations aims at increasing numerical accuracy of the simulation in an economic way. By adapting an initial grid according to the flow features, a better local resolution can be obtained. Improved convergence behavior is reached by means of reducing numerical approximation errors through fine grid spacings

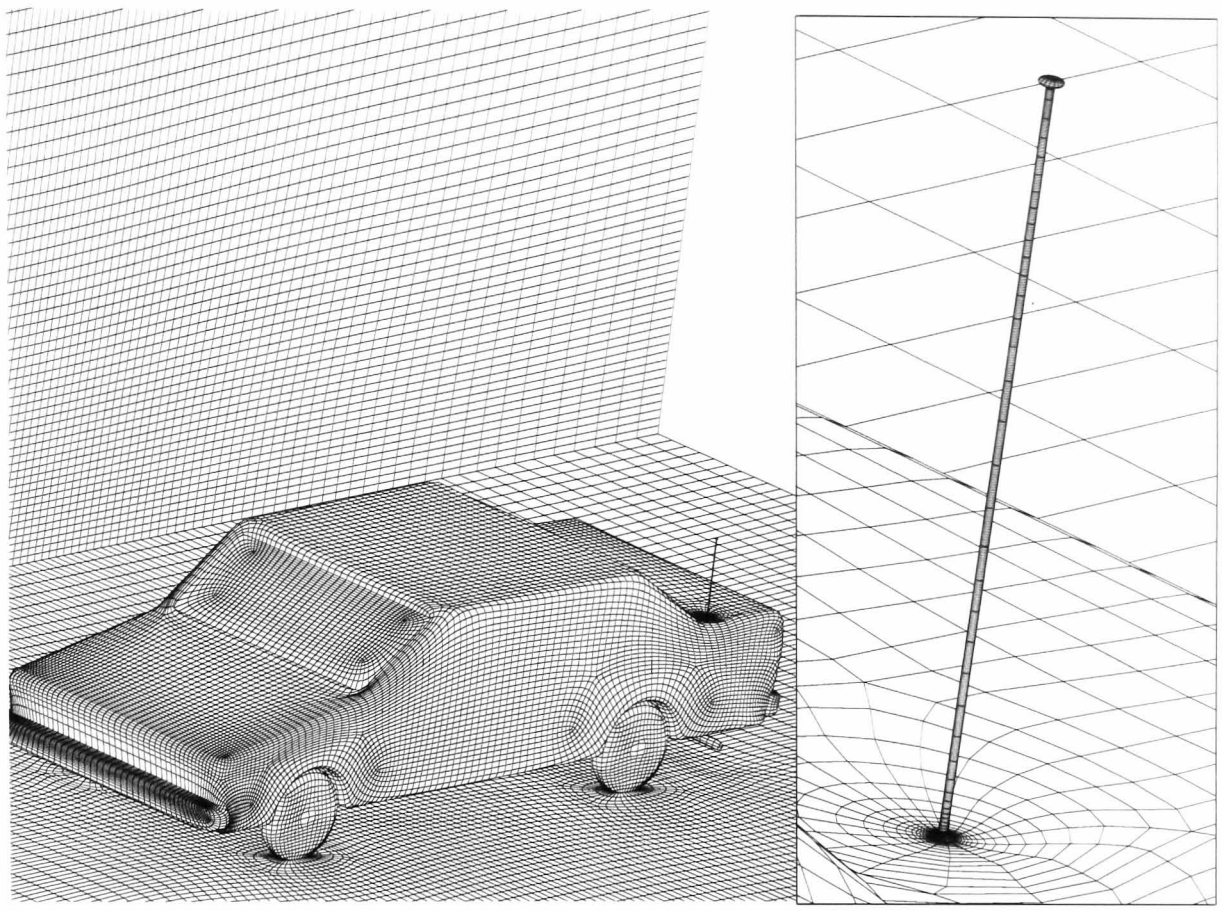


Figure 1.8: **A fine grid distribution is concentrated in special regions while the grid density in their environment is much lower. This separation is realized by using block encapsulation techniques.**

in these regions. This kind of grid adaptation is termed *active grid adaptation*, since grid density will be changed according to the flow solution.

The choice of a redistribution scheme in the present work is justified by the fact that such a scheme requires constant memory only and is well suited for flow simulations with an extremely large number of grid points. The main issues of grid adaptation using a redistribution scheme are addressed as follows.

- ▷ **Fixed boundary**. Shape features on fixed boundaries have to be retained during an adaptation process.
- ▷ **Cell quality**. Skewness or sheared cells are caused by unconstrained spatial movement of grid points.
- ▷ **Block interface**. Unsmoothness on block interfaces could be caused by an implementation failing in calculating the common points of both blocks.

There are geometric and topological restrictions for grid adaptation, i.e., grid points have their degrees of freedom according to their positions in the solution domain. The core of the strategy for multiblock grid adaptation is to use different methods to adapt grid points with respect to these constraints.

Chapter 2

Geometry Description for Grid Generation

The boundary of a meshing domain is generally termed *physical boundary.* It is represented by a set of geometry surfaces. Creation of the geometric surfaces used for defining the physical boundary of a meshing domain is termed *geometry description.* The topics of geometry description contain geometry modelling using CAD tools, CAD data conversion e.g., extraction of geometry surface data from a CAD solid model and data generation of geometry surfaces.

Not every geometric configuration can be used as a meshing object. In three–dimensional cases, for instance, it is required that a meshing domain must be a continuous solid and that its physical boundary can be described by a set of closed geometry surfaces. The first question to be answered in this chapter is:

- *Which geometric configuration is available for building a meshing domain?*

In cases of complex configurations, geometry surfaces are mostly generated by a set of line segments, curves, or surfaces. In order to reflect its complexity, a block wireframe should be built with respect to the geometric features. Dividing a complete geometry surface into a set of pieces, local block topologies can describe the shape features at a lower level of the geometric complexity. The question arising from this consideration is:

- *What measures are there for the decomposition of geometry surfaces?*

2.1 Geometric Objects for Grid Generation

In order to represent a geometric object which is available for building a solution domain, a formal description of a geometric object and its properties is given in the following. Only three–dimensional cases are discussed, while two–dimensional cases are considered as a reduction of the dimensionality.

2.1.1 Definition of Geometric Object

In $\mathbb{R}^3$, a set of points represents a collection of regions. Each region is continuous, and is bounded by a piece–wisely differentiable surface. The complete region is defined as a *geometric object* or *geometric model*. Each of pieces with its boundary is termed a *face segment*. A curve segment, results from two or more face segment intersection, is termed *edge*. A common point, at which two face segments joint, or a point in which two or more edges meet, is termed *isolated point*. At each point within a face segment, two unit vectors normal to the surface exit. One points into the region, the another points the outside of the region. An object is representable, when it has the following properties [78]:

- ▷ An object must occupy a finite volume and must have an invariant form, i.e., its shape is independent of the location or orientation.
- ▷ The boundary of an object must determine the content of the inside unambiguously.
- ▷ A complete solid must be dimensionally homogeneous, i.e., its boundary has not dangling edges or dangling faces.

Such an object can be used for building a solution domain. The geometry of the object is represented by its surface, described by a set of face segments. The face segments have to satisfy the following conditions, termed *validation conditions for meshing boundary representation*:

- ▷ A face segment of an object is a subset of the boundary of the object.
- ▷ The union of all face segments of an object gives the complete boundary of the object.
- ▷ A face must have a finite area and can be described in analytical or discrete form.

Fig. 2.1 shows two solids connected at an isolated point, and along an edge. The boundaries in both cases are not be a closed surface. Therefore they can not build a meshing domain.

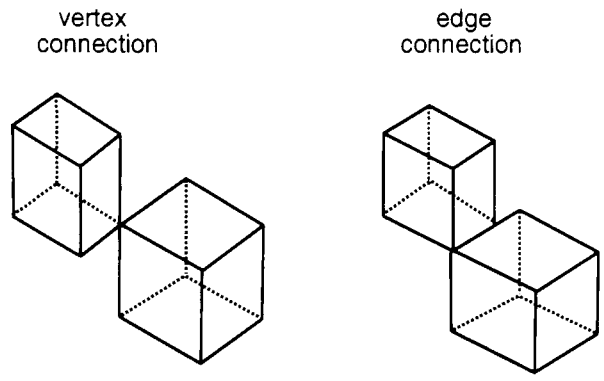


Figure 2.1: **Unavailable connections among solids.**

2.1.2 Representation of Complex Geometries

Geometric objects used for building a meshing domain are often simplified by modifying some geometric components or shape features. Moreover, it is desirable to model geometric objects using a set of ruled surfaces with respect to some quantitative information of shape features. To do this, the surfaces are classified into *geometry primitive* and *free–form surface.*

2.1.2.1 Geometry Primitive

Many engineering or manufacturing parts have similar shapes. A change of dimensions by transformation operators has an influence on the geometry, but not on the topology of their previous shapes. These shapes are termed *parametrized shapes.* Grouping them into families of some generic forms, termed *geometry primitives*, the individual instances of a family can be generated in an easy manner. The following geometry surfaces belong to the geometry primitives.

- ▷ **Analytical surface**. An analytical surface is described by a set of analytical equations.
- ▷ **Translational sweep surface**. A two–dimensional set A lying in a plane in space builds a surface. Let G be the solid swept by A, along a line segment B which is perpendicular to the plane A and has an endpoint on the plane, when translated parallel to its plane, along the trajectory B. The surface of the solid is termed *translational sweep surface* [78].
- ▷ **Rotational sweep surface**. A one–dimensional set A lies in a plane in the space. A line segment B is co–planar to A, and builds an axis. Let G be the solid swept by A, when it is rotated about B. The surface of the solid is termed *rotational sweep surface* [78].
- ▷ **General sweep surface**. A two–dimensional set A lying in a plane in space builds a surface. Let G be the solid swept by A, along some arbi-

trary path B, while the surface built by A may change its size, shape and orientation [65].

These sweep surfaces are shown in Fig. 2.2. The geometry primitives have some properties, and their geometry surfaces can be generated using algebraic methods.

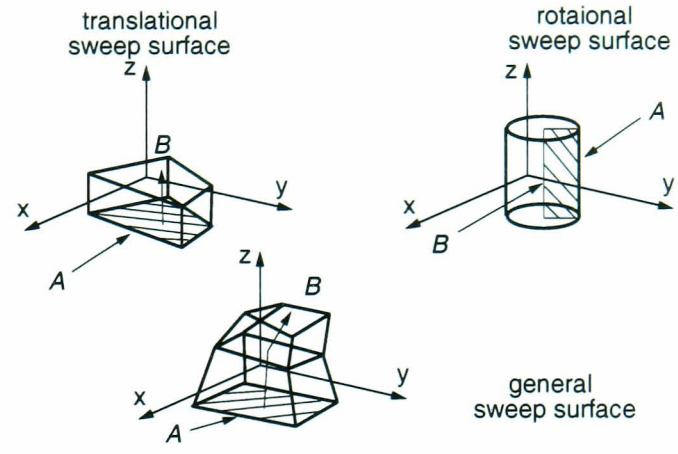


Figure 2.2: **Different types of sweep surfaces.**

2.1.2.2 Free–Form Surface

Many geometry surfaces can not be represented by geometry primitives. The geometries of these surfaces are irregular and are represented using piecewise discrete surfaces, or by means of computer tomography. They are termed *free–form surfaces.* Fig. 2.3 shows an example of a generic animal. The geometry surface of the animal is represented by a triangulated mesh, while its eyes are represented using geometry primitives.

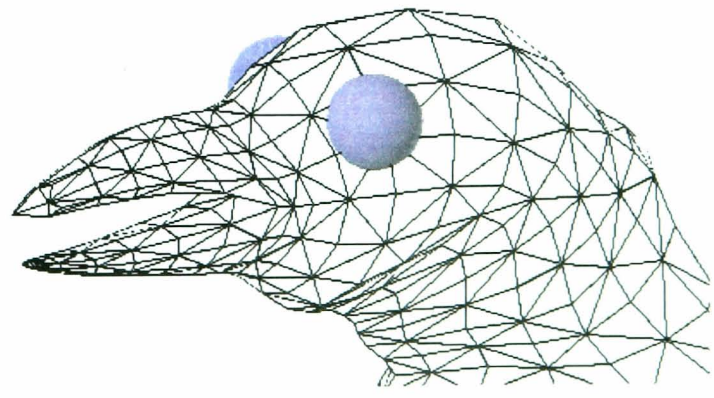


Figure 2.3: **Geometry surface of a generic animal, represented by a free–form surface in the form of triangulated mesh.**

2.2 Data Preparation for Geometry Surface

A meshing object is bounded by its physical boundary. In grid generation, the geometry surface of this boundary is used for describing the solution domain. A general way to generate the geometry data is to model a solid by means of CAD tools and generate its surface mesh. The surface mesh of a solid model is then used as *surface description* or *boundary description* in grid generation.

A closed surface mesh may have a highly geometric complexity, since shape features of the solid are implicitly contained in the mesh. On the basis of the above classification of geometry surfaces, a complex solid is considered as a collection of components, which are represented either by some geometry primitives, or by free–form surfaces.

2.2.1 Partitioning of a Complex Object

A complex meshing object is intentionally split into a set of parts. Some engineering geometric features are used for representing these parts:

- ▷ **Planar part**. The part is partitioned, if it is represented by a plane.
- ▷ **Sweep part**. The part is partitioned, if its geometry surface belongs to a sweep surface.
- ▷ **Spherical part**. The part is partitioned, if it is described by a sphere or an ellipsoid.

The remaining parts are represented by some curved and sculptured surfaces. The practical benefit of this engineering feature–based (also *object–oriented*) decomposition of the meshing object permits to generate geometry surface data using the geometry primitives as well as free–form surfaces for a complex configuration. The relations among parts split represent some shape features, which should be accurately reflected by a block topology.

Fig. 2.4 shows an example of extracting a cylinder from a given geometric object. The complete configuration is divided into two parts, because the important shape feature, namely, the intersection curve is the cutter path of the cylinder and the base surfaces.

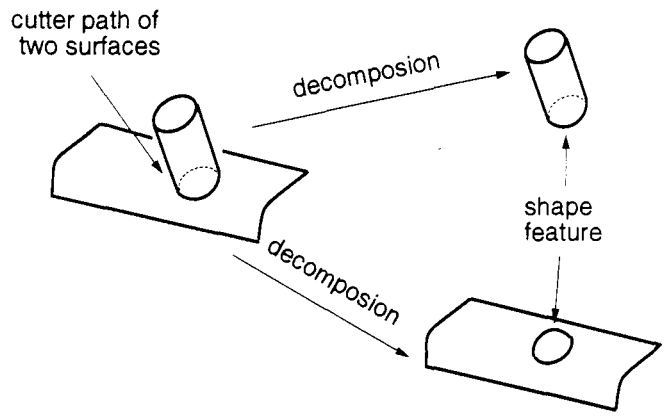


Figure 2.4: **Decomposition of a complete configuration along surface intersection.**

2.2.2 Generation of Geometry Surface

The surface data for geometry primitives are generated using analytical or parametrized methods. Free–form surfaces are generated using some spline surfaces. For a complex configuration like an aircraft, the geometry is modelled by CAD tools. Discrete formats of a CAD output may have three types:

- ▷ **Triangulated surface**. A triangulated surface consists of a node list and an element list.
- ▷ **Quadrilateral surface**. A quadrilateral surface consists of a node list and an element list.
- ▷ **Patch surface**. A patch surface consists of a set of bilinear surfaces, and each bilinear surface is a two–dimensional parametrized surface.

In this work, all patches have two–dimensional structured forms. A multi–patch surface is obtained, if patches are connected to each other in a one–to–one manner. Fig. 2.5 shows an example of data generation of geometry surface. The process of data generation contains the following steps:

- ▷ **Sketch design object**. A design object is coarsely sketched with accurate indication of size and position of all components. In Fig. 2.5, a scaled sketch is given in three views.
- ▷ **Partition the geometry surface**. The complete geometry surface is divided into a set of parts. They are modelled by geometry primitives or free–from surfaces. The parts to be modelled by free–form surfaces should have a rectangular topology, such that its geometry surface can be generated in the form of spline surface functions, as shown in Fig. 2.6.
- ▷ **Model geometry surface**. Geometry surfaces are generated in the form of surface meshes. The mesh process is performed in a piecewise manner. In order to model curved surfaces, some adequate control points have to be selected.

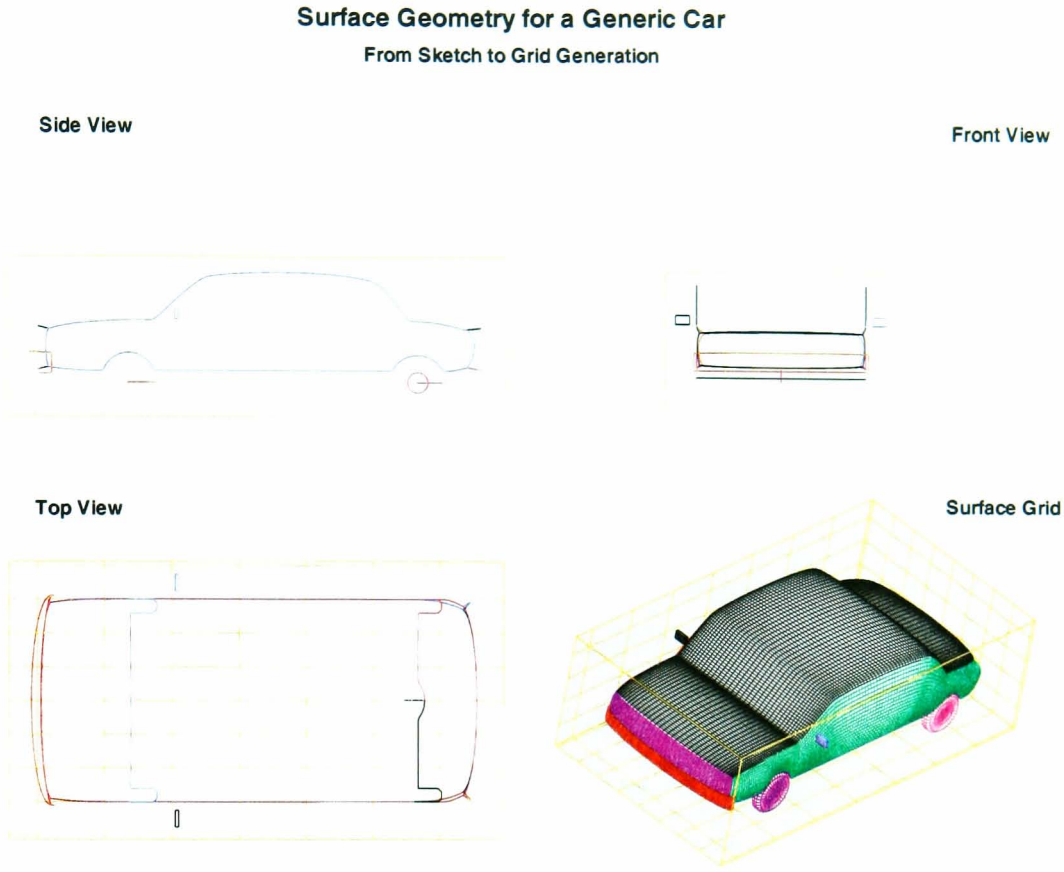


Figure 2.5: **Geometry data is generated in the form of patch surfaces.**

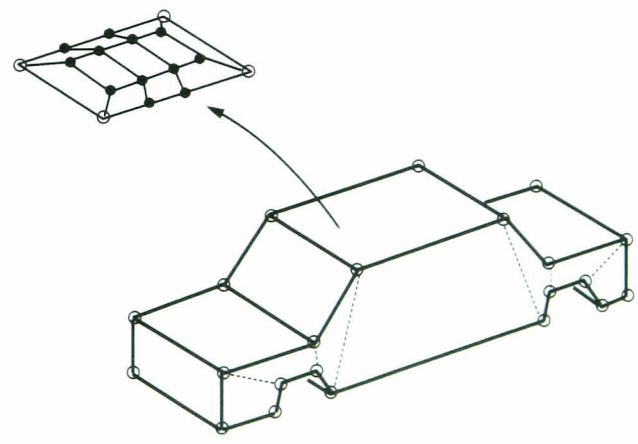


Figure 2.6: **Geometry surface of a sketch is divided into a set of parts. Each part has a rectangular topology.**

According to the above method, data for the geometry surface for a generic car is generated, as shown in Table 2.1.

Table 2.1: Data generation of geometry surface

component	*partition*	*geometry representation*	*shape type*
chassis	many parts	free–form surface	
wheel		geometry primitive	cylinder
mirror		geometry primitive	cube
exhaust		geometry primitive	sweep surface
antenna		geometry primitive	cylinder

2.2.3 Repair of Geometry Surface Data

The validation conditions for meshing boundary representation in section 2.1.1 require a closed boundary for a solution domain. This surface representation must be unique and unambiguous. If the complete boundary of a solid model is described by a set of geometry surfaces, these surfaces are connected to each other in a trimmed manner. Three cases are considered as ill–connected, as shown in Fig. 2.7:

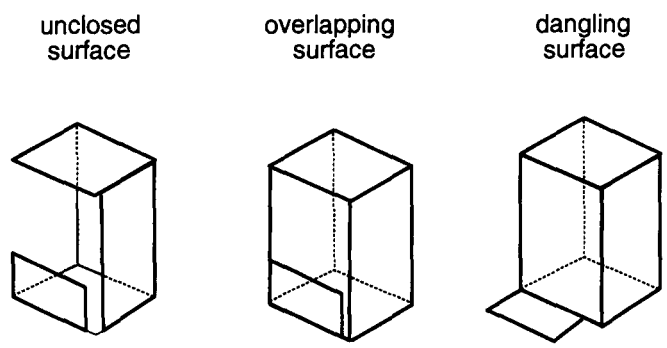


Figure 2.7: **Three cases of ill–connected surfaces: surface gap, surface overlap, and surface dangling.**

- ▷ **Surface gap**. The connecting edges of two faces are not stitched exactly. The boundary of the surface is unclosed.
- ▷ **Surface overlap**. Two surfaces are connected to each other. The surface data is partially duplicated, when the surfaces are overlapping.
- ▷ **Surface dangling**. Some part of a surface is jutting out of the solid object. The dangling surface does not describe the boundary of a solution

domain [78].

In any of these cases, the missing or ambiguous geometric information has to be completed or removed.

2.3 Chapter Summary

The main contribution of the present chapter is that a method for describing the boundary of a meshing domain is presented. The essential core of this method is that geometry surfaces are classified into two types based on shape features of geometric configurations, i.e., *geometry primitives* and *free–form surfaces*. The classification enables the designer to simplify a complex configuration in analytical, parametric as well as discrete forms. In addition, the method for generating surface description is explained. It contains partitioning of a complex object, and surface data generation. The issue of surface data repair is addressed.

Chapter 3

Numerical Grid Generation

The concept of a boundary (or body) fitted grid or structured grid was first published by Winslow [101]. It was systematically further developed in the work of Thompson [88], [89]. The grids generated using this method have a high degree of smoothness of the grid lines. In addition, a boundary fitted grid accurately matches curved boundaries, and its grid lines can be generated almost orthogonal at boundaries. This advantage is one of the major reasons for the preference of a structured grid in the solution of the Navier–Stokes equations, since the simulation demands that the grid is closely clustered near the body to describe the physics of the boundary layer. In this case, the specification of boundary conditions becomes more difficult, when the equations are solved in a coordinate system that does not match the shape of the solid boundary [98].

The present thesis does not directly deal with numerical methods for grid generation. Therefore, only a brief description of structured grid generation using elliptic partial differential equations is presented in this chapter.

3.1 Fundamentals of Numerical Grid Generation

Scalar and vector notations are used in presenting the grid generation equations.

3.1.1 Coordinate Systems and Vectors

In the Euclidean space $\mathbb{R}^n$, with Cartesian basis unit vectors $\mathbf{l}_1, \ldots, \mathbf{l}_n$, and using the *Kronecker symbol* δ_{ij}, an orthogonal system of vectors is given by

$$\mathbf{l}_i \cdot \mathbf{l}_j = \delta_{ij} = \begin{cases} 1 & \text{for } i = j \\ 0 & \text{for } i \neq j \end{cases} \tag{3.1}$$

The coordinate systems used for the derivation of grid generation equations are Cartesian and general curvilinear coordinate systems. Cartesian coordinates are denoted as $x^1, x^2, \ldots, x^n$, while general curvilinear coordinates are denoted as $\xi^1, \xi^2, \ldots, \xi^n$.

In three–dimensions, a vector $\mathbf{r}$ can be written as

$$\mathbf{r} = x^1\mathbf{l}_1 + x^2\mathbf{l}_2 + x^n\mathbf{l}_n \tag{3.2}$$

In cases of continuous and differentiable transformation functions, three numerical values specified with ξ^1, ξ^2, and ξ^3 result in the corresponding values of the Cartesian coordinates x^1, x^2, and x^3.

A position vector, shown in Fig. 3.1, is written as

$$\mathbf{r} = x^1(\xi^1, \xi^2, \xi^3)\mathbf{l}_1 + x^2(\xi^1, \xi^2, \xi^3)\mathbf{l}_2 + x^3(\xi^1, \xi^2, \xi^3)\mathbf{l}_3 \tag{3.3}$$

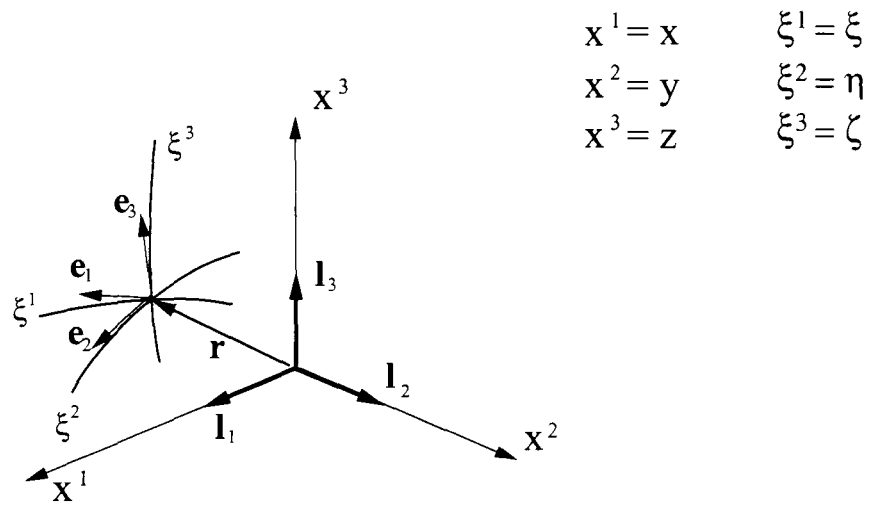


Figure 3.1: **Curvilinear nonorthogonal coordinate system. The covariant basis vectors $\mathbf{e}_i$ point in the tangential directions but are not unit vectors. Vectors $\mathbf{l}_i$ are Cartesian unit vectors.**

In order to simply describe equations, the variant of the summation convention is employed, in which an index is summed over, if it is repeated once as a subscript

and once a superscript in any term of an equation. The set of the covariant basis vectors is defined by

$$\mathbf{e}_i = \frac{\partial \mathbf{r}}{\partial \xi^i} \tag{3.4}$$

The vector $\mathbf{e}_i$ is the tangent to the curve labeled ξ^i. A curve can be represented by specifying either the Cartesian coordinates x^i or the curvilinear coordinates ξ^i as functions of some parameter s, namely

$$x^i = x^i(s) \quad \text{or} \quad \xi^i = \xi^i(s) \tag{3.5}$$

Taking s to be the distance along the curve from some arbitrarily chosen point, the vector tangent to the curve is obtained by

$$\mathbf{T} = \frac{\partial \mathbf{r}}{\partial s} = \frac{\partial \mathbf{r}}{\partial \xi^i}\frac{d\xi^i}{ds} = \mathbf{e}_i \frac{d\xi^i}{ds} \tag{3.6}$$

Since $\mathbf{T}$ is a unit vector, it satisfies the relation

$$\mathbf{T} \cdot \mathbf{T} = 1 = \mathbf{e}_i \cdot \mathbf{e}_j \frac{d\xi^i}{ds}\frac{d\xi^j}{ds} \tag{3.7}$$

From the above equation, one obtains

$$ds^2 = g_{ij} d\xi^i d\xi^j \tag{3.8}$$

where the covariant components of the metric tensor g_{ij} are defined as

$$g_{ij} = \mathbf{e}_i \cdot \mathbf{e}_j \tag{3.9}$$

The set of the contravariant basis vectors, denoted by $\mathbf{e}^i$, is defined by

$$\mathbf{e}^i \cdot \mathbf{e}_j = \delta^i_j = \begin{cases} 1 & \text{for } i = j \\ 0 & \text{for } i \neq j \end{cases} \tag{3.10}$$

Because a vector can be expressed as $\mathbf{v} = v^i \mathbf{e}_i$, the vector $\mathbf{e}^i$ can be expressed using the contravariant components of the metric tensor g^{ij}

$$\mathbf{e}^i = g^{ij} \mathbf{e}_j \tag{3.11}$$

The relation between the co– and contravariant components of the metric tensor is expressed by

$$g^{ij} = \frac{g_{ij}}{|g_{ij}|} \tag{3.12}$$

where $|g_{ij}|$ is the determinant of g_{ij}, denoted by a simple form g. In three–dimensional case, the determinant is in the form of

$$g = \begin{vmatrix} g_{11} & g_{12} & g_{13} \\ g_{21} & g_{22} & g_{23} \\ g_{31} & g_{32} & g_{33} \end{vmatrix} \tag{3.13}$$

3.1.2 Derivatives of Vectors and Scalars

Moving a point from a position with coordinates ξ^i to a neighboring point with coordinates $\xi^i + d\xi^i$, the vectors $\mathbf{e}_i$ are incremented by $d\mathbf{e}_i$. Referring to the basis $\mathbf{e}_i$, the coefficients are linear in the differentials $d\mathbf{e}_i$, that is

$$d\mathbf{e}_i = \Gamma^k_{ij} d\xi^j \mathbf{e}_k \tag{3.14}$$

where the *Christoffel symbol* Γ^k_{ij} denotes the components of affine connection. The above equation can be written as

$$\frac{\partial \mathbf{e}_i}{\partial \xi^j} = \Gamma^k_{ij} \mathbf{e}_k \tag{3.15}$$

3.2 Grid Generation Using Elliptic Partial Differential Equations

In order to use body fitted grids, all equations are expressed in general curvilinear coordinate systems. The one–to–one transformation from the Cartesian coordinate system to the computational coordinate system is given by

$$x^j = x^j(\xi^i), \quad \xi^i = \xi^i(x^j)$$

For the derivation of the transformed grid generation equations, the Poisson equations are used [58], [98]

$$\nabla^2 \xi^n = -g^{ij}\Gamma^n_{ij} \tag{3.16}$$

In order to solve Eq. (3.16) in a convenient manner, the equation is multiplied with $\mathbf{e}_n$

$$\mathbf{e}_n \nabla^2 \xi^n = -g^{ij}\Gamma^n_{ij}\mathbf{e}_n \tag{3.17}$$

Using the relation (3.15), one can rewrite this equation as

$$\mathbf{e}_n \nabla^2 \xi^n = -g^{ij}\frac{\partial \mathbf{e}_i}{\partial \xi^j} \tag{3.18}$$

Since

$$\mathbf{e}_n = \frac{\partial \mathbf{r}}{\partial \xi^n}, \quad \text{and} \quad \frac{\partial \mathbf{e}_i}{\partial \xi^j} = \frac{\partial^2 \mathbf{r}}{\partial \xi^i \partial \xi^j} \tag{3.19}$$

the following form of the Poisson equation can be obtained

$$g^{ij}\frac{\partial^2 \mathbf{r}}{\partial \xi^i \partial \xi^j} + \frac{\partial \mathbf{r}}{\partial \xi^n}\nabla^2 \xi^n = 0 \tag{3.20}$$

where $\nabla^2 \xi^n$ are source terms, often denoted as

$$\nabla^2 \xi^1 = P, \quad \nabla^2 \xi^2 = Q, \quad \nabla^2 \xi^3 = R \tag{3.21}$$

They are very often used to control grid line distribution or generate solution–adaptive grid [28], [35], [89]. Eq. (3.20) provides a convenient approach to formulating the elliptic partial differential equations for grid generation. Using the relation (3.3), the elliptic equations in two dimensions obtained from Eq. (3.20) have the form

$$\begin{aligned} g^{11}\frac{\partial^2 x}{\partial \xi \partial \xi} + 2g^{12}\frac{\partial^2 x}{\partial \xi \partial \eta} + g^{22}\frac{\partial^2 x}{\partial \eta \partial \eta} + P\frac{\partial x}{\partial \xi} + Q\frac{\partial x}{\partial \eta} = 0 \\ g^{11}\frac{\partial^2 y}{\partial \xi \partial \xi} + 2g^{12}\frac{\partial^2 y}{\partial \xi \partial \eta} + g^{22}\frac{\partial^2 y}{\partial \eta \partial \eta} + P\frac{\partial y}{\partial \xi} + Q\frac{\partial y}{\partial \eta} = 0 \end{aligned} \tag{3.22}$$

For simplicity, the following symbols are introduced

$$x_{\xi\xi} = \frac{\partial^2 x}{\partial\xi\partial\xi},\ x_{\xi\eta} = \frac{\partial^2 x}{\partial\xi\partial\eta},\ x_{\eta\eta} = \frac{\partial^2 x}{\partial\eta\partial\eta}$$
$$y_{\xi\xi} = \frac{\partial^2 y}{\partial\xi\partial\xi},\ y_{\xi\eta} = \frac{\partial^2 y}{\partial\xi\partial\eta},\ y_{\eta\eta} = \frac{\partial^2 y}{\partial\eta\partial\eta}$$

and Eq. (3.22) is rewritten as

$$\begin{aligned} g^{11}x_{\xi\xi} + 2g^{12}x_{\xi\eta} + g^{22}x_{\eta\eta} + x_\xi P + x_\eta Q = 0 \\ g^{11}y_{\xi\xi} + 2g^{12}y_{\xi\eta} + g^{22}y_{\eta\eta} + y_\xi P + y_\eta Q = 0 \end{aligned} \tag{3.23}$$

For three–dimensions, the elliptic equations take the form

$$\begin{aligned} g^{11}x_{\xi\xi}+2g^{12}x_{\xi\eta}+2g^{13}x_{\xi\zeta}+g^{22}x_{\eta\eta}+2g^{23}x_{\eta\zeta}+g^{33}x_{\zeta\zeta}+x_\xi P+x_\eta Q+x_\zeta R=0 \\ g^{11}y_{\xi\xi}+2g^{12}y_{\xi\eta}+2g^{13}y_{\xi\zeta}+g^{22}y_{\eta\eta}+2g^{23}y_{\eta\zeta}+g^{33}y_{\zeta\zeta}+y_\xi P+y_\eta Q+y_\zeta R=0 \\ g^{11}z_{\xi\xi}+2g^{12}z_{\xi\eta}+2g^{13}z_{\xi\zeta}+g^{22}z_{\eta\eta}+2g^{23}z_{\eta\zeta}+g^{33}z_{\zeta\zeta}+z_\xi P+z_\eta Q+z_\zeta R=0 \end{aligned} \tag{3.24}$$

The above equations are generally solved in the computational space. Using the relation (3.12), the contravariant components of metric tensor are substituted by the corresponding covariant components. The elliptic partial differential equations for grid generation, for instance, in two–dimensional cases, can be expressed by

$$g^{11} = \frac{g_{22}}{g},\quad g^{12} = -\frac{g_{21}}{g},\quad g^{22} = \frac{g_{11}}{g} \tag{3.25}$$

where Eq. (3.23) can be rewritten as

$$\begin{aligned} g_{22}x_{\xi\xi} - 2g_{12}x_{\xi\eta} + g_{11}x_{\eta\eta} + g(x_\xi P + x_\eta Q) = 0 \\ g_{22}y_{\xi\xi} - 2g_{12}y_{\xi\eta} + g_{11}y_{\eta\eta} + g(y_\xi P + y_\eta Q) = 0 \end{aligned} \tag{3.26}$$

Numerical solution of Eq. (3.26) requires the specification of control functions P and Q, as well as adequate boundary conditions. The detailed method for the numerical solution can be found in [35]. These equations are formulated for a single block. In cases of a multiblock grid, there is a need for updating grid data on block interfaces to their neighboring blocks [99].

3.3 Chapter Summary

The main contribution of the present chapter is that an outline of the basic knowledge of numerical grid generation is given. It contains the method for representation of vectors and scalars in different coordinate systems and their transformation relations. In order to provide the designer a clear mathematical understanding of numerical grid generation, the elliptic partial differential equations for numerical grid generation are derived using the methods in [58], [89], [98], and [101].

Chapter 4

Representation of a Multiblock Grid

A structured multiblock grid can be considered as an ordered set of points in $\mathbb{R}^3$. It divides a given physical domain into a set of structured subdomains, termed *block*, which has a hexahedral shape. In a topological sense, it can be considered as a cube, in which grid points are arranged in regular three–dimensional order. In general, such a structured multiblock grid is represented at two levels. At the micro–level, grid points are arranged in a regular manner. At the macro–level, connectivity relations among blocks are described by means of connecting block faces to neighboring blocks.

The connectivity relations among blocks can be very complex. This complexity may impede the designer's understanding of a multiblock grid. Moreover, a standard method for representing multiblock grids does not exist in the literature. Various representations of multiblock grids probably cause misunderstanding and incorrect interpretation of data structures during data exchanges. In the industrial environment, because information is to be reused, it is expected to define a data structure in an object–oriented fashion. Since such a grid is block structured but is topologically unstructured, a formal description of representation of grids, including structures and properties both at macro– and micro–levels, is given in this chapter.

In grid generation, the main focus of the designer is often on domain decomposition at the macro–level, i.e., his work concentrates on a wireframe building using grid elements such as *vertex*, *edge*, and *face*, while a subdomain discretization is generated by solving some governing equations at the micro–level, namely, at block level. Similar terms *vertex*, *edge*, and *face* are used in geometry, topology

or computer–aided design. In order to prevent confusion of terminologies, a set of definitions for grid elements and a formal description are given in this chapter.

The formal description of a multiblock grid contains:

- ▷ description of data structure of subdomains at micro–level, namely, a description of block data structure, and
- ▷ description of data structure of subdomains at macro–level, namely, a description of relations among blocks.

4.1 Grid Structure at Micro–Level

A structured grid is generated based on the ideas of regular domain decomposition. A physical domain with complex geometry in $\mathbb{R}^3$ is divided into a set of subdomains that consist of a set of points. The subdomains form a finite number of regions being occupied by hexahedrons, called *volume cells* or *voxels*. Each voxel is compact and contiguous, and joins together with its neighboring voxels. A collection of voxels is bounded by six surfaces. Eight vertices and twelve edges result from surface intersections. Such a subdomain is termed a *block*. Block boundaries are the interfaces of subdomains to adjacent blocks.

Using such a multiblock grid in flow simulation, the computation is usually performed block–to–block in the computational space. Although the position of a grid point in the solution domain is uniquely given, it can be represented in two different ways in the physical or computational space. In the physical space, its position can be expressed by Cartesian coordinates, termed *physical position*, while its position in the computational coordinate system is given by a subdomain number and an index of a three–dimensional array. This position is also termed *storage position*, since the position indicates a sequence of grid points ordered in a storage.

Definition 4.1: Block. A set of points is given in the form

$$B = \{p_{i,j,k}(\mathbf{x}) \mid i,j,k\} \subset \mathbb{R}^3$$

where $\mathbf{x}=x,y,z$, and the index set of the points satisfies

$$\mathcal{I} = \{(i,j,k) \mid i = 1,\ldots,I;\ j = 1,\ldots,J;\ k = 1,\ldots,K\};\ i,j,k \in \mathsf{N}$$

The index I denotes the point number of a line in the first dimension. J denotes the line number in the second dimension, and K is the number of the three–dimensional array in the third dimension. This set of points is defined as a *block*.

A block boundary is an interface to adjacent blocks, which is coupling of geometric and physical values among blocks. Since a block has a hexahedral shape, its boundary is divided into six pieces, termed *faces*. The definition of faces is given as follows.

Definition 4.2: Face. Given a block

$$B = \{p_{i,j,k}(\mathbf{x}) \mid i,j,k \in \mathsf{N}\}$$

where $i = 1,\ldots,I; j = 1,\ldots,J; k = 1,\ldots,K$. A subset of the block is in the form

$$F_n = \{p_{i,j,k}(\mathbf{x}) \mid i = 1,\ldots,I; j = 1,\ldots,J; k = 1,\ldots,K\} \subset B$$

This subset of the block is defined as *faces*. The subscript n denotes the face number. Faces of a block are numbered as follows:

$$n = \begin{cases} 1 & : \quad k = 1,\ i = 1,\ldots,I;\ j = 1,\ldots,J \\ 2 & : \quad j = 1,\ i = 1,\ldots,I;\ k = 1,\ldots,K \\ 3 & : \quad i = 1,\ j = 1,\ldots,J;\ k = 1,\ldots,K \\ 4 & : \quad i = I,\ j = 1,\ldots,J;\ k = 1,\ldots,K \\ 5 & : \quad j = J,\ i = 1,\ldots,I;\ k = 1,\ldots,K \\ 6 & : \quad k = K,\ i = 1,\ldots,I;\ j = 1,\ldots,J \end{cases}$$

Fig. 4.1 depicts a standard block. Faces are indicated by arrows in accordance with their definitions, namely, face 1 bottom ($\zeta = 1$), face 2 left ($\eta = 1$), face 3 back ($\xi = 1$), face 4 front ($\xi = I$), face 5 right ($\eta = J$), face 6 top ($\zeta = K$). This definition supplies a simplified expression for block boundary. For instance, face 1 is expressed by

$$F_1 = \{p_{i,j,1}(\mathbf{x}) \mid i = 1,\ldots,I; j = 1,\ldots.J\}.$$

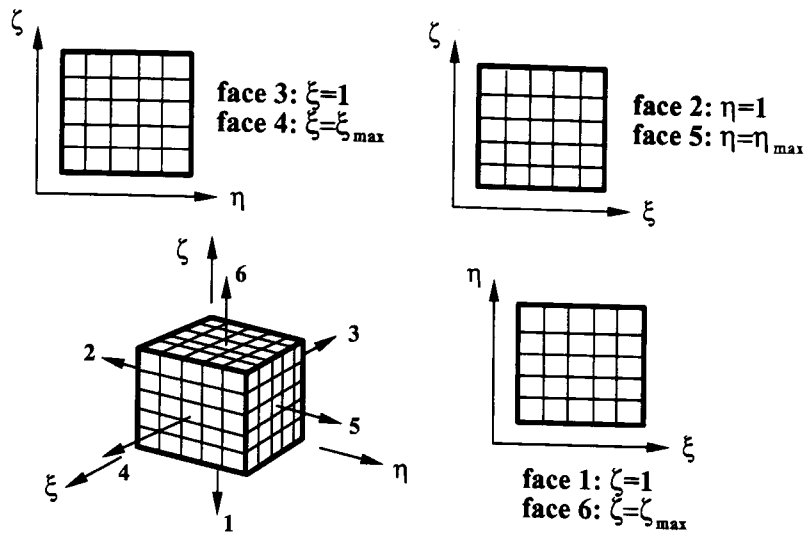


Figure 4.1: **Standard block in computational space. Each block has it own local coordinate system. The arrows indicate the normal directions of faces [41].**

4.1.1 Data Structure of a Block

Grid points of a block are considered as a set of points B in $\mathbb{R}^3$, expressed by

$$B = \{p_{i,j,k}(\mathbf{x}) \mid i = 1, \ldots, I; j = 1, \ldots, J, k = 1, \ldots, K\} \subset \mathbb{R}^3$$

Its elements, i.e., grid points are arranged as a three–dimensional array with indices $i, j, k \in \mathsf{N}$, which correspond to the computational coordinates $\xi, \eta, \zeta \in \mathsf{N}$. Grid points are saved in such a manner that the i direction is treated first, followed by the j and k directions, respectively, namely

$$\begin{aligned} B \;=\; & \{p_{1,1,1}(\mathbf{x}), p_{2,1,1}(\mathbf{x}), \ldots, p_{I,1,1}(\mathbf{x}), \ldots, p_{1,2,1}(\mathbf{x}), p_{2,2,1}(\mathbf{x}), \ldots, \\ & p_{I,J,1}(\mathbf{x}), \ldots, p_{1,1,2}(\mathbf{x}), p_{2,1,2}(\mathbf{x}), \ldots, p_{I,J,K}(\mathbf{x})\} \end{aligned}$$

The sequence of grid points ordered in computational space is shown in Fig. 4.2. Since three coordinates (ξ, η, ζ) are orthogonal in computational space, and form a right–handed coordinate system, each block has its own right–handed coordinate system. The physical position of a point is given by its coordinate values (x, y, z), which is indicated by the index (i, j, k) in the (ξ, η, ζ) directions, respectively.

4.2 Grid Structure at Macro–Level

A complete solution domain is a collection of subdomains, which are represented by a set of standard blocks. In cases of high complexity of block topology, a

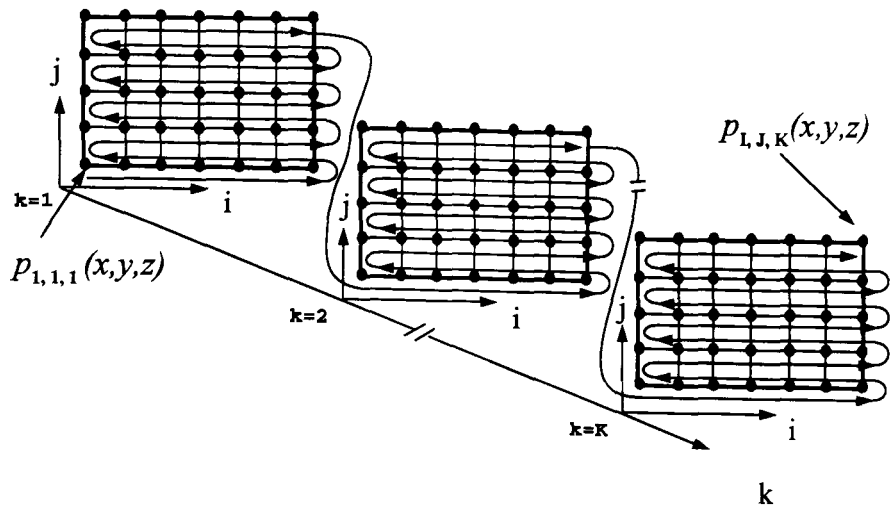


Figure 4.2: **Grid points are arranged in the sequence indicated by the arrow.**

multiblock grid can not be represented in a computational space by a one–to–one mapping. In order to understand multiblock grid structure at the macro–level, a formal description of block connectivity is given in the following section.

4.2.1 Block Connectivity

Relationships among blocks are generally termed *block connectivity.* In the present thesis, this connectivity is restricted to a one–to–one connectivity across block boundaries. That means, an arbitrary face of a block can only be connected to a single face of another block. Block connectivity is defined as follows.

Definition 4.3: Block connectivity. $P = \{p_i(x,y,z) \mid i \in \mathsf{N}\}$ and $Q = \{q_j(x,y,z) \mid j \in \mathsf{N}\}$ represent two sets of grid points of faces $F_{r,m}$ and $F_{s,n}$ from different blocks B_r and B_s, where the subscripts r and s denote block numbers, and m and n denote face numbers, respectively (see definitions 4.1 and 4.2), i.e., $P \in F_{r,m}$ and $Q \in F_{s,n}$. The following metric and combinatorial conditions between both sets are satisfied:

1. Both faces have the same number of points as well as the same number of quadrilateral elements.
2. The equivalent relation $(\forall_{\mathtt{F_{r,m}}} P)(\exists_{\mathtt{F_{s,n}}} Q) : P \rightarrow Q$ is met [6].

Such two faces from distinct blocks have a neighboring relation.

However, this definition only requires the same number of elements and a one–to–one mapping of grid points of two neighboring faces. In order to couple grid points of a common face from neighboring blocks, it is necessary to know how grid points of different blocks are matched on neighboring faces.

4.2.2 Block Matching

Determination of a pair of face connectivity gives only a neighboring relation between two faces. For two standard blocks, there are eventually four possibilities to connect any face of a neighboring block. Suppose that two blocks are connected to each other, as depicted in Fig. 4.3. In addition, both faces connected have the same point number in ξ– and ζ–directions. Let block 1 be a reference block, and keep it fixed. The pair of face connectivity for block 1 is face 6, and for block 2 face 1. Block 2 is rotated about the ζ–axis by $\theta = \frac{\pi}{2}i$, with i=0, 1, 2, 3, respectively, and then four pairs of vertices are matched, one obtains four cases, as shown in Fig. 4.4. The cases are termed *block matching* or *block matching orientation.*

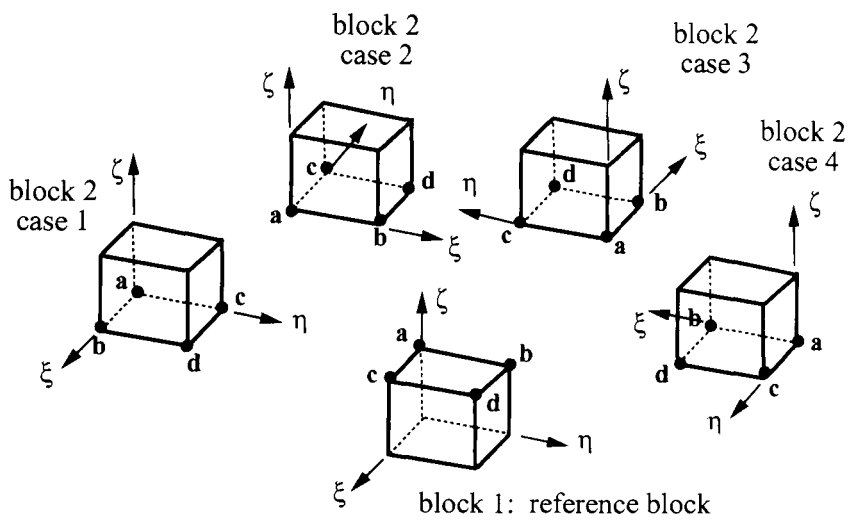


Figure 4.3: **Two connected blocks with the common faces. Different cases of block matching are obtained by fixing the reference block and rotating its neighboring block.**

Clearly, a graphic representation of block connectivity and block matching is unlikely to describe a grid topology with a large number of blocks.

4.2.3 Representation of Block Interface

With increase of block number and complexity of block connectivity, one needs a method to represent such a grid topology unambiguously and uniquely. The block structure such as number of grid points and their order in each block, and relations among blocks have to be contained in a grid topology representation. To this end, a block connectivity file (*interface to flow solvers*) is formalized in this thesis. To avoid using mixed local coordinate systems, all blocks are of the

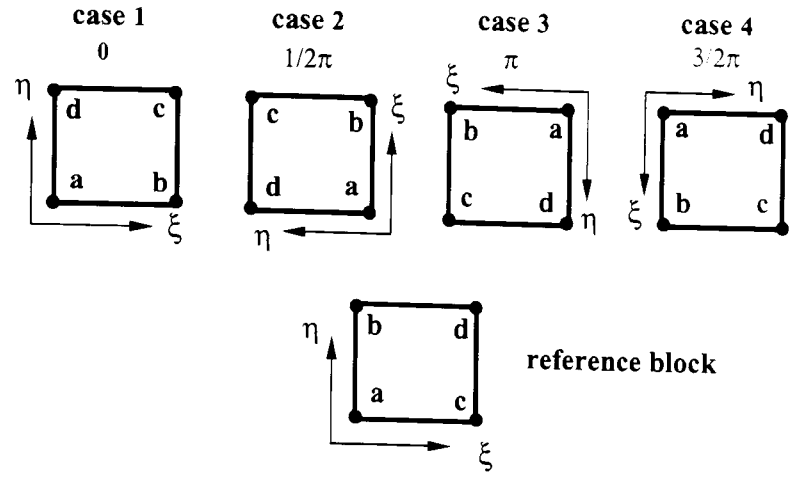


Figure 4.4: **The four possible block matching of a pair of face connectivity. Four cases are obtained by successive rotations about ζ–axis of block 2 by** 0, $\frac{1}{2}\pi$, π, **and** $\frac{3}{2}\pi$.

right–handed systems in the block connectivity file.

Since a multiblock grid is structured at the block level, each block is declared as an object, specified by object string, block number, dimensions, block connectivity and block matching in the following form

```
BLOCK blockId
I J K
f1  fc  nb  nf  i  j  nc1  nc2
f2  fc  nb  nf  i  k  nc1  nc2
f3  fc  nb  nf  j  k  nc1  nc2
f4  fc  nb  nf  j  k  nc1  nc2
f5  fc  nb  nf  i  k  nc1  nc2
f6  fc  nb  nf  i  j  nc1  nc2
```

where, the key word `BLOCK` is a control string and gives the object name. This is followed by block number, denoted by `blockId`, and block dimensions in ξ–, η– and ζ–directions, denoted by `I J K`. For each face there is one line with face–specific information. `f1`,...,`f6` are face numbers. The key word `fc` implies face condition, e.g., a physical boundary or an internal boundary. This is specified by an integer number 0 for a physical boundary and 1 for an internal boundary. Block connectivity is specified by the neighboring block number `nb`, and the neighboring face number `nf`. Face matching is specified by relations of two local coordinate systems. The fifth and sixth columns indicate the first and second computational coordinates of the present faces of reference block. $\texttt{nc}_1$ and $\texttt{nc}_2$ are the computational coordinates of the faces of the neighboring block.

In [36], the format of a structured grid topology file is presented for the first time. which gives the matching orientation implicitly. The matching cases are given by integer numbers. In this thesis, matching cases are given in an explicit manner. i.e., the cases are represented by indicating computational coordinates of faces instead of integer numbers.

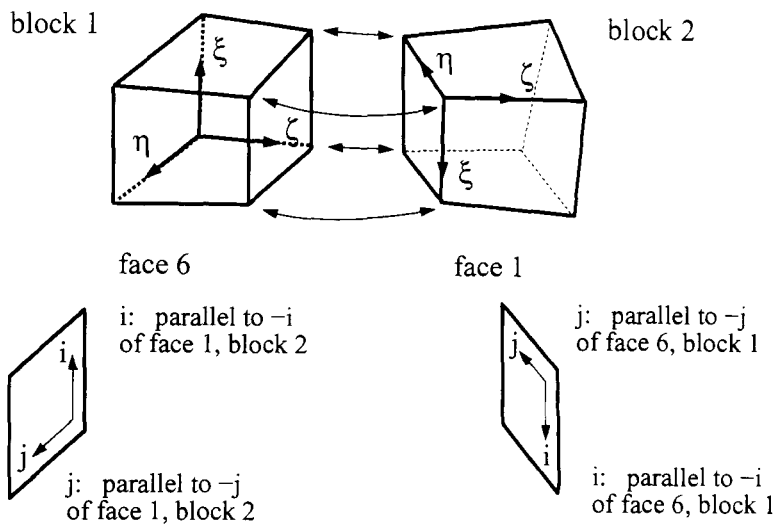


Figure 4.5: **A block neighboring relation is given by face connectivity and matching orientation.**

Example of a block connectivity file. The block connectivity of this example is depicted in Fig. 4.5. It is supposed that these two blocks have the same dimensions $I = 5$, $J = 5$, and $K = 5$, in ξ–, η– and ζ–directions, respectively. The face 6 of block 1 and the face 1 of block 2 are connected to each other. The block connectivity file has the form

```
BLOCK 1
5 5 5
1  0  0  0  i  j  0  0
2  0  0  0  i  k  0  0
3  0  0  0  j  k  0  0
4  0  0  0  j  k  0  0
5  0  0  0  i  k  0  0
6  1  2  1  i  j -i -j
BLOCK 2
5 5 5
1  1  6  1  i  j -i -j
2  0  0  0  i  k  0  0
3  0  0  0  j  k  0  0
4  0  0  0  j  k  0  0
5  0  0  0  i  k  0  0
6  0  0  0  i  j  0  0
```

In the above block connectivity file, one finds the following information

- ▷ The grid consists of two blocks. Both blocks have the dimension $I = 5$, $J = 5$, and $K = 5$.
- ▷ Face 6 of block 1 is a non–physical boundary, `fc` is 1. It is connected to the face 1 of block 2.
- ▷ Face 1 of block 2 is a non–physical boundary, `fc` is 1. It is connected to the face 6 of block 1.
- ▷ On face 6 of block 1, the first and the second computational coordinates indicate ξ– and η–directions, denoted by i and j. At the view port of block 1, the parallel axes of the face 1 of block 2 to its ξ– and η–axes are the negative ξ– and η–axes, respectively, denoted by $-i$ and $-j$, comparing in Fig. 4.4.
- ▷ On the face 1 of block 2, the first and the second computational coordinates indicate ξ– and η–directions, denoted by i and j. At the view port of block 2, the parallel axes of the face 6 of block 1 to its ξ– and η–axes are the negative ξ– and η–axes, respectively, denoted by $-i$ and $-j$, comparing in Fig. 4.4.

In general, a block connectivity file can be generated by comparing coordinates of grid points selected from two faces of different blocks. If the conditions of the definition 4.3 are satisfied, a pair of block connectivity is determined.

The block connectivity file provides the necessary as well as sufficient information about block interfaces to a flow solver. It is, however, not sufficient for a grid adaptation process using a redistribution scheme. This type of grid adaptation requires that the original geometry on physical boundary is retained. Therefore, geometric features on block boundaries have to be regarded in an adaptive grid generation.

4.2.4 Geometric Features on Physical Boundary

According to their roles in describing geometric shapes, grid elements such as faces, edges and vertices fall under different geometric constraints. In grid adaptation process, grid points have to be treated subject to the geometry surface of the physical boundaries. To distinguish them, the faces of physical boundaries are termed *external faces*, while other faces are called *internal faces*. The grid points of external or internal faces are called *external* or *internal points*, respectively.

Extracting external faces from a volume grid, these faces are connected to each other on four sides. Since such a block topology is two–dimensional, connectivity relations among external faces are termed *face connectivity.*

4.2.5 Face Connectivity

The neighboring relations among external faces are represented by connectivity of sides. Fig. 4.6 shows three blocks that are connected to each other. The geometric features at common sides have to be specified before a grid adaptation process is started, since different methods will be employed to move grid points with different degrees of freedom.

There is no clear and strict definition of a measure for a sharp edge. However, such a geometric feature must be specified by a quantity, such as an angle between two intersection faces.

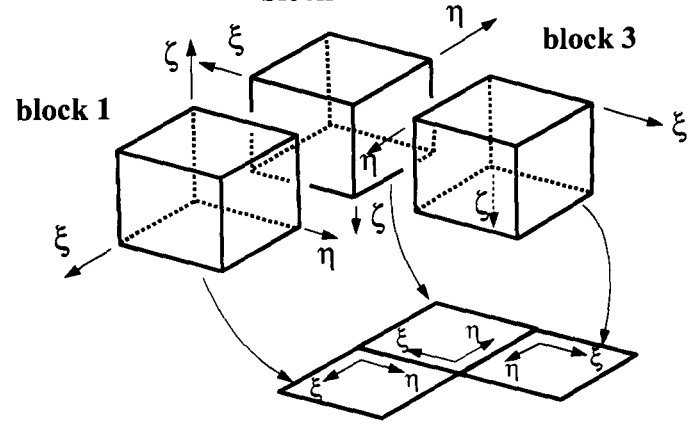


Figure 4.6: **Three faces are connected to each other. At any two common sides, the geometric features are important for a grid adaptation process.**

4.2.6 Face Matching

Two faces are matched at their common side, if the one–to–one relations of points along the the common side are found, and every pair of the common points is in the same position, as shown in Fig. 4.7.

Similar to the block matching, face matching is represented using the relative

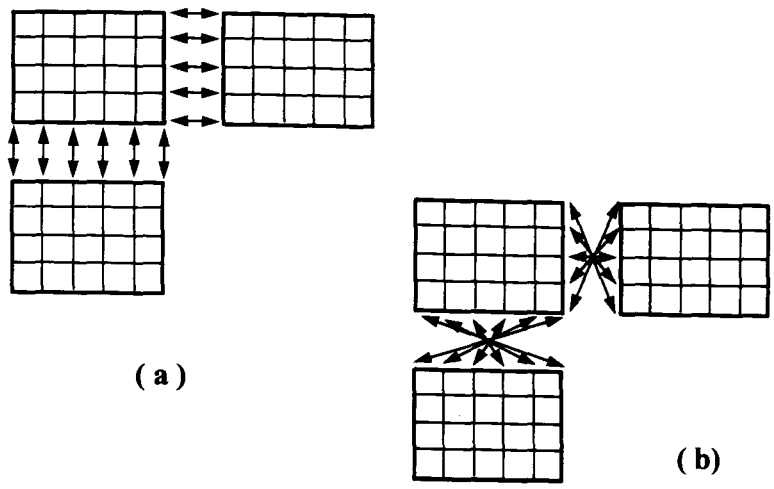


Figure 4.7: **Two cases of face matching: (a) Faces are matched along the common sides; (b) faces are not matched along the common sides.**

coordinate direction of two common sides.

4.2.7 Representation of Grid Physical Boundary

The information about external or internal face types is directly obtained from a block connectivity file, while edge and vertex types can be derived using the wireframe topology. Besides a block connectivity file, a face connectivity file is formalized to describe block topology of a grid and its geometric features on fixed boundary. Its format is given as follows.

```
BLOCK blockId
I J K
f1
I_min sc nb nf ns gf
I_max sc nb nf ns gf
J_min sc nb nf ns gf
J_max sc nb nf ns gf
f2
I_min sc nb nf ns gf
I_max sc nb nf ns gf
K_min sc nb nf ns gf
K_max sc nb nf ns gf
f3
J_min sc nb nf ns gf
J_max sc nb nf ns gf
K_min sc nb nf ns gf
K_max sc nb nf ns gf
f4
J_min sc nb nf ns gf
J_max sc nb nf ns gf
```

K_{min} sc nb nf ns gf
K_{max} sc nb nf ns gf
f5
I_{min} sc nb nf ns gf
I_{max} sc nb nf ns gf
K_{min} sc nb nf ns gf
K_{max} sc nb nf ns gf
f6
I_{min} sc nb nf ns gf
I_{max} sc nb nf ns gf
J_{min} sc nb nf ns gf
J_{max} sc nb nf ns gf

where sc denotes *side condition*; ns and gf denote *neighboring side* and *geometric feature*. The method for determination of edge features will be explained in section 5.3.1.

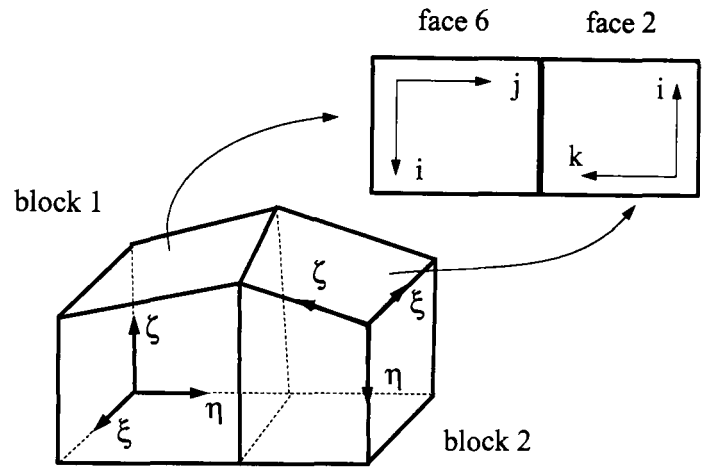


Figure 4.8: **An example for explanation of a fixed boundary file.**

Example of a face connectivity file. Fig. 4.8 is depicted to explain a fixed boundary file. Suppose that face 6 of block 1 and face 2 of block 2 are on physical boundary. The common side of both faces describes a sharp edge. In a face connectivity file, the topological relation and geometric feature are given in the following form

BLOCK 1
I_1 J_1 K_1
...
f6
I_{min} 0 0 0 0 0
I_{max} 0 0 0 0 0
J_{min} 0 0 0 0 0

```
Jmax 1 2 2 -K 1
BLOCK 2
I2 J2 K2
...
f2
Imin 0 0 0  0 0
Imax 0 0 0  0 0
Kmin 0 0 0  0 0
Kmax 1 1 6 -J 1
...
```

The above face connectivity file provides the following information:

- ▷ Face 6 of block 1 as well as face 2 of block 2 are on the physical boundary.
- ▷ For block 1: J_{max}–side of face 6 is a common side of K_{max}–side of face 2 of block 2. The minus symbol for the K_{max}–side means to reverse the sequence of grid points along this side.
- ▷ For block 2: K_{max}–side of face 2 is a common side of J_{max}–side of face 6 of block 1. The minus symbol for the J means to reverse the sequence of grid points along this side.
- ▷ The common side describes a sharp edge on physical boundary, denoted by the integer 1.

4.3 Wireframe Topology

Theoretically, one can visualize all grid points to represent the complete solution domain. However, it is extremely expensive when grid point numbers are large. A practical simplification of a complete grid is to extract the grid elements from the complete grid, which represent the complete block boundaries, and to build the model of block boundaries. The model can be imaged as being made out of a deformable material that can be stretched or continuously deformed in any manner other than separating or tearing. The model built by selected elements is termed *wireframe model.*

4.3.1 Elements of Wireframe Topology

The wireframe model simplifies a block in the form of a hull, and is used to specify shape features of a block. Fig. 4.9 shows a wireframe model extracted from a block. It can be seen that a block whose boundaries are formed by six

curved faces is converted to a cube. A formal description of terminologies for the elements of a wireframe model is given as follows.

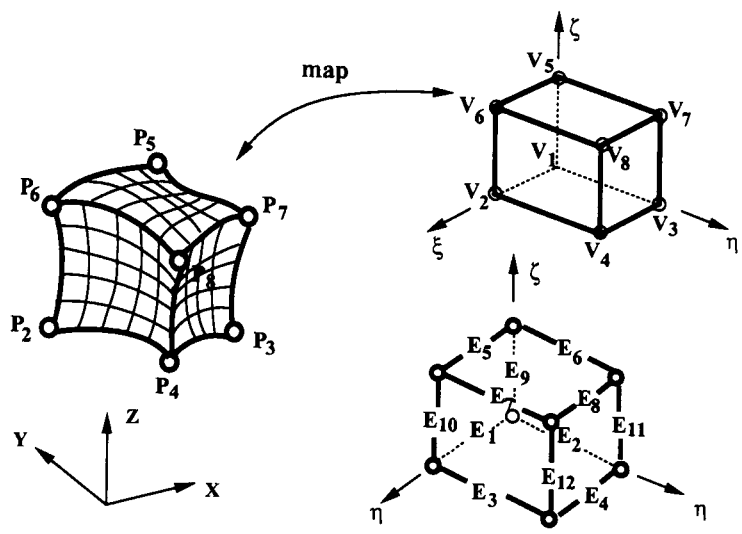


Figure 4.9: **Vertices, edges, faces and block are extracted from a block in order to build a wireframe model that is used to represent a multiblock grid.**

Definition 4.4: Vertex of wireframe model. A vertex of a wireframe model is one of the eight corners of a block, denoted by $\mathcal{V}$.

Definition 4.5: Edge of wireframe model. An edge of a wireframe model is a pair of vertices, denoted by $\mathcal{E}$. It represents one of the twelve curved edges of a block, and can be written by $\mathcal{E}\{\mathcal{V}_1, \mathcal{V}_2\}$.

Definition 4.6: Face of wireframe model. A face of a wireframe model is a closed loop of four edges, denoted by $\mathcal{F}$. It represents one of the six curved surfaces of a block, and can be written by $\mathcal{F}\{\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3, \mathcal{E}_4\}$. According to the above definitions, a face of a wireframe model can also be expressed by $\mathcal{F}\{\mathcal{E}_1\{\mathcal{V}_1, \mathcal{V}_2\}, \mathcal{E}_2\{\mathcal{V}_1, \mathcal{V}_3\}, \mathcal{E}_3\{\mathcal{V}_2, \mathcal{V}_4\}, \mathcal{E}_4\{\mathcal{V}_3, \mathcal{V}_4\}\}$.

Definition 4.7: Cube of wireframe model. A cube of a wireframe model is a simplified solid model of a block. It consists of six faces, twelve edges and eight vertices of wireframe model, denoted by $\mathcal{B}$. According to the above definition, it can be expressed by $\mathcal{B}\{\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3, \mathcal{F}_4, \mathcal{F}_5, \mathcal{F}_6\}$.

The topological relations of a single block among the grid elements defined above are given in Table 4.1.

It is convenient to use a wireframe model to visualize a complex grid topology in space. A visualization tool for this application needs only minimal storage to represent a complete grid.

Table 4.1: The topological relations between the elements of wireframe model of a single block.

$\mathcal{B}$	$\mathcal{F}$	$\mathcal{E}$	$\mathcal{V}$
1	1	1, 2, 3, 4	1, 2, 3, 4
	2	1, 5, 7, 9	1, 2, 5, 6
	3	2, 6, 9, 11	1, 3, 5, 7
	4	3, 7, 10, 12	2, 4, 6, 8
	5	4, 8, 11, 12	3, 4, 7, 8
	6	5, 6, 7, 8	5, 6, 7, 8

Moreover, the elements are divided into the external and internal types. The former are the elements at physical boundary, while the latter the other elements. Hiding the internal elements, a block structure on the surface of the physical boundary is given.

4.4 Chapter Summary

The main contribution of the present chapter is that a set of terminologies and definitions is introduced to represent a grid. Using these terminologies and definitions, a formal description of a multiblock grid is given. Based on a standardized block data structure, the method for representing a multiblock grid is developed. Given a data structure for single block, a multiblock grid can be represented by connectivity relations among blocks. These relations are termed *block topology*, and are formalized in a *block connectivity file*. In addition, a set of definitions for a simplified model, called *wireframe model*, used for visualizing a multiblock grid is given.

Chapter 5

Automatic Identification of Grid Topology

The central issues in grid topology identification are the determination of block connectivity and matching cases, as well as the identification of geometric features at physical boundaries of a given grid. The former process supplies the information about block neighbors, including block connectivity and matching orientation. These build the interface of a multiblock grid to a flow solver. However, they do not contain any geometric features of the grid.

For a process of grid adaptation, description of a grid topology based only by these neighboring relations is rough. The geometric characteristics on fixed boundaries have to be incorporated into an adaptation computation in order to prevent deformation or destruction of surface geometry of fixed boundaries.

A complete representation of a structured grid consists therefore of connectivity relations and geometric features of block boundaries. With the increase in block numbers, it is becoming more important to generate the grid topology automatically, since the complexity of the grid makes it impossible to identify and specify block relations and features manually.

The objective of the present chapter is to describe the strategy and algorithms developed for automatically identifying the block topology. It contains the algorithms for detecting block connectivity, determining matching orientation, and finding geometric features on fixed boundaries.

5.1 Block Connectivity among Blocks

A multiblock grid has a standard form, defined in the previous chapter. It requires that all local coordinate systems are right–handed, and a one–to–one mapping of grid points of two neighboring faces. The algorithm developed for automatic grid topology identification contains the block standardization, recognition of block connectivity, and determination of matching orientation.

5.1.1 Generation of the Standard Cube

The data structure for a standard block, described in section 4.1.1, is defined based on a local right–handed coordinate systems. For this reason, a pre–process is designed to ensure that all blocks have local right–handed coordinate systems, such that block topology will be uniquely and unambiguously interpreted.

Determination of local coordinate systems. Suppose, a voxel in physical space has the form depicted in Fig. 5.1. The normal vector given by

$$\mathbf{n}(\xi,\eta) = \frac{\mathbf{r}_\xi \times \mathbf{r}_\eta}{|\mathbf{r}_\xi \times \mathbf{r}_\eta|} \tag{5.1}$$

is orthogonal to the plane spanned by vectors $\mathbf{r}_\xi$ and $\mathbf{r}_\eta$. The unit vector

$$\mathbf{e}_\zeta = \frac{\mathbf{r}_\zeta}{|\mathbf{r}_\zeta|} \tag{5.2}$$

it points in the direction along the vector $\mathbf{r}_\zeta$. The angle between $\mathbf{n}(\xi,\eta)$ and $\mathbf{e}_\zeta$ is expressed by

$$cos\beta = \frac{\mathbf{n}(\xi,\eta)\cdot\mathbf{e}_\zeta}{|\mathbf{n}(\xi,\eta)|\cdot|\mathbf{e}_\zeta|} \tag{5.3}$$

Rewriting the above equation

$$cos\beta = \frac{(\mathbf{r}_\xi \times \mathbf{r}_\eta)\cdot\mathbf{r}_\zeta}{|(\mathbf{r}_\xi \times \mathbf{r}_\eta)|\cdot|\mathbf{r}_\zeta|} \tag{5.4}$$

where $cos\beta$ is a measure of the angle between the ζ–axis and the the normal vector of the plane spanned by ξ– and η–axes.

The voxel forms fall into three classes:

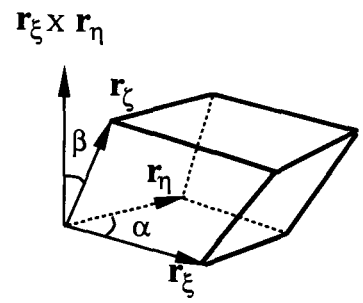


Figure 5.1: **A model of an arbitrary voxel and three vectors built by $\mathbf{r}_\xi$, $\mathbf{r}_\eta$ and $\mathbf{r}_\zeta$.**

- ▷ $cos\beta$ is negative, a voxel has a local left–handed coordinate system.
- ▷ $cos\beta$ is equal to zero, the ζ–axis lies in the plane spanned by the ξ– and η–axes.
- ▷ $cos\beta$ is positive, a voxel has a local right–handed coordinate system.

Theoretically, the quantity $cos\beta$ can be employed to detect a local coordinate system. In practical application, it should be avoided that the value of divisor may occasionally be very small, since the robustness of computation may be affected in this case.

Algorithm 5.1: Generation of the Standard Cube. An alternative of the algorithm provides more robustness for practical applications. The determination of the type of local coordinate system is performed in two stages:

- ▷ Stage 1: compute the divisor of Eq. (5.4). A non–zero divisor indicates that vector $\mathbf{r}_\zeta$ is not linear dependent on vectors $\mathbf{r}_\xi$ and $\mathbf{r}_\eta$, and
- ▷ Stage 2: since the absolute value of divisor is non–negative, the denominator is sufficient to check the type of local coordinate system.

Suppose that an arbitrary grid point within a block is chosen as a reference point, depicted in Fig.5.2. This can be expressed as

$$P_{i,j,k} = \{p_{i,j,k}(x, y, z) \mid i, j, k \in N\} \subset \mathbb{R}^3$$

The three neighboring points of the reference point in ξ–, η–, and ζ–directions are selected to build three vectors with respect to the reference point. A cross product of two vectors, namely, $\mathbf{r}_\xi \times \mathbf{r}_\eta$, gives the normal vector for the ξ–η plane. The local coordinate system of the voxel is determined by the following relation

$$D_{[\mathbf{r}_\xi,\mathbf{r}_\eta,\mathbf{r}_\zeta]} = (\mathbf{r}_\xi \times \mathbf{r}_\eta)\cdot\mathbf{r}_\zeta = \begin{cases} < 0 \ : \ left-handed \\ = 0 \ : \ deformed\ voxel \\ > 0 \ : \ right-handed \end{cases} \tag{5.5}$$

where $D_{[\mathbf{r}_\xi,\mathbf{r}_\eta,\mathbf{r}_\zeta]}$ is termed the determinant, and the subscript denotes the dot product of $\mathbf{n}(\xi,\eta)$ and $\mathbf{e}_\zeta$. The vectors are constructed from the points

$$\begin{aligned} \mathbf{r}_\xi &= p_{i+1,j,k}(x,y,z) - p_{i,j,k}(x,y,z) \\ \mathbf{r}_\eta &= p_{i,j+1,k}(x,y,z) - p_{i,j,k}(x,y,z) \\ \mathbf{r}_\zeta &= p_{i,j,k+1}(x,y,z) - p_{i,j,k}(x,y,z) \end{aligned} \tag{5.6}$$

In physical space, a block may occupy a region described by curved boundaries. Voxels deformed or skewed can also be detected using this method.

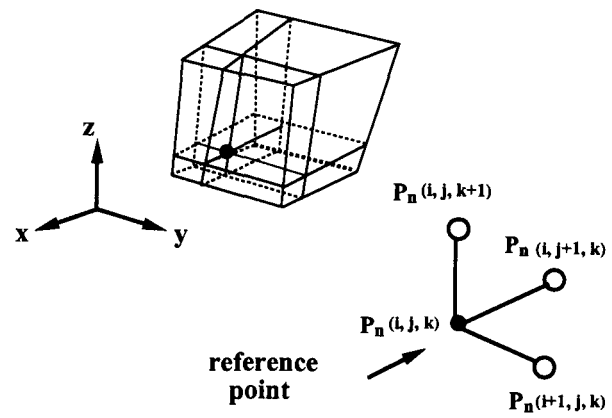


Figure 5.2: **A reference point of a block is selected, and a dot product is built to determine the type of local coordinate system.**

Implementation of the algorithm 5.1. The algorithm serves as a detector to identify the type of local coordinate systems. It is used as a preprocess to unify local coordinate system of all blocks. Meanwhile, it provides a simple quality control of bad voxels and their locations. The algorithm contains the following steps:

- ▷ Step 1. Read a block $B=\{p_{i,j,k}(x,y,z)\}$ and assign all points into a three–dimensional array (i,j,k).
- ▷ Step 2. Build a three–level loop to check voxels and their local coordinate systems. Referring to a point $p_{i,j,k}(x,y,z)$ three vectors are constructed using Eq. (5.6), and their dot product given by Eq. (5.4), is computed. A non–zero value allows the process to proceed.
- ▷ Step 3. Compute the determinant $D_{[\mathbf{r}_\xi,\mathbf{r}_\eta,\mathbf{r}_\zeta]}$ of Eq. (5.5) and determine the type of local coordinate system.

▷ Step 4. Record the block number and location of bad voxels for cases 1 and 2 in Eq. (5.5). All left–handed voxels are converted into right–handed.

5.1.2 Generation of Block Connectivity

Block connectivity is determined by identifying the positions of two sets of points that are selected from two faces of different blocks. If these two sets have a one–to–one mapping and the coordinates of the respective points are the same within a specified tolerance, these two faces are neighbored. However, this identification method might not be practical for implementation.

The approach implemented to determining face connectivity is based on a face data structure. An arbitrary face is represented by a single point, and its coordinates give the position of the face uniquely in space. This representative position of a face is employed to speed up this process.

Definition 5.1: Face simplex. A face of a block consists of a set of points, forming a set of quadrilateral elements. It is supposed that the quadrilateral elements have a homogeneous material property with a virtual and constant density and thickness. The center of mass of the face is represented by a single point uniquely and unambiguously in $\mathbb{R}^3$, this point is defined as *face simplex*, denoted by $X_{b,f}(x, y, z)$, where indices b and f indicate block and face numbers, respectively.

A face simplex serves as a comparative quantity to find a neighboring face. The algorithm tracks the shortest distance between a pair of face simplices. Once a neighboring relation of two faces is determined, the matching orientation will be identified by comparing positions of four pairs of vertices.

Building a face simplex. Suppose that a face with $I \times J$ grid points has a material property with a virtual and constant density and thickness, denoted by $\rho_{i,j}$ and thickness $t_{i,j}$ at each grid point. The concept of the center of mass of the face is used to evaluate a face simplex

$$X_{b,f} = \frac{\sum_{i=1}^{I-1} \sum_{j=1}^{J-1} \overline{p}_{i,j} A_{i,j} t_{i,j} \rho_{i,j}}{\sum_{i=1}^{I-1} \sum_{j=1}^{J-1} A_{i,j} t_{i,j} \rho_{i,j}}$$

where $A_{i,j}$ denotes the area of an element built by four points $p_{i,j}(x, y, z)$, $p_{i+1,j}(x, y, z)$, $p_{i,j+1}(x, y, z)$, and $p_{i+1,j+1}(x, y, z)$, and $\overline{p}_{i,j}(x, y, z)$ is the average value of the four points. Obviously, the above equation can be simplified to

$$X_{b,f} = \frac{\sum_{i=1}^{I-1}\sum_{j=1}^{J-1}\overline{p}_{i,j}A_{i,j}}{\sum_{i=1}^{I-1}\sum_{j=1}^{J-1}A_{i,j}} \tag{5.7}$$

Once a large amount of points on a face is reduced into a single point, the problem of determining a connectivity relation is devoted to finding the minimal distance function between a pair of face simplices.

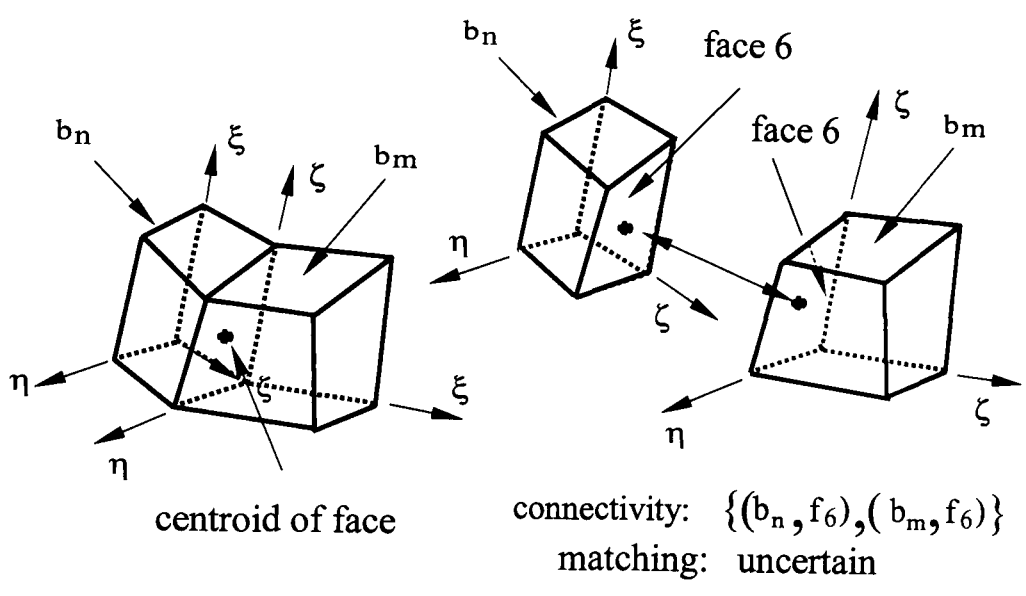


Figure 5.3: **Face simplex is employed to determine face connectivity.**

Fig. 5.3 depicts two connected blocks. The boundary of a block is bounded by six faces, and the centroids of these faces possess their unique locations. In the entire mesh domain, there are no penetrating blocks. A face simplex, represented by the centroid of the face, gives the position of the face, which is used for checking its common face simplex in another block. Table 5.1 gives the pseudocode of the algorithm employed to determine block connectivity.

Description of the algorithm 5.2. The algorithm for determination of block connectivity detects a face connectivity in two stages. First, the positions of faces are replaced by their centers of mass. Second, if two centers of mass of two different faces are equal, the corresponding blocks might be connected. The sufficient condition is that the vertices of two faces satisfy a one–to–one mapping.

The algorithm begins with calculation of the face simplex. Before a loop of comparing two face simplices of different blocks is started, all face simplices are initialized with a face condition *non–connectivity* (lines *[6]* to *[7]* in the table).

Table 5.1: Pseudocode of Algorithm 5.2: Determine block connectivity

```
[ 1]:   read a grid, save face data in two stores f_{N,6} and g_{N,6}
[ 2]:   assign face connectivity to F_{N,6} and G_{N,6}
[ 3]:   computing face simplices, and save them in X_{N,6}, Y_{N,6}
[ 4]:   for (b:=1 to N)
[ 5]:     for (f:=1 to 6)
[ 6]:       F_{b,f} ← non–connectivity;
[ 7]:       G_{b,f} ← non–connectivity;
[ 8]:       X_{b,f} ← (centroid of face)
[ 9]:       Y_{b,f} ← (centroid of face)
[10]:   compare face simplices
[11]:   for (m:=1 to N)
[12]:    for (f_1:= 1 to 6)
[13]:      if (F_{m,f_1}) == non–connectivity)
[14]:        for (n:= 1 to N)
[15]:          if (m ≠ n)
[16]:            for (f_2:=1 to 6)
[17]:              if (F_{n,f_2} == non–connectivity)
[18]:                if (X_{m,f_1} == Y_{n,f_2})
[19]:                  connectivity : block m face f_1 → block n face f_2;
[20]:                  connectivity : block n face f_2 → block m face f_1;
[21]:                  out of m − loop : F_{m,f_1} ← connectivity;
[22]:                  out of m − loop : F_{n,f_2} ← connectivity;
```

Then, two loops for blocks m and n and their sub–loops for faces f_1 and f_2 are executed in order to detect block connectivity. Once a block connectivity is detected, a pair of neighboring faces are recorded (lines *[19]* to *[20]*), and the face condition is changed to *connectivity* (lines *[21]* to *[22]*), i.e., both faces are out of the loop.

- ▷ Step 1. A grid with N blocks is read. Its vertices are extracted from blocks and saved duplicated as two variables $f_{b,f}$ and $g_{b,f}$, where indices $b = 1, \ldots, N$, and $f = 1, \ldots, 6$ denote *block* and *face number*, respectively.

- ▷ Step 2. Coordinates of face simplices are constructed by computing the face centroids according to Eq. (5.7), and are stored using a two–dimensional array that indicates block and face number $X_{b,f}$ and $Y_{b,f}$.

- ▷ Step 3. Before a loop of comparing face simplices is started, variables $F_{b,f}$ and $G_{b,f}$ for face condition is introduced. At the beginning, all faces are assigned to *non–connectivity*. If a pair of face connectivity is found, both faces are assigned to *connectivity*.

- ▷ Step 4. To find a pair of block connectivity, simplices from distinct blocks are compared, as shown in Fig. 5.4. If the condition $|X_{b_1,f_1} - Y_{b_2,f_2}|$ for $b_1 \neq b_2$ is met, a pair of block connectivity is determined.

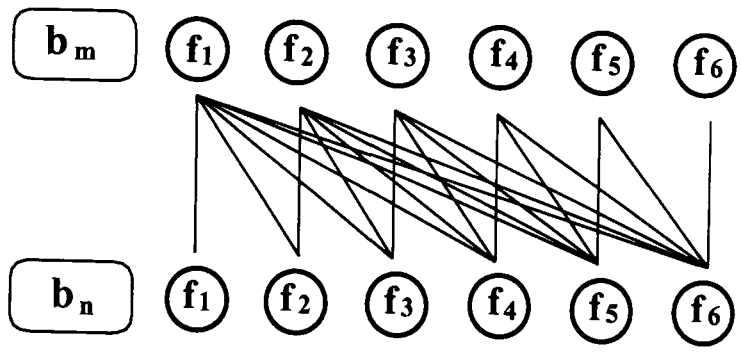


Figure 5.4: **The possibilities of block connectivity of two neighboring blocks amount $\sum_{i=1}^{6} i$.**

5.1.3 Generation of Block Matching Orientation

Data exchange on block interface of two neighboring blocks can only be performed correctly, if the relation of block matching orientation is identified. Similar to the generation of a block connectivity file, the matching orientation is determined by comparing vertex positions of two neighboring faces.

The algorithm used to automatically generate the matching orientations consists of three main steps.

- ▷ Step 1. Read a grid file, and extract the vertices from the blocks. Every four vertices are assigned to their corresponding block and face number.
- ▷ Step 2. Read the block connectivity file. Coordinates of four vertices of two faces neighbored are compared to determine their block matching orientation.
- ▷ Step 3. If a block matching orientation for two neighboring faces is determined, their matching case is added to the block connectivity file (see section 4.2.3).

5.2 Face Connectivity among External Faces

Face connectivity deals with the connectivity relations of the external faces. These relations are important for a grid adaptation process, since the degrees of freedom of grid points along the common edges of faces connected will be determined by their shape features.

A face connectivity file should provide the information about the neighboring relations among external faces, and the shape features of the common sides. For instance, in cases that common sides describe sharp edges, grid points will be moved along these sides.

5.2.1 Generation of Face Connectivity

In order to speed up this process, the existing block connectivity file is employed to separate the external faces from the internal faces. The problem is to find the common sides on the external boundary of a grid. One can use a representative quantity, e.g., the center of a side, to compare the position and orientation of two common sides from different faces. Since the principle of generating a face connectivity can be found in the algorithm for generating a block connectivity, the main steps of the algorithm are explained without the pseudocode.

- ▷ Step 1. Read a block connectivity file, and save all vertices of the internal faces, and the information about their block, face and side numbers.
- ▷ Step 2. Compute the geometric centers of all sides, and save the information about their block, face and side numbers.
- ▷ Step 3. Compare the distance between two centers of two different faces, and find the face connectivity relation, and then compare the vertices of the side to determine the face matching.
- ▷ Step 4. Assign the face connectivity and face matching in a face connectivity file.

5.3 Shape Features on Physical Boundary

The fixed boundary file explained in section 4.2.7 is formalized in order to store the geometric shape features, i.e., sharp edges. Since shape features appear only on a physical boundary, it is sufficient to identify the shape features on the external faces.

5.3.1 Determination of Shape Feature

There is no clear and strict definition of a measure for a sharp edge. However, such a geometric feature should be specified using a quantity according to the geometric accuracy required. Fig. 5.5 shows three blocks that are connected to each other. In case 1, face f_i of block b_1 and face f_j of block b_2, as well as face f_j of block b_2 and face face f_k of block b_3 are neighbored. The common edges of f_i and f_j as well as f_j and f_k have different shape feature than that of the common edge of face f_l and face f_m.

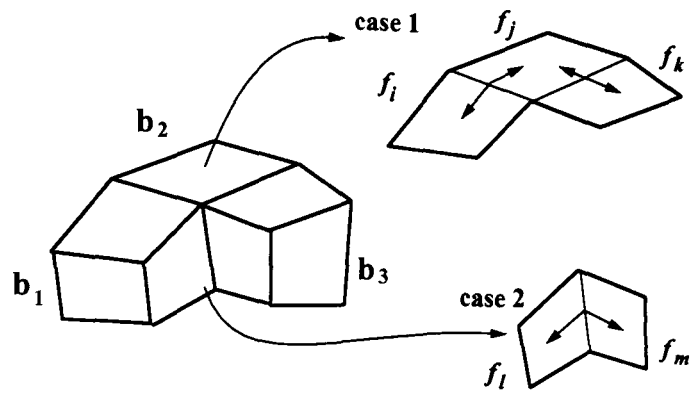


Figure 5.5: **Three blocks are connected each other. Two types of face connectivities are possible. (1) A sharp edge is formed by the boundary of two faces connected. (2) The common boundary of two faces do not build a sharp edge.**

A sharp edge is often formed by an intersection curve of two surfaces. To explain the method for identification of the geometric feature, the following definition is introduced.

Definition 5.3: Offset point. Let $p(x,y,z) \subset \mathbb{R}^3$ be a point on a parametric surface $S(u,v) \subset \mathbb{R}^3$. A point $q(x,y,z) \subset \mathbb{R}^3$ is obtained by projecting $p(x,y,z)$ in the normal direction of the surface with the distance $d(p(x,y,z), q(x,y,z))$

$$q(x,y,z) = p(x,y,z) + d \cdot \mathbf{n}$$

where $\mathbf{n}$ denotes the normal vector of surface $S(u,v)$ at point $p(x,y,z)$. Point $q(x,y,z)$ is defined as *offset point* of point $p(x,y,z)$ on surface $S(u,v)$,

Let two external faces $S_1(u_1,v_1)$ and $S_2(u_2,v_2)$ be connected, as shown in Fig. 5.6. Along their common side, points are one–to–one connected, i.e., any two points connected have the same coordinates. A pair of points p_1, p_2 ($p_1 \subset S_1$ and $p_2 \subset S_2$) is chosen, and two normal vectors $\mathbf{n}_1$ at p_1, and $\mathbf{n}_2$ at p_2 are calculated. Specifying a distance d, and projecting the points along the two normal vectors with this distance, one obtains two offset points q_1 and q_2. The angle θ between $\overline{p_1q_1}$ and $\overline{p_2q_2}$ is given by

$$cos\theta = \frac{(p_1 - q_1) \cdot (p_2 - q_2)}{|p_1 - q_1||p_2 - q_2|} \tag{5.8}$$

An edge feature is determined by comparing this value with the value prescribed for a sharp edge.

The algorithm for angle computation contains the following steps.

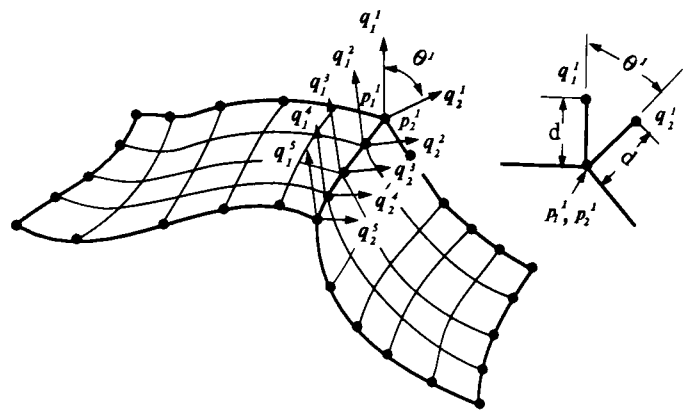


Figure 5.6: **Using a reference point at an edge intersection, from the both faces this point will projected along both normals by a prescribed distance.**

- ▷ Step 1. Based on the given geometric features, a value of a maximal angle θ_{max} is specified.
- ▷ Step 2. A pair of common edges is chosen. Offset points for both faces are generated. The mean value of the intersection angles for all pairs of the offset points θ_{mean} is computed. Comparing the mean value with the value specified, the shape feature of the common edge is determined and specified in the face connectivity file.
- ▷ Step 3. The process continues to the next common edges, until all common edges are calculated.

The information about geometric features indicates the degrees of freedom of grid points on physical boundary, and is stored in a face connectivity file for grid adaptation processing.

5.4 Chapter Summary

The main contribution of the present chapter is that the algorithms for determination of block connectivity, face connectivity, and shape features on fixed boundary are developed.

1. Block connectivity. In order to determine block connectivity, a quantity *face simplex* is built. This quantity represents the position of a face uniquely and unambiguously. Comparing face simplices of different blocks, the neighboring relations among blocks can be found. In addition, corresponding four pairs of

vertices of two neighboring faces are compared in order to determine the cases of matching orientation.

2. Face connectivity. In order to generate data exchange on fixed boundary, the connectivity relations among the faces of the fixed boundary must be determined. Similar to the above method, this type of connectivity relation will be determined using a comparative quantity, built from a side of a face, while their matching orientation is determined by comparing two pairs of vertices of two faces connected to each other.

3. Shape features on physical boundary. Sharp edges must be identified, so that the geometric feature on the physical boundaries can be retained during a grid adaptation process. Since there is no strict measure for a sharp edge, the detection of shape features on fixed boundary requires user specification.

Chapter 6

Grid Design Strategy and Construction Rules

There is presently no theory or set of construction rules for a topology design methodology for block structured grids, consisting of a general process of wireframe building, and the control of grid quality.

It is well known that efficiency in grid generation depends not only on a mesh tool, but also on a comprehensive knowledge of topology concepts and graphic imagination. From this standpoint, it is expected that a general design methodology provides both the theoretical and practical foundations for meshing complex geometries in reasonable time and effort. The questions often in grid generation practices are

- *How can one understand the design object to be meshed?*

and

- *Is it possible to establish a set of grid construction rules, such that a structured grid will be efficiently generated?*

The objective of the present chapter is to answer the above questions with a new strategy for grid design, and a set of grid construction rules. The strategy deals with the general methods for decomposing a complex meshing task into a set of sub–tasks, and their grid topology building. The simplified forms of these sub–tasks provide the designer with fewer abstract images of his grid design object. The grid construction rules are experience–based methods to support grid design reasonably and efficiently. In contrast to the grid construction rules for automatic generation of block structured grids in [68] and [70], the grid construction rules in this thesis are not presented in a mathematical sense.

6.1 Grid Design Strategy

A grid design process usually contains human interaction as well as data processing. Human interaction deals with defining a valid geometric object (see section 2.1.1), generating its geometry surface and block structure, and specifying boundary conditions for grid generation, while a data processing deals with numerical solution of grid generation equations, e.g., the elliptic partial differential equations, explained in section 3.2.

To simplify an abstract meshing object, a complete meshing task is divided into sub–tasks using certain rules. These sub–tasks occupy some spaces, and are parts of the complete solution domain. Their block topologies represent therefore the local block structures, called *local grid topologies*. Due to their simpler forms than the complete block topology, the local block topologies are built at lower levels of geometric and topological complexity.

The conceptual model of a grid generation process is shown in Fig. 6.1 and can be formulated as follows.

- ▷ **Analysis of a meshing task**. A meshing task may contain special requirements and restrictions. Based on the known information, a complete meshing task will be divided into a set of sub–tasks, called *meshing objects*. They will be individually treated at lower levels of geometric and topological complexity.
- ▷ **Determination of relations among meshing objects**. For the integration of all meshing objects, one needs to determine the geometric and topological relations among them.
- ▷ **Data processing**. First, wireframe models are individually built for meshing objects. Assembling these wireframe models with respect to their relations, grid topology building is completed.
- ▷ **Evaluation of design result**. The initial design result is controlled by criteria of quality. New requirements and constraints are specified according to the result of evaluation.

6.1.1 Analysis of a Meshing Task

The attempt of analysis of meshing task is to more accurately declare some requirements and restrictions for a design objective. Based on this analysis, a design planning or working schedule can be established. In cases of complex geometries, a precise analysis of meshing task might avoid mistakes in the

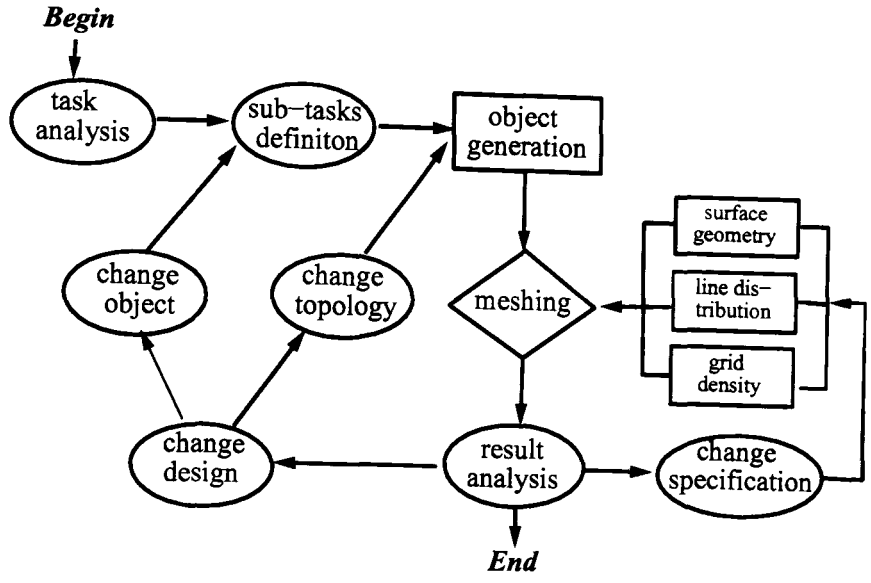


Figure 6.1: **The conceptual model of a grid generation process. In this diagram, human interaction, process and data are represented by symbols ellipsoid, parallelogram and rectangle, respectively.**

meshing objective specified.

6.1.1.1 Meshing Requirements

In the example **a generic aircraft in subsonic flow**, shown in Fig. 6.2, the grid generation consists of the following topics:

- ▷ description of the complete aircraft geometry;
- ▷ definition of an outer flow boundary;
- ▷ building of block topology, and
- ▷ generation of the grid.

Since meshing requirements and restrictions are determined based on the information about engineering (structural) and geometric features, as well as flow physics, all relevant information is collected as follows:

- ▷ **Structural features**. There are three components, namely, fuselage, wing, and vertical fin.
- ▷ **Geometric features**. In a Cartesian coordinate system, geometric shapes of different objects are expressed in a generic form.
- ▷ **Flow field**. Flow is subsonic flow. The outer flow boundary should be far away from the aircraft.

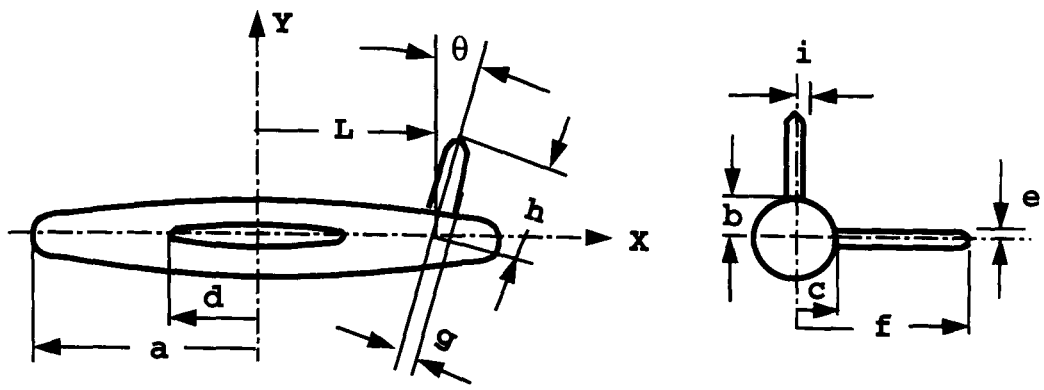


Figure 6.2: **A generic aircraft consists of fuselage, wing and vertical fin. The dimensions and positions of all components are given in the drawing.**

6.1.1.2 Definition of Meshing Objects

Three components have their own structural features, and are considered as three individual parts. Complete geometric configuration is decomposed into a set of parts or sub–tasks. The meshing task is transformed into a set of sub–tasks, called *meshing objects*. Using transformation and rotation operators, their positions in the given coordinate system are determined. Their simplest shapes of generic forms can be approximated at a lower degree of geometric complexity.

The grid construction begins with building grid topologies for meshing objects, and then relations among them are generated. Dealing with a complete meshing task, these relations have to be clearly and correctly described for the final assembly of all objects.

6.1.2 Determination of Relations among Meshing Objects

Individual generation of meshing objects reduces the complexity of geometry and topology. However, it may be difficult to assembly all meshing objects together. In order to describe these relations exactly, a graph of a network system, called *object tree*, is introduced.

6.1.2.1 Generation of an Object Tree

Similar to the graph *geometry tree* used in engineering design [24], the relations among meshing objects is represented by a graph called *object tree*. In an object tree, the sub–tasks are represented by nodes, while the relations between sub–tasks are represented by routes between nodes.

Fig. 6.3 shows the object tree of the example in Fig. 6.2. The root of the tree is the meshing task. Each branch from the root represents one meshing object. Introducing an orientation vector, denoted by $V = [x_0, y_0, z_0, \alpha_x, \alpha_y, \alpha_z]^T$, the object *vertical fin* O_3 is placed to its position.

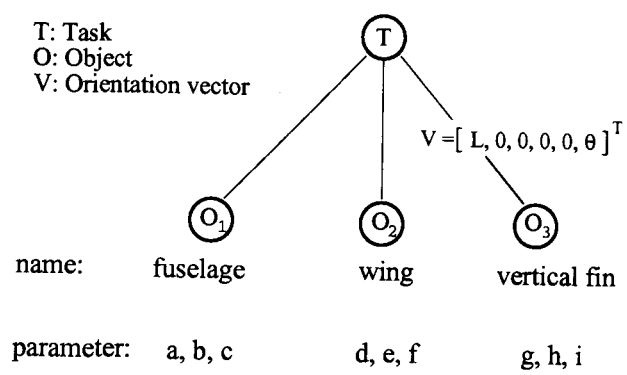


Figure 6.3: **An object tree represents relations of coordinates and locations among sub–tasks of the example in Fig. 6.2.**

The above object tree supports the understanding of a complex meshing task and the relationship between the meshing task and meshing objects.

6.1.3 Data Processing

First, geometric configuration given in a construction drawing is modelled using a CAD tool, and its output file is converted into a suitable boundary description of the solid model. The process of geometry data generation, called *generation of surface description*, is explained in section 2.2. A more complicated data processing procedure involves the wireframe building for meshing objects. We employ the object–oriented method for wireframe building. That is, any of the wireframe structures represents a partial block structure in the solution domain. It is important to assign and specify their interfaces to other meshing objects. Then, the wireframe models are assembled according to the sequence specified

with respect to their interface relations.

6.1.3.1 Generation of Local Grid Topologies

The essence of an object–oriented grid topology building is to encapsulate all the information of each component within an individual package, described by a data structure defined for the object. It is so–called *object hiding*. The complete meshing task is considered as being made up of meshing objects and their relations, represented in an object tree. The nodes are instances of objects, and will be treated locally. Combination of all local nodes is the *global topology generation*.

A simple approach to encapsulating a local topology is to generate a set of cubes connected in a structured form, called *framework*.

> **Definition 6.1: Framework**. A wireframe model is built in the manner that its vertices are ordered in a three–dimensional array (i, j, k), where i, j, and k denote the vertex numbers in three directions, respectively. There are $(i-1) \times (j-1) \times (k-1)$ cubes arranged in the form of a box. This wireframe model is defined as *framework*.

For instance, a framework with the index $(2, 2, 4)$ means that this framework consists of 2 vertices in the first dimension, 2 vertices in the second dimension, and 4 vertices in the third dimension. Introducing a function `framework(i,j,k)`, where arguments i, j, and k denote the number of vertices in three dimensions, the framework is represented by `framework(2,2,4)`.

Using this definition, the fuselage and wing of the example of Fig. 6.3 are separately described by frameworks `framework(2,2,4)` and `framework(4,2,2)`, respectively, as shown in Fig. 6.4. Using translation, rotation or deformation operators, the meshing objects can be assembled together in an easy manner.

6.1.3.2 Generation of Global Grid Topology

Since there is no mesh tool that automatically completes a meshing task, human participation is still necessary. The human participation in an informal basis will solve the issues such as selection of topological structures, or variation of local block structures, while a data process, i.e., mesh generation, is operated by the formalized input data and the rules applied.

Building of the objects does not obey a certain sequence. However, an assembly of objects is often sequence–dependent [55]. These two important facts have to be considered in grid design. Since an object tree can only indicate the relations of mesh objects in its finally assembled form, and does not contain any information about a sequence of assembly, the global topology building is the generation of interfaces of objects.

In order to ensure the correctness of a global topology building, the designer should have a complete outline of the meshing task, as well as the relations among the meshing objects.

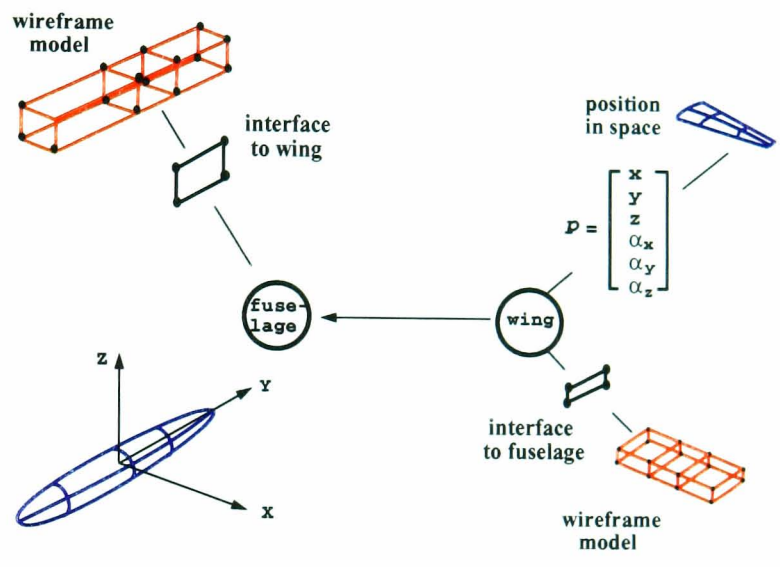


Figure 6.4: **The assembly graph gives a geometric and topological description of the connectivity relation between these two objects.**

Fig. 6.4 gives a geometric and topological description of the connectivity relation among the wing and fuselage. The meshing objects, their symbols of the object tree, and the assembly sequence directed by the arrow, are illustrated. However, it is more difficult to sketch an assembly graph for a complex geometry. A short form for representation of an assembly is introduced, called *assembly function*. The objects fuselage, wing and vertical fin are substituted by the symbol O_f, O_w, and O_v. Introducing a notation $s(O_i, O_j)$ to express a sub–assembly of an object O_i to O_j, and an assembly function $A\{s_1(O_w, O_f), s_2(O_v, O_f)\}$, the sequence and direction of a complete assembly process are indicated in a simple form. In reality, an assembly of meshing objects could be substantially complicated due to the complexity of a geometric configuration as well as the topological structures.

6.1.4 Evaluation of Results

The initial grid result is evaluated in order to determine whether the design objective was achieved. Upon failure, part of the design refinement is required. Backtracking is necessary to find out the refinement steps that were responsible for the mismatch between design specification and its result. The refinement might be expressed as a selection of a subset of all known rules which are already being considered applicable.

The basic requirements of a structured multiblock grid are in two aspects. Firstly, the accuracy of geometry model must be ensured, such that flow features can be simulated. Since this accuracy handles the shape and geometry of a grid, it is called *accuracy of grid geometry.* Secondly, cells of a grid must have a good quality, such that numerical errors in flow computation are reduced. The grid quality associated with cell is called *cell quality.*

6.1.4.1 Accuracy of Grid Geometry

Accuracy of grid geometry can only be estimated for some special configurations, e.g., a rectangle, a circle, the geometric accuracy can be determined. Due to geometric complexity in most cases, it is unlike to measure a tolerance between a given geometric configuration and a surface geometry of a mesh. It is suggested to measure the accuracy of geometry modelling using some representative geometric features.

- ▷ **Sharp edge**. The form of a sharp edge will give a clear sign whether the shape feature required is modelled.
- ▷ **Intersection curve**. An intersection curve should be modelled by the grid elements edges and vertices. This ensures that the curve shape can be used to keep the geometric given feature.
- ▷ **Location of a large curvature**. The geometric feature at this location will be accurately modelled, if a high grid density is generated. A general rule is: *the larger the curvature, the larger the grid density.*

6.1.4.2 Cell Quality

The cell quality is evaluated by three main criteria. They are: orthogonality of grid lines, cell smoothness, and cell aspect ratio.

Orthogonality. A general discussion of orthogonal systems on planar and curved surfaces, as well as various generation procedures are given in [27]. The measure of orthogonality of the grid lines closed to the fixed boundaries is that the angles between mesh lines is not larger than 45 degrees in the domain, and continuity of the mesh slopes and volumes. Small deviation of the mesh lines from one cell to the next one, an it should not exceed 10 to 15 degrees [49].

Cell smoothness. This quality is also considered as *regularity*. It requires that grid lines are uniformly distributed in all curvilinear coordinate directions of a grid [50], [51].

Cell aspect ratio. An aspect ratio is measured by calculating the length proportion between two arc lengths along a computational coordinate direction. A measure of cell aspect ratio could be: small volume rate between adjacent cells is kept between 0.7 and 1.3 in critical areas [49].

6.1.4.3 Local Cell Repair

A special tool for cell repair is developed in [72]. The cell qualities of a grid are checked using the above three criteria. The positions and the values of bad cells are indicated in the form of a list. The user can invoke these cells as well as their neighboring cells to repair them locally.

6.2 Concept of a Topology Database

In section 2.1.2 some surface types were defined as primitives. The formal representation of the primitives provides the possibility to vary their forms through parameters. The question arises of whether topological structures can be incorporated in geometric configurations, such that special wireframe models can be stored or reused as parametrized components. Based on the idea of knowledge–based system in manufacturing design [56], the concept of a *topology database* is developed. The goal is that the information about geometry and topology of an object is structured at a generic level [39], and both geometric and topological statements of an object are saved in a predefined form. For instance, positions, sizes, and wireframe descriptions of a geometric configuration are saved in a standard form, and can be varied by corresponding parameters, they

supply a flexibility and re–usability for grid topology building. In a topology database, these statements are called *topology entities.*

Fig. 6.5 shows the principle of a topology database. The user invokes entities, and inputs their parameters. The information about shapes and wireframe models of entities is processed according to their definitions in a topology database. An initial model is visualized, such that the user can modify his design parameters. The user interaction will be iteratively performed until the result is satisfactory, and output as the final model.

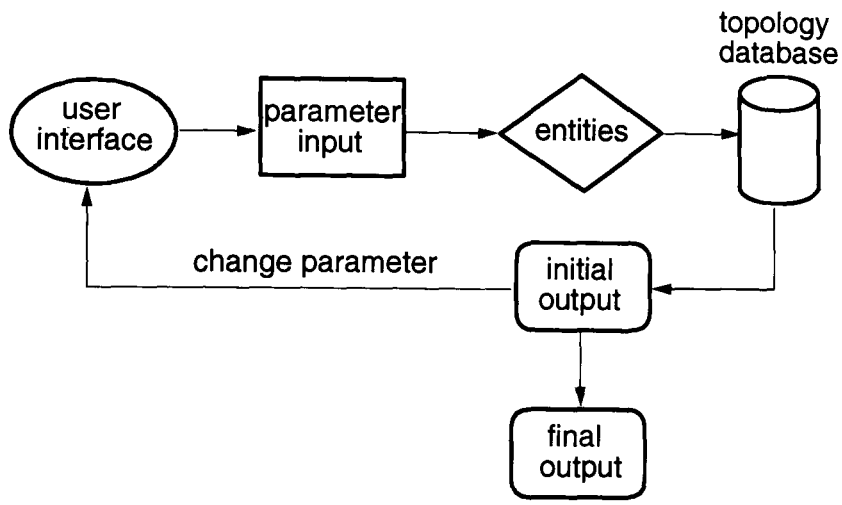


Figure 6.5: **The user interaction for generating an object using a topology database. In this diagram, the human interaction, input, process and output are represented by ellipsoid, rectangle, parallelogram and rectangles with rounded off vertices, respectively.**

6.2.1 Topology Entities

In a topology database, each entity is defined associated with its corresponding geometric description, which are restricted to explicit and implicit algebraic equations, and wireframe models predefined. Their shapes, sizes and positions can be varied by using scaling, translation or rotation operators. Variables of an entity are:

- ▷ **Vertices**. Vertices are numbered according to the definition of entities. They are ordered in a sequence.
- ▷ **Position of vertices**. The positions of vertices, represented by its parameter, are contained in an entity.
- ▷ **Surface**. A vertex has not only its number and position, it also belongs to a certain surface. The assignments of vertices to surfaces are contained in an entity.

▷ **Linkage of vertices.** The link relations among vertices are given in a backward manner, namely, the vertices of higher numbers are linked to those of lower numbers.

6.2.1.1 Standard Topology Entities

As an example, the topology entity *torus* is given. Shape and wireframe model of a torus are saved at a generic level. Invoking this entity and specifying shape parameters r for radius of cross section, and R for radius of torus, its surface is generated by

$$x = (R + rcos\phi)cos\theta,\ y = (R + rcos\phi)sin\theta,\ z = rsin\phi\ \ ,(R > 2r) \qquad (6.1)$$

where ϕ and θ are the sweep angles, respectively, as shown in Fig. 6.6.

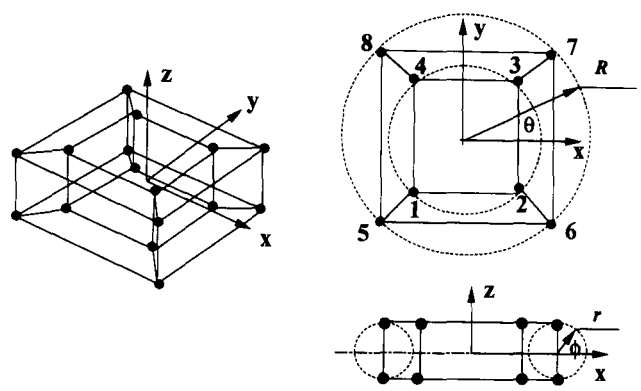


Figure 6.6: **The information about the entity *torus* is saved in a topology database. It can be invoked by inputing required parameters.**

The vertices, assigned to their corresponding surfaces, are numbered, and ordered in a sequence. In addition, the linkage relation among vertices is indicated, as shown in Table 6.1. For instance, vertex 1 has connectivity with vertices 2. The backward link relation means that the connectivity is interpreted by *vertex 2 to vertex 1.* This information is found in the table by row 2.

6.2.1.2 User–Defined Topology Entities

The most complex geometry surfaces are described by non–geometry primitives. In modern industrial design environments, geometry shapes of design objects are incorporated with optimization processes. The need for special templates of design objects leads to an extension of the standard topology entities. These templates have unchanged block topologies, while the corresponding geometry surfaces are variable. This type of templates is called *user–defined entities.*

Table 6.1: Data structure of topology entity for a generic torus in a database

V	$p(x, y, z)$	surface #	vertex linkage
1	$-(R-r)\sqrt{2}^{-1}, -(R-r)\sqrt{2}^{-1}, -r$	1	
2	$(R-r)\sqrt{2}^{-1}, -(R-r)\sqrt{2}^{-1}, -r$	1	1
3	$(R-r)\sqrt{2}^{-1}, (R-r)\sqrt{2}^{-1}, -r$	1	2
4	$-(R-r)\sqrt{2}^{-1}, (R-r)\sqrt{2}^{-1}, -r$	1	1,3
5	$-(R+r)\sqrt{2}^{-1}, -(R+r)\sqrt{2}^{-1}, -r$	1	1
6	$(R+r)\sqrt{2}^{-1}, -(R+r)\sqrt{2}^{-1}, -r$	1	2,5
7	$(R+r)\sqrt{2}^{-1}, (R+r)\sqrt{2}^{-1}, -r$	1	3,6
8	$-(R+r)\sqrt{2}^{-1}, (R+r)\sqrt{2}^{-1}, -r$	1	4,5,7
9	$-(R-r)\sqrt{2}^{-1}, -(R-r)\sqrt{2}^{-1}, r$	1	1
10	$(R-r)\sqrt{2}^{-1}, -(R-r)\sqrt{2}^{-1}, r$	1	2,9
11	$(R-r)\sqrt{2}^{-1}, (R-r)\sqrt{2}^{-1}, r$	1	3,10
12	$-(R-r)\sqrt{2}^{-1}, (R-r)\sqrt{2}^{-1}, r$	1	4,9,11
13	$-(R+r)\sqrt{2}^{-1}, -(R+r)\sqrt{2}^{-1}, r$	1	5,9,13
14	$(R+r)\sqrt{2}^{-1}, -(R+r)\sqrt{2}^{-1}, r$	1	6,10,14
15	$(R+r)\sqrt{2}^{-1}, (R+r)\sqrt{2}^{-1}, r$	1	7,11,15
16	$-(R+r)\sqrt{2}^{-1}, (R+r)\sqrt{2}^{-1}, r$	1	8,12,13,15

The concept of a topology database is implemented in [29]. Like a geometry library in many CAD systems, the knowledge–based topology database supplies the user to improve his grid design.

6.3 Grid Construction Rules

The immediate purpose or motivation of developing a set of construction rules is twofold, that is first, to obtain objective and scientific knowledge on grid generation activities and second, to optimize or make the grid generation process more efficient and economical. The construction rules based on the experience of grid generation have been successfully applied in grid design over several years. The core of the construction rules can be summarized as follows.

- ▷ **Reduction of degrees of complexity**. Adequate geometric and topological information is available and contributes to finding the way to assemble and disassemble the objects from and into sub–assemblies and parts respectively.
- ▷ **Object–oriented grid generation**. Designers can see and inspect the shape to whatever degree of geometric complexity at which they are

working.

6.3.1 Definition of Solution Domain

The domain over which the mathematical model is applied is often limited to a portion of the complete domain. Definition of a solution domain idealizes a geometric object through removing some microscopic geometric features that are not needed for a model building. However, these idealizations add errors, i.e., a simplification of the computing model reduces the geometric accuracy. Details of how geometric simplification of the domain are introduced to the model can be critical to the solution reliability. Since there is no existing rule to explain the criteria for model simplification, the execution of model simplification should regard the requirements of flow simulation. Usually, microscopic components that do not have influence on the particular flow are removed. For instance, it is not meaningful to model geometric components like screws, bolts or nuts in the simulation process of a spacecraft. Suppose that a given geometric object is simplified as necessary. The next step is to define a solution domain for grid generation. Two types of meshing domain are defined as follows, as shown in Fig. 6.7.

Definition 6.2: The first type of meshing domain. Let $\{D : p(x, y, z)\} \subset \mathbb{R}^3$ be a domain. Its boundary is represented by a continuous and closed boundary $\{S : p(x, y, z)\}$. If the meshing area can be expressed by the set $\{\Omega : D \cap S\} \subset \mathbb{R}^3$, the domain is defined as *the first type of meshing domain*, denoted by Ω_1.

Definition 6.3: The second type of meshing domain. Let $\{D_1 : p(x, y, z)\} \subset \mathbb{R}^3$ and $\{D_2 : p(x, y, z)\} \subset \mathbb{R}^3$ be two domains. Their boundaries are represented by two continuous and closed boundaries $\{S_1 : p(x, y, z)\}$ and $\{S_2 : p(x, y, z)\}$. If the meshing area can be expressed by the set $\{\Omega : (D_2 - D_1) \cap S_1\} \subset \mathbb{R}^3$, the domain is defined as *the second type of meshing domain*, denoted by Ω_2. The boundary S_1 is called *internal mesh boundary*. while S_2 is the *external mesh boundary*.

During a grid generation process, grid points on physical boundaries are considered as known initial and boundary conditions, while other points are calculated by solving governing grid generation equations. Similar to this process, the physical boundary of a meshing domain is considered as a known condition for grid topology building. That is, a grid topology building begins with generating wireframe models for a set of the known boundaries of geometric components. The

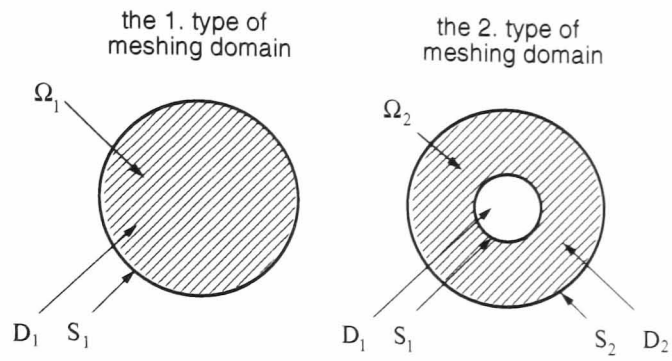


Figure 6.7: **Two types of meshing domain.**

wireframe models for meshing objects are locally generated. They will be extended to other regions of the solution domain, so that a global grid topology is completed. An important question is how to use this idea to carry out the whole wireframe building.

Definition 6.4: Meshing direction. A meshing direction is referred to as the direction from a start wireframe model to complete wireframe model. There are two types of meshing direction, used in building a wireframe grid topology, namely, inward and outward meshing directions. If a wireframe model is built starting from the outer boundary and going inward to the body surface, it is defined as *inward meshing direction.* It the wireframe model is built starting from the body surface and going outward to the outer boundary, it is defined as *outward meshing direction.*

In case of the first type of meshing domain (see Definition 6.2), there exists only an external boundary. To represent the shape features of the surface geometry accurately, a wireframe topology should be built first for the surface geometry of the external boundary. This wireframe model, represented by a set of continuous two–dimensional blocks, covers the whole surface of the physical boundary. Extending this two–dimensional layer of wireframe model inward to the given meshing domain, and varying its block structures in location, where a space is not filled with block structures, the complete grid topology is obtained.

In case of the second type of meshing domain (see Definition 6.3), there are an external boundary as well as a set of internal boundaries. For instance, the solid boundary of the spacecraft X–33 is of highly geometric complexity. It consists of lifting body, vertical fins, aerospike engine etc., while an outer flow boundary is defined in the form of an ellipsoid or box, as shown in Fig. 6.8. The geometric complexity of the meshing objects on the internal boundary is much higher than that of the outer boundary. A wireframe building is considered as being

a process of extension of a two–dimensional wireframe model into the third dimension with constant or varying block structures. A layer–wise variation of block structure is easily done if the extension begins with the surface whose block structure is of the highest topological complexity. As a result, the following rule for selecting a meshing direction is established.

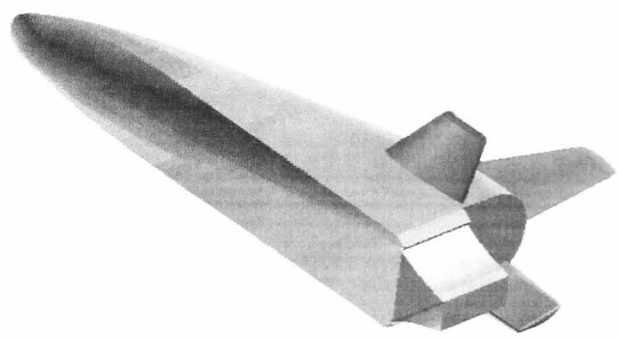


Figure 6.8: **The internal boundary of the meshing domain is described by the surface geometry of the X–33 vehicle. Due to its geometric complexity, the outward meshing direction is selected.**

Grid construction rule 1: Selection of a meshing direction. Based on the complexity of the two–dimensional topology, the surface with the most complex geometry is chosen and its two–dimensional topology is continued in three–dimensional space following the direction of its normal vectors. Thus, it is ensured that the wireframe building goes in a direction that does not increase the topological complexity of the wireframe model.

Fig. 6.9 shows a solution domain defined by both an internal and an external boundary. Suppose that the internal boundary represents the geometry to be meshed, while the external boundary is an artificial outer boundary. The outer boundary is described by an ellipsoid. A meshing direction should be selected, so that the topological complexity of block structures is reduced during the extension of wireframe structure. Conversely, the designer is facing against an increase in the complexity of his wireframe structure from the outer boundary toward the meshing objects.

Another reason for the choice of the outward meshing direction is based on the spatial visibility during wireframe building. Extending a two–dimensional wireframe model inward to the internal boundary, the front of a wireframe model is not visible due to the interference of the views. The designer may lose his

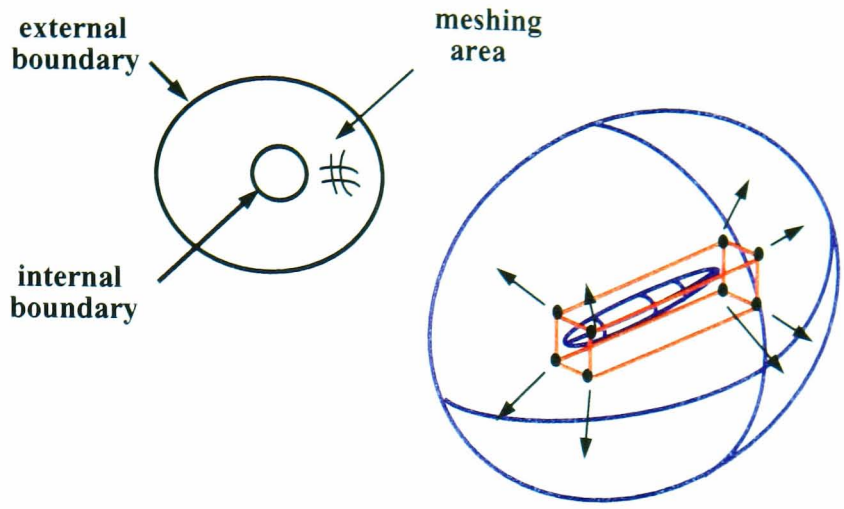


Figure 6.9: **An outward meshing scheme is used in case that a solution domain is described by an external and a set of internal boundaries.**

understanding of the global block topology.

6.3.2 Generation of Surface Description

In order to describe the boundary of a meshing area, a CAD output is processed in the manner that the surface geometry of a solid model is described by a set of geometry surfaces. They can have analytical, parametric or discrete data formats. In case of parametric or discrete data formats, a complete geometry surface may be C^1–discontinuous. The abrupt changes of normals of elements imply geometric features. In wireframe model building, there is need for specially treating these features.

> **Grid construction rule 2: Partitioning of a surface along curves that are not C^1–continuous**. Any surface is partitioned into a set of patches that are C^1 or slope continuous. A curve, connecting any two patches, may be C^0–continuous. The grid generation process has to account for this geometric feature by reducing the degree of freedom for the movement of grid points.

The goal of partitioning a surface description along curves that are not C^1–continuous is to control slope discontinuous points. For instance, a surface description of an arbitrary airfoil can be roughly divided into three pieces. The first piece gives the geometry of the leading edge. The other two pieces represent the profile of both windward and leeward sides, respectively. At the trailing edge, the slope is not uniquely determined. Curvatures along the line segments

are taken in account in a wireframe model building. Fig. 6.10 shows two alternatives of block topologies. The first block topology neglects the geometric feature of the trailing edge, while the second one fixes the trailing edge through the singularity point of four blocks.

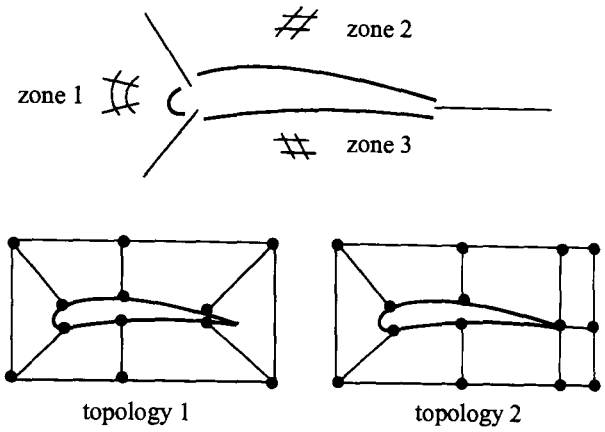


Figure 6.10: **A closed surface geometry is separated into three parts due to large changes of surface curvatures.**

6.3.3 Wireframe Building

A complex wireframe model is made of active elements (procedures) and passive elements (data). They comprise primitive topological structures and higher level constructs created by the repetitive invoking of predefined entities and employing binding rules to those at lower levels. The construction rules, developed for wireframe topology building with complex geometries, are summarized in the following sections.

6.3.3.1 Block Topology on Solid Boundaries

A three–dimensional grid topology can be considered as being an extension of a two–dimensional block structure into the third dimension. The wireframe topology on a physical boundary has to cover all surface region with a set of quadrilateral block structures, shown in Fig. 6.11. Partitioning a complete physical boundary into a set of regions, and projecting them to a two–dimensional plane, the wireframe block topology for each region must reflect a two–dimensional block topology in the transformed plane.

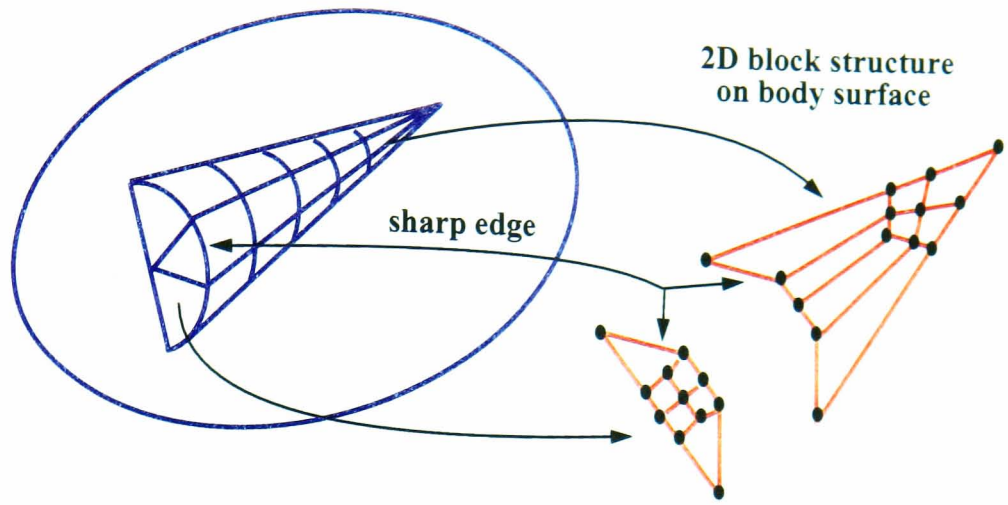


Figure 6.11: **On a solid boundary, block topology must have a two–dimensional block structure. This rule is employed to check whether all regions of the solid boundary are covered by the structured block topology.**

Grid construction rule 3: Block structure on a solid boundary. On a solid boundary, represented by a closed surface, a block structure is devised for the three–dimensional surface. As a surface, it must have a two–dimensional block structure.

6.3.3.2 Interfaces among Objects

Generation of connectivity relations among objects and assembly form of them is the key issue of completing a grid topology. The topological structures of the outer vertices of objects are considered as interfaces. A wireframe model that provides all objects geometric and topological environment, in which the objects have their unambiguous locations, and their interfaces have unique connections to the local block structures of the wireframe model, is called a *background wireframe model*. In order to generate an available interface and background wireframe, the following rules are established.

Grid construction rule 4: Interface of an object. The interface of an object is represented by its outer block topology. In order to integrate the object into the background wireframe model, the interface must have the same topology as the local topology of the background wireframe model.

As an example, the interface of a generic fuselage is generated. The wireframe of the fuselage is illustrated in Fig. 6.12. The interface of the fuselage is built by a $(2, 2, 4)$ wireframe model. An assembly of the object into a background wireframe model is only possible, if the interface structure is a part of the

background wireframe model.

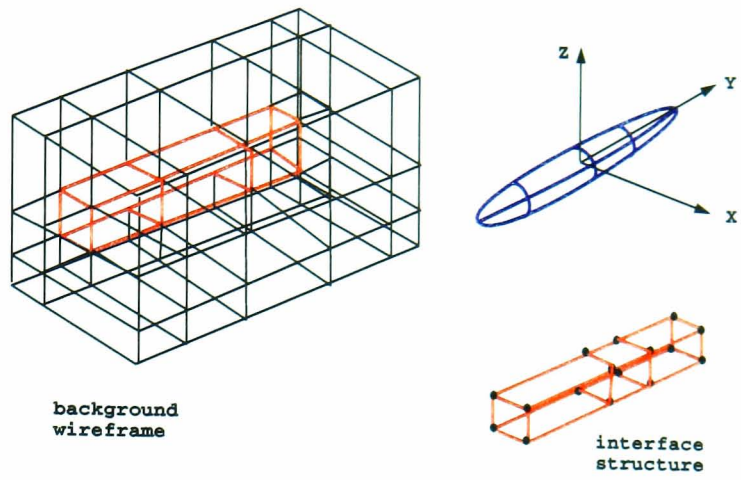


Figure 6.12: **The interface of fuselage is generated by a $2\times2\times4$ wireframe. This structure is part of the the background wireframe model.**

In general, the interface of an object is built in a simple form, i.e., the external boundary of its wireframe model is represented by a framework (see Definition 6.1), when it is possible. A complex interface increases the complexity of the background wireframe model. This will make an assembly of objects into a background wireframe model difficult.

6.3.3.3 Background Wireframe Topology

A background wireframe model is filled into the entire solution domain. It covers all regions, where objects will be inserted.

> **Grid construction rule 5: Generation of a background wireframe**. A background wireframe has to be generated such that it can accommodate the topologies of all objects in the object tree together with their topological interfaces to neighboring objects. The region, where an object will be inserted needs to have the same topological structure as the visible (external) topology of this object along with its set of interfaces describing the interference with neighboring objects.

6.4 Chapter Summary

The main contribution of the present chapter consists of the following three parts.

1. Object–oriented grid generation. Grid design strategy presented in this chapter is developed based on the object–oriented method. The essential core

of this method is to divide a complete meshing task into a set of sub–tasks, called *meshing objects*, so that the meshing task can be treated at lower levels of geometric and topological complexity. The relations among meshing objects are described in an object tree. The meshing objects in the object tree are generated individually. Their block topologies represent parts of the complete grid topology, and are called *local topologies*. Assembling all local topologies, the complete grid topology (*global topology*) is finished.

2. Concept of topology database. A topology database consists of a set of topology entities defined. They store analytic and parametrized surface geometries and their corresponding block topologies. It serves as a workbench for the designer. Invoking these entities with suitable parameters, some meshing objects are generated.

3. Grid construction rules. In order to provide the CFD engineer technical supports in his grid design process, a set of grid construction rules is summarized from the practices of grid generation over several years. They cover the following issues:

- ▷ **Grid construction rule 1: Selection of a meshing direction**. Based on the complexity of the two–dimensional topology, the surface with the most complex geometry is chosen and its two–dimensional topology is continued in three–dimensional space following the direction of its normal vectors. Thus, it is ensured that the wireframe building goes in a direction that does not increase the topological complexity of wireframe model.

- ▷ **Grid construction rule 2: Partitioning of a surface along curves that are not C^1–continuous**. Any surface is partitioned into a set of patches that are C^1 or slope continuous. A curve, connecting any two patches, may be C^0–continuous. The grid generation process has to account for this geometric feature by reducing the degree of freedom for the movement of grid points.

- ▷ **Grid construction rule 3: Block structure on a solid boundary**. On a solid boundary, represented by a closed surface, a block structure is devised for the tow–dimensional surface. Therefore, it must have a two–dimensional block structure.

- ▷ **Grid construction rule 4: Interface of an object**. The interface of an object is represented by its outer block topology. In order to integrate the object into the background wireframe model, the interface must have the same topology as a local topology of the background wireframe model.

- ▷ **Grid construction rule 5: Generation of a background wireframe**. A background wireframe has to be generated such that it can accommodate

the topologies of all objects in the object tree together with their topological interfaces to neighboring objects. The region, where an object will be inserted needs to have the same topological structure as the visible (external) topology of this object along with its set of interfaces describing the interference with neighboring objects.

Chapter 7

Grid Example for a Complex Topology

The grid generation strategy explained in the previous chapter stems from the numerous ideas of the engineering design theory [105]. As it is mentioned, the essence of the object–oriented method for grid generation is to reduce geometric and topological complexity by dividing a complete meshing task into a set of sub–tasks. The sub–tasks are the meshing objects. In wireframe building, each meshing object will be individually treated with respect to its relations to other objects. Since they are at the lower level of geometric complexity than the main task, wireframe models of sub–tasks are easily generated. Another advantage of using this method is that the objects are built in a manner that their interfaces to other objects are described by simple topologies. The geometric complexity of an object can be encapsulated with its local topology.

The work described here has been used over several years to generate numerous grids with complex geometries and topologies. The example presented in this chapter is chosen from a multiblock grid for the Ariane 5 launcher, which is one of the most important International Launch Vehicles [74]. The grid is generated for simulating its flight under subsonic and transsonic flow conditions. A detailed description of the strategy as well as the application of the construction rules are given below.

7.1 Analysis of the Meshing Task

Ariane 5 is the most heavy–lift launcher in Europe [22], [23]. Its first stage comprises liquid hydrogen oxygen engine, and upper stages. The segmented

solid–fuel boosters augment its lift–off capability. It is able to launch 18.000 kg into low earth orbit, and 7,000 kg into geostationary transfer orbit. The diameter of the payload fairing amounts to 5 meters with the useful diameter of 4.8 meters. Because of its capability of lofting a large payload, two satellites can be transported in one mission to orbit. The lower satellite will be mounted in a bulbous cocoon–like structure referred to as *SPELTRA* and *SPILMA* with the latter mounted on top of the *SPELTRA* [74].

The goal of the flow simulation is twofold. Firstly, the flow separation effect was observed during the Ariane 502 mission. This phenomenon causes vibrations, which may damage the instruments to be transported to orbit. In order to reduce these vibrations, the construction is modified by adding components to stabilize the structure. These components increase the weight of the launcher, and change aerodynamic behavior of the launcher. Secondly, according to the data measured during the Ariane 502 mission, an unexpected rotation of the complete launcher around the main stage axis was observed.

7.1.1 Meshing Requirement

Flow simulation is concentrated both on a subsonic and on a transsonic flow condition. Requirement for grid generation is specified as follows.

- ▷ **Flow separation**. In order to explain the phenomenon of flow separation the components such as attachments, helium tank, have to be accurately modelled.

The geometry of the Ariane 5 launcher is modeled by the CAD system CATIA [21]. The original CAD file contains the detailed geometry design, shown in Fig. 7.1. The complete CAD output consists of several hundreds of IGES entities [67]. The micro–components, such as screws or nuts, have no influence on the flow phenomena to be simulated. They are removed to considerably simplify the mesh generation process.

7.1.2 Definition of Meshing Object

The components taken into account are the sub–tasks. They are:

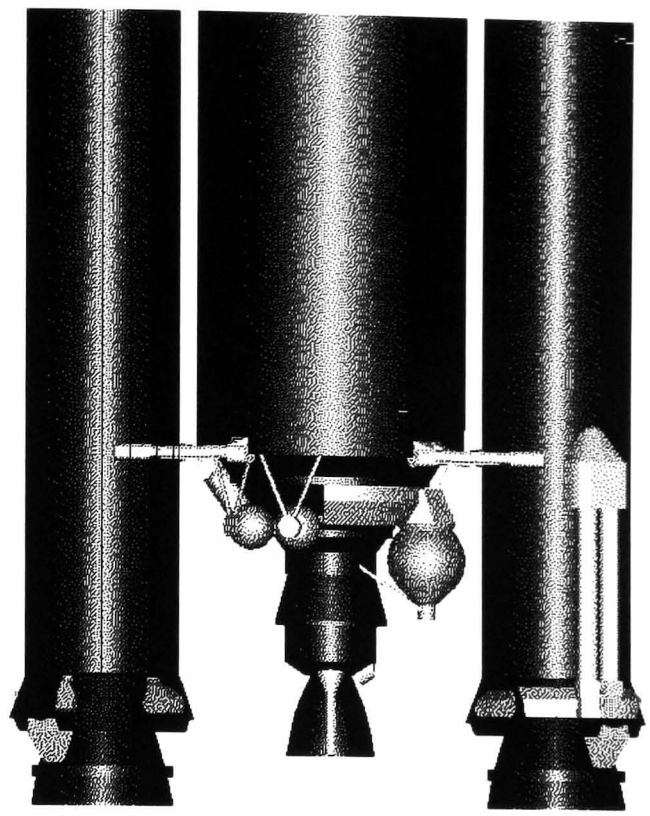


Figure 7.1: **Partial CAD model for the Ariane 5 launcher.**

- ▷ **Cryogenic main stage (EPC)**. It has the largest size among all components and an axisymmetric configuration. Many sharp edges on its surface may cause local flow separation.
- ▷ **Boosters**. Two boosters are placed at both sides of the main stage. Many sharp edges on their surface may cause local flow separation.
- ▷ **DAV: forward attachment hardware**. Two DAV attachments connect the main stage and boosters together. The flow separation may be caused by these components.
- ▷ **DAAR: aft attachment struts**. They connect the main stage and boosters. The flow separation may be caused by these components.
- ▷ **Helium tank**. Its position changes the geometric symmetry of the configuration. The helium tank is important to model flow separation effects.
- ▷ **GATT: high–pressure capacity for the the cryogenic main stage thrust–vector control**. They are attached at the boosters. Since they are not symmetric to a neutral axis (x–axis in this case), they may have an influence on the rotation of the launcher during a mission.
- ▷ **Hydraulic accumulator**. Due to its position and size, it may have influences on flow separation effects.
- ▷ **Nozzle of the main stage**. At the launcher lift–off, the reflection of the exhaust from the nozzle may have an acoustic influence on the launcher structure.

- ▷ **Booster nozzles**. The shock wave and the acoustic effects caused at lift–off from the exhaust of the nozzles are tasks for the CFD simulations.

The surface geometry of meshing object is described by geometry primitives as well as by the free–form surfaces. The choice of surface types is based on the geometric feature of objects.

- ▷ **Surface description using geometry primitives**. The objects such as main stage, nozzles and DAAR attachment are axisymmetric. They are described by the geometry primitives using the rotational sweep surfaces.
- ▷ **Surface description using free–form surface**. All other objects are described using free–form surfaces.

Surface description is in the form of a set of rotational sweep surfaces, and triangular or quadrilateral surfaces, as shown in Fig. 7.2. In addition, the outer boundary is described by the surface of a cylinder whose top and bottom are the inflow and outflow boundaries, respectively.

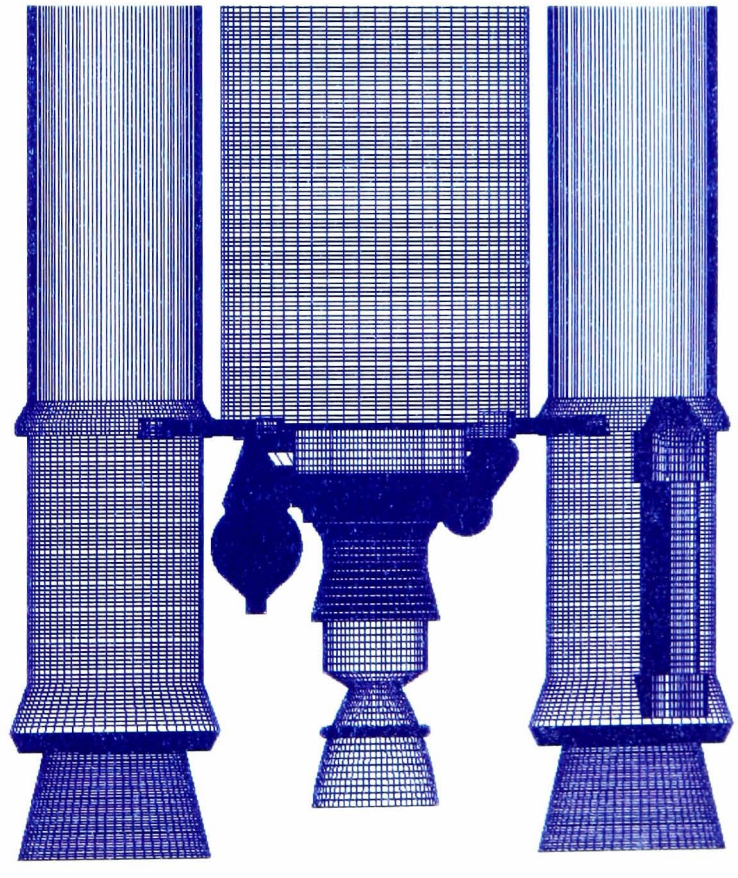


Figure 7.2: **Partial surface description of the Ariane 5 launcher.**

7.2 Determination of Relations among Meshing Objects

The relation between the main task and sub–tasks as well as the relations among the sub–tasks are depicted in the object tree, as shown in Fig. 7.3.

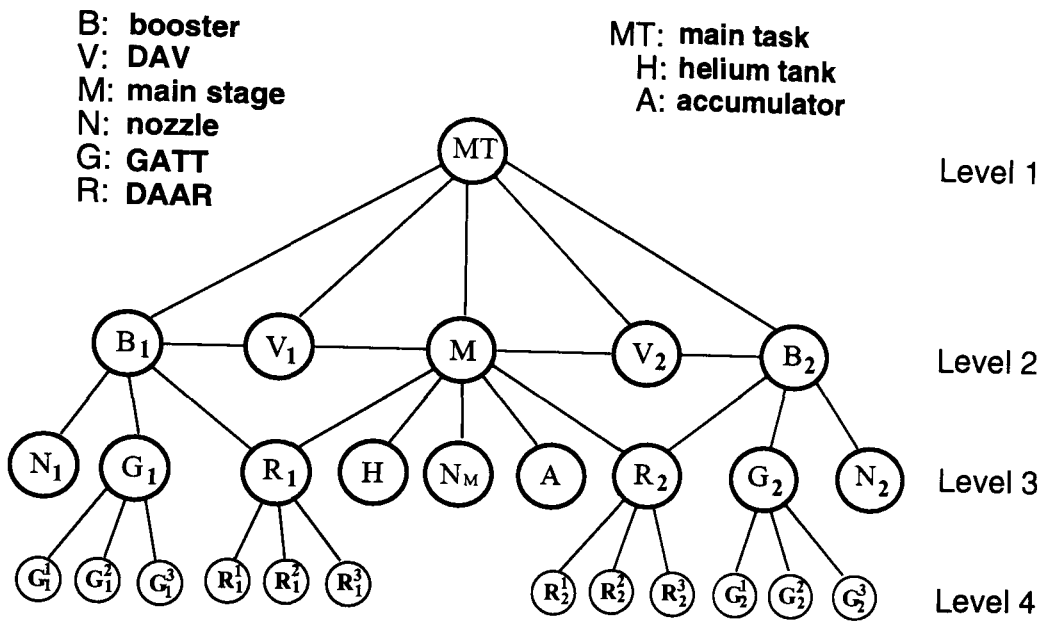


Figure 7.3: **Object tree of the meshing objects of the Ariane 5 launcher.**

There are four levels of the object tree for the decomposition of the complete meshing task. The main task, denoted by `MT`, is at the highest level. On other levels, the topological relations among the sub–tasks are indicated by edges between objects.

7.3 Data Processing

The grid topology is built in three stages. Firstly, a background wireframe model, consisting of a set of frameworks, is generated. It covers the complete solution domain, and its local frameworks provide enough space for all other objects. Secondly, for each object a local topology is generated. Each of them has its own wireframe model. Interfaces of these wireframes to other objects are represented by their external block topologies. In order to simplify connectivity relations between the individual objects and their wireframe environments, the external block topology of an object has a box–like form when ever possible, while its internal block topology may be highly complex. The external topology

encapsulates a highly complex internal block topology.

7.3.1 Background Wireframe Model

The outer boundary of the solution domain is described by a cylinder. Its top and bottom represent inflow and outflow boundaries, respectively. The internal boundary of the solution domain is given by a closed surface resulting from the union of all objects. A background wireframe model divides the solution domain into a set of regions with respect to the configuration of the solution domain and the other meshing objects, as shown in Fig. 7.4. It can be seen that all objects are placed in a box–like framework. If their external topologies are also represented by box–like topological structures, they can be easily inserted into the background wireframe model.

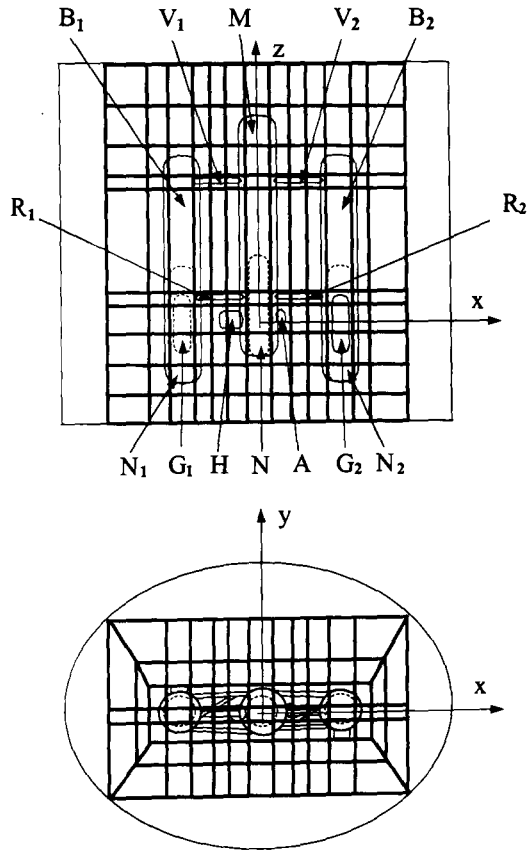


Figure 7.4: **A background wireframe model is generated with respect to the configuration of the solution domain and positions of the meshing objects.**

7.3.2 Local Topology Building at Level 4

Two objects, GATT and DAAR, denoted by `G` and `R` are treated individually at level 4. Their geometry and block topology are saved in a general form. Using

the orientation vectors, they will be duplicated and positioned in the positions required.

7.3.2.1 Local Topology Building for Object G

The GATT consists of three components, $\mathtt{G}^1$, $\mathtt{G}^2$ and $\mathtt{G}^3$, where a superscript denotes the number of the component. Since the GATT is connected to the object booster, its local topology is built with respect not only to its geometric and topological feature, but also to it relation to the booster, as shown in Fig. 7.5.

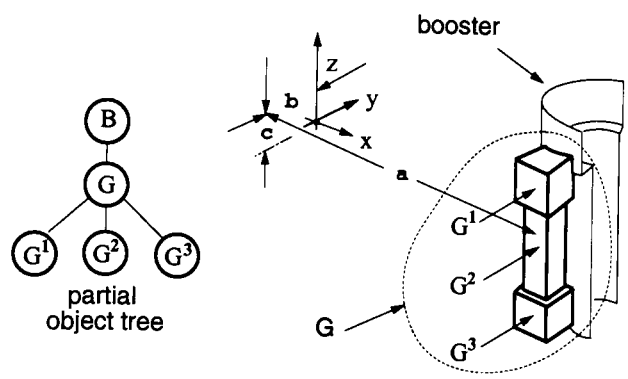


Figure 7.5: **GATT consists of three components. Its geometry and wireframe model are illustrated.**

One of the GATTs is selected for topology building. Its orientation vector is expressed by $V_{\mathtt{G}} = [a, b, c, 0, 0, 0]^T$, where a, b, and c denote the coordinates of the center of geometry. Another GATT will be obtained by rotating this object about the z–axis by 180^o.

In order to simplify the interface of the object G to the object B, three components $\mathtt{G}^1$, $\mathtt{G}^2$ and $\mathtt{G}^3$, are encapsulated by the wireframe model, termed *object hiding*. That means, the external boundary of the wireframe for G has the form of a framework, while its internal block topology could be very complex, as shown in Fig. 7.6.

The encapsulation of objects using the object hiding enables easy assembly. The background wireframe model is built by connecting a set of frameworks together, which are generated in the form of box–like frameworks. It is easy to select a location that is covered by a framework. An object encapsulated will be assembled in this location.

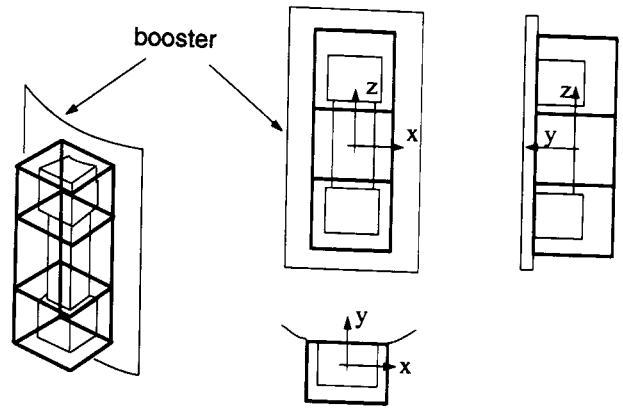


Figure 7.6: **The object GATT is assembled into the background wireframe.**

Fig. 7.7 shows the target region covered by a local background wireframe and the interface of the object G. The external boundaries of the object and the region targeted for this object have the same topology. Assembly of the object in the target region is accomplished by generating a one–to–one vertex connection.

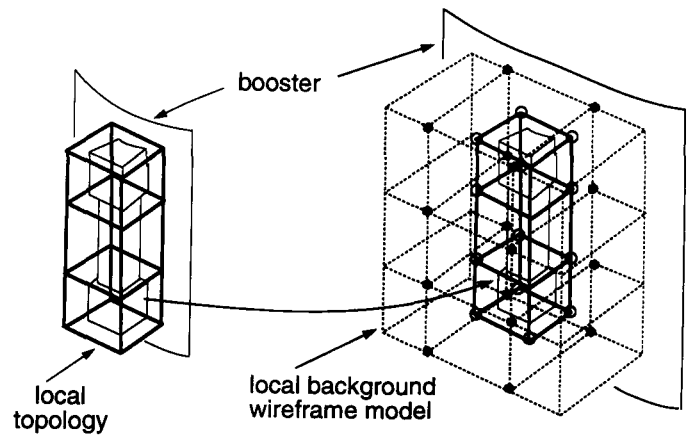


Figure 7.7: **The object GATT is assembled into the background wireframe.**

7.3.2.2 Local Topology Building for Object R

The object DAAR consists of three components that attach the object main stage M and the objects booster B. In the object tree, they are represented by three ending nodes, R^1, R^2 and R^3, where the superscript denotes the number of DAAR's component, as shown in Fig. 7.3. In wireframe building, its neighboring objects are sketched in a partial object tree in order to generate interfaces among objects accurately, as shown in Fig. 7.8.

One of the DAARs, R_1, is selected to generate the local topology in a standard form. The other object R_2 can be obtained by duplicating R_1 and rotating it

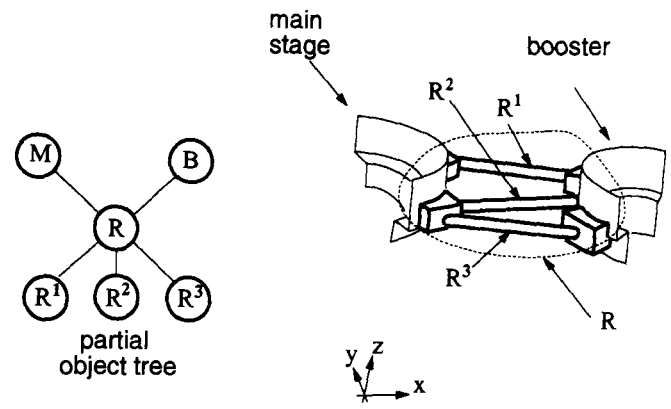


Figure 7.8: **Object DAAR and its relation to other objects.**

about the z–axis by 180^o.

The difficulty is to generate an adequate block topology for the Y–joint, built by the intersection of the components $\mathtt{R}^1$ and $\mathtt{R}^2$. The external wireframe topology of the complete R should have a simple form, such that its interface to the targeted region in the background wireframe model can be represented by a framework.

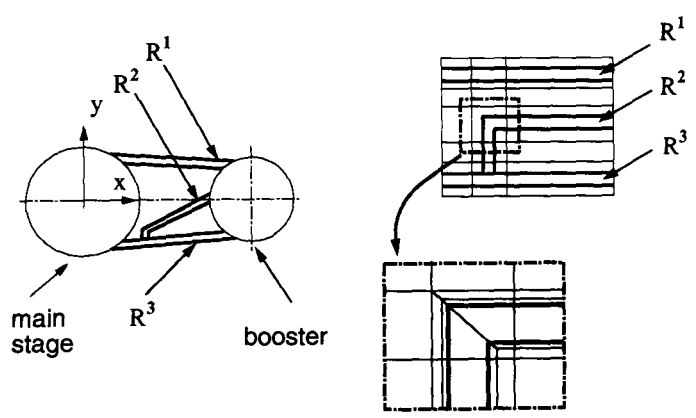


Figure 7.9: **The simplified geometry of the DAAR object and its topology building.**

To encapsulate the complete R, a framework that covers all DAAR components is first generated. Object $\mathtt{R}^2$ has the form of a Γ–type tube. Then, the blocks which cover the region of $\mathtt{R}^2$ are selected. The internal blocks are generated within the region selected, which represent the Γ–type component $\mathtt{R}^2$, as shown in Fig. 7.9.

7.3.3 Object Building at Level 3

The objects to be generated at level 3 are the nozzle of the main stage N_M, the nozzles of booster N_1 and N_2, the attachment struts R_1 and R_2, the helium tank H and the accumulator A. Similar to the objects GATT and DAAR, the local topologies for these objects are generated with respect to their geometric features and their relations to other objects as well as to the environments of the background wireframe model.

7.3.3.1 Local Topology Building for Object N_M

The nozzle is considered as an internal component of the main stage. The geometries of these two objects are originally designed as separate CAD models. If the nozzle geometry is varied, the CAD model will be easily modified by updating the nozzle geometry. Regarding modification of the nozzle geometry, the grid topology is generated by separating the nozzle from the main stage. The geometry of the object N_M is described by an axisymmetric surface. The object and its relations to other objects are shown in Fig. 7.10.

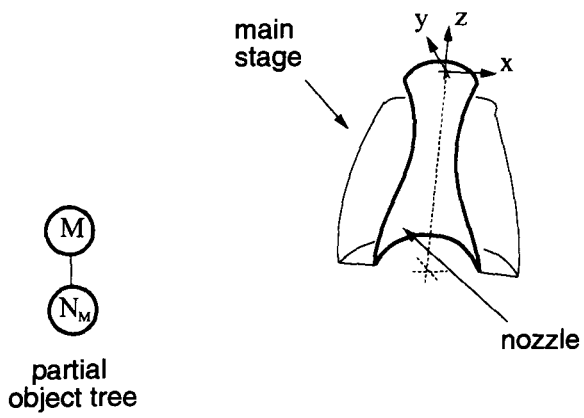


Figure 7.10: **The geometry of the object N_M and its relation to the object M.**

The objects N_M and M are connected at the nozzle exit. Since the object N_M is an internal component of the main stage, they will be assembled by merging their common interface.

In order to generate the common interface of two objects, both objects are selected. The local topology for the object N_M is generated with respect to the main stage. The topology of the common interface of the object N_M is employed as a part of the local topology of the main stage, as shown in Fig. 7.11.

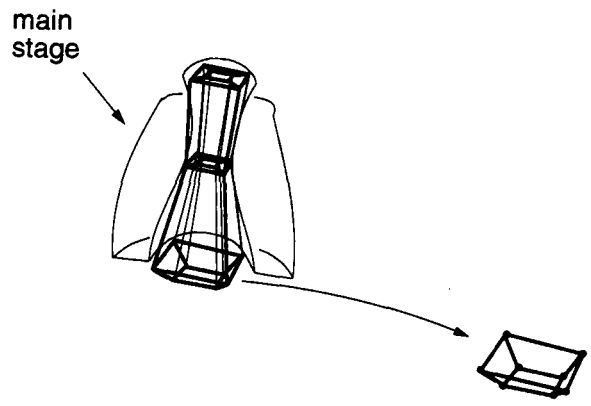


Figure 7.11: **The wireframe topology of the object $\mathtt{N_M}$ and its interface to the object M.**

7.3.3.2 Local Topology Building for Object N

The geometry of object N is axisymmetric about the z–axis. Its surface geometry is described by a rotational sweep surface. The position of the object $\mathtt{N_1}$ is expressed by the orientation vector $V_{\mathtt{N_1}} = [a, 0, 0, 0, 0, 0]^T$, where a gives the distance between the z–axis and the rotational axis of the nozzle. Obviously, the position of the object $\mathtt{N_2}$ is expressed by $V_{\mathtt{N_2}} = [-a, 0, 0, 0, 0, 0]^T$, as shown in Fig. 7.12.

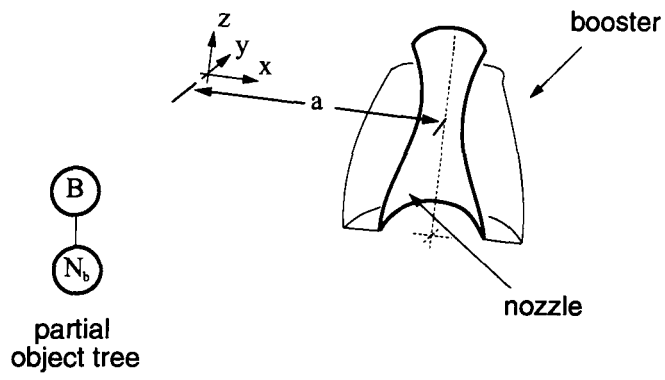


Figure 7.12: **The geometry of the object $\mathtt{N_b}$ and its relation to the object booster.**

The nozzle is considered as an internal component of the booster. In general, one can generate a booster together with its nozzle. With respect to modification of the nozzle geometry, the nozzle is treated separately.

The vertices at the bottom of the wireframe model of the nozzle are connected to the booster. The interface of the local topology to the booster is therefore

generated in a simple form, as shown in Fig. 7.13.

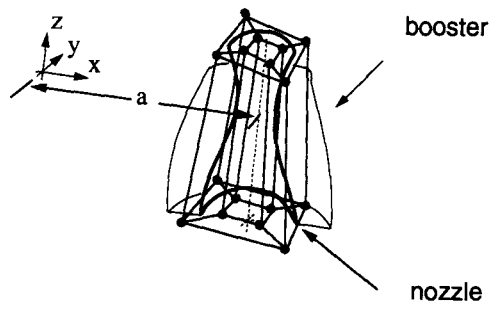


Figure 7.13: **The wireframe topology of the object N_b and its interface to the booster.**

7.3.3.3 Local Topology Building for Object H

The object helium tank, denoted by H, is attached to the main stage. The relation of the object to the main stage M is shown in Fig. 7.3. The geometry of the helium tank is described by a rotational sweep surface. Its geometric features as well as its relation to the object M are depicted in Fig. 7.14.

Since the object H will not be duplicated, it is generated in its original size and position in the physical space.

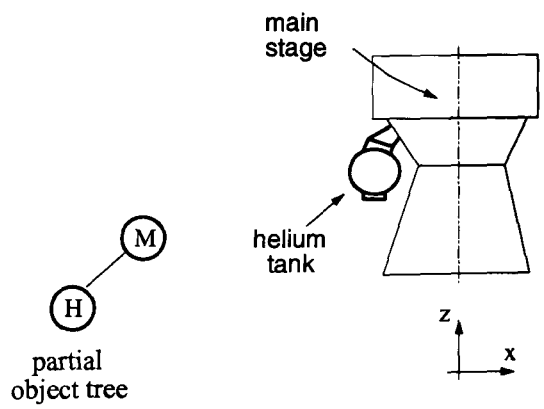


Figure 7.14: **The geometry of the object helium tank H and its relation to the main stage.**

The object H has a block topology similar to a Y–joint. In order to encapsulate the helium tank within a framework, the local topology is generated according to the following steps. Firstly, a framework of the region that covers the helium

tank is selected. Secondly, an internal block topology is built for the helium tank. Finally, the connection between the internal topology and the selected framework is generated.

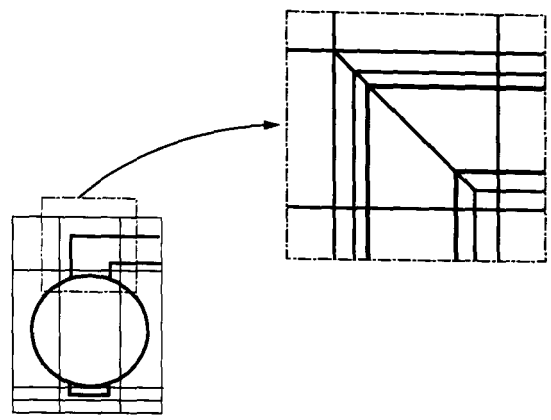


Figure 7.15: **Topology building of object H.**

7.3.3.4 Local Topology Building for Object A

The object accumulator, denoted by A, is attached to the main stage. The relation of the object to the main stage is shown in Fig. 7.3. The geometry of the object A is described by a free–form surface. Its shape and position in physical space is depicted in Fig. 7.16.

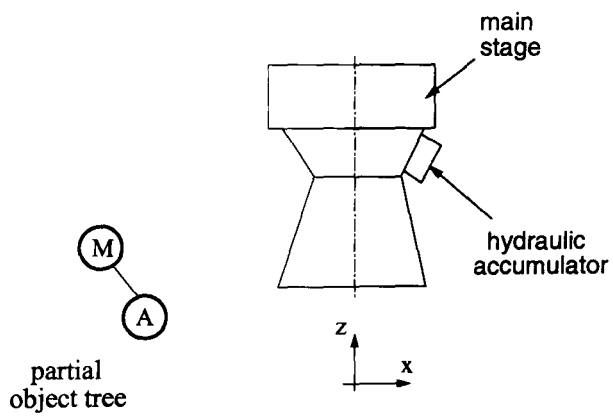


Figure 7.16: **The shape of the object A and its position in physical space.**

Object A is generated in original size and position in physical space. The local topology of this object is built as shown in Fig. 7.17. Due to its simple topology, a detailed description of the wireframe model building is omitted.

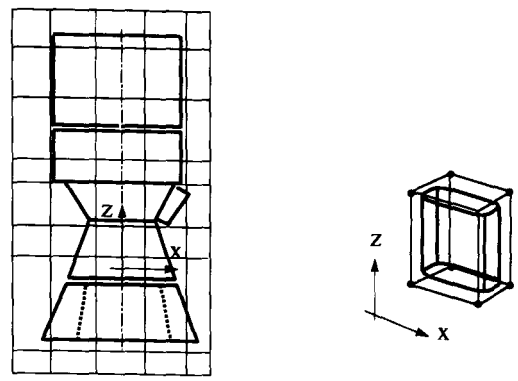


Figure 7.17: **The local topology of object A.**

7.3.4 Object Building at Level 2

The main stage, two boosters and their attachments DAV are defined as the meshing objects at level 2. The main stage and boosters have the largest sizes among all objects. The two DAVs are attachments between the main stage and booster.

7.3.4.1 Local Topology Building for Object M

The object main stage, denoted by M, is axisymmetric. The major requirement for topology building is to accurately model the sharp edges, since these geometric features are important for simulating flow separation. The complete geometry is described by a set of axisymmetric surfaces. The intersections between these surfaces give the sharp edges, as shown in Fig. 7.18.

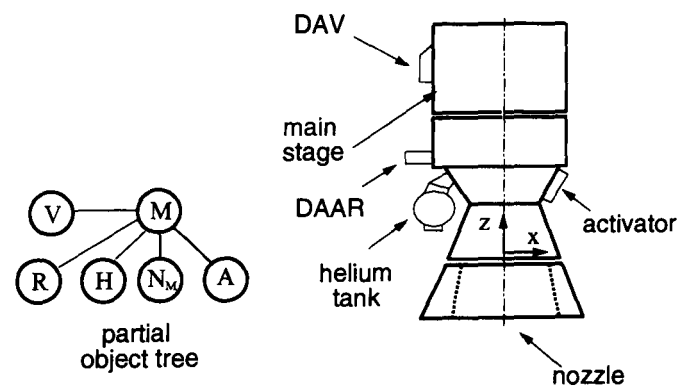


Figure 7.18: **The object main stage M and its relations to other objects.**

The object, connected to the main stage on the same level, is DAV V. Other objects connected to the main stage at lower levels are DAAR R, helium tank H, nozzle N_M, and accumulator A, as shown in Fig. 7.3. The topology for object M is

generated regarding the geometric shape of the object M, as well as its relations to other objects.

The wireframe topology of object M must provide sufficient places (frameworks) for other objects to be connected. In addition, it should have a simple form to the environment of the background wireframe model. To meet these requirements, all local topologies of the relevant objects are placed in their respective positions. A coarse framework that encloses all objects is generated. This framework is refined, until all local topologies can be enclosed by their corresponding frameworks, as shown in Fig. 7.19.

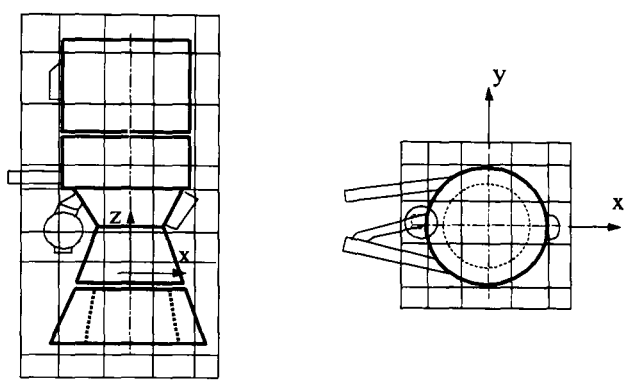


Figure 7.19: **The local topology of the object M.**

Through the refinement of the coarse framework, the locations where other objects are placed, are decomposed by the local frameworks that have the same topologies as the external topologies of the objects to be integrated. In this manner, all objects are assembled.

7.3.4.2 Local Topology Building for Object B

The major requirement for topology building of the object booster B is to model the sharp edges. The complete geometry is described by a set of rotational sweep and axisymmetric surfaces. Intersections between surfaces give the sharp edges. The orientation vector of the object B is expressed by $V_{B_1} = [a, 0, 0, 0, 0, 0]^T$, where the a denotes the distance between the z–axis and the main axis of the booster, as shown in Fig. 7.20. Another object B_2 is obtained by rotating the object B_1 around the z–axis by 180^o.

In topology building, the attachment components, such as GATT G, DAAR R,

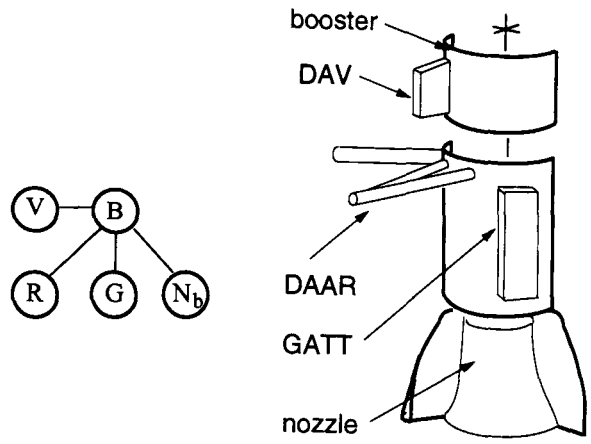


Figure 7.20: **The object B and its relations to other objects.**

nozzle $\mathtt{N_b}$ and DAV V, are placed in their positions. A framework for the object B is generated with respect to these objects, as shown in Fig. 7.21.

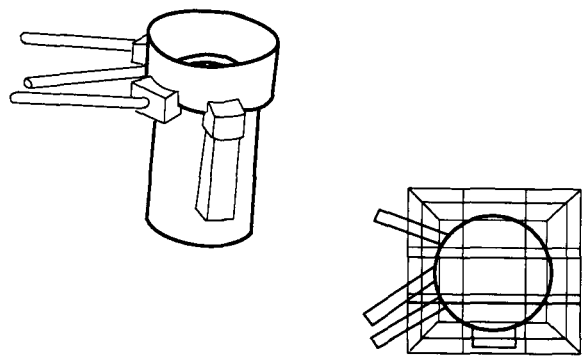


Figure 7.21: **The local topology of the object booster B.**

7.3.4.3 Local Topology Building for Object V

The last object, DAV, on this level, denoted by V, attaches the main stage and booster. Its geometry is described by a set of parametric surfaces in the form of quadrilateral elements. The geometric shape of object V and its relation to other objects are depicted in Fig. 7.22.

The orientation vector of the object V is expressed by $V_{\mathtt{V_1}} = [a, 0, c, 0, 0, 0]^T$, where the a and c denote the distances between the x– and z–coordinates of the geometry center of the object V and the x– and z–axes. Another object $\mathtt{B_2}$ is obtained by rotating the object $\mathtt{B_1}$ about the z–axis by 180^o. The interface of the object V to other objects is generated as in Fig. 7.23.

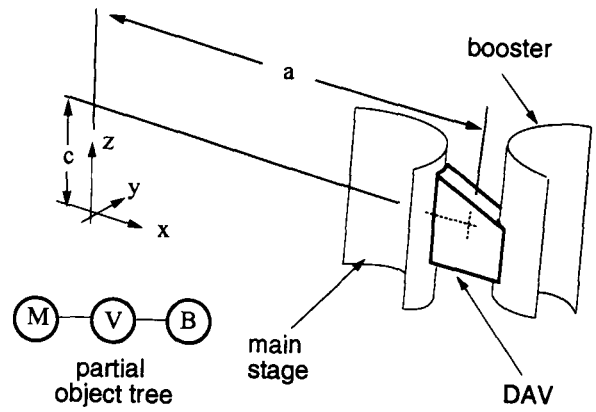


Figure 7.22: **The object DAV V and its relations to other objects.**

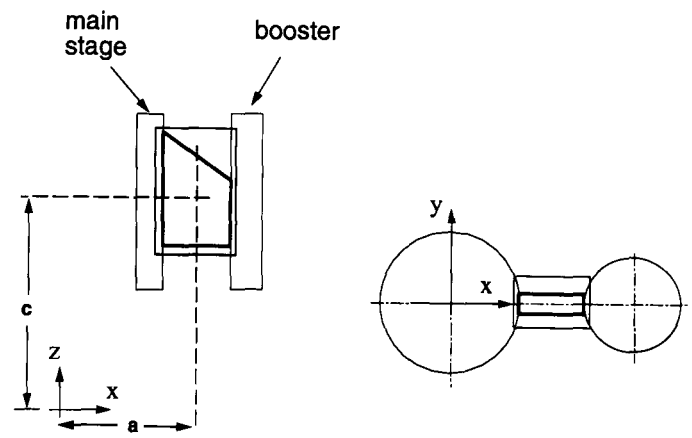
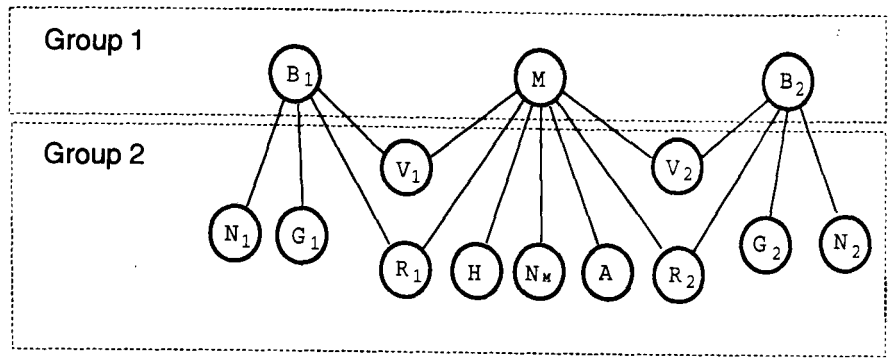


Figure 7.23: **The local topology of the object DAV V.**

7.3.5 Generation of the Final Grid

To complete the meshing task, all objects are assembled into the background wireframe model. The method employed for the assembly process is that the objects are inserted into the background wireframe model in sequence.

In general, an assembly process begins with inserting objects into the background wireframe model. It is desirable to obtain a grid after every assembly. However, attention should be paid to the sequence of object assembly. In the present example, the objects are divided into two groups according to their surface description. The main stage and boosters are described by closed surfaces. Inserting them into the background wireframe model, together with an outer boundary, the *first type of meshing domain* can be found (section 6.3.1). The objects, whose surface geometries are described by open surfaces, belong to the second group. Since an open surface does not define a subdomain, the objects of the second group have to be assembled after the assembly of the first group, as shown in Fig. 7.24.



Group 1: Objects are described by closed surfaces.

Group 2: Objects are described by open surfaces.

Figure 7.24: **The objects are assigned to two groups in order to specify a sequence of object assembly.**

7.3.5.1 The Sequence of Assembly

The assembly sequence can be expressed by an assembly graph as well as by an assembly function. The assembly function is employed to represent the sequence of assembly, since it is expressed in a simple form.

Object assembly of the group 1. The objects of the group 1 are assembled into the background wireframe model.

- ▷ 1. Assembly A_1. The main stage M is assembled to the background wireframe model F, denoted by the assembly function

$$A_1 = \{s_1(\mathtt{M},\ \mathtt{F})\}$$

- ▷ 2. Assembly A_2. The boosters $\mathtt{B}_1$ and $\mathtt{B}_2$ are assembled to the background wireframe model F, denoted by the assembly function

$$A_2 = \{s_1(\mathtt{B}_1,\ \mathtt{F}),\ s_2(\mathtt{B}_2,\ \mathtt{F})\}$$

Object assembly of the group 2. Three objects of the group 2 are assembled into the background wireframe model.

- ▷ 3. Assembly A_3. The DAVs $\mathtt{V}_1$ and $\mathtt{V}_2$ are assembled into the background wireframe model. They build the attachment between the main stage M and boosters $\mathtt{B}_1$ and $\mathtt{B}_2$, denoted by the assembly function

$$A_3 = \{s_1(\mathtt{V}_1,\ \mathtt{B}_1),\ s_2(\mathtt{V}_1,\ \mathtt{M}),\ s_3(\mathtt{V}_2,\ \mathtt{B}_2),\ s_4(\mathtt{V}_2,\ \mathtt{M})\}$$

- ▷ 4. Assembly A_4. The nozzles $\mathtt{N}_\mathtt{M}$, $\mathtt{N}_1$ and $\mathtt{N}_2$ are assembled into the background wireframe model F. They are connected to the main stage M. and boosters $\mathtt{B}_1$, $\mathtt{B}_2$. It is expressed by

$$A_4 = \{s_1(\mathtt{N}_\mathtt{M},\ \mathtt{M}),\ s_2(\mathtt{N}_1,\ \mathtt{B}_1),\ s_3(\mathtt{N}_2,\ \mathtt{B}_2)\}$$

▷ 5. Assembly A_5. The DAARs $\mathtt{R}_1$ and $\mathtt{R}_2$ are assembled into the background wireframe model. They are connected to the main stage M and boosters $\mathtt{B}_1$. $\mathtt{B}_2$. It is expressed by

$$A_5 = \{s_1(\mathtt{R}_1,\ \mathtt{M}),\ s_2(\mathtt{R}_1,\ \mathtt{B}_1),\ s_3(\mathtt{R}_2,\ \mathtt{M}),\ s_4(\mathtt{R}_2,\ \mathtt{B}_2)\}$$

▷ 6. Assembly A_6. The GATTs $\mathtt{G}_1$ and $\mathtt{G}_2$ are assembled into the background wireframe model. They are connected to the boosters $\mathtt{B}_1$ and $\mathtt{B}_2$. It is expressed by

$$A_6 = \{s_1(\mathtt{G}_1,\ \mathtt{B}_1),\ s_2(\mathtt{G}_2,\ \mathtt{B}_2)\}$$

▷ 7. Assembly A_7. The helium tank H is assembled into the background wireframe model, and is connected to the main stage M. It is expressed by

$$A_7 = \{s_1(\mathtt{H},\ \mathtt{M})\}$$

▷ 8. Assembly A_8. The accumulator A is assembled into the background wireframe model, and is connected to the main stage M. It is expressed by

$$A_8 = \{s_1(\mathtt{A},\ \mathtt{M})\}$$

The assembly functions given above serve as a plan for completing the main meshing task. The assembly process, a so–called *object–oriented sequence of assembly*, provides the designer with a grid containing the objects already assembled. The method enables a step–by–step control of grid topology as well as grid quality. In the following the object assembly is explained.

7.3.5.2 Assembly of Objects

The main stage M is inserted into the background wireframe model F, denoted by the assembly function $A_1 = \{s_1(\mathtt{M},\ \mathtt{F})\}$. First, the the background wireframe model is placed in its position. In the region, where the main stage is inserted, the framework is selected. Deleting all internal vertices of the framework, the hull box of the framework has the same topology as the external topology of the object main stage. Generating the one–to–one connection between them, the object main stage is assembled into the background wireframe model, as shown in Fig. 7.25.

The above example shows the principle of an assembly. Since all objects are generated in the same way, a detailed description of the complete assembly process is omitted.

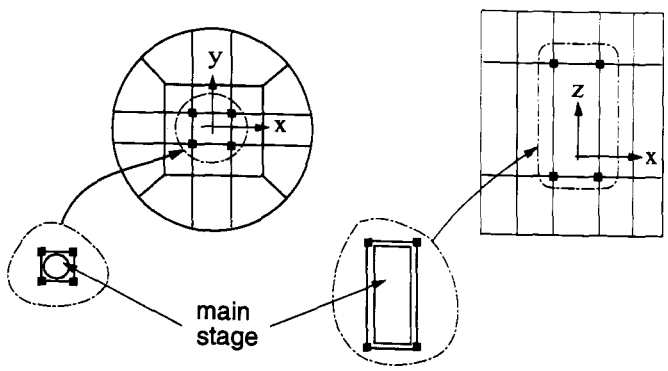


Figure 7.25: **The assembly of the main stage into the background wireframe model.**

7.4 Evaluation of Results

Results of grid generation are evaluated by comparing the grids from each assembly with the requirements for geometric accuracy and grid quality. If these requirements are satisfied, the assembly process will be continued. In the following, the results of the example are given in the form of a series of grids.

7.4.1 DAAR, GATT, and Hydraulic Accumulator

Attachment struts DAAR. The grid quality of the attachment struts DAAR is evaluated by the following two major points. First, two struts build a Y–joint, where the cutter path must be accurately modeled. This region is considered as a sensible region, i.e., a reduced convergent behavior of grid lines may occur in this region, if block topology or surface description are of insufficient quality. Second, in region of the Y–joint, a high density of grid lines is required for resolving viscous flow.

The strategy to prevent these problems is to optimize the local wireframe model. That means, before it is assembled into the global grid topology, vertices of the wireframe model are put in optimal positions, where a good initial solution is provided. Such a test of a local grid topology enables the designer to control grid quality at different levels. The result shows that both requirements are satisfied, as shown in Fig. 7.26.

GATT. The local topology of the object GATT encapsulates it three components. The test result of the local wireframe topology provides a high grid quality. The grid lines of surface grid are almost uniformly and orthogonally

distributed. In addition, the cutter path between the GATT and booster is exactly modeled, as shown in Fig. 7.26.

Hydraulic accumulator. The grid quality of the object hydraulic accumulator can be evaluated by the smoothness and uniform distribution of grid lines on its surface. Since the object is attached on the main stage, where sharp edges exist, the trajectory of the intersection curve is the major difficulty of grid topology building. Around the object, a boundary layer topology is generated, which ensures that the cutter path is clearly modeled. The result shows the satisfactory grid quality, as shown in Fig. 7.26.

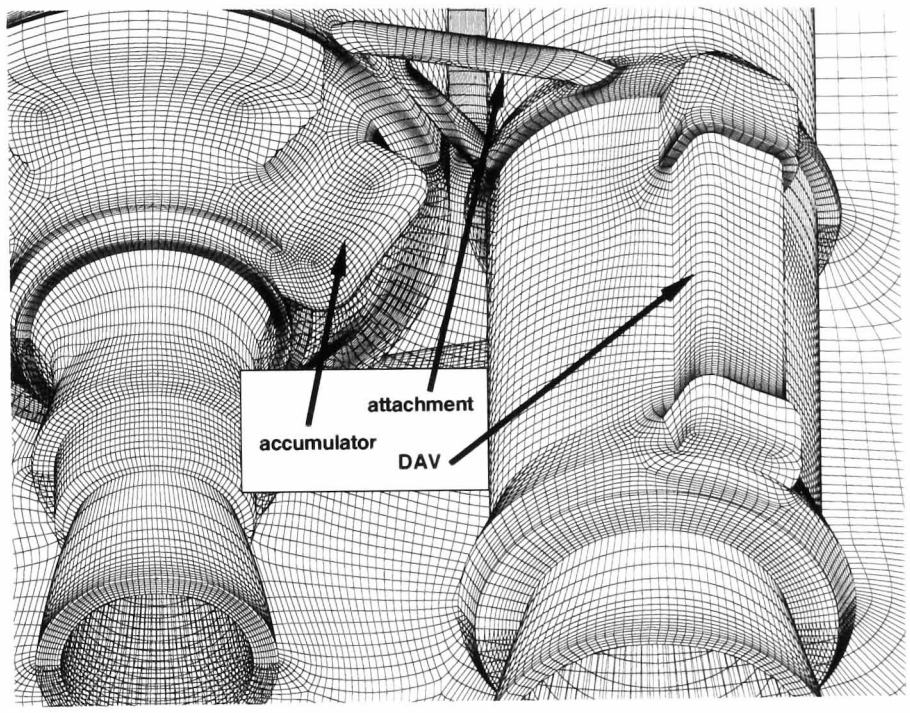


Figure 7.26: **Local grid of the objects attachment struts, GATT, and hydraulic accumulator.**

7.4.2 Helium Tank

The helium tank is an important component for simulating flow separation, and eventual the flow separation. Therefore, its size and geometry should be accurately modeled. The main difficulty of meshing this object is to build an appropriate block topology for the complete object and its interface to the main stage.

The result is shown in Fig. 7.27. The cutter path between the object and the main stage is modeled very clearly. Grid lines on the surface of the helium tank are of high smoothness. This satisfies the requirement for an accurate flow

simulation in region of interest.

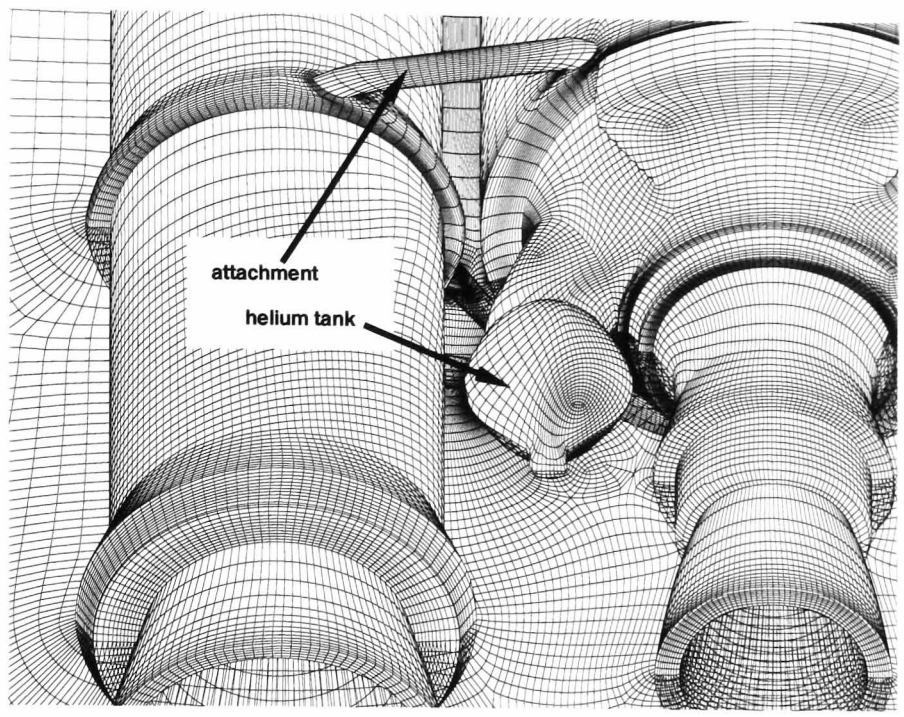


Figure 7.27: **Local grid of the object helium tank.**

7.4.3 Nozzles

It is required to build one–dimensional boundary layer topologies for three nozzles, such that an eventual enrichment of grid lines for Navier–Stokes computation can be generated using cluster tools. In addition, the sharp edges at nozzle exist are important for simulating flow separation.

The result is shown in Fig. 7.28. Grid lines are almost uniformly and orthogonally distributed with the nozzles. The boundary layer topology enables a one–dimensional clustering of grid lines wrapped around nozzles. The geometric features are accurately meshed.

7.4.4 GATT Attachment

In order to simulate flow separation, caused by GATT attachments, which may have some influence on shock–shock interaction with other flow separation, the shape features should be modeled accurately. In topology design, the boundary layer topology is selected to wrap these attachment components. This provides the possibility to change grid density around the GATT surfaces.

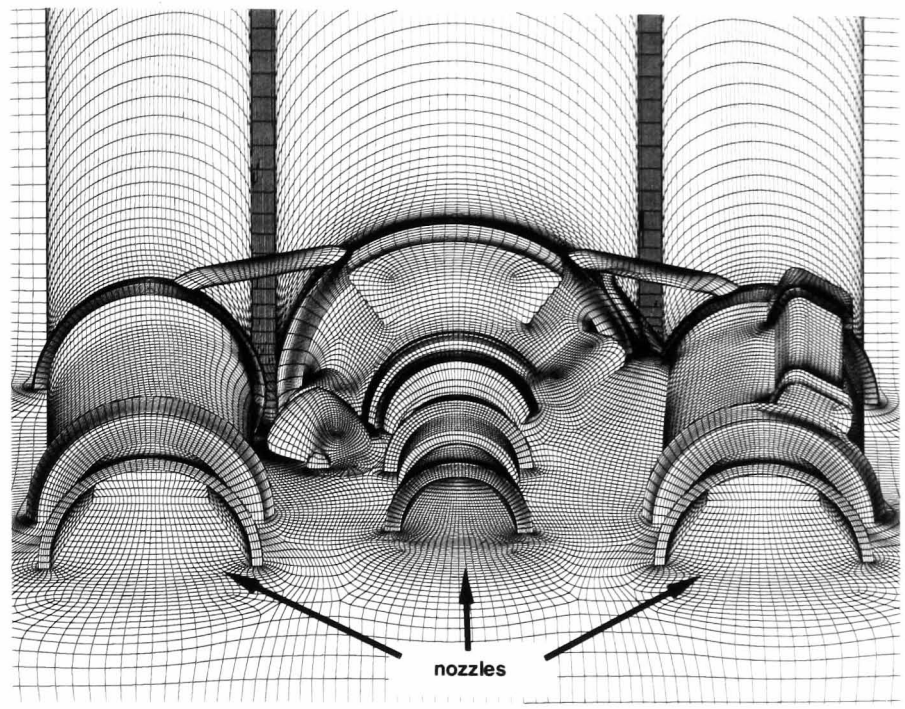


Figure 7.28: **Local grid of the object nozzle.**

The result is shown in Fig. 7.29. The geometric features of GATT attachments are sufficiently meshed. However, the distance of the first lines to GATT surface is much larger than it is required. This can be improved to use control functions, which force grid lines within a desirable range.

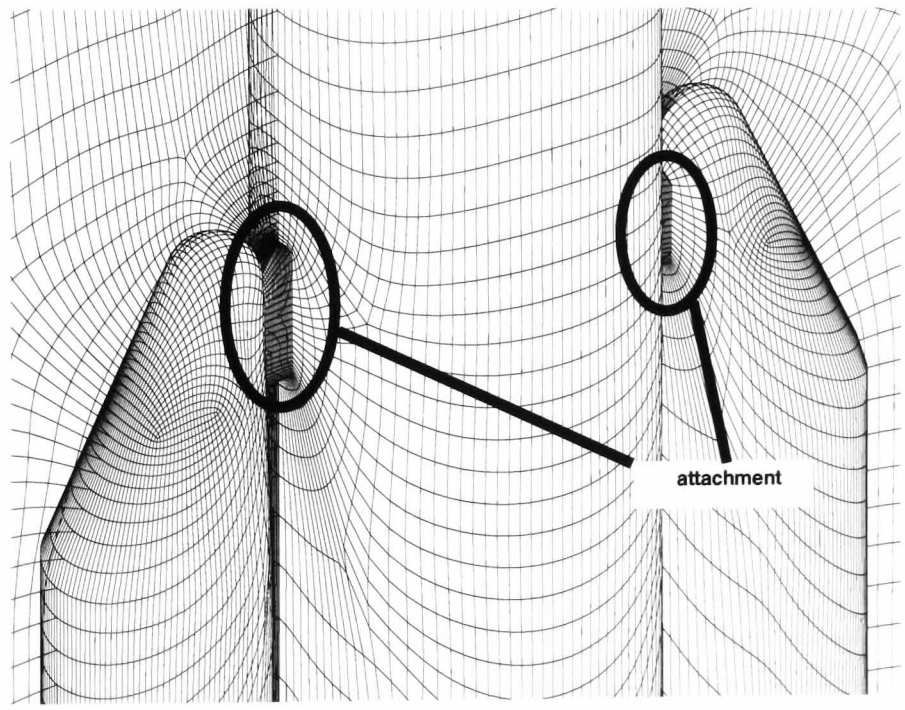


Figure 7.29: **Local grid of the object GATT Attachment.**

7.4.5 Complete Grid

The final grid is shown in Figs. 7.30 and 7.31. In order to examine the precision of the meshing geometry, the geometry surface of the Ariane 5 grid is visualized

in shaded view. The cut plane on the x–plane is given in the form of a mesh. The grid consists of about 5000 blocks with about 700,000 grid points.

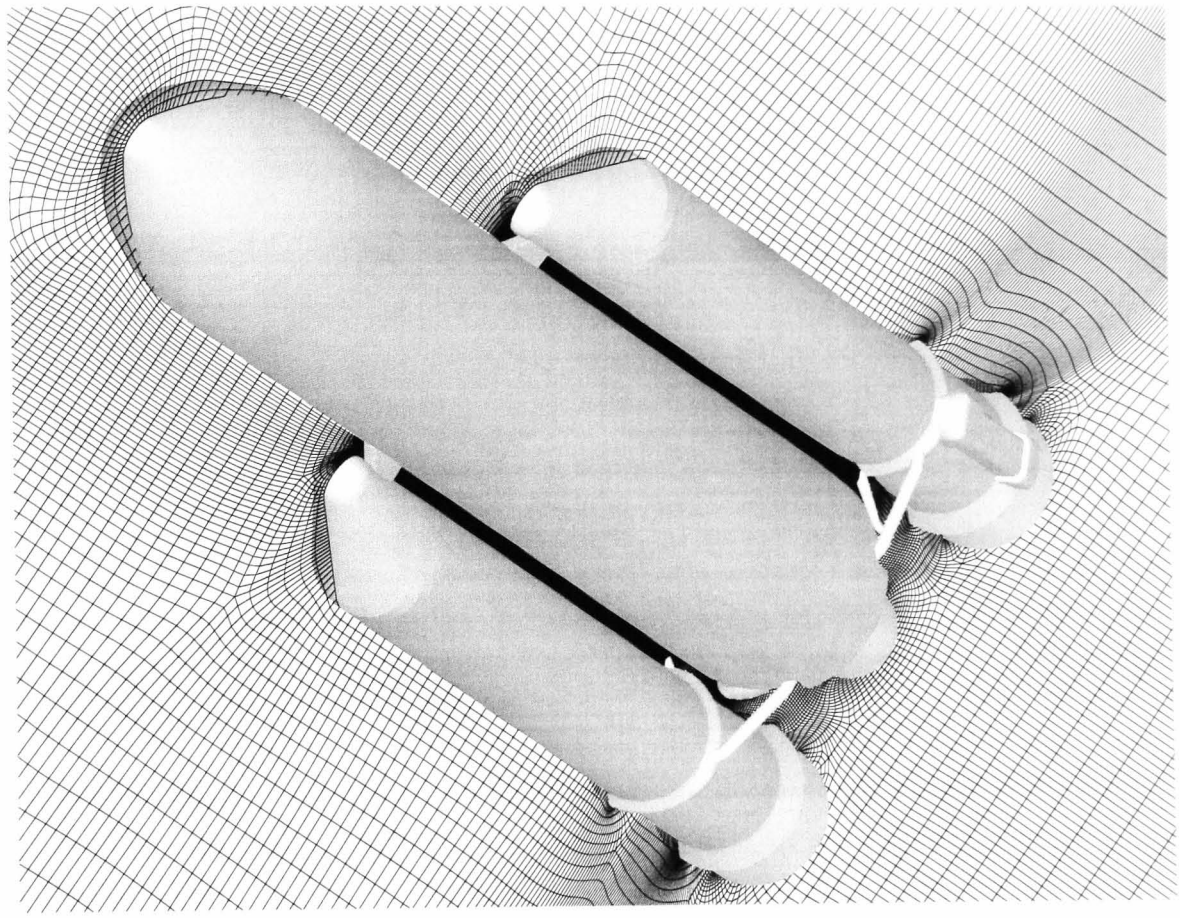


Figure 7.30: **The final grid of the Ariane 5 launcher (1).**

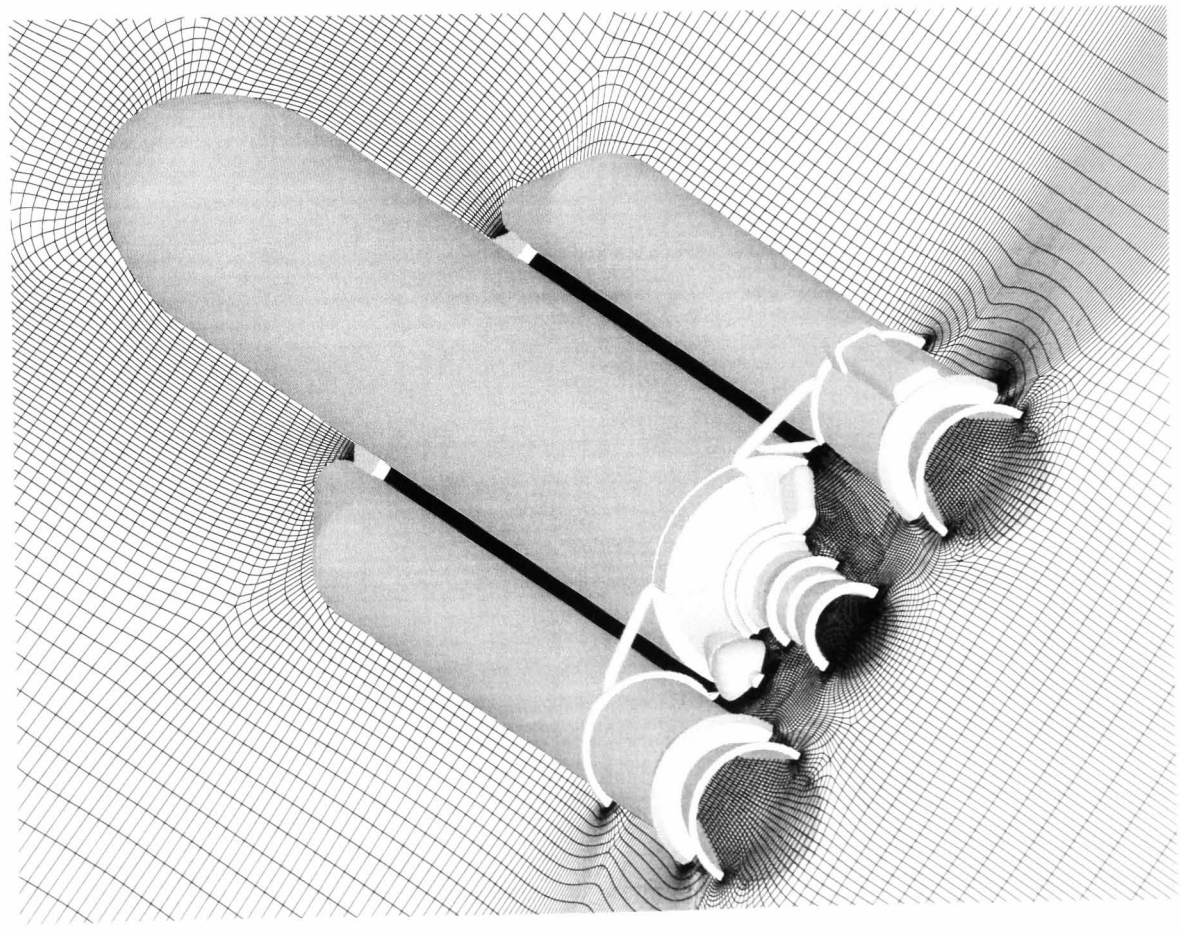


Figure 7.31: **The final grid of the Ariane 5 launcher (2).**

7.5 Chapter Summary

The main contribution of the present chapter is to describe the practical application of the object–oriented grid design method in complex cases. The *Ariane 5 launcher* is chosen as the example. The complete grid generation process is explained in the following parts:

- ▷ 1. A complete meshing task is divided into a set of sub–tasks, termed *meshing objects.*
- ▷ 2. Relations among the main meshing tasks and meshing objects are represented in an object–tree.
- ▷ 3. A background wireframe model is generated for the complete solution domain, in which there are enough spaces for each meshing object.
- ▷ 4. Wireframe topologies for meshing objects are locally generated with respect to their neighboring relations to other objects and environments in the background wireframe model.
- ▷ 5. Meshing objects are inserted to the background wireframe model.

Chapter 8

Topology–Based Grid Adaptation

There is no formula to determine the optimal grid density using, for instance, a quantity like *cell number per unit volume.* A requirement for the optimal cell number for a grid is likely to be an estimate based on the experience of the designer.

A homogeneous distribution is advantageous to reduce numerical errors, it is, however, not economical in many cases. In general, a high grid point distribution is generated in regions of interest, where either large gradients of flow variables are observed, or a high grid point distribution to resolving flow discontinuities is required.

A heterogeneous distribution is based on the efficient flow simulation with a reduced number of cells. It is sufficient to have a lower grid density in regions of less interest with respect to the flow physics. Usually, a heterogeneous distribution is obtained by using solution–oriented grid adaptation.

A solution–independent approach to grid clustering is the change of local block structures to retain a high number of grid points with a marked region without employing a flow solution. This adaptation is a topology–based adaptation and thus is solution–independent. In the present thesis this type of grid adaptation is termed *passive grid adaptation.*

Another important reason for introducing topology–based grid adaptation is the following. Grid generation for a highly complex configuration is often time–consuming. It is hoped that a modification of local grid structure, grid density, or grid line distribution does not substantially affect the existing grid structure and grid quality. Based on a rough estimate of a high grid density

in regions of interest, grid topology can be adapted in the manner that a fine block structure will improve grid density in these regions. The idea inspires the strategy development for a solution–independent grid adaptation, i.e., block topology–based grid adaptation.

8.1 Strategy of Passive Grid Adaptation

Passive grid adaptation as developed in this thesis consists of three main parts: *block encapsulation*, *boundary layer enrichment* and *smart topology*. All of these adaptations are accomplished by building special block structures. The terminology *passive adaptation* is used, since a local grid density is increased without reasonably regarding accurate flow physics. Passive grid adaptation will be employed in the following cases.

- ▷ **Block encapsulation**. In order to separate a fine distribution of grid lines from a coarse one, the region of this fine distribution is encapsulated by a set of blocks. Their outer boundary, termed *encapsulated surface* restricts the degrees of freedom of grid points. The grid line distributions at both sides of the encapsulated surface are separated.
- ▷ **Boundary layer enrichment**. In case of generating a N–S grid, an extremely fine distribution of grid lines to resolving the boundary layer is required. A block structure, consisting of a set of blocks, builds a wrap around the fixed body, termed *boundary layer topology*. Grid line enrichment can be carried out within this block layer.
- ▷ **Smart topology** Refinement of grid points is achieved by inserting a set of blocks into an existing grid. Through this block enrichment, termed *smart topology*, a high local grid density is generated [30].

The core of the above three adaptation methods is summarized as *scaling line distribution* (block encapsulation), *separate generation of grid clustering* (boundary layer enrichment), and *block enrichment* (smart topology). They are aimed at providing an optimal grid point distribution associated with requirements for flow simulation.

8.2 Concept 1: Block Encapsulation

A homogeneous grid line distribution may be generally regarded as an optimal domain decomposition. It is, however, neither rational nor economic for flow

simulation in many cases. Consider the two–dimensional example of Fig. 1.8 in section 1.3.2 (see page 13). Suppose that the diameter of antenna is D, and the length of car is L, as shown in Fig. 8.1, and the ratio of D/L is some $1/2 \times 10^{-3}$. There is a need for resolving the micro–aerodynamic flow phenomenon around the antenna with 4 points in each direction. If the number of cells needed for the area of the antenna tip amounts to $N = 4 \times 4$, then the area of the solution domain A_{Ω_2} is approximated by

$$\begin{aligned} A_{\Omega_2} &\approx (3L) \times (2.5L) - L \times (0.5L)^2 \\ &= 7.25L^2 \end{aligned}$$

Using the relation

$$\begin{aligned} \frac{N_{\Omega_2}}{N} &= \frac{7.25L^2}{D^2} = 7.25 \times (\frac{2000}{1})^2 \\ &= 2.9 \times 10^7 \end{aligned}$$

one can estimate the total number of cells in the solution domain to be $N_{\Omega_2} = N \times 2.9 \times 10^7 = 4.64 \times 10^8$.

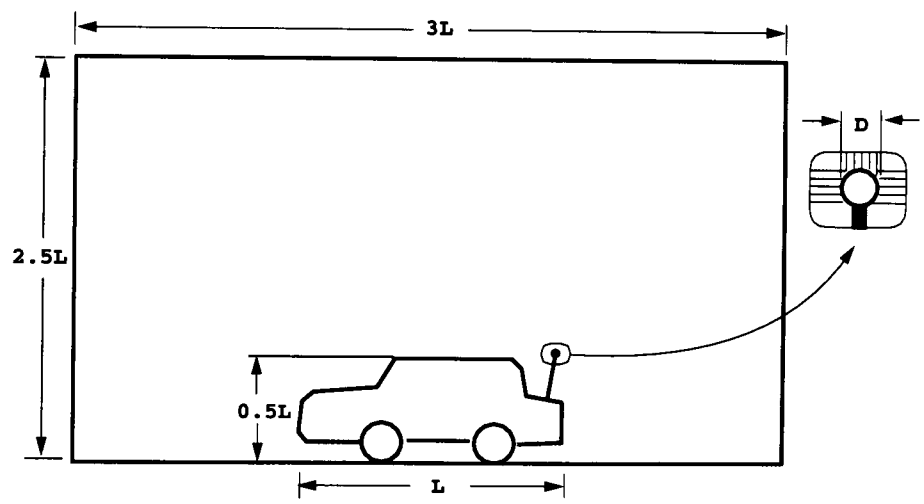


Figure 8.1: **The sizes of the meshing objects can be significantly different. A globally homogeneous grid density may cause a large number of grid points.**

Similar to solution–adaptive grid generation, it is desirable to obtain a globally heterogeneous grid line distribution. In order to retain high grid density only in regions of physical interest, block encapsulation separates the region of the antenna from its environment. In section 6.3.1 (see page 69), two types of meshing domain are defined. The subdomain, enveloped by an internal boundary, is part of the meshing domain, but differs from a general subdomain. Therefore the following definition is employed to describe this type of subdomains.

Definition 8.1: The third type of meshing domain. A subdomain is continuous, and its boundary is represented by a set of closed surfaces S_3, which is used for reducing the degrees of freedom of grid points. This domain is defined as *the third type of meshing domain*, denoted by Ω_3. The boundary of the Ω_3 is termed *internal boundary*.

The idea of block encapsulation is to prevent free movement of grid points. On an internal boundary of Ω_3, also termed *internal surface*, grid points are restricted by its geometry. At both sides of the internal surface different aspect ratios of grid lines can be generated, as shown in Fig. 8.2.

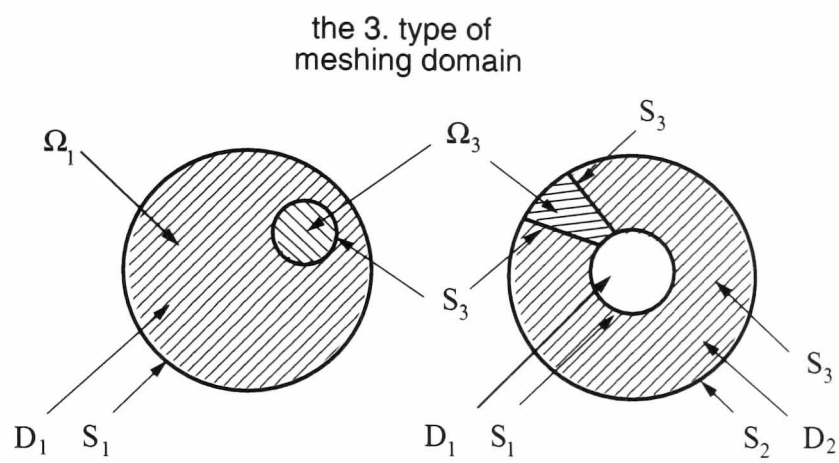


Figure 8.2: **The third type of meshing domain.**

8.2.1 Block Encapsulation Building

A subdomain, represented by a set of blocks, is selected from an existing wireframe model. The outer boundary of the subdomain is described by a set of closed surfaces. A subdomain of type Ω_3 is bounded by its outer boundary S_3. The grid elements (vertices, edges and faces) on the boundary of Ω_3 boundary are assigned to the S_3.

Fig. 8.3 shows an example of block encapsulation. Let a solution domain be defined by the box S_1 and the circle S_2. Suppose that a large grid density is required at the both sides of the line $y = 2x$. Introducing the internal surfaces S_3, expressed by two parallel lines

$$y = 2x \pm |a|$$

They intersect the physical boundaries S_1 and S_2, respectively, and result in two Ω_3 subdomains. Two blocks are generated for both Ω_3 subdomains. During grid

generation, the grid density within the blocks encapsulated is separated from the Ω_2 domains.

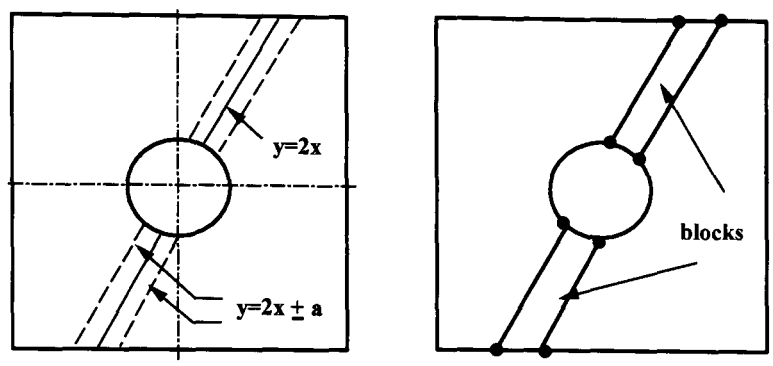


Figure 8.3: **Block encapsulation is a method for separating grid density of a subdomain Ω_3 from the Ω_2 domain.**

The application of block encapsulation is aimed to improve local resolution. For instance, if a certain region is of special interest, it will be accurately simulated in two stages. First, the whole solution domain is computed until a stationary solution is obtained. Second, since the region is encapsulated by its local block topology, it can be extracted from the solution domain. Using the known stationary solution as boundary condition, a more accurate computation for this subdomain can be performed.

8.3 Concept 2: Boundary Layer Enrichment

A Navier–Stokes grid is considered as a special form of an Euler grid. Clustering grid lines on body surfaces of an Euler grid, a Navier–Stokes grid is generated. However, attention must be paid to boundary layer topology. An inadequate topology may lead to an extension of the fine grid line distribution into the far field. Unnecessary small cells will fill the solution domain where the boundary layer does not exist.

Fig. 8.4 depicts a grid topology for an Euler computation. The enrichment of grid lines on the body surface of the cone yields a two–dimensional clustering. The area of the smallest cell is proportional to

$$A_{cell} \propto \delta_\xi \delta_\eta$$

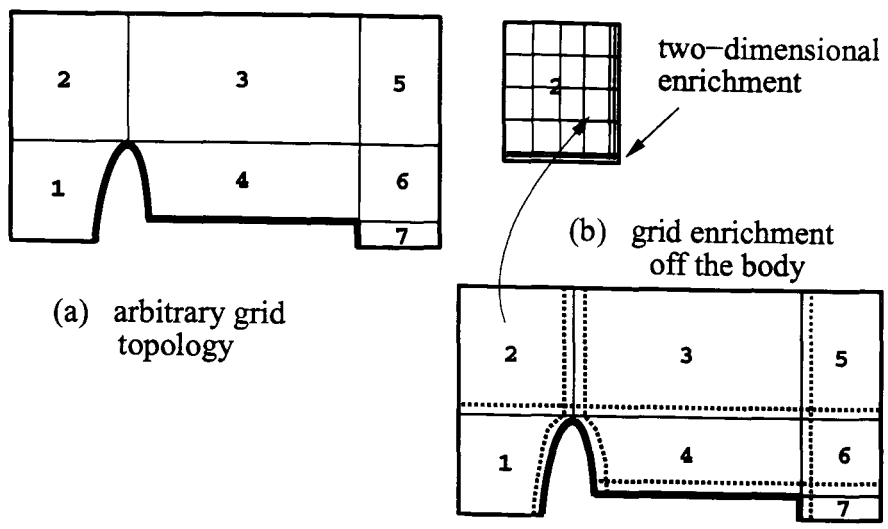


Figure 8.4: **This grid topology causes an extension of high grid density into regions where this large density is not wanted [40].**

where δ_ξ and δ_η denote the distances between the first and second lines of enriched surfaces in both ξ– and η–directions. Extremely small distances in both directions result in small cell sizes, which could cause problems like reduced convergence. Table 8.3 shows the extension of enrichment and its influence on cell size of the grid topology of the example in Fig. 8.4.

Table 8.1: The influence of grid topology on enrichment extension of the example in Fig. 8.4.

block #	*enrichment direction*	*extension direction*	*clustering dimension*
1	ξ_{max}		1
2	ξ_{max}, η_{min}		2
3	ξ_{min}, η_{min}		2
4	ξ_{min}, η_{min}		2
5		ξ_{min}, η_{min}	2
6		ξ_{min}, η_{min}	2
7			1

The concept of one dimensional enrichment is developed to overcome the disadvantage of multi–dimensional enrichment. In many cases, a fine grid line distribution of a Navier–Stokes mesh is needed on the fixed boundary, and an extension of grid line enrichment into the far field should be prevented.

Definition 8.2: Boundary layer topology. A set of blocks is connected to each other. They completely cover a fixed boundary, on which an enrichment of grid lines is required. If one and only one side (in two–dimensional case) or face (in three–dimensional case) of each of these blocks contacts the fixed boundary, this block topology is termed *boundary layer topology.*

A boundary layer topology ensures one–dimensional enrichment, and is therefore termed *boundary layer topology.*

8.3.1 Building for Boundary Layer Topology

In two–dimensional cases, building a boundary layer topology contains the following steps, depicted in Fig. 8.5:

- ▷ **Generate a start boundary**. A set of vertices $P = \{p_i(x, y)|i = 1, 2, \ldots, I\}$ is selected around a body surface. They are connected in sequence in the manner that no more than two edges have an end in common. The sequence of vertices is topologically a one–dimensional line. It is used as a start boundary.

- ▷ **Determine a sweep direction for vertices**. In order to determine the sweep direction for an arbitrary vertex, the normal $\mathbf{n}_i$ is calculated. This normal indicates the sweep direction.

- ▷ **Generate an offset boundary**. The offset boundary is generated by duplicating the start boundary in the sweep direction. The offset boundary consists of the same numbers of vertices and line segments.

- ▷ **Generate block loop**. A one–to–one connection of the corresponding vertices on both boundaries is generated. On the start boundary a one–dimensional enrichment can be generated when required.

In three–dimensional cases, a start boundary must have a two–dimensional block topology. An algebraic method is employed to generated an offset boundary. In order to explain this method, the following terminologies are introduced.

Definition 8.3: Dangling vertex. A vertex is defined as a *dangling vertex*, if it belongs to one edge only.

Definition 8.4: Undirected vertex. A vertex is defined as a *undirected vertex*, if it belongs two edges in common.

Definition 8.5: Junction vertex. A vertex is defined as a *junction vertex*, if it belongs to at least three edges.

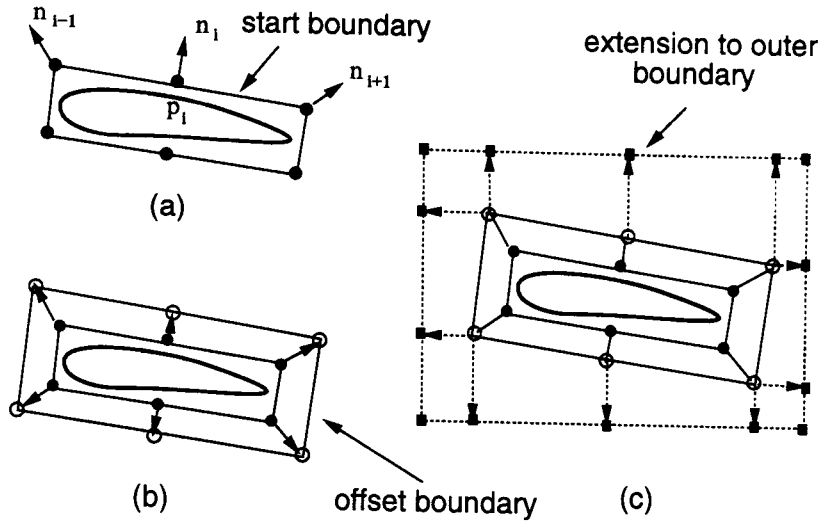


Figure 8.5: **Principle of building a one–dimensional enrichment topology.**

A dangling vertex has only one edge, and does no satisfy a tow–dimensional topology. Undirected and junction vertices are employed in building for boundary layer topology.

It is supposed that a start boundary is generated in three–dimensional space. It satisfies a two–dimensional block topology (see *grid construction rule 3* in section 6.3.3, page 74). Duplicating and projecting the vertices along certain directions, the two–dimensional block topology is converted into a three–dimensional one, as shown in Fig. 8.6.

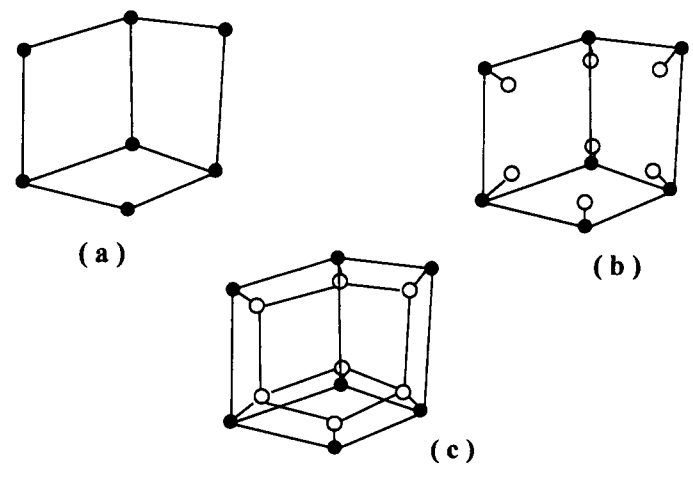


Figure 8.6: **The principle of generating a spatial boundary layer topology: (a) generation of a start boundary with two–dimensional block topology; (b) calculation of the projection directions for all vertices; and (c) generate the new vertices and their connectivity.**

The algorithm to generate this topology contains the general steps:

▷ **Calculate the shortest length of edges** l_{min}. The goal of finding the shortest edge length is to determine a measure for projection, termed projection radius r. A larger projection radius will substantially deform the geometric shape of an offset boundary. It is suggested that r is kept in the range $0 < r \leq 0.3 l_{min}$.

▷ **Calculate sweep direction for all vertices.** This key step can be accomplished using a simple method, depicted in Fig. 8.7. Using a projection radius r, three intersection points Q_1, Q_2 and Q_3 are found. The geometry center of these three points Q_c is the position of the offset vertex of the junction vertex. Note, in cases that edges of a junction vertex are at one plane, or in case of a undirected vertex, the offset vertices are at the same plane as the plane of their junction or indirection vertices. The positions of the offset vertices should be inadequate in their adequate positions.

▷ **Generate the offset boundary**. All vertices are projected in their normal directions. The one–to–one vertex connectivity is generated for the start offset boundary.

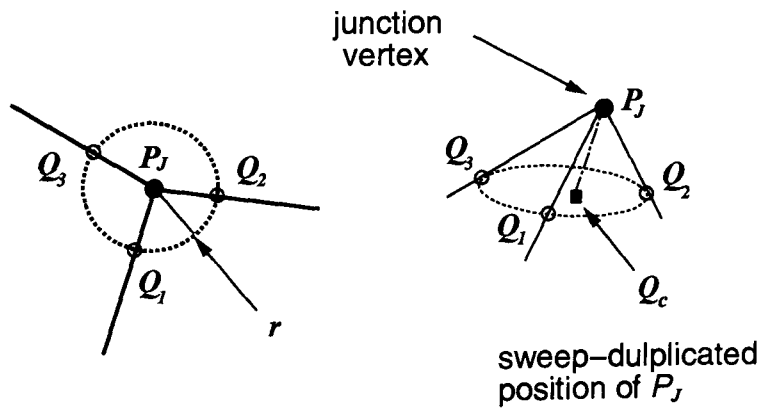


Figure 8.7: **Using a projection radius r three intersection points Q_1, Q_2 and Q_3 are obtained at three non–planar edges. The sweep–duplicated position is the geometric center of these points.**

As a summary, the construction rule for generating a boundary layer topology is given as follows.

Grid construction rule 6: Boundary layer topology. A boundary layer topology is built by generating a start boundary and its sweep–duplicative wireframe. The requirements of block topology for a start boundary are: (1) in two–dimensional cases, a start boundary topology is one–dimensional; (2) in three–dimensional cases, a start boundary topology is two–dimensional.

8.4 Concept 3: Smart Topology

It is expected that a local grid density can be varied without using a grid generator. This requires that blocks added to an existing grid are partially or totally merged or encapsulated by the blocks of the initial grid. This type of block topology used for local refinement of grid density is termed *smart topology.* Two types of smart topologies are explained as follows.

Block–merged smart topology. A set of blocks employed to enrich local grid density is inserted into a chosen region. The outer block topology of the region, i.e., face and vertex numbers remain unchanged. That is, new blocks are completely merged into the interior of the selected region. This type of smart topology is termed *block–merged smart topology.* Fig. 8.8 (a) shows a two–dimensional example of a block–merged smart topology. The original grid topology consists of four blocks, which are added in order to provide a local block refinement. An increase of grid density is encapsulated within the region chosen.

Side–merged smart topology. A local block refinement is generated by adding a set of blocks into the selected region. On the outer boundary of the region, the original block topology will be changed, since the new blocks are not completely merged into the interior of the region. That means, face and vertex numbers of the outer boundary of the selected region will be increased, shown in Fig. 8.8 (b). This topology is termed *side–merged smart topology* or *face–merged topology* for two– or three–dimensional cases, respectively.

8.4.1 Topology Building of a Smart Topology

The local wireframe topology, selected for a smart topology building, is termed *parent–structure.* A smart topology to be inserted into the parent–structure is

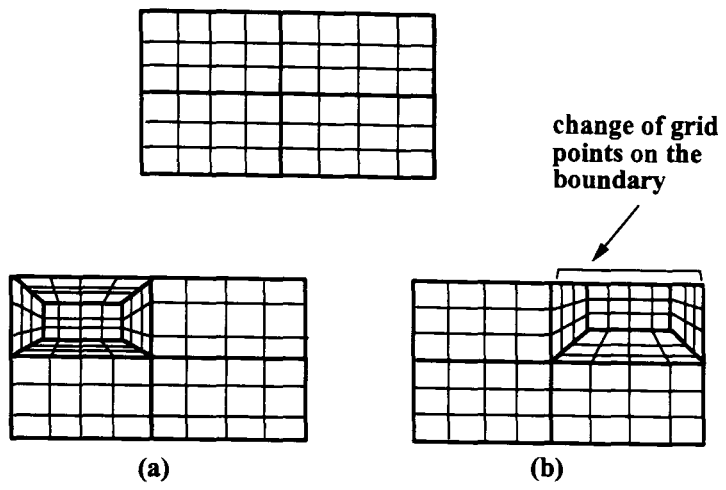


Figure 8.8: **Two types of smart topologies: (a) block–merged smart topology, and (b) side–merged smart topology.**

termed *child–structure*. According to requirements of grid density, a block– or a side–merged topology generation has the following steps, as shown in Fig. 8.9:

- ▷ **Select a local wireframe topology as parent–structure**. A region is selected for block refinement. In the region all grid points are deleted.
- ▷ **Generate a child–structure**. A block refinement either with block–merged or with face–merged type is generated. Both structures must have the same block topology.
- ▷ **Link two topologies**. Two block topologies are linked by generating one–to–one vertex connectivity.
- ▷ **Generate new blocks**. New blocks are generated using the block topology resulted from the parent– and child–structures.

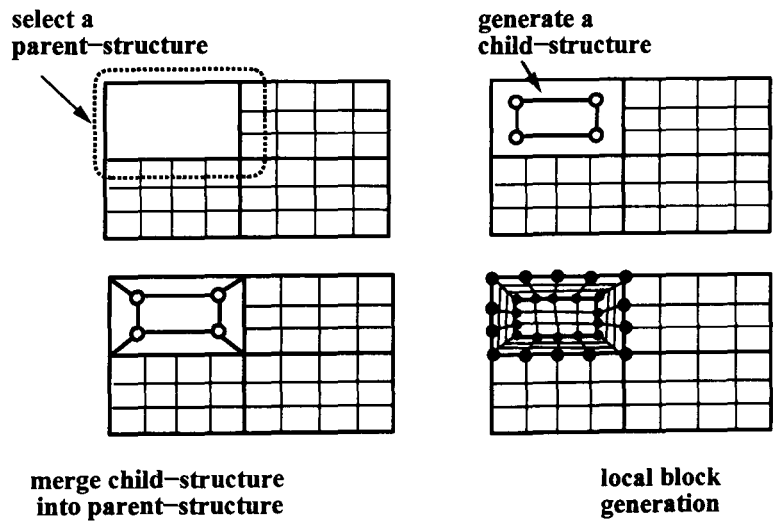


Figure 8.9: **A smart topology is constructed in four steps: (1) select region of block refinement; (2) build internal block topology; (3) combine two boundaries; (4) generate new blocks.**

The following rule gives the topological relation between these two structures.

> **Grid construction rule 7: Similarity of child–structure with parent–structure.** A smart topology, termed *child–structure*, is generated within an existing block topology, termed *parent–structure*. Both structures must have the same block topology.

A more useful application of a smart topology is to change a local grid topology for the increase of grid density. Fig. 8.10 shows two methods for the local refinement of block topology.

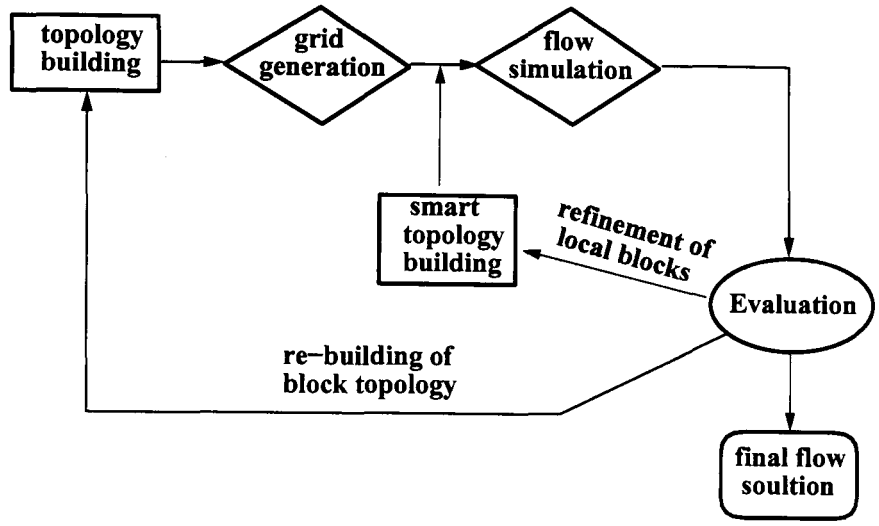


Figure 8.10: **Two methods for the local refinement of block topology.**

8.5 Examples for Passive Grid Adaptation

The examples are selected from the grid generation practices over several years. Although all grids are generated using GridPro [29] as well as Grid⋆ [36], the strategy of passive grid adaptation can be used to generate structured multiblock grids independent of the grid generator.

8.5.1 Block Encapsulation

The grid *generic car with micro–aerodynamic components* is selected as an advanced example. Suppose that a fine resolution of the region of the antenna tip is required, shown in Fig. 8.11, and a fine distribution of grid lines has to be kept in this region. The topology encapsulation is generated according to the steps explained above. First, a region is selected from the coarse grid topology. The dimension of the region will be determined in accordance with

the requirements of flow simulation. Second, an ellipsoid which encapsulates a third type of meshing domain (see Definition 8.1 in section 8.2) is introduced to represent the boundary of the subdomain. In the last step, a fine block topology is generated for the antenna tip. The grid elements on the subdomain boundary are assigned to the surface of ellipsoid.

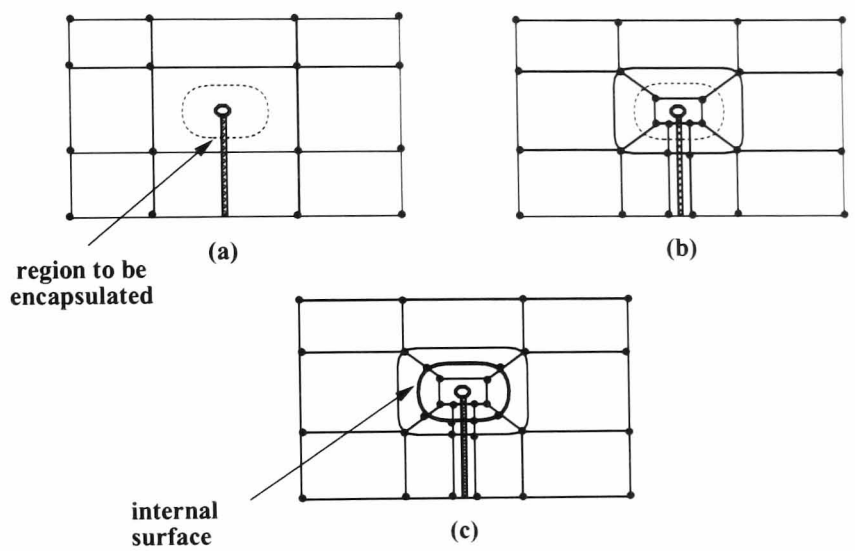


Figure 8.11: **Block encapsulation for the region of an antenna.**

Fig. 8.12 shows a close up of the grid *generic car with micro–aerodynamic components*. It can be seen that a high grid density is concentrated on region of the antenna tip. This is carried out by block encapsulation.

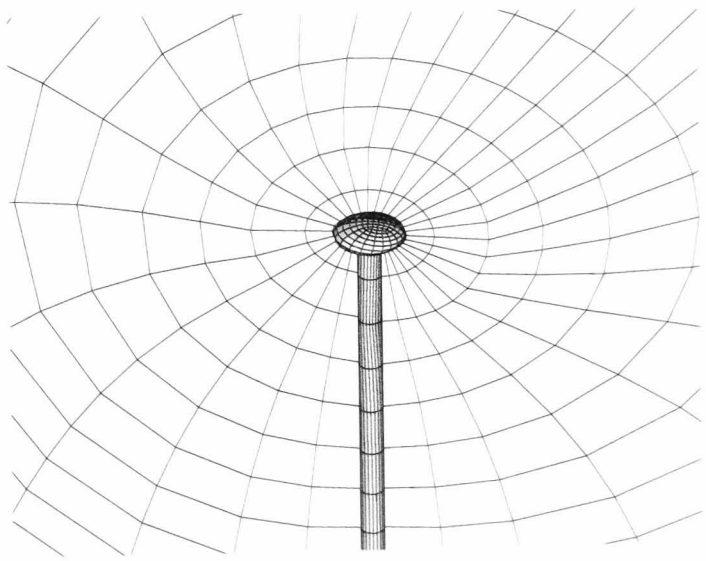


Figure 8.12: **Close up of the grid** *generic car with micro–aerodynamic components*. **A high grid density near the antenna tip is reached using block encapsulation.**

8.5.2 Boundary Layer Topology

Fig. 8.13 shows the principle of constructing the boundary layer topology of the example shown in Fig 8.4. Four additional blocks are introduced into the solution domain, which build a closed channel for one–dimensional enrichment. This enrichment topology avoids the disadvantages of unnecessary extension of grid lines into the far field. Grid generation and grid enrichment processes can be treated separately. The idea stems from the work in application in unstructured grid generation [61]. In [7], [38], [40], [48] and [82], the concept for structured grid are presented in order to generate a Navier–Stokes grid using user defined enrichment parameter. The strategy to separate a Navier–Stokes grid generation from an Euler grid generation has a large potential especially in three–dimensional cases.

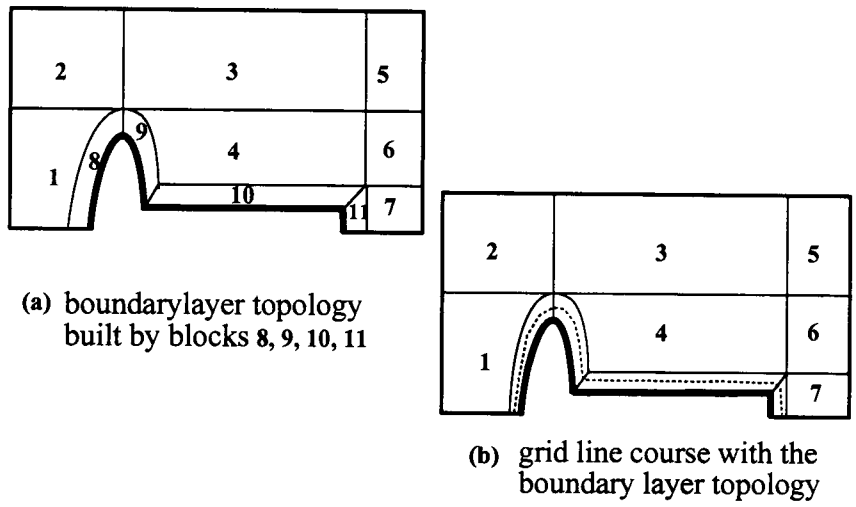


Figure 8.13: **A boundary layer topology is generated to enrich grid lines in one dimension [40].**

8.5.3 Smart Topology

In three–dimensional cases the block–wise refinement provides a high flexibility of local grid modification. A three–dimensional example stems from the joint–project in cooperation with the European Space Agency. The European space shuttle model Halis is simulated with the body flap under a given reflection angle. It is important to simulate the thermodynamic behavior of the body surface under the body flap. The smart topology is generated for the region under the body flap. Fig. 8.14 shows the building of block enrichment. The block is selected as a parent–structure, whose one face is on the windward side of the flap. Its eight vertices are denoted by A, B, C and D, respectively. The child–structure, built by one cube $\mathcal{B}\{a, b, c, d, a', b', c', d'\}$, has a face–merged

topology. Connecting the corresponding vertices, the smart topology of the region provides a local increase of grid density.

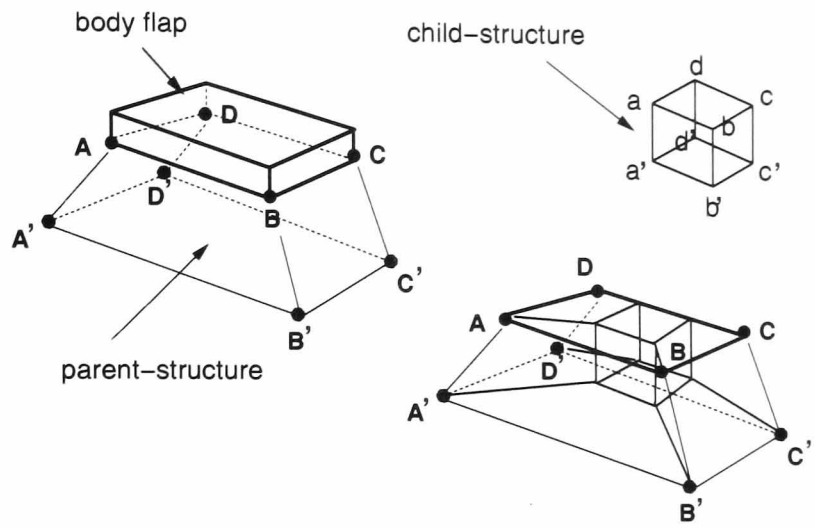


Figure 8.14: **Smart topology is generated on the windward side of the body flap in order to locally increase grid density in this region.**

Fig. 8.15 shows the local enrichment using a three–dimensional smart topology. Within the closed block loop, grid lines are clustered directly below the body flap where high numerical accuracy is required.

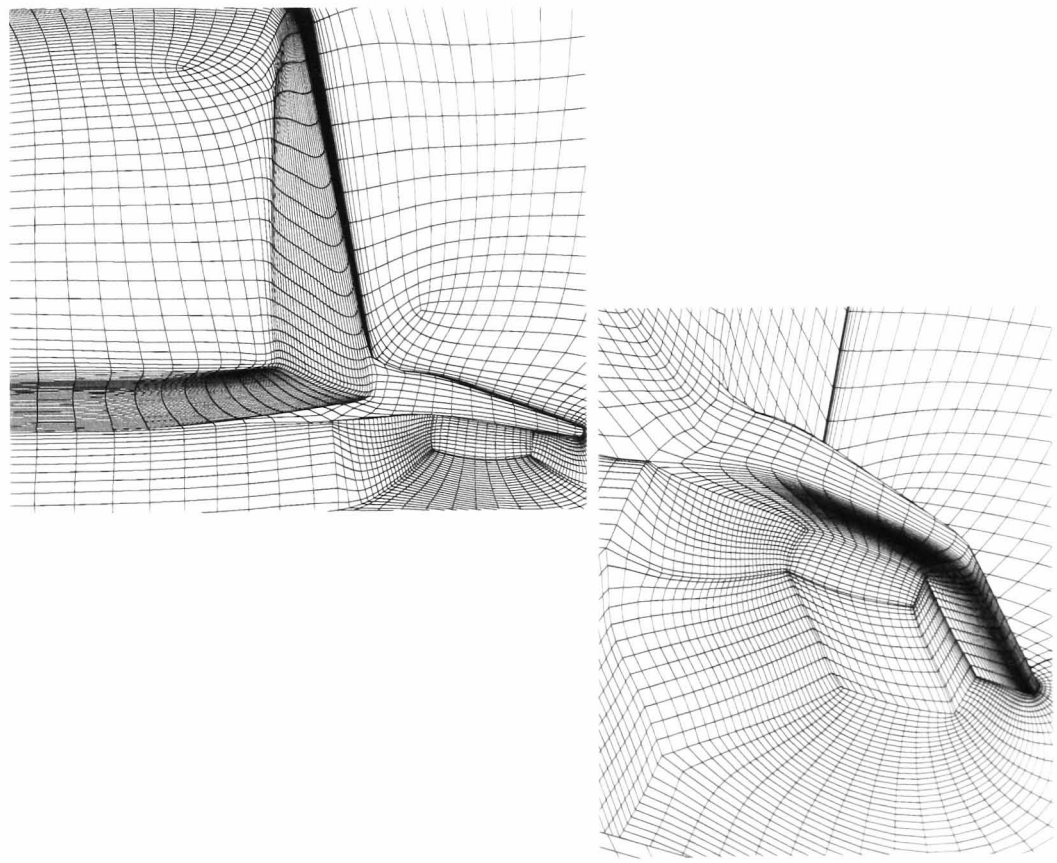


Figure 8.15: **Below the body flap a smart topology is generated to increase local grid density. The smart topology does not considerably change the original grid topology [40].**

8.6 Chapter Summary

The main contribution of the present chapter is that a new strategy for topology–based grid adaptation is developed. This type of grid adaptation is independent of a flow solution, it is therefore termed *passive grid adaptation.* The passive grid adaptation consists of three methods. They are *block encapsulation*, *boundary layer enrichment*, and *smart topology.* All these methods are used for improvement of local grid density.

Block encapsulation. A subdomain is encapsulated by a set of encapsulated surfaces. Grid points on these surfaces have reduced degrees of freedom. Grid line distributions are separated at both sides of the encapsulated surfaces.

Boundary layer enrichment. On body surface, a block structure, consisting of a set of blocks wrapped around the body surface, ensures a one–dimensional enrichment of grid lines off the body surface. This grid topology is used for generating fine line distribution for a Navier–Stokes grid.

Smart topology. Grid density is increased by local block refinement. Inserting blocks into an existing grid, grid density is improved.

Chapter 9

Solution–Based Grid Adaptation

Solution–based grid generation is also known as solution adaptation [5], [57]. In comparison with topology–based grid adaptation, which is realized by block refinement, or grid line enrichment within a boundary layer topology, a grid is adapted using a flow solution. In this thesis, the term *active grid generation* is used for this kind of approach, in order to distinguish it from *passive grid adaptation* based on block topology. In general, there are two major schemes to actively adapt block–structured grids, namely *refinement* and *redistribution* schemes.

The former is performed either by introducing new cells into an initial grid or by splitting existing cells in regions of large gradients of flow variables [18]. An essential advantage of grid refinement is that the primary positions of initial grid points are not changed during a refinement process. Problems such as slope discontinuity on block interfaces, geometric deformation of a physical boundary do not exist. However, a refinement scheme causes an increase in cell numbers. This requires a dynamic allocation of memory storage. The connectivity relations among cells, grid topology, as well as data structure of nodes and their linkage relations become more complex with the increasing number of cells. In addition, for a Navier–Stokes computation, an excessive number of grid points may be inserted by refinement tools in the boundary layers. The need for one–dimensional refinement in such regions necessitates the use of hexahedral elements in the boundary layer. Cell splitting in this region may not be more effective than grid line enrichment in a one–dimensional fashion [10].

The latter is performed by grid point redistribution. In comparison to a refinement scheme, a redistribution scheme does not change grid topology or the

total number of cells. After an adaptation process, an initial flow solution can be updated with respect to the spatial relation between the original and adaptive positions of grid points, so that the computation can be quickly continued. since there is no need for generating a new block topology.

In general, structured grids provide sophisticated means both for clustering and adaptation using redistribution or local refinement techniques. A comparison of these two approaches is given in [19] and [20], where local refinement gives somewhat better results. However, application of a refinement can be more expensive due to increasing memory demand and computing intensity. In case that a fine resolution for a bow shock or a shock reflection is required, the alignment of the grid through a redistribution can provide more accurate solution than refining the grid with an increasing large number of smaller and smaller cells. The choice of a redistribution scheme in this thesis is based on the following arguments.

- ▷ Since the computational grids used are block structured, and the solver developed is based on an object–oriented method, the emphasis of this research work is therefore focused on adaptation of structured grids.
- ▷ With increasing geometric complexity, a grid may have several thousands of blocks with millions of cells. A constant number of cells enables the user to estimate the available memory capacity, and to utilize his hardware resource more effectively.

The present chapter describes the strategy for multiblock adaptation using a redistribution scheme as well as the adaptation algorithm developed based on the previous papers of grid optimization [12], [50]. The core of the strategy is to adapt grid points with respect to their geometric and topological constraints. In extension of the algorithm for mono–block adaptation to multiblock case, the strategy concentrates on finding answers of the following two questions:

- *How can we prevent C^1–discontinuity on block interfaces?*

and

- *How will grid points on a fixed boundary be moved without destroying its geometry?*

The basic idea of optimizing an adaptive grid is to establish an objective function, which describes the relationship between grid point redistribution

and weight functions. Grid points are redistributed by the minimization of objective functions. In an objective function for grid adaptation, some measures for grid quality are formulated as penalty functions. A combination of the penalty functions with the objective function improves adaptive grid quality and convergence behaviors of numerical solutions.

9.1 Strategy Development for Grid Adaptation

In order to successfully adapt multiblock grids, the general requirements for grid adaptation and the strategy developed in this thesis are summarized as follows:

- ▷ **1. Computational coordinate system**. In practical applications, grids are generated for complex geometries using different block topologies. Grid sizes and shapes should not play a role in a grid adaptation process.

To ensure that an algorithm for grid adaptation will work in a robust manner, and is available independent of grid sizes and shapes, grids should be transformed into a computational space, where the computation for grid adaptation is performed in a dimensionless way.

- ▷ **2. Weight functions**. Flow features are formulated in the form of some weight functions to adapt a grid. For a grid adaptation process, grid points are clustered according to flow gradients, while the magnitudes of flow variables should not play a role in a grid adaptation process.

Flow variables used for weight functions should be dimensionless, such that an adaptation algorithm will work independently of the magnitudes of flow variables. To meet this requirement, it is reasonable to norm these quantities.

- ▷ **3. Adaptation of the internal grid points**. To adapt a single grid point, influences of spatial positions as well as weight functions of neighboring grid points have to be considered in a grid adaptation computation.

Grid points within a block are connected in a structured manner. Grid points should be moved with respect to their relations to neighboring points. This requirement can be met by building *cell coupling*, i.e., an internal grid point and its neighboring points and the corresponding weight functions should be formulated as independent variables in a stencil equation. Grid adaptation is performed by solving a set of these equations.

▷ **4. Adaptation of grid points on block boundaries**. There are some geometric constraints of the grid points on physical boundary during grid adaptation. They have to be adapted with respect to the surface geometry of the physical boundary.

It necessitates special treatment of these grid points. Grid adaptation should be performed in different stages with respect to degrees of the freedom of grid points on physical boundary. Moreover, it must ensure that a movement of grid points does not change the surface geometry of the fixed boundary.

▷ **5. Adaptation of grid points on block interfaces**. Blocks are connected to each other through their interfaces. That is, the grid points on block interfaces have some topological constraints during the adaptation process.

It necessitates a data coupling of grid points among adjacent blocks to ensure that grid lines are C^1–continuous on block interfaces. This can be improved by generating block overlaps, i.e., a block interface is extended using the grid points of its neighboring block.

▷ **6. Adaptive grid quality**. In cases of complex geometries and block topologies, the quality of adaptive grids is unlikely to be visually evaluated in an easy manner. An algorithm for grid adaptation is required in controlling grid quality.

The strategy used in this thesis is to formulate the measures for grid quality control in the adaptation equation through penalty functions. The efficiency of an adaptation algorithm must be tested in different cases.

▷ **7. Multiblock grid adaptation**. The core of a successful adaptation of multiblock grids is to design an adaptation procedure with different degrees of freedom have to be specified in this procedure.

First, internal grid points, which have three degrees of freedom, are adapted. After this, all grid points on block interfaces are updated, so that all pairs of common grid points are placed in their corresponding unique positions. Second, grid points on a physical boundary are classified into two groups according to their degrees of freedom. Grid points with two degrees of freedom are adapted on the geometry surface. Third, grid points with one degree of freedom are adapted along edges. Last, it must be sure that the singularity points, having zero degree of freedom, are not moved anywhere.

9.1.1 Description of a Solution Domain

Grid adaptation is computed in parametric coordinate system. In three–dimensional space, a grid in physical space, represented by $P = \{p_{i,j,k}(\mathbf{x}) \mid i = 1, \ldots, I; j = 1, \ldots, J; k = 1, \ldots, K\}$ is transformed to a parametric space using the following relation

$$\begin{aligned} t_{i,j,k} &= \frac{\sum_{i=2}^{I} |p_{i,j,k}(\mathbf{x}) - p_{i-1,j,k}(\mathbf{x})|}{\sum_{l=2}^{I} |p_{l,j,k}(\mathbf{x}) - p_{l-1,j,k}(\mathbf{x})|}(I-1) \\ u_{i,j,k} &= \frac{\sum_{j=2}^{J} |p_{i,j,k}(\mathbf{x}) - p_{i,j-1,k}(\mathbf{x})|}{\sum_{m=2}^{J} |p_{i,m,k}(\mathbf{x}) - p_{i,m-1,k}(\mathbf{x})|}(J-1) \\ v_{i,j,k} &= \frac{\sum_{k=2}^{K} |p_{i,j,k}(\mathbf{x}) - p_{i,j,k-1}(\mathbf{x})|}{\sum_{n=2}^{K} |p_{i,n,k}(\mathbf{x}) - p_{i,n-1,k}(\mathbf{x})|}(K-1) \end{aligned} \tag{9.1}$$

where $t_{i,j,k}$, $u_{i,j,k}$ and $v_{i,j,k}$ denote the coordinates in the parameter space. The grid in the parameter space is denoted by

$$\hat{P} = \{\hat{p}_{i,j,k}(\mathbf{t}) \mid i = 1, \ldots, I; j = 1, \ldots, J; k = 1, \ldots, K\} \tag{9.2}$$

where the symbol " ˆ " denotes the parameter space, and $\mathbf{t} = (t, u, v)$. Specifying the index (i, j, k), the parameter coordinates are obtained from Eq. (9.1). The parameters give relations among grid spacings in a dimensionless manner.

Solution domain of the objective function is described by

$$\hat{P} = \{\hat{p}_{i,j,k}(\mathbf{t}) \mid i = 2, \ldots, I-2; j = 2, \ldots, J-2; k = 2, \ldots, K-2\} \tag{9.3}$$

Grid points on a block boundary are considered as stationary during an adaptive computation, i.e., the first and second partial derivatives with respect to weight functions and parameter coordinates at these grid points vanish.

9.1.2 Determination of Flow Features

Different methods can be employed to extract flow features from a flow solution. For instance, the *van Albada* limiter function is an efficient shock sensor to detecting the position of a shock wave [95]. It is mostly applied in time–accurate simulations or adaptation [5]. In case of a stationary solution, flow features can be determined by gradients of flow variables.

Scalar quantities such as Mach number, density, pressure, and temperature, selected as quantities to build a weight function, are termed *weight variables.*

9.1.2.1 Norm of Weight Variables

The values of the scalar quantities of a flow solution have different magnitudes. Their orders of magnitudes can be significantly different. This leads to a peak value in the weight function at a location of the flow discontinuity. In order to calculate weight functions within a reasonable range, weight variables are normalized by

$$\psi_{i,j,k}(b) = \frac{\phi_{i,j,k}(b) - \phi_{min}}{\phi_{max} - \phi_{min}} \tag{9.4}$$

where $\psi_{i,j,k}(b)$ denotes the normalized weight variable at the node (i,j,k) of block b, $\phi_{i,j,k}(b)$ is the weight variable to be normalized, and ϕ_{min} and ϕ_{max} are the global maximal and minimal values of the weight variable of the entire solution domain. Normalized weight variables are dimensionless, and their magnitudes are in the range $0 \leq \psi_{i,j,k}(b) \leq 1$.

Both the parametric coordinate system and the weight variables are dimensionless. In the development of an adaptation algorithm, these dimensionless quantities for grid adaptation ensure that a grid adaptation is independent of the magnitudes of weight variables.

9.1.2.2 Weight Functions

Weight functions are the measure for grid point redistribution. Many different types of numerical weight functions are discussed in [57] and [92]. Theoretically, it is possible to construct a weight function using a smooth function to improve the quality of the weight function, i.e., to reduce the numerical noise near a shock wave [8]. However, the computational stability of the grid adaptation may be costly due to the algorithmic complexity.

McRae suggested to formulate a weight function as a combination of the superposition of the first and second derivatives of flow variables [62]. The sensitivity and effectiveness of flow gradients and curvatures are controlled by means of filter functions. The most time consuming problem of implementing

this algorithm is the computation of the second derivatives of the flow solution, because this computation may require values from vertices in the diagonal direction outside a block.

A more general and representative formulation of a weight function is a linear combination of a set of non–negative gradients in the following form [28], [36]

$$w(x) = 1 + \sum_{i=1}^{n} \alpha_i \left| \frac{\partial \psi^i}{\partial x} \right| \tag{9.5}$$

where $x = x, y, z$ denotes the physical coordinates, α_i and $\left|\frac{\partial \psi^i}{\partial x}\right|$ denote coefficients and gradients of weight variables, respectively.

9.1.3 Adaptation of Internal Points

The basic equation for grid adaptation was derived from the variational functional formulation in [25]. In the one–dimensional case, it has the following form

$$w x_\xi = c \tag{9.6}$$

where w denotes a weight function, x_ξ the mesh spacing between two neighboring points in physical space, and c is a constant. This statement expresses the constant product of the mesh spacing and weight function one–dimensionally in physical space [1].

The computation using this formulation is performed in the physical space. The efficiency of an adaptation algorithm could be dependent upon sizes and shapes of grids. Moreover, there is no penalty function to limit movement of grid points within a restricted region, in which no cell skewness occurs.

In [11], [12], [51], [66], [80] and [94], the quality of adaptive grids is controlled using variational or functional methods. The essence of grid quality control is to generate an adaptive grid using a measure for grid quality derived from flow features. Based on this idea, an optimization method is developed in this thesis. This method incorporates measures for grid quality into the basic adaptation equation and formulates the adaptation function in the form of an objective function. The results from the minimization of the objective function will give the local optimal grid adaptation. In the following, the method is

derived in two–dimensions, while three–dimensional problems are considered as an extension of the two–dimensional equation to the third dimension.

9.1.3.1 A General Equation for Grid Adaptation

Eiseman presented the least–squares statement for grid adaptation using the mid–point weights [28]. In one–dimensional case with N grid points, a grid is adapted by the minimization of the weighted sum of squares

$$f = \sum_{i=1}^{N-1} w_{i+\frac{1}{2}} [p_{i+1}(\mathbf{x}) - p_i(\mathbf{x})]^2 \tag{9.7}$$

where $p_i(\mathbf{x})$ denotes the position of a grid point. Regarding the weights as constants, the minimization of the equation gives the same statement as Eq. (9.6). Adaptive positions of grid points are obtained by solution of a tridiagonal equation system.

In this thesis, the quadratic statement of Eq. (9.6) is used based on the following arguments. First, a grid point in space is connected to its neighboring grid points. Moving this grid point, the weight functions and grid spacings of its neighboring grid points should be considered. Second, grid points should be adapted in certain optimal positions, where grid clustering represent flow features, and cells are of good quality. It is required to add some control terms for grid quality to an adaptation statement. In a linear equation system, these additional control terms will change the linearity of the equation system.

Based on the idea of the equidistribution statement of Eq. (9.6), we use its quadratic formulation as an *objective function of grid adaptation*, and apply it in multi–dimensional grid adaptation. The model equation for a general objective function has the following form

$$f = (w_1 s_1)^2 + (w_2 s_2)^2, \ldots, (w_n s_n)^2 + \text{control terms} \tag{9.8}$$

where s_i denotes the length of a line segment. The *control terms* are constructed using the measures for grid quality, such as orthogonality, cell aspect ratio, and cell smoothness.

Consider a simple one–dimensional problem. Grid points are ordered in sequence x_0, x_1, x_2 and x_3, where x_0 and x_3 are fixed points. Line segments are weighted

by $w_{\frac{1}{2}}$, $w_{1+\frac{1}{2}}$ and $w_{2+\frac{1}{2}}$ at the mid–points of three line segments, respectively, as shown in Fig. 9.1.

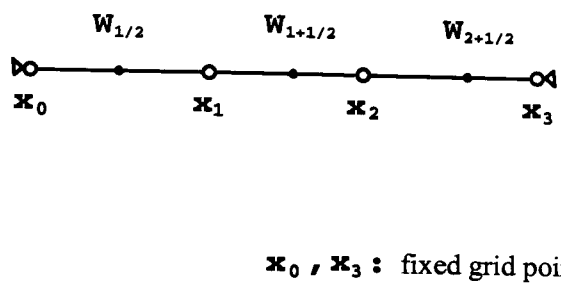


Figure 9.1: **Example of one–dimensional grid adaptation. The points x_0 and x_3 are fixed points, while the new positions of x_1 and x_2 are to be determined.**

Neglecting control terms, the weighted sum of squares is expressed by

$$f = w_{\frac{1}{2}}^2(x_1 - x_0)^2 + w_{1+\frac{1}{2}}^2(x_2 - x_1)^2 + w_{2+\frac{1}{2}}^2(x_3 - x_2)^2 \tag{9.9}$$

Let $w_{\frac{1}{2}} = 1$, $w_{1+\frac{1}{2}} = 1$ and $w_{2+\frac{1}{2}} = 1$, $x_0 = 0$, $x_3 = 1$. Within the unit square, function f describes a convex surface, as shown in Fig. 9.2. It is obvious that the minimum of Eq. (9.9) is reached, when the grid points x_1 and x_2 are distributed equidistantly.

9.1.3.2 Objective Function for Grid Adaptation

The model considered in this section for the derivation of the objective function is a part of the two–dimensional grid, as shown in Fig. 9.3, where solid circles denote grid points. It is assumed that weight functions are located at the mid–point of line segments, denoted by a square. A point $p_{i,j}(\mathbf{x})$ is coupled to its four neighboring points, $p_{i+1,j}(\mathbf{x})$, $p_{i,j+1}(\mathbf{x})$, $p_{i-1,j}(\mathbf{x})$, and $p_{i,j-1}(\mathbf{x})$. This system is termed *five–point coupling*. It is supposed that the coupling has a strong influence on the grid point $p_{i,j}(\mathbf{x})$ and that the influences from other grid points on it is weak. Therefore neglected.

The basic objective function for adapting the point $p_{i,j}(\mathbf{x})$ is

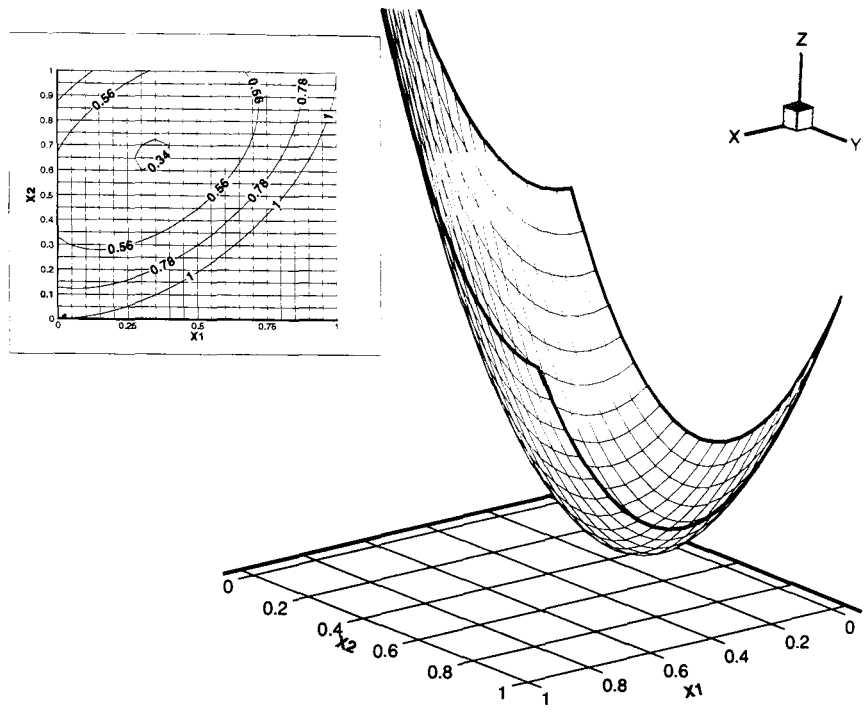


Figure 9.2: **Eq. 9.9 is expressed by a convex surface. In the x–y plane, two independent variables x_1 and x_2 are given, while the z–axis denotes the function values.**

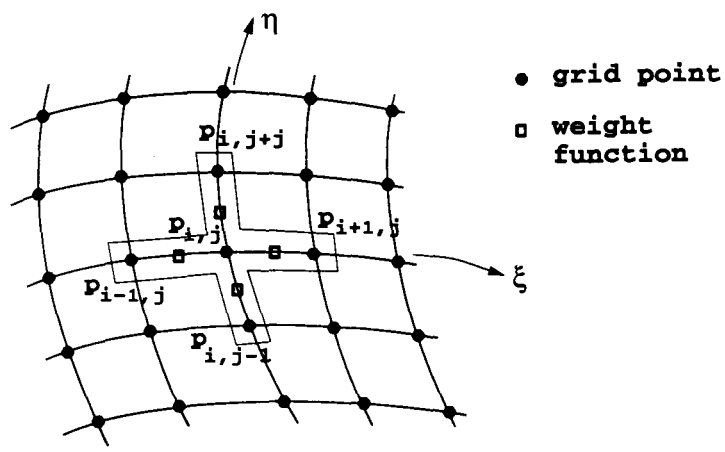


Figure 9.3: **The point $p_{i,j}(\mathbf{x})$ is directly coupled with its four neighboring points.**

$$
\begin{aligned}
f^a_{i,j}(\mathbf{x}) \;=\; & w^2_{i,j-\frac{1}{2}}\left[(x_{i,j-1}-x_{i,j})^2+(y_{i,j-1}-y_{i,j})^2\right]+ \\
& w^2_{i-\frac{1}{2},j}\left[(x_{i-1,j}-x_{i,j})^2+(y_{i-1,j}-y_{i,j})^2\right]+ \\
& w^2_{i+\frac{1}{2},j}\left[(x_{i+1,j}-x_{i,j})^2+(y_{i+1,j}-y_{i,j})^2\right]+ \\
& w^2_{i,j+\frac{1}{2}}\left[(x_{i,j+1}-x_{i,j})^2+(y_{i,j+1}-y_{i,j})^2\right] \qquad (9.10)
\end{aligned}
$$

where the superscript a denotes *adaptation*, and i and j denote the ξ–, and η–directions, respectively.

In three–dimensional cases, the point $p_{i,j,k}(\mathbf{x})$ is coupled by its six neighboring points, namely, $p_{i-1,j,k}(\mathbf{x})$, $p_{i+1,j,k}(\mathbf{x})$ in ξ–direction, $p_{i,j-1,k}(\mathbf{x})$, $p_{i,j+1,k}(\mathbf{x})$ in η–direction, and $p_{i,j,k-1}(\mathbf{x})$, $p_{i,j,k+1}(\mathbf{x})$ in ζ–direction, termed *seven–point coupling*, as shown in Fig. 9.4. The new position of $p_{i,j,k}(\mathbf{x})$ is determined by the minimization of the following objective function

$$
\begin{aligned}
f^a_{i,j,k}(\mathbf{x}) \;=\; & w^2_{i,j-\frac{1}{2},k}\left[(x_{i,j-1,k}-x_{i,j,k})^2+(y_{i,j-1,k}-y_{i,j,k})^2\right]+ \\
& w^2_{i-\frac{1}{2},j,k}\left[(x_{i-1,j,k}-x_{i,j,k})^2+(y_{i-1,j,k}-y_{i,j,k})^2\right]+ \\
& w^2_{i+\frac{1}{2},j,k}\left[(x_{i+1,j,k}-x_{i,j,k})^2+(y_{i+1,j,k}-y_{i,j,k})^2\right]+ \\
& w^2_{i,j+\frac{1}{2},k}\left[(x_{i,j+1,k}-x_{i,j,k})^2+(y_{i,j+1,k}-y_{i,j,k})^2\right]+ \\
& w^2_{i,j,k-\frac{1}{2}}\left[(x_{i,j,k-1}-x_{i,j,k})^2+(y_{i,j,k-1}-y_{i,j,k})^2\right]+ \\
& w^2_{i,j,k+\frac{1}{2}}\left[(x_{i,j,k+1}-x_{i,j,k})^2+(y_{i,j,k+1}-y_{i,j,k})^2\right] \qquad (9.11)
\end{aligned}
$$

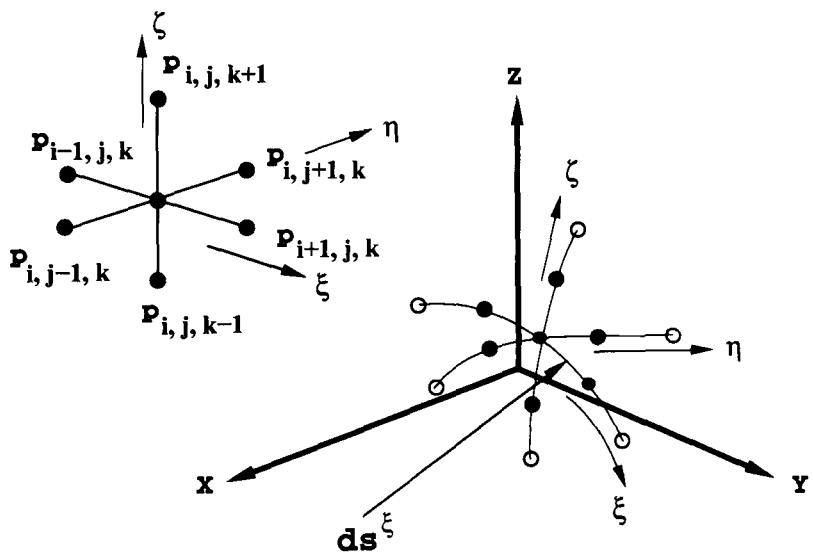


Figure 9.4: **The point $p_{i,j,k}(\mathbf{x})$ is coupled with its six neighboring points, termed** *seven–point coupling*.

Test case of the objective function. The first example is a single block grid with the dimension 3×3, as shown in Fig. 9.5. The coordinates of grid points and weight functions on grid points are given as follows

$$\begin{array}{lll} p_{i-1,j-1}(-2,-3), & p_{i,j-1}(1,-3), & p_{i+1,j-1}(3,-3) \\ p_{i-1,j}(-3,\ 0), & p_{i,j}(0,\ 0), & p_{i+1,j}(3,-1) \\ p_{i-1,j+1}(-1,\ 3), & p_{i,j+1}(2,\ 3), & p_{i+1,j+1}(4,\ 2) \end{array}$$

$$\begin{array}{lll} w_{i-1,j-1} = 0.5, & w_{i,j-1} = 1.0, & w_{i+1,j-1} = 1.0 \\ w_{i-1,j} = 0.0, & w_{i,j} = 1.0, & w_{i+1,j} = 1.0 \\ w_{i-1,j+1} = 0.0, & w_{i,j+1} = 0.0, & w_{i+1,j+1} = 0.5 \end{array}$$

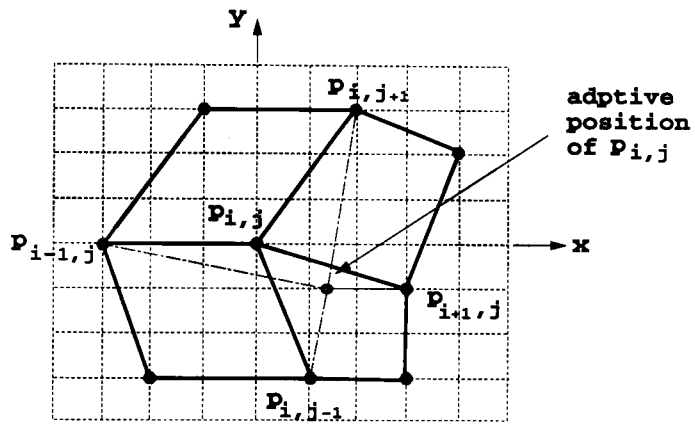


Figure 9.5: **A 3×3 grid is the first example to test the objective function of Eq. (9.10).**

Weight functions, linearly interpolated onto the mid–points, have the values $w_{i,j-\frac{1}{2}} = 1.0$, $w_{i-\frac{1}{2},j} = 0.5$, $w_{i+\frac{1}{2},j} = 1.0$, and $w_{i,j+\frac{1}{2}} = 0.5$. Four neighboring points are on the block boundary, and are therefore stationary during adapting the interior grid point $p_{i,j}(\mathbf{x})$. The minimum of the objective function is iteratively found using Newton method [69]

$$\mathbf{x}^{(n+1)} = \mathbf{x}^{(n)} - [\mathbf{H}(\mathbf{x}^{(n)})]^{-1} \nabla f^a(\mathbf{x}^{(n)}) \tag{9.12}$$

where $\mathbf{x} = [x, y]^T$, the superscript n denotes iteration number; $\mathbf{H}(\mathbf{x})$ denotes the square and symmetric Hessian matrix; the gradient vector $\nabla f^a(\mathbf{x})$ is the first partial derivatives of the objective function $f^a(\mathbf{x})$. Eq. (9.12) is rewritten as

$$\begin{bmatrix} x_{i,j}^{(n+1)} \\ y_{i,j}^{(n+1)} \end{bmatrix} = \begin{bmatrix} x_{i,j}^{(n)} \\ y_{i,j}^{(n)} \end{bmatrix} - \begin{bmatrix} \frac{\partial^2 f_{i,j}^a(\mathbf{x})}{\partial x_{i,j}^{(n)} \partial x_{i,j}^{(n)}} & \frac{\partial^2 f_{i,j}^a(\mathbf{x})}{\partial x_{i,j}^{(n)} \partial y_{i,j}^{(n)}} \\ \frac{\partial^2 f_{i,j}^a(\mathbf{x})}{\partial y_{i,j}^{(n)} \partial x_{i,j}^{(n)}} & \frac{\partial^2 f_{i,j}^a(\mathbf{x})}{\partial y_{i,j}^{(n)} \partial y_{i,j}^{(n)}} \end{bmatrix}^{-1} \begin{bmatrix} \frac{\partial f_{i,j}^a(\mathbf{x})}{\partial x_{i,j}^{(n)}} \\ \frac{\partial f_{i,j}^a(\mathbf{x})}{\partial y_{i,j}^{(n)}} \end{bmatrix} \tag{9.13}$$

The first and second partial derivatives of Eq. (9.10) with respect to variables $x_{i,j}$ and $y_{i,j}$ are obtained by

$$\begin{aligned} \frac{\partial f_{i,j}^a(\mathbf{x})}{\partial x_{i,j}} &= 2a_{i,j}x_{i,j} - 2b_{i,j}(x) \\ \frac{\partial f_{i,j}^a(\mathbf{x})}{\partial y_{i,j}} &= 2a_{i,j}y_{i,j} - 2b_{i,j}(y) \\ \frac{\partial^2 f_{i,j}^a(\mathbf{x})}{\partial x_{i,j}^2} &= 2a_{i,j} \\ \frac{\partial^2 f_{i,j}^a(\mathbf{x})}{\partial y_{i,j}^2} &= 2a_{i,j} \\ \frac{\partial^2 f_{i,j}^a(\mathbf{x})}{\partial x_{i,j} \partial y_{i,j}} &= 0 \end{aligned} \tag{9.14}$$

where

$$\begin{aligned} a_{i,j} &= w_{i,j-\frac{1}{2}}^2 + w_{i-\frac{1}{2},j}^2 + w_{i+\frac{1}{2},j}^2 + w_{i,j+\frac{1}{2}}^2 \\ b_{i,j}(x) &= w_{i,j-\frac{1}{2}}^2 x_{i,j-1} + w_{i-\frac{1}{2},j}^2 x_{i-1,j} + w_{i+\frac{1}{2},j}^2 x_{i+1,j} + w_{i,j+\frac{1}{2}}^2 x_{i,j+1} \\ b_{i,j}(y) &= w_{i,j-\frac{1}{2}}^2 y_{i,j-1} + w_{i-\frac{1}{2},j}^2 y_{i-1,j} + w_{i+\frac{1}{2},j}^2 y_{i+1,j} + w_{i,j+\frac{1}{2}}^2 y_{i,j+1} \end{aligned} \tag{9.15}$$

Inserting the relations (9.15) into Eq. (9.14), and using them in Eq. (9.13), one obtains

$$\begin{bmatrix} x_{i,j}^{(n+1)} \\ y_{i,j}^{(n+1)} \end{bmatrix} = \begin{bmatrix} x_{i,j}^{(n)} \\ y_{i,j}^{(n)} \end{bmatrix} - \begin{bmatrix} 2a_{i,j} & 0 \\ 0 & 2a_{i,j} \end{bmatrix}^{-1} \begin{bmatrix} 2a_{i,j}x_{i,j}^{(n)} - 2b_{i,j}(x^{(n)}) \\ 2a_{i,j}y_{i,j}^{(n)} - 2b_{i,j}(y^{(n)}) \end{bmatrix} \tag{9.16}$$

In case of $a_{i,j} > 0$, the Hessian of Eq. (9.16) is positive–definite. The objective function reaches the minimum by the stationary position of the grid point $p_{i,j}(\mathbf{x})$. Since the Hessian is fixed for quadratic functions, this quadratic function exactly reaches the solution in one step from any starting search point.

Inserting the known values into Eq. (9.15), one obtains $a_{i,j} = 2.5$, $b_{i,j}(x) = 3.5$, and $b_{i,j}(y) = -2.5$. The solution is obtained using the starting point $\mathbf{x}^{(0)} = [0,\ 0]^T$

$$\begin{bmatrix} x^1_{i,j} \\ y^1_{i,j} \end{bmatrix} = \begin{bmatrix} 0.0 \\ 0.0 \end{bmatrix} - \begin{bmatrix} 5.0 & 0.0 \\ 0.0 & 5.0 \end{bmatrix}^{-1} \begin{bmatrix} -7.0 \\ 5.0 \end{bmatrix}$$
$$= \begin{bmatrix} 1.4 \\ -1.0 \end{bmatrix}$$

However, there is no restriction in constraining the movement of point $p_{i,j}(\mathbf{x})$ to a restricted region in the objective function. This may result in the cause of cell skewness where $p_{i,j}(\mathbf{x})$ lies outside the marked area (dashed lines in Fig. 9.6). Control functions are required to prevent cell skewness or cell shear.

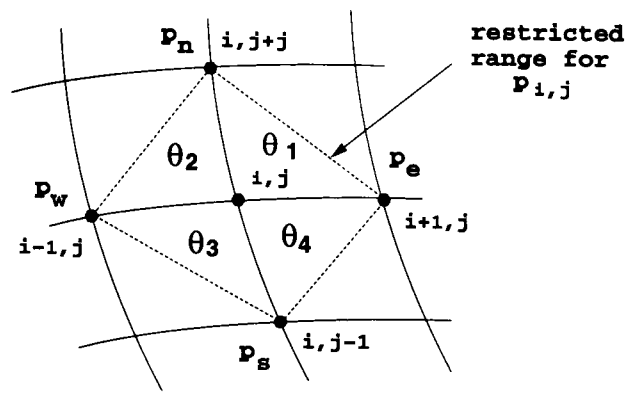


Figure 9.6: **The dashed line represents the available region for the movement of the grid point $p_{i,j}(\mathbf{x})$.**

In general, grid quality is measured by orthogonality, aspect ratio and smoothness. Various methods for grid quality control were discussed in [26], [27], [58], [60], [66] and [94]. In the following, three control functions for orthogonality, cell aspect ratio and smoothness are used as penalty functions to improve adaptive grid quality.

9.1.3.3 Control Function for Orthogonality

The control function for orthogonality serves as a penalty function to avoid cell skewness. In two–dimensional case, four angles intersected by two families of grid lines through the point $p_{i,j}(\mathbf{x})$ are denoted by $\theta_1, \ldots, \theta_4$, shown in Fig. 9.6. The control function for orthogonality is formulated by

$$f^o_{i,j}(\mathbf{x}) = cos^2\theta_1 + cos^2\theta_2 + cos^2\theta_3 + cos^2\theta_4 \tag{9.17}$$

where the subscript o denotes *orthogonality*. This equation is rewritten as

$$\begin{aligned} f^{o}_{i,j}(\mathbf{x}) = & \left[\frac{(p_{i+1,j}(\mathbf{x}) - p_{i,j}(\mathbf{x})) \cdot (p_{i,j+1}(\mathbf{x}) - p_{i,j}(\mathbf{x}))}{|p_{i+1,j}(\mathbf{x}) - p_{i,j}(\mathbf{x})| \cdot |p_{i,j+1}(\mathbf{x}) - p_{i,j}(\mathbf{x})|}\right]^2 + \\ & \left[\frac{(p_{i,j+1}(\mathbf{x}) - p_{i,j}(\mathbf{x})) \cdot (p_{i-1,j}(\mathbf{x}) - p_{i,j}(\mathbf{x}))}{|p_{i,j+1}(\mathbf{x}) - p_{i,j}(\mathbf{x})| \cdot |p_{i-1,j}(\mathbf{x}) - p_{i,j}(\mathbf{x})|}\right]^2 + \\ & \left[\frac{(p_{i-1,j}(\mathbf{x}) - p_{i,j}(\mathbf{x})) \cdot (p_{i,j-1}(\mathbf{x}) - p_{i,j}(\mathbf{x}))}{|p_{i-1,j}(\mathbf{x}) - p_{i,j}(\mathbf{x})| \cdot |p_{i,j-1}(\mathbf{x}) - p_{i,j}(\mathbf{x})|}\right]^2 + \\ & \left[\frac{(p_{i,j-1}(\mathbf{x}) - p_{i,j}(\mathbf{x})) \cdot (p_{i+1,j}(\mathbf{x}) - p_{i,j}(\mathbf{x}))}{|p_{i,j-1}(\mathbf{x}) - p_{i,j}(\mathbf{x})| \cdot |p_{i+1,j}(\mathbf{x}) - p_{i,j}(\mathbf{x})|}\right]^2 \end{aligned} \tag{9.18}$$

In case of a control term with higher order, such as the control term for cell orthogonality, a search process using Newton method could be expensive due to the computation of an inverse Hessian matrix.

The method used for minimization of an objective function contains the following steps:

▷ **Step 1**. Set a termination criterion ϵ, for instance $\epsilon = 10^{-13}$, and select starting points $\mathbf{x}^{(0)}$ for the first search. The gradient of the objective function $f(\mathbf{x})$ is computed by

$$\mathbf{g}^{(n)} = \nabla f(\mathbf{x}^{(n)}) \tag{9.19}$$

where $\mathbf{g}$ denotes *gradient*, and the superscript n denotes the iteration number. Let $n=0$, and the direction of the negative gradient of the grid point at its initial position be the search direction, i.e.

$$\mathbf{S}^{(0)} = -\mathbf{g}^{(0)} \tag{9.20}$$

where $\mathbf{S}$ denotes *search direction*.

▷ **Step 2**. Compute the Hessian matrix $\mathbf{H}[(\mathbf{x}^{(n)})]$ of the control function, and determine the optimal step length $\alpha^{(n)}$ using the following equation

$$\alpha^{(n)} = -\frac{[\nabla f(\mathbf{x}^{(n)})]^T \mathbf{S}^{(n)}}{[\mathbf{S}^{(n)}]^T [\mathbf{H}(\mathbf{x}^n)] \mathbf{S}^{(n)}} \tag{9.21}$$

▷ **Step 3**. A set of new points $\mathbf{x}^{(n+1)}$ is calculated by

$$\mathbf{x}^{(n+1)} = \mathbf{x}^{(n)} + \alpha^{(n)} \mathbf{S}^{(n)} \tag{9.22}$$

▷ **Step 4**. Calculate the gradient with the new points using Eq. (9.19), and check the termination criterion

$$|\mathbf{g}^{(n+1)}|^2 \leq \epsilon \tag{9.23}$$

If this condition is satisfied, the search process is terminated. Else:

▷ **Step 5**. The search direction is corrected by

$$\beta^{(n)} = \frac{|\mathbf{g}^{(n+1)}|^2}{|\mathbf{g}^{(n)}|^2} \tag{9.24}$$

where $\beta^{(n)}$ is the correct factor for an optimal search direction in step $n+1$.

▷ **Step 6**. The next search direction is determined by

$$\mathbf{S}^{(n+1)} = -\mathbf{g}^{(n+1)} + \beta^{(n)}\mathbf{g}^{(n)} \tag{9.25}$$

The iteration turns to the **Step 2**, i.e., the next computation begins with determination of the search step.

Step 5 computes a correct factor for the further search direction. This method is found in [32], termed *conjugate gradient method.* Both loops avoid to compute inverse Hessian matrix of an objective function.

Test case for the control function for orthogonality. Using the initial guess estimates $\mathbf{x}^{(0)} = [1.5, -0.25]^T$ and $\mathbf{x}^{(0)} = [1.2, 0]^T$, the minimization processes needs 5 and 8 search steps, respectively, as shown in Tables 9.1 to 9.2. From the initial guess $\mathbf{x}^{(0)} = [10^4, 10^4]^T$, the minimum can not be found.

Table 9.1: Numerical solution of the control function for orthogonality with the initial guess estimate $\mathbf{x}^{(0)} = [1.5, -0.25]^T$.

n	$x^{(n)}$	$y^{(n)}$	$\theta_1^{(n)}[\,^o]$	$\theta_2^{(n)}[\,^o]$	$\theta_3^{(n)}[\,^o]$	$\theta_4^{(n)}[\,^o]$
1	1.26868	–0.754816	87.0391	90.9936	93.2035	88.7638
2	1.37904	–0.733368	89.8976	89.9361	90.0137	90.1526
3	1.37878	–0.729715	90.0088	89.9951	89.9892	90.0070
4	1.37838	–0.729707	90.0007	90.0003	89.9998	89.9992
5	1.37838	–0.729730	90.0000	90.0000	90.0000	90.0000
6	1.37838	–0.729730	90.0000	90.0000	90.0000	90.0000

Table 9.2: Numerical solution of the control function for orthogonality with the initial guess estimate $\mathbf{x}^{(0)} = [1.2, 0]^T$.

n	$x^{(n)}$	$y^{(n)}$	$\theta_1^{(n)}[^o]$	$\theta_2^{(n)}[^o]$	$\theta_3^{(n)}[^o]$	$\theta_4^{(n)}[^o]$
1	1.07072	–1.214830	71.2127	85.8169	104.348	98.6223
2	1.36196	–1.118970	77.0406	84.4175	93.4958	105.046
3	1.37231	–0.689526	91.1441	90.6932	89.8079	88.3548
4	1.37595	–0.730754	89.9172	90.0155	90.0737	89.9936
5	1.37803	–0.729406	90.0031	90.0094	90.0065	89.9810
6	1.37836	–0.729739	89.9994	90.0001	90.0005	90.0001
7	1.37837	–0.729728	90.0000	90.0001	90.0001	89.9999
8	1.37838	–0.729730	90.0000	90.0000	90.0000	90.0000

The test cases show that the initial guess influences the convergence of the numerical solutions. In order to ensure a convergent solution of objective functions, initial guess estimates can be chosen in their restricted ranges. To improve the convergence behavior, other control functions are needed.

9.1.3.4 Control Function for Cell Aspect Ratio

Consider an algebraic relation

$$\frac{s_1^2 + s_2^2}{2 s_1 s_2} \geq 1 \tag{9.26}$$

where $s_1, s_2 > 0$. In case the sum of s_1 and s_2 is constant, the minimum is obtained by $s_1 = s_2$. This relation is extended to measuring the cell aspect ratio. Substituting s_1 and s_2 by two pieces of neighboring line segments (see Fig. 9.6), the control function for the cell aspect ratio is constructed as follows

$$f_{i,j}^{l}(\mathbf{x}) = \quad a_1 + a_2 + a_3 + a_4 \tag{9.27}$$

where a_i denotes the *aspect ratio*, superscript l denotes *length* control, and

$$\begin{aligned}
a_1 &= \frac{|p_{i+1,j}(\mathbf{x}) - p_{i,j}(\mathbf{x})|^2 + |p_{i,j+1}(\mathbf{x}) - p_{i,j}(\mathbf{x})|^2}{2|p_{i+1,j}(\mathbf{x}) - p_{i,j}(\mathbf{x})| \cdot |p_{i,j+1}(\mathbf{x}) - p_{i,j}(\mathbf{x})|} \\
a_2 &= \frac{|p_{i,j+1}(\mathbf{x}) - p_{i,j}(\mathbf{x})|^2 + |p_{i-1,j}(\mathbf{x}) - p_{i,j}(\mathbf{x})|^2}{2|p_{i,j+1}(\mathbf{x}) - p_{i,j}(\mathbf{x})| \cdot |p_{i-1,j}(\mathbf{x}) - p_{i,j}(\mathbf{x})|} \\
a_3 &= \frac{|p_{i-1,j}(\mathbf{x}) - p_{i,j}(\mathbf{x})|^2 + |p_{i,j-1}(\mathbf{x}) - p_{i,j}(\mathbf{x})|^2}{2|p_{i-1,j}(\mathbf{x}) - p_{i,j}(\mathbf{x})| \cdot |p_{i,j-1}(\mathbf{x}) - p_{i,j}(\mathbf{x})|} \\
a_4 &= \frac{|p_{i,j-1}(\mathbf{x}) - p_{i,j}(\mathbf{x})|^2 + |p_{i+1,j}(\mathbf{x}) - p_{i,j}(\mathbf{x})|^2}{2|p_{i,j-1}(\mathbf{x}) - p_{i,j}(\mathbf{x})| \cdot |p_{i+1,j}(\mathbf{x}) - p_{i,j}(\mathbf{x})|}
\end{aligned} \tag{9.28}$$

Test case for the control function for cell aspect ratio. The same initial guess as in the previous example is selected to compare the efficiency of the control function. The results are shown in Tables 9.3 and 9.4. Minimizing the control function requires a good initial solution. Using the initial guess $\mathbf{x}^{(0)} = [10^4, 10^4]^T$, the minimum of the penalty function can not be found.

Table 9.3: Numerical solution of the control function for cell aspect ratio using the initial guess $\mathbf{x}^{(0)} = [1.5, -0.25]^T$.

n	$x^{(n)}$	$y^{(n)}$	AR_1	AR_2	AR_3	AR_4
1	0.812292	0.123764	1.000000	1.097836	1.019649	1.029158
2	0.244614	0.322348	1.000000	1.002095	1.000967	1.005911
3	0.189239	0.296000	1.000000	1.000586	1.001683	1.004257
4	0.195155	0.279429	1.000000	1.000784	1.001325	1.004148
5	0.196160	0.279399	1.000000	1.000808	1.001305	1.004168
6	0.196119	0.279459	1.000000	1.000807	1.001306	1.004168
7	0.196127	0.279467	1.000000	1.000807	1.001306	1.004169

Table 9.4: Numerical solution of the control function for cell aspect ratio using the initial guess $\mathbf{x}^{(0)} = [0, 0]^T$.

n	$x^{(n)}$	$y^{(n)}$	AR_1	AR_2	AR_3	AR_4
1	0.252320	0.189525	1.000000	1.003565	1.000015	1.004050
2	0.207643	0.280827	1.000000	1.001104	1.001109	1.004429
3	0.195715	0.281237	1.000000	1.000789	1.001338	1.004185
4	0.196112	0.279531	1.000000	1.000806	1.001307	1.004169
5	0.196132	0.279473	1.000000	1.000807	1.001306	1.004169
6	0.196126	0.279468	1.000000	1.000807	1.001306	1.004169

The results from both control functions satisfy the requirements for grid quality control. However, the critical selection of starting search points restricts their practical application. Therefore, an additional penalty function is introduced.

9.1.3.5 Control Function of Cell Smoothness

The property of grid smoothness is measured by the continuous changes of grid spacing along each computational coordinate direction [28]. In the present work, the numerical solution of the objective function is based on a five–point stencil for two–dimensional cases. Therefore, grid smoothness is locally evaluated locally for the five–point stencil.

The derivation of the local smoothness of a five–point stencil is based on the following assumption. Suppose that five grid points have weights, holding them in an equilibrium. The central grid point $p_{i,j}(\mathbf{x})$ has a negative weight $-\overline{\omega}$, while its four neighbors have the positive weights $\frac{\overline{\omega}}{4}$. This configuration is stationary, because the sum of all weights is zero. The condition of the equilibrium for this five–point system is

$$p^*_{i,j}(\mathbf{x}) = \frac{\frac{1}{4}\overline{\omega}p_{i,j-1}(\mathbf{x}) + \frac{1}{4}\overline{\omega}p_{i-1,j}(\mathbf{x}) + \frac{1}{4}\overline{\omega}p_{i+1,j}(\mathbf{x}) + \frac{1}{4}\overline{\omega}p_{i,j+1}(\mathbf{x}) - \overline{\omega}p_{i,j}(\mathbf{x})}{\frac{1}{4}\overline{\omega} + \frac{1}{4}\overline{\omega} + \frac{1}{4}\overline{\omega} + \frac{1}{4}\overline{\omega} + \overline{\omega}} \tag{9.29}$$

where $p^*_{i,j}$ denotes the equilibrium position of grid point $p_{i,j}(\mathbf{x})$ in this system. Note that these weights are different from a weight function used for grid adaptation.

The quadratic expression of the above equation is used as the control function for cell smoothness. It has the following form

$$f^s_{i,j}(\mathbf{x}) = \left(\frac{p_{i-1,j}(\mathbf{x}) + p_{i+1,j}(\mathbf{x}) + p_{i,j-1}(\mathbf{x}) + p_{i,j+1}(\mathbf{x}) - 4p_{i,j}(\mathbf{x})}{8} \right)^2 \tag{9.30}$$

where superscript s denotes *smoothness*. The function serves as a penalty function to ensure that the minimum of the objective function will be found when the search is within the restricted region of grid point movement.

Test case for the control function for cell smoothness. The same initial guess estimates as in the previous example are chosen. They are $\mathbf{x}^{(0)} = [1.5, -0.25]^T$, $\mathbf{x}^{(0)} = [0, 0]^T$, and $\mathbf{x}^{(0)} = [10^4, 10^4]^T$. The numerical solutions of the quadratic equation are obtained in one step, independent of the initial conditions, as shown in Tables 9.5 to 9.7.

Table 9.5: Minimum solution of the control function for cell smoothness using the initial guess $\mathbf{x}^{(0)} = [1.5 - 0.25]^T$.

n	$x^{(n)}$	$y^{(n)}$	$\theta_1^{(n)}[\,^o]$	$\theta_2^{(n)}[\,^o]$	$\theta_3^{(n)}[\,^o]$	$\theta_4^{(n)}[\,^o]$
1	0.75	–0.25	87.3974	107.223	99.0085	66.3706

Table 9.6: Minimum solution for the control function of cell smoothness using the initial guess $\mathbf{x}^{(0)} = [0, 0]^T$.

n	$x^{(n)}$	$y^{(n)}$	$\theta_1^{(n)}[\,^o]$	$\theta_2^{(n)}[\,^o]$	$\theta_3^{(n)}[\,^o]$	$\theta_4^{(n)}[\,^o]$
1	0.75	–0.25	87.3974	107.223	99.0085	66.3706

Table 9.7: Minimum solution for the control function of cell smoothness using the initial guess $\mathbf{x}^{(0)} = [10^4, 10^4]^T$.

n	$x^{(n)}$	$y^{(n)}$	$\theta_1^{(n)}[\ ^o]$	$\theta_2^{(n)}[\ ^o]$	$\theta_3^{(n)}[\ ^o]$	$\theta_4^{(n)}[\ ^o]$
1	0.75	–0.25	87.3974	107.223	99.0085	66.3706

9.1.3.6 Objective Function for Grid Quality Control

It is assumed that the final objective function for grid adaptation by control functions is a combination of all control functions f^o, f^l, and f^s. In two–dimensional cases, the final objective function for adapting an internal point $p_{i,j}(\mathbf{x})$ has the following form

$$f_{i,j}(\mathbf{x}) = f^a_{i,j}(\mathbf{x}) + \gamma_1 f^o_{i,j}(\mathbf{x}) + \gamma_2 f^l_{i,j}(\mathbf{x}) + \gamma_3 f^s_{i,j}(\mathbf{x}) \tag{9.31}$$

where γ_1, γ_2, and γ_3 are penalty factors. In grid adaptation process, the values of these factors are increased with iteration number, i.e., $\gamma_i^{(0)} < \gamma_i^{(1)}, \ldots, < \gamma_i^{(n)}$. This slows down the search, when a grid point is close to the minimum.

In the previous sections, the objective function $f^a(\mathbf{x})$ and the three control functions $f^o(\mathbf{x})$, $f^l(\mathbf{x})$, and $f^s(\mathbf{x})$ are separately tested. The same example is used to test the final objective function. Four intersection angles are listed, because the adaptive grid quality can be more intuitively evaluated by this measure than by other measures.

Test case for the final objective function. Using the initial guess estimates $\mathbf{x}^{(0)} = [1.5, -0.25]^T$, $\mathbf{x}^{(0)} = [0, 0]^T$ and $\mathbf{x}^{(0)} = [10^{13}, -10^{13}]^T$, the final objective function is minimized. The results are shown in Tables 9.8 to 9.10.

Table 9.8: Minimum solution for the final objective function using the initial guess $\mathbf{x}^{(0)} = [1.5, -0.25]^T$.

n	$x^{(n)}$	$y^{(n)}$	$\theta_1^{(n)}[\ ^o]$	$\theta_2^{(n)}[\ ^o]$	$\theta_3^{(n)}[\ ^o]$	$\theta_4^{(n)}[\ ^o]$
1	1.22514	–0.832209	83.9697	90.2882	95.2133	90.5288
2	1.24167	–0.820501	84.6022	90.2787	94.6206	90.4985
3	1.24116	–0.820125	84.6042	90.2908	94.6314	90.4737
4	1.24116	–0.820142	84.6036	90.2905	94.6317	90.4742
5	1.24116	–0.820142	84.6036	90.2905	94.6316	90.4742

In the final objective function, initial guess estimates do not play an important role in convergence of solution. This property enables an implementation for

Table 9.9: Minimum solution for the final objective function using the initial guess $\mathbf{x}^{(0)} = [0, 0]^T$.

n	$x^{(n)}$	$y^{(n)}$	$\theta_1^{(n)}[\,^o]$	$\theta_2^{(n)}[\,^o]$	$\theta_3^{(n)}[\,^o]$	$\theta_4^{(n)}[\,^o]$
1	1.29604	–0.851293	84.6292	89.1501	93.3637	92.8570
2	1.24078	–0.820154	84.5965	90.2949	94.6426	90.4660
3	1.24116	—0.820147	84.6035	90.2904	94.6316	90.4745
4	1.24116	–0.820142	84.6036	90.2905	94.6317	90.4742
5	1.24116	–0.820142	84.6036	90.2905	94.6316	90.4742

Table 9.10: Minimum solution for the final objective function using the initial guess $\mathbf{x}^{(0)} = [10^{13}, -10^{13}]^T$.

n	$x^{(n)}$	$y^{(n)}$	$\theta_1^{(n)}[\,^o]$	$\theta_2^{(n)}[\,^o]$	$\theta_3^{(n)}[\,^o]$	$\theta_4^{(n)}[\,^o]$
1	1.29722	–0.880554	83.7474	88.6849	93.5976	93.9702
2	1.24019	–0.820529	84.5751	90.2959	94.6631	90.4659
3	1.24117	–0.820145	84.6037	90.2904	94.6313	90.4746
4	1.24116	–0.820142	84.6036	90.2905	94.6316	90.4742
5	1.24116	–0.820142	84.6036	90.2905	94.6316	90.4742

grid adaptation of single blocks.

9.1.3.7 Single–Block Grid Adaptation

In the derivation of the objective function for multi–dimensional grid adaptation, the physical space is used, since the coordinate transformation in this stage is not regarded. Later on, the computation is performed in parameter space in a dimensionless manner. In the following, two–dimensional grid adaptation is explained.

In parameter space, a grid is given by

$$\hat{P} = \{\hat{p}_{i,j}(\mathbf{u}) \mid i = 1, \ldots, I; j = 1, \ldots, J\} \tag{9.32}$$

where the symbol " ˆ " denotes the parameter space, and $\mathbf{u} = (u, v)$. Specifying the index (i, j), the parameter coordinates are obtained from Eq. (9.1).

Grid points in the solution domain are represented by

$$\hat{P} = \{\hat{p}_{i,j}(\mathbf{u}) \mid i = 2, \ldots, I-2; j = 2, \ldots, J-2\} \tag{9.33}$$

Grid points on block boundaries are considered stationary. The first and second partial derivatives with respect to the coordinates at these grid points vanish.

The objective function is written in stencil form for each interior grid point. A stencil objective function can be written in the following forms:

- ▷ **Stencil objective function of the first type**. In this case, the central grid point $p_{i,j}(\mathbf{u})$ is adapted with regard to the influences of the weight functions of its four neighboring points. The first and second partial derivatives of an objective function with respect to the coordinates of these four grid points vanish.
- ▷ **Stencil objective function of the second type**. In this case, the central grid point $p_{i,j}(\mathbf{u})$ is adapted regarding the influences of both weight functions and coordinates of its four neighboring points.

In the solution domain, there are $(I-2)\times(J-2)$ objective functions to be minimized in total. The grid is adapted in the parameter space using

$$\mathbf{u}^{(n+1)} = \mathbf{u}^{(n)} + \alpha^{(n)}\mathbf{S}^{(n)} \tag{9.34}$$

Solution of the objective function is obtained explicitly in a successive manner. For instance, after one search process at a grid point $\hat{p}_{i,j}(\mathbf{u})$, the search process turns to the next grid point. The second search process starts, when the first search process is finished for all interior grid points in the solution domain.

After one search process, the grid point $\hat{p}_{i,j}(\mathbf{u})$ changes to its new position $\overline{p}_{i,j}(\mathbf{u})$ in parameter space. The new position influences the search direction as well as the search step of the next iteration. In order to prevent a large jump of grid point movement from one search step to the next one, the adaptive position is relaxed by

$$\hat{p}_{i,j}^{(k+1)}(\mathbf{u}) = \hat{p}_{i,j}^{(k)}(\mathbf{u}) + \omega\left[\overline{p}_{i,j}^{(k+1)}(\mathbf{u}) - \hat{p}_{i,j}^{(k)}(\mathbf{u})\right] \tag{9.35}$$

where the relaxation factor is chosen by $0 < \omega < 1$.

The actual state (i.e. position and associated weight functions) of the grid point in the physical space can not be represented by its initial position $p_{i,j}^{(k)}(\mathbf{x})$ and weight function $w_{i,j}^{(k)}(\mathbf{x})$. It is necessary to use the updated values for the next iteration. Using the differential relations, one obtains the new values

$$
\begin{aligned}
x_{i,j}^{(k+1)} &= x_{i,j}^{(k)} + \frac{\Delta x_{i,j}^{(k)}}{\Delta u_{i,j}^{(k)}}(u_{i,j}^{(k+1)} - u_{i,j}^{(k)}) + \frac{\Delta x_{i,j}^{(k)}}{\Delta v_{i,j}^{(k)}}(v_{i,j}^{(k+1)} - v_{i,j}^{(k)}) \\
y_{i,j}^{(k+1)} &= y_{i,j}^{(k)} + \frac{\Delta y_{i,j}^{(k)}}{\Delta u_{i,j}^{(k)}}(u_{i,j}^{(k+1)} - u_{i,j}^{(k)}) + \frac{\Delta y_{i,j}^{(k)}}{\Delta v_{i,j}^{(k)}}(v_{i,j}^{(k+1)} - v_{i,j}^{(k)}) \\
w_{i,j}^{(k+1)} &= w_{i,j}^{(k)} + \frac{\Delta w_{i,j}^{(k)}}{\Delta u_{i,j}^{(k)}}(u_{i,j}^{(k+1)} - u_{i,j}^{(k)}) + \frac{\Delta w_{i,j}^{(k)}}{\Delta v_{i,j}^{(k)}}(v_{i,j}^{(k+1)} - v_{i,j}^{(k)})
\end{aligned} \tag{9.36}
$$

where

$$
\begin{aligned}
\frac{\Delta x_{i,j}}{\Delta u_{i,j}} &= \frac{x_{i+1,j} - x_{i-1,j}}{u_{i+1,j} - u_{i-1,j}} \\
\frac{\Delta y_{i,j}}{\Delta u_{i,j}} &= \frac{y_{i+1,j} - y_{i-1,j}}{u_{i+1,j} - u_{i-1,j}} \\
\frac{\Delta w_{i,j}}{\Delta u_{i,j}} &= \frac{w_{i+1,j} - w_{i-1,j}}{u_{i+1,j} - u_{i-1,j}} \\
\frac{\Delta x_{i,j}}{\Delta v_{i,j}} &= \frac{x_{i,j+1} - x_{i,j-1}}{v_{i,j+1} - v_{i,j-1}} \\
\frac{\Delta y_{i,j}}{\Delta v_{i,j}} &= \frac{y_{i,j+1} - y_{i,j-1}}{v_{i,j+1} - v_{i,j-1}} \\
\frac{\Delta w_{i,j}}{\Delta v_{i,j}} &= \frac{w_{i,j+1} - x_{i,j-1}}{v_{i,j+1} - v_{i,j-1}}
\end{aligned} \tag{9.37}
$$

A stencil objective function of the first type does not describe the relations of coordinate coupling among a five–point system. The equation system can be solved using the Newton method. A stencil objective function of the second type describes the coupling relations among five grid points. The methods for solution of this equation system are explained in the following.

The first method for the solution of objective functions. Objective functions are solved in parameter space. An objective function for point $\hat{p}_{i,j}(\mathbf{u})$ has the form

$$
f_{i,j}(\mathbf{u}) = (u_1, \ldots, u_n) \tag{9.38}
$$

In a two–dimensional solution domain, the interior grid points are ordered in a natural sequence, i.e., $(\hat{p}_{2,2}(\mathbf{u}), ..\hat{p}_{I-2,2}(\mathbf{u}), \hat{p}_{2,3}(\mathbf{u}), ..\hat{p}_{I-2,J-2}(\mathbf{u}))$. as shown in Fig. 9.7. Using the notation s, w, e and n for the *south*, west. east. and north neighboring points of an interior grid point $\hat{p}_{i,j}(\mathbf{u})$. the second partial derivatives

of an objective function is ordered in a Hessian matrix in the form shown in Fig. 9.7.

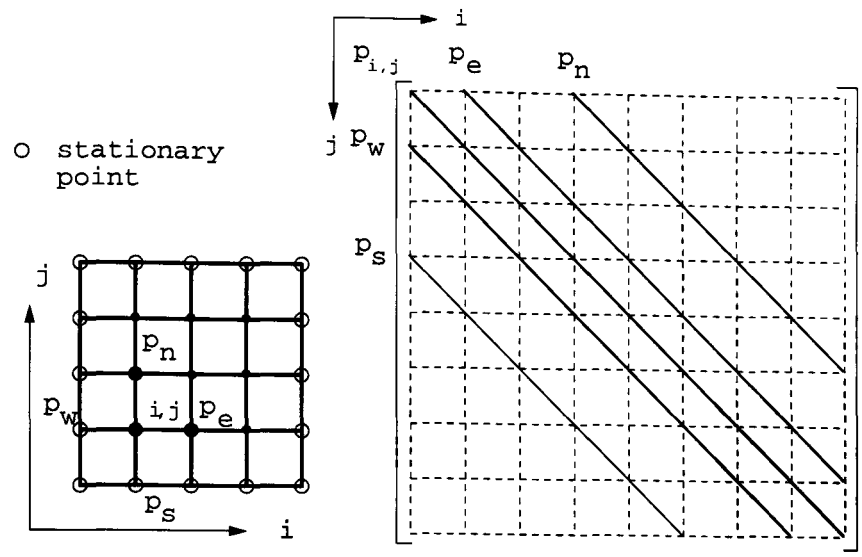


Figure 9.7: **Using the first method for solution of an objective function, the second derivatives can be ordered in the Hessian matrix as shown.**

The search direction and step are determined using all interior grid points, Therefore, this method belongs to an *implicit* method. However, there are some unsolved issues using this solution method. First, the search step $\alpha^{(k)}$ of the k–th iteration for this system is computed by Eq. (9.21). It is denoted as *global step*, and is used for all interior grid points. A global search step as well as a global search direction are dependent upon the number of grid points. Second, this method is expensive because of its large memory demand. Having $M \times N$ interior grid points, about $5 \times (2 \times M \times N)$ elements are stored in the Hessian matrix.

Test case using the first method. The test grid has 41×41 points and represents a square. The analytical weight function describes a circle in the square. The solution domain has 39×39 grid points. Eq. (9.8) with a control term for cell smoothness is used. A global search step is computed by

$$\alpha_g = c_1 \frac{\alpha}{number\ of\ grid\ points} \tag{9.39}$$

where α and α_g denote *local* and *global* search steps, respectively, and c_1 is the coefficient to vary the magnitude of the search step. In this example, $c_1 = 50$ is chosen. In practice, the relation between search step and number of grid points is more complicated. Therefore, this solution method is not used in the later

examples. Fig. 9.8 shows the result after one search step.

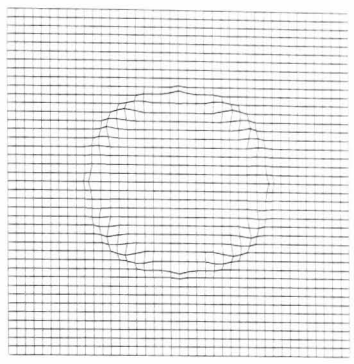


Figure 9.8: **The objective function with control term for cell smoothness is minimized in one step.**

The second method for the solution of objective functions. An arbitrary interior point $\hat{p}_{i,j}(\mathbf{u})$ is coupled with its four neighboring points $\hat{p}_{i,j-1}(\mathbf{u})$, $\hat{p}_{i-1,j}(\mathbf{u})$, $\hat{p}_{i+1,j}(\mathbf{u})$ and $\hat{p}_{i,j+1}(\mathbf{u})$. To simplify the derivation, these five points are numbered and denoted by $\hat{p}_1(\mathbf{u})$, $\hat{p}_2(\mathbf{u})$, $\hat{p}_3(\mathbf{u})$, $\hat{p}_4(\mathbf{u})$, and $\hat{p}_5(\mathbf{u})$ for the grid points $\hat{p}_{i,j-1}(\mathbf{u})$, $\hat{p}_{i-1,j}(\mathbf{u})$, $\hat{p}_{i,j}(\mathbf{u})$, $\hat{p}_{i+1,j}(\mathbf{u})$ and $\hat{p}_{i,j+1}(\mathbf{u})$, respectively, as shown in Fig. 9.3. Neglecting the influences of grid points that are not coupled with the grid point $\hat{p}_{i,j}(\mathbf{u})$, search directions and steps are determined locally for a five–point stencil.

In computation of a local search direction and step for an interior grid point, its four neighbors are considered as stationary points. Therefore, this method belongs to an *explicit* method.

In the test examples using control functions for both orthogonality and cell aspect ratio (see section 9.1.3.6, page 138), four neighboring grid points are stationary. In this case, the control functions for cell orthogonality and aspect ratio are expressed by Eq. (9.17) (only for one angle) and Eq. (9.26). The objective function has the form

$$f_{i,j}(u_1, u_2, u_3, u_4, u_5, v_1, v_2, v_3, v_4, v_5, \hat{\theta}, \hat{s}_1, \hat{s}_2)$$

where $\hat{\theta}$, $\hat{s}_1$ and $\hat{s}_2$ are dependent variables, and are expressed by

$$cos\hat{\theta} = \frac{(\hat{p}_{i+1,j}(\mathbf{u}) - \hat{p}_{i-1,j}(\mathbf{u})) \cdot (\hat{p}_{i,j+1}(\mathbf{u}) - \hat{p}_{i,j-1}(\mathbf{u}))}{|\hat{p}_{i+1,j}(\mathbf{u}) - \hat{p}_{i-1,j}(\mathbf{u})| \cdot |\hat{p}_{i,j+1}(\mathbf{u}) - \hat{p}_{i,j-1}(\mathbf{u})|} \tag{9.40}$$

and

$$\hat{s}_1 = |\hat{p}_{i,j}(\mathbf{u}) - \hat{p}_{i-1,j}(\mathbf{u})| + |\hat{p}_{i+1,j}(\mathbf{u}) - \hat{p}_{i,j}(\mathbf{u})|$$
$$\hat{s}_2 = |\hat{p}_{i,j}(\mathbf{u}) - \hat{p}_{i,j-1}(\mathbf{u})| + |\hat{p}_{i,j+1}(\mathbf{u}) - \hat{p}_{i,j}(\mathbf{u})| \tag{9.41}$$

These relations are depicted in Fig. 9.9.

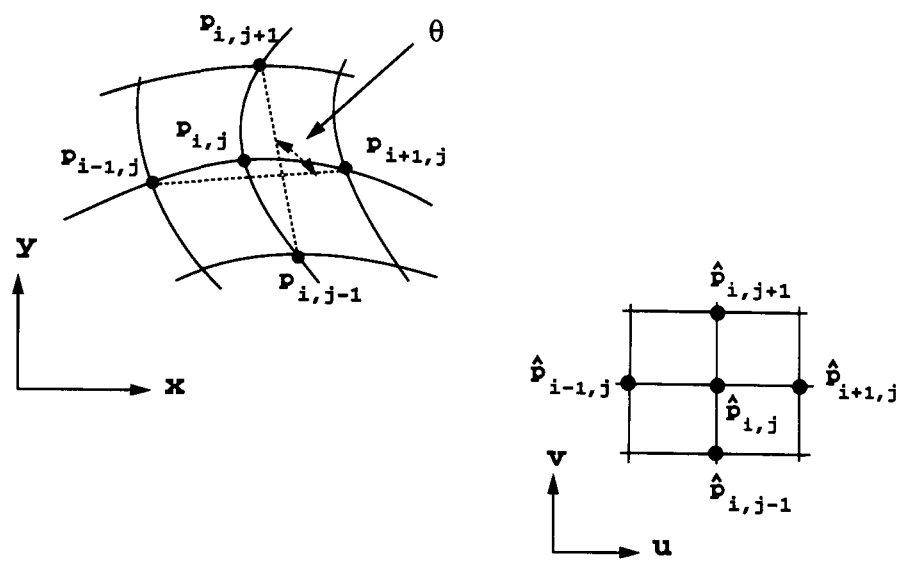


Figure 9.9: **Cell orthogonality and aspect ratio are measured by the intersection θ and length relation of s_1 and s_2.**

Example of solution of the system of equations. This example explains the solution process of the objective function with smoothness control. The first partial derivatives of the objective function with respect to parameter coordinates are written in the form of a row vector

$$\nabla f(\mathbf{u}) = \left[\frac{\partial f(\mathbf{u})}{\partial u_1}, \ldots, \frac{\partial f(\mathbf{u})}{\partial u_5}, \frac{\partial f(\mathbf{u})}{\partial v_1}, \ldots, \frac{\partial f(\mathbf{u})}{\partial v_5}\right] \tag{9.42}$$

The second partial derivatives of the objective function with respect to parameter coordinates are written in a 10×10 Hessian matrix

$$H(\mathbf{u})=\begin{bmatrix}
\frac{\partial^2 f(\mathbf{u})}{\partial u_1^2} & \frac{\partial^2 f(\mathbf{u})}{\partial u_1 \partial u_2} & \frac{\partial^2 f(\mathbf{u})}{\partial u_1 \partial u_3} & \frac{\partial^2 f(\mathbf{u})}{\partial u_1 \partial u_4} & \frac{\partial^2 f(\mathbf{u})}{\partial u_1 \partial u_5} & \frac{\partial^2 f(\mathbf{u})}{\partial u_1 \partial v_1} & \frac{\partial^2 f(\mathbf{u})}{\partial u_1 \partial v_2} & \frac{\partial^2 f(\mathbf{u})}{\partial u_1 \partial v_3} & \frac{\partial^2 f(\mathbf{u})}{\partial u_1 \partial v_4} & \frac{\partial^2 f(\mathbf{u})}{\partial u_1 \partial v_5} \\
\frac{\partial^2 f(\mathbf{u})}{\partial u_2 \partial u_1} & \frac{\partial^2 f(\mathbf{u})}{\partial u_2^2} & \frac{\partial^2 f(\mathbf{u})}{\partial u_2 \partial u_3} & \frac{\partial^2 f(\mathbf{u})}{\partial u_2 \partial u_4} & \frac{\partial^2 f(\mathbf{u})}{\partial u_2 \partial u_5} & \frac{\partial^2 f(\mathbf{u})}{\partial u_2 \partial v_1} & \frac{\partial^2 f(\mathbf{u})}{\partial u_2 \partial v_2} & \frac{\partial^2 f(\mathbf{u})}{\partial u_2 \partial v_3} & \frac{\partial^2 f(\mathbf{u})}{\partial u_2 \partial v_4} & \frac{\partial^2 f(\mathbf{u})}{\partial u_2 \partial v_5} \\
\frac{\partial^2 f(\mathbf{u})}{\partial u_3 \partial u_1} & \frac{\partial^2 f(\mathbf{u})}{\partial u_3 \partial u_2} & \frac{\partial^2 f(\mathbf{u})}{\partial u_3^2} & \frac{\partial^2 f(\mathbf{u})}{\partial u_3 \partial u_4} & \frac{\partial^2 f(\mathbf{u})}{\partial u_3 \partial u_5} & \frac{\partial^2 f(\mathbf{u})}{\partial u_3 \partial v_1} & \frac{\partial^2 f(\mathbf{u})}{\partial u_3 \partial v_2} & \frac{\partial^2 f(\mathbf{u})}{\partial u_3 \partial v_3} & \frac{\partial^2 f(\mathbf{u})}{\partial u_3 \partial v_4} & \frac{\partial^2 f(\mathbf{u})}{\partial u_3 \partial v_5} \\
\frac{\partial^2 f(\mathbf{u})}{\partial u_4 \partial u_1} & \frac{\partial^2 f(\mathbf{u})}{\partial u_4 \partial u_2} & \frac{\partial^2 f(\mathbf{u})}{\partial u_4 \partial u_3} & \frac{\partial^2 f(\mathbf{u})}{\partial u_4^2} & \frac{\partial^2 f(\mathbf{u})}{\partial u_4 \partial u_5} & \frac{\partial^2 f(\mathbf{u})}{\partial u_4 \partial v_1} & \frac{\partial^2 f(\mathbf{u})}{\partial u_4 \partial v_2} & \frac{\partial^2 f(\mathbf{u})}{\partial u_4 \partial v_3} & \frac{\partial^2 f(\mathbf{u})}{\partial u_4 \partial v_4} & \frac{\partial^2 f(\mathbf{u})}{\partial u_4 \partial v_5} \\
\frac{\partial^2 f(\mathbf{u})}{\partial u_5 \partial u_1} & \frac{\partial^2 f(\mathbf{u})}{\partial u_5 \partial u_2} & \frac{\partial^2 f(\mathbf{u})}{\partial u_5 \partial u_3} & \frac{\partial^2 f(\mathbf{u})}{\partial u_5 \partial u_4} & \frac{\partial^2 f(\mathbf{u})}{\partial u_5^2} & \frac{\partial^2 f(\mathbf{u})}{\partial u_5 \partial v_1} & \frac{\partial^2 f(\mathbf{u})}{\partial u_5 \partial v_2} & \frac{\partial^2 f(\mathbf{u})}{\partial u_5 \partial v_3} & \frac{\partial^2 f(\mathbf{u})}{\partial u_5 \partial v_4} & \frac{\partial^2 f(\mathbf{u})}{\partial u_5 \partial v_5} \\
\frac{\partial^2 f(\mathbf{u})}{\partial v_1 \partial u_1} & \frac{\partial^2 f(\mathbf{u})}{\partial v_1 \partial u_2} & \frac{\partial^2 f(\mathbf{u})}{\partial v_1 \partial u_3} & \frac{\partial^2 f(\mathbf{u})}{\partial v_1 \partial u_4} & \frac{\partial^2 f(\mathbf{u})}{\partial v_1 \partial u_5} & \frac{\partial^2 f(\mathbf{u})}{\partial v_1^2} & \frac{\partial^2 f(\mathbf{u})}{\partial v_1 \partial v_2} & \frac{\partial^2 f(\mathbf{u})}{\partial v_1 \partial v_3} & \frac{\partial^2 f(\mathbf{u})}{\partial v_1 \partial v_4} & \frac{\partial^2 f(\mathbf{u})}{\partial v_1 \partial v_5} \\
\frac{\partial^2 f(\mathbf{u})}{\partial v_2 \partial u_1} & \frac{\partial^2 f(\mathbf{u})}{\partial v_2 \partial u_2} & \frac{\partial^2 f(\mathbf{u})}{\partial v_2 \partial u_3} & \frac{\partial^2 f(\mathbf{u})}{\partial v_2 \partial u_4} & \frac{\partial^2 f(\mathbf{u})}{\partial v_2 \partial u_5} & \frac{\partial^2 f(\mathbf{u})}{\partial v_2 \partial v_1} & \frac{\partial^2 f(\mathbf{u})}{\partial v_2^2} & \frac{\partial^2 f(\mathbf{u})}{\partial v_2 \partial v_3} & \frac{\partial^2 f(\mathbf{u})}{\partial v_2 \partial v_4} & \frac{\partial^2 f(\mathbf{u})}{\partial v_2 \partial v_5} \\
\frac{\partial^2 f(\mathbf{u})}{\partial v_3 \partial u_1} & \frac{\partial^2 f(\mathbf{u})}{\partial v_3 \partial u_2} & \frac{\partial^2 f(\mathbf{u})}{\partial v_3 \partial u_3} & \frac{\partial^2 f(\mathbf{u})}{\partial v_3 \partial u_4} & \frac{\partial^2 f(\mathbf{u})}{\partial v_3 \partial u_5} & \frac{\partial^2 f(\mathbf{u})}{\partial v_3 \partial v_1} & \frac{\partial^2 f(\mathbf{u})}{\partial v_3 \partial v_2} & \frac{\partial^2 f(\mathbf{u})}{\partial v_3^2} & \frac{\partial^2 f(\mathbf{u})}{\partial v_3 \partial v_4} & \frac{\partial^2 f(\mathbf{u})}{\partial v_3 \partial v_5} \\
\frac{\partial^2 f(\mathbf{u})}{\partial v_4 \partial u_1} & \frac{\partial^2 f(\mathbf{u})}{\partial v_4 \partial u_2} & \frac{\partial^2 f(\mathbf{u})}{\partial v_4 \partial u_3} & \frac{\partial^2 f(\mathbf{u})}{\partial v_4 \partial u_4} & \frac{\partial^2 f(\mathbf{u})}{\partial v_4 \partial u_5} & \frac{\partial^2 f(\mathbf{u})}{\partial v_4 \partial v_1} & \frac{\partial^2 f(\mathbf{u})}{\partial v_4 \partial v_2} & \frac{\partial^2 f(\mathbf{u})}{\partial v_4 \partial v_3} & \frac{\partial^2 f(\mathbf{u})}{\partial v_4^2} & \frac{\partial^2 f(\mathbf{u})}{\partial v_4 \partial v_5} \\
\frac{\partial^2 f(\mathbf{u})}{\partial v_5 \partial u_1} & \frac{\partial^2 f(\mathbf{u})}{\partial v_5 \partial u_2} & \frac{\partial^2 f(\mathbf{u})}{\partial v_5 \partial u_3} & \frac{\partial^2 f(\mathbf{u})}{\partial v_5 \partial u_4} & \frac{\partial^2 f(\mathbf{u})}{\partial v_5 \partial u_5} & \frac{\partial^2 f(\mathbf{u})}{\partial v_5 \partial v_1} & \frac{\partial^2 f(\mathbf{u})}{\partial v_5 \partial v_2} & \frac{\partial^2 f(\mathbf{u})}{\partial v_5 \partial v_3} & \frac{\partial^2 f(\mathbf{u})}{\partial v_5 \partial v_4} & \frac{\partial^2 f(\mathbf{u})}{\partial v_5^2}
\end{bmatrix} \tag{9.43}$$

In three–dimensional cases, the 21 × 21 Hessian matrix is for a seven–point stencil. A search is explicitly determined for an interior grid point.

Validation of the method for single–block grid adaptation. An adaptation algorithm is available, if two fundamental requirements are satisfied. First, grid size (range of solution domain in physical space), grid dimension (number of grid points), and grid shape (Cartesian or curvilinear form) do not play a role in an adaptation process. Second, magnitudes of weight variables (pressure or density chosen as weight variables) does not play a role in an adaptation process. In the following, some test examples are given for the purpose of validating the adaptation algorithm.

Test case 1. Consider the example in section 9.1.3.7. The question to be answered is

- *Can one obtain the satisfactory results, if part of the solution domain will be adapted?*

Two subdomains are extracted from this example. They have 20 × 20 and 27 × 27 grid points, respectively, as shown in Fig. 9.10. Using the same objective function with control term for cell smoothness, grids are adapted separately with five iterations, respectively. The relaxation factor is $\omega = 1.0$.

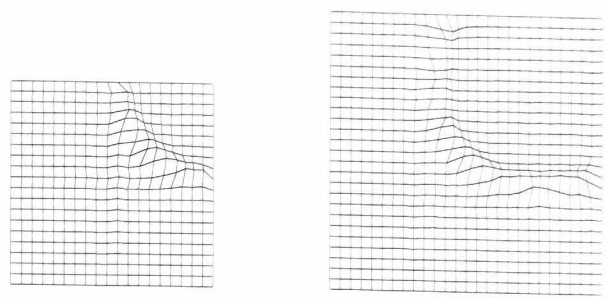


Figure 9.10: **Independent of block dimensions and number of grid points, both adaptive grids represent the features of weight functions.**

The objective functions are solved using the explicit method, i.e., search direction and step for an interior grid point are locally determined by a five–point stencil. Although these two blocks have different dimensions, the adaptive grids show the satisfactory quality. In both cases, grid points are clustered towards the weight functions, as shown in Fig. 9.10.

Test case 2. To implement an adaptation algorithm for multiblock grid adaptation, it should be ensured that block number and block size do not play a role in an adaptation process. For this purpose, this test example is constructed. The grid of the example of Fig. 9.8 (see page 143) is divided into four blocks with different block sizes, as shown in Fig. 9.11. Four blocks have the dimensions with 27×27, 27×17, 17×17, and 27×17 grid points, respectively. The objective function with smoothness control is minimized using the explicit method.

Four blocks are adapted separately, as shown in Fig. 9.12 (see page 147). Locally, blocks are adapted with satisfactory grid quality. Globally, the problem is the C^1–discontinuity on block interfaces. The reason for this discontinuity is the lack of data coupling among blocks.

Test case 3. The next validation is aimed at answering the following question

- *Can one obtain the satisfactory results, if a solution domain is meshed by curvilinear grid?*

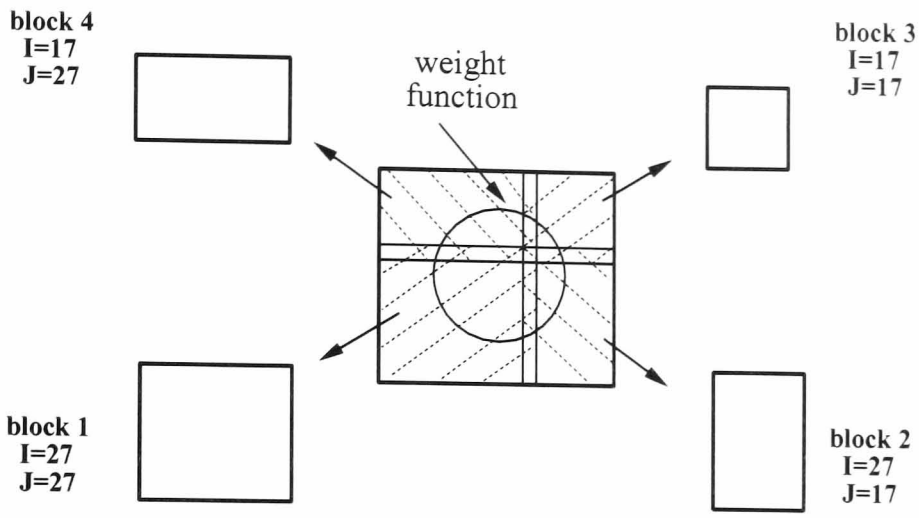


Figure 9.11: **A** 41×41 **grid is divided into four blocks having different block sizes. The test with subdomains decomposed is important for validation of an adaptation algorithm being implemented for multi-block cases.**

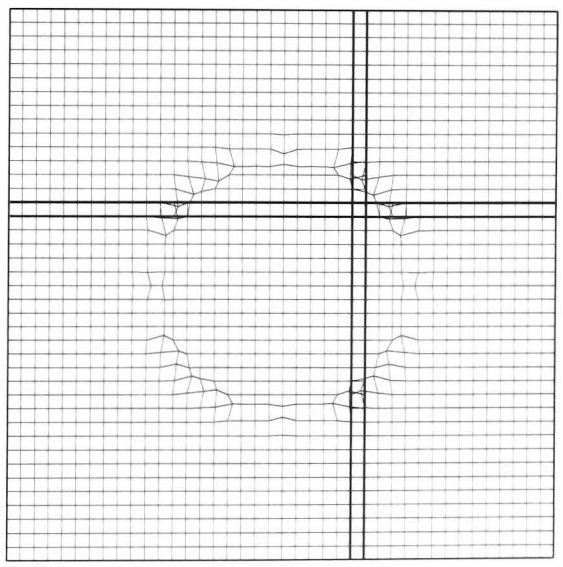


Figure 9.12: **Four blocks are adapted separately in one search step. Due to the lack of data coupling among block boundaries, grid lines are no longer** C^1**–continuous across block boundaries.**

The adapted grid is generated using the following analytical equation

$$\hat{P} = \{\hat{p}_{i,j}(x, y) \mid i = -20, \ldots, 20; j = -20, \ldots, 20\}$$

where

$$\begin{aligned} x &= (2i + 8j + 1)cos[(i + \sqrt{j} - 1)\theta] \\ y &= (2i + 8j + 1)sin[(i + \sqrt{j} - 1)\theta], \quad \theta = 3^o \end{aligned}$$

Weight functions are generated in computational space using the same analytical equation as that in previous example.

The result from one search step is shown in Fig. 9.13, where a small square represents the grid boundary of the first test case. This grid is much larger than the one in the example in section 9.1.3.7 (see Fig. 9.8, page 143), and the grid lines are curved. It can be seen that the size and shape of grids do not substantially affect the efficiency of the adaptation process.

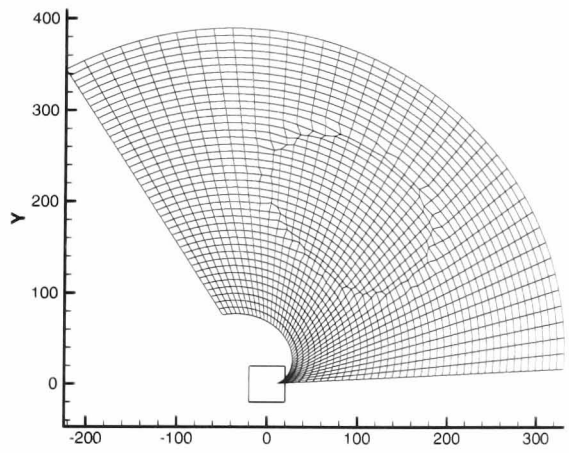


Figure 9.13: **The objective function with smoothness control is minimized in one search step. The small square represents the block boundary of the test example in section 9.1.3.7.**

Test case 4. In the above three test cases, weight variables are generated analytically. The next question to be answered is

- *Can one obtain the satisfactory results, if a weight function is generated using a flow solution?*

The examples is chosen from a flow simulation case *forward facing step*. Shock reflection is simulated under a supersonic flow condition. The goal of this example is to test the efficiency of the algorithm for real applications.

The result is shown in Fig. 9.14. Along the shock reflection, grid points are clustered. In this test case, there is no special treatment of flow variables, such as filtering of flow variables. The normalized weight variables are directly used as weight variables.

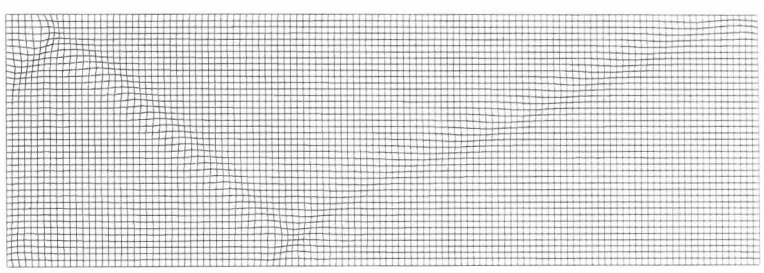


Figure 9.14: **Part of forward facing step is selected in order to test efficiency of adaptation algorithm in real cases.**

The next question to be answered is

- *Can one obtain the satisfactory results, if the algorithm is used in three–dimensional cases?*

Test cases 5. A single block grid with $41 \times 41 \times 41$ grid points is generated. It represents a cube in the range $x, y, z \in [-20, 20]$. Weight variables are analytically generated in computational space, and describe a sphere in the cube. In parameter space, an internal grid point $p_{i,j,k}(\mathbf{t})$ is coupled with its six neighboring grid points $p_{i,j-1,k}(\mathbf{t})$, $p_{i-1,j,k}(\mathbf{t})$, $p_{i+1,j,k}(\mathbf{t})$, $p_{i,j+1,k}(\mathbf{t})$, $p_{i,j,k-1}(\mathbf{t})$ and $p_{i,j,k+1}(\mathbf{t})$. Ordering these grid points $p_i(\mathbf{t})$, $i = 1, \ldots, 6$, the 7×7 Hessian matrix of the objective function for grid adaptation is in the form

$$H^a(\mathbf{t}) = 2\begin{bmatrix} w_1^2 & 0 & 0 & 0 & 0 & 0 & -w_1^2 \\ 0 & w_2^2 & 0 & 0 & 0 & 0 & -w_2^2 \\ 0 & 0 & w_3^2 & 0 & 0 & 0 & -w_3^2 \\ 0 & 0 & 0 & w_4^2 & 0 & 0 & -w_4^2 \\ 0 & 0 & 0 & 0 & w_5^2 & 0 & -w_5^2 \\ 0 & 0 & 0 & 0 & 0 & w_6^2 & -w_6^2 \\ -w_1^2 & -w_2^2 & -w_3^2 & -w_4^2 & -w_5^2 & -w_6^2 & -\sum_{i=1}^{6} w_i^2 \end{bmatrix} \tag{9.44}$$

where the superscript a denotes the objective function for *adaptation*. The 7×7 Hessian matrix of the control function for cell smoothness is in the form

$$H^s(\mathbf{t}) = \frac{1}{72}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & -6 \\ 1 & 1 & 1 & 1 & 1 & 1 & -6 \\ 1 & 1 & 1 & 1 & 1 & 1 & -6 \\ 1 & 1 & 1 & 1 & 1 & 1 & -6 \\ 1 & 1 & 1 & 1 & 1 & 1 & -6 \\ 1 & 1 & 1 & 1 & 1 & 1 & -6 \\ -6 & -6 & -6 & -6 & -6 & -6 & 36 \end{bmatrix} \tag{9.45}$$

where the superscript s denotes the control function for *smoothness*. After one search step, the result is shown in Fig. 9.15.

Both objective and control functions for smoothness are quadratic, the minimization of the objective function could be finished in few search steps.

Test case 6. A hollow sphere with radii $r_1 = 1$ and $r_2 = 2$, is generated using a 6–block topology. Each block has $41 \times 41 \times 41$ grid points. One of the blocks is chosen as test case. Weight variables are analytically generated in computational space. It describes a sphere in the solution domain.

The size of solution domain of this example is much smaller in comparison with the previous test case. The complete solution domain possesses about eight voxels of the previous example. After one search step, the result is visualized in Fig. 9.16.

Above two test cases show that this adaptation method is available for both Cartesian and curvilinear grids. Although two grids are different in size and

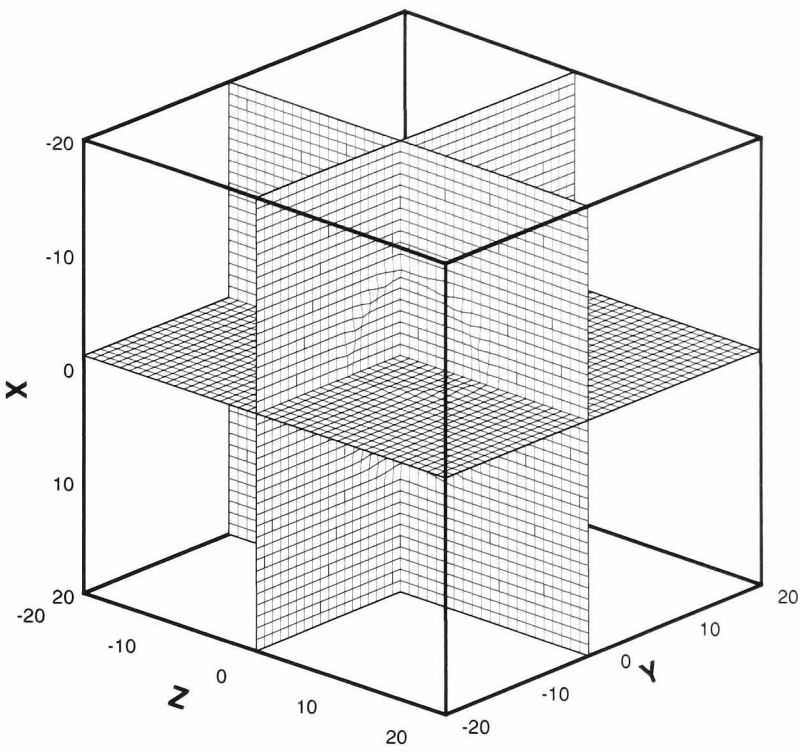


Figure 9.15: **The grid represents a cube. Weight functions are generated using analytical equation. The objective function with smoothness control is minimized in one step.**

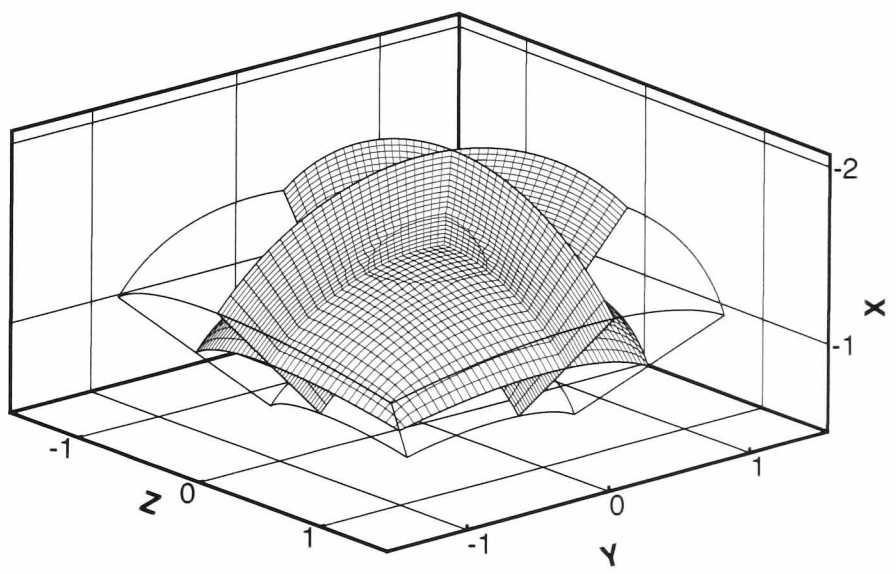


Figure 9.16: **Part of a hollow sphere is meshed with** $41 \times 41 \times 41$ **grid points. In parameter space, weight variables are generated in the same way as the previous example. The objective function with smoothness control is minimized in one step.**

shape, the adaptation results are satisfactorily obtained.

Termination criterion of block adaptation. The termination criterion used in test cases is to compare the rate of gradients of objective functions from two search steps. It is locally available for a single grid point $p_{i,j}(\mathbf{t})$.

In the following example, the termination criterion $\epsilon = 10^{-13}$ is specified for all grid points. The search is terminated for an interior grid point, when this condition is met at this point. The search process is iteratively continued, until this condition is met at all interior grid points.

However, this condition is globally specified. A bad solution at any interior grid point leads to the next search throughout the entire block. Fig. 9.17 shows three solutions. Although the minimization of objective function amounts to 500 steps in total, the global termination condition is still not satisfied.

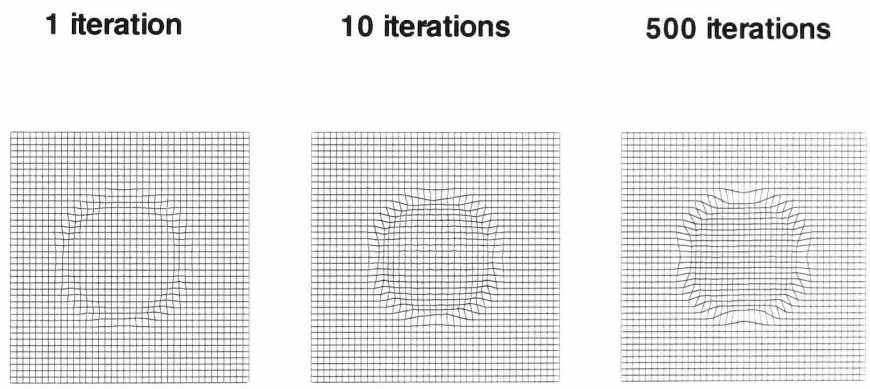


Figure 9.17: **The minimization process will continue, when a global termination condition is not satisfied at any interior grid point. Although a large number of iterations, the global termination condition is not met.**

Algorithm for minimization of objective function. The algorithm describes

the process of single–block grid adaptation in two–dimensional cases. The extension of the algorithm to three–dimensions, stencil equations are in the form of a seven–point system instead. The main steps are as follows:

- ▷ **Step 1**. Read a single block, and allocate grid points $p_{i,j}(\mathbf{x})$ and weight variables $\phi_{i,j}(\mathbf{x})$ in two–dimensional arrays with $i = 1, \ldots, I$, and $j = 1, \ldots, J$, respectively.

- ▷ **Step 2**. For all grid points: normalize the weight variables using Eq. (9.4), and save the normalized values $\psi_{i,j}$ in a two–dimensional array; build weight functions for all grid points in parameter space $w_{i,j}$, and save them in a two–dimensional array.

- ▷ **Step 3**. For all grid points: transform the physical coordinates into parameter coordinates using Eq. (9.1), and save the grid points $\hat{p}_{i,j}(\mathbf{u})$ in a two–dimensional array, where $i = 1, \ldots, I-1$ and $j = 1, \ldots, J-1$. The interior grid points $\hat{P} = \{\hat{p}_{i,j}(\mathbf{u}) \mid i = 2, \ldots, I-2; j = 2, \ldots, J-2\}$ describe the solution domain.

- ▷ **Step 4**. For all interior grid points: weight functions are interpolated onto the mid–points of line segments. For instance, at the grid point $\hat{p}_{i,j}(\mathbf{u})$, four weights are calculated by

$$w_s = \frac{1}{2}(w_{i,j-1} + w_{i,j}), \quad w_w = \frac{1}{2}(w_{i-1,j} + w_{i,j})$$
$$w_e = \frac{1}{2}(w_{i+1,j} + w_{i,j}), \quad w_n = \frac{1}{2}(w_{i,j+1} + w_{i,j}) \tag{9.46}$$

- ▷ **Step 5**. For all interior grid points: calculate first partial and second derivatives of the objective function for all five–point stencil. Generate the Hessian matrix using Eq. (9.43), and set the values of the first and second derivatives at the boundary points by zero.

- ▷ **Step 6**. For all interior grid points: choose the directions of the negative gradients as search directions; calculate the search steps using Eq. (9.21).

- ▷ **Step 7**. For all interior grid points: select their actual positions as starting search points, and adapt them using Eq. (9.22); correct the adaptive position using Eq. (9.35), where the relaxation factor is in range $0 < \omega < 1$.

- ▷ **Step 8**. For all interior grid points: update their position and weight functions in physical space using Eq. (9.36).

After each iteration, penalty factors will be increased. The computation repeats **Step 3** to **Step 8**, until the termination condition is met.

9.1.4 Grid Points with Geometric Constraints

In grid adaptation process, internal grid points are moving in three–dimensional space, while the movement of external grid points is subject to specific geometric constraints, as shown in Fig. 9.18. Based on the degrees of freedom of their movement, geometric constraints of grid points fall into the following categories:

- ▷ **Face constraint**. Grid points on a physical boundary, which can be moved in a two–dimensional parametric net $\mathbf{r}(u,v)=\{x(u,v),y(u,v),z(u,v)\}$. The geometric constraints of these points are termed *face constraint.*
- ▷ **Edge constraint**. Grid points on a physical boundary, which are movable along a curve $P=\{p(x(t),y(t),z(t))\}$. Geometric constraints of these points are termed *edge constraint.*
- ▷ **Vertex constraint**. Grid points on a physical boundary, which are fixed. Geometric constraints of these points are termed *vertex constraint.*

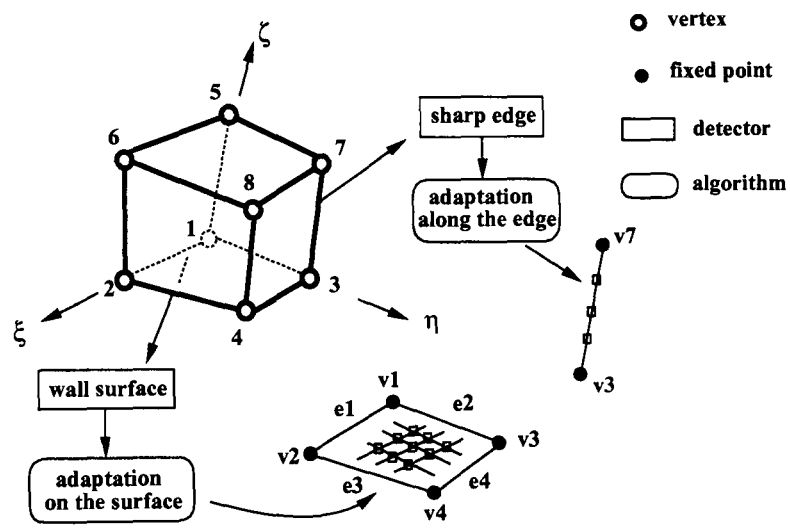


Figure 9.18: **Grid points are moved with respect to their geometric constraints. They are represented by the degrees of freedom.**

In order to retain the geometric features on a physical boundary, grid points with these geometric constraints must be adapted using different methods.

9.1.4.1 Adaptation of Grid Points with Face Constraints

Movement of these points is subject to the geometric surface. The adaptive computation is performed in parameter space. Grid points are adapted using two–dimensional objective functions with control functions. In order to find the positions of the new grid points in physical space, the following differential relations are used

$$\begin{aligned} dx &= x_u du + x_v dv \\ dy &= y_u du + y_v dv \\ dz &= z_u du + z_v dv \end{aligned} \tag{9.47}$$

In practice, the surface of a solid boundary is most often given in discrete form. The positions of the adapted points in physical space are therefore approximated by the following discrete form

$$\begin{aligned} x_{i,j}^{(k+1)} &= x_{i,j}^{(k)} + \frac{\Delta x_{i,j}^{(k)}}{\Delta u_{i,j}^{(k)}}(u_{i,j}^{(k+1)} - u_{i,j}^{(k)}) + \frac{\Delta x_{i,j}^{(k)}}{\Delta v_{i,j}^{(k)}}(v_{i,j}^{(k+1)} - v_{i,j}^{(k)}) \\ y_{i,j}^{(k+1)} &= y_{i,j}^{(k)} + \frac{\Delta y_{i,j}^{(k)}}{\Delta u_{i,j}^{(k)}}(u_{i,j}^{(k+1)} - u_{i,j}^{(k)}) + \frac{\Delta y_{i,j}^{(k)}}{\Delta v_{i,j}^{(k)}}(v_{i,j}^{(k+1)} - v_{i,j}^{(k)}) \\ z_{i,j}^{(k+1)} &= z_{i,j}^{(k)} + \frac{\Delta z_{i,j}^{(k)}}{\Delta u_{i,j}^{(k)}}(u_{i,j}^{(k+1)} - u_{i,j}^{(k)}) + \frac{\Delta z_{i,j}^{(k)}}{\Delta v_{i,j}^{(k)}}(v_{i,j}^{(k+1)} - v_{i,j}^{(k)}) \end{aligned} \tag{9.48}$$

where the superscripts k denote iteration numbers.

9.1.4.2 Adaptation of Grid Points with Edge Constraints

In parametric space, grid points moving along a sharp edge are expressed in a one–dimensional manner, $\hat{P} = \{\hat{p}_i(t) \mid i = 1, \ldots, I\}$. The objective function is of the following form

$$f_i^a(t) = w_{i-\frac{1}{2}}^2 \left[\hat{p}_{i-1}(t) - \hat{p}_i(t)\right]^2 + w_{i+\frac{1}{2}}^2 \left[\hat{p}_{i+1}(t) - \hat{p}_i(t)\right]^2 + \text{control term}$$

where a control term is constructed to prevent that the middle grid point $\hat{p}_i(t)$ is moved to its neighboring grid points. This occurs, in case one of the weight functions vanishes.

The control function for cell orthogonality does not exist. The control function for cell smoothness is used as control term. This is

$$f_i^s = \left(\frac{\hat{p}_{i-1}(t) + \hat{p}_{i+1}(t) - 2\hat{p}_i(t)}{4}\right)^2 \tag{9.49}$$

The objective function is therefore in the form

$$f_i = f_i^a + f_i^s \tag{9.50}$$

Another method for grid point adaptation with one degree of freedom employs a spline function. It is derived from a mechanical model, as shown in Fig. 9.19.

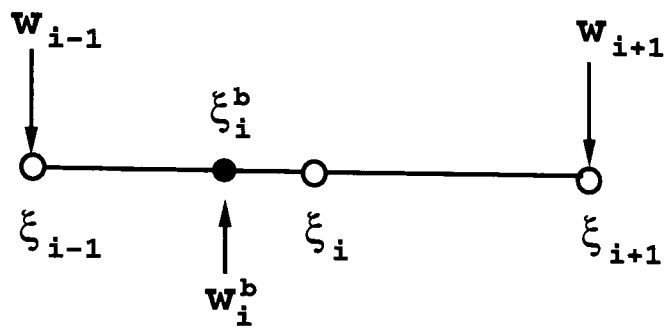


Figure 9.19: **A simple mechanical model of moment equilibrium at the pivot.**

In computational space, the line is represented by a strut, neglecting its own weight. It is supposed that weights w_{i-1} of point ξ_{i-1} and w_{i+1} of point ξ_{i+1} attract point ξ_i. For $w_{i-1} < w_{i+1}$, point ξ_i is moved towards ξ_{i+1} direction, and *vice versa*. Initially, the weight w_i of point ξ_i is ignored.

Suppose that a weight w_i^b of point ξ_i^b is to counter–balance the attractions from w_{i-1} and w_{i+1}, where superscript b denotes *balance*. The equilibrium conditions for the system with respect to the point ξ_{i-1} are expressed by

$$(\xi_{i+1} - \xi_{i-1})w_{i+1} - (\xi_i^b - \xi_{i-1})w_i^b = 0 \tag{9.51}$$

and

$$w_{i-1} + w_{i+1} - w_i^b = 0 \tag{9.52}$$

One obtains the balance position from above equations:

$$\xi_i^b = \frac{w_{i-1}\xi_{i-1} + w_{i+1}\xi_{i+1}}{w_{i-1} + w_{i+1}} \tag{9.53}$$

The displacement of point ξ_i^b with respect to point ξ_i is calculated by

$$\varepsilon = \frac{\xi_i^b - \xi_{i-1}}{\xi_i + \xi_{i-1}} \tag{9.54}$$

where ε is a measure for the relative displacement of point ξ_1 in the computational space. Suppose that the curve in the physical space is described through three neighboring points by a spline function. Three points $p_{i-1}(\mathbf{x})$, $p_i(\mathbf{x})$, and $p_{i+1}(\mathbf{x})$ are used to determine the new position of the ξ_i^b in the physical space [44]

$$p_i^b(\mathbf{x}) = \frac{(1-\varepsilon)^2 w_{i-1} p_{i-1}(\mathbf{x}) + 2\varepsilon(1-\varepsilon) w_i p_i(\mathbf{x}) + \varepsilon^2 w_{i+1} p_{i+1}(\mathbf{x})}{(1-\varepsilon)^2 w_{i-1} + 2\varepsilon(1-\varepsilon) w_i + \varepsilon^2 w_{i+1}} \tag{9.55}$$

9.1.5 Vertices of a Block

Block connectivity can be very complicated in three dimensional cases. Because of high complexity of block connectivity, it is difficult to find a desirable method for adapting the common vertices shared by different blocks. It could be happen that a vertex has no unique position in the space after an adaptation process. In order to avoid the problem, the positions of all vertices are checked after grid adaptation. In case of different positions of a vertex, they are replaced to the point resulted from the mean value of their coordinates.

9.2 Multiblock Grid Adaptation

The extension of above algorithms for a single block to multiblock cases requires regarding topological specialty of multiblock grids. The major problems are summarized as follows.

- ▷ **Data coupling of block interfaces**. Most multiblock grids can not be merged into a single superblock, in which points are arranged in the manner of a three–dimensional array. Multiblock grid adaptation is therefore performed on a block by block basis. In order to retain C^1–continuity of grid lines across block boundaries, a data coupling among blocks must exist.
- ▷ **Design of adaptation procedure**. Multiblock grid adaptation must be performed automatically. This requires a logical sequence of data computation and updating for weight variables, since each movement of grid points will change the weight variables of these points.

Extending single block adaptation to multiblock cases, the major focus is on answering two questions:

- *How can one build a coupling among blocks, such that the algorithm for a single block can be extended into multiblock cases?*

- *Which sequence of data processing is required, such that a multiblock grid adaptation can be performed automatically?*

9.2.1 Data Coupling among Blocks

Grid data of different blocks is coupled by generating overlap layers. According to the degrees of freedom of grid points, the overlap is divided into two types:

▷ **Block overlap in solution domain**. This type of overlap layers is generated among neighboring blocks. Due to data coupling among neighboring blocks, C^1–continuity of grid lines is retained during grid adaptation.

▷ **Face overlap on physical boundary**. This type of overlap layers is generated among neighboring faces on physical boundary. This data coupling allows to move grid points over common edges.

9.2.1.1 Block Overlap

Suppose that two blocks are connected to each other, as shown in Fig. 9.20. Face 4 ($j = J$) of block 1 and face 5 ($i = I$) of block 2 build the block interface to each other. Blocks are overlapped in the following manner:

▷ For block 1: block dimension in j–direction is extended to $J + 1$, and grid points for $J + 1$ plane are copied by the $I - 1$ plane of block 2 with respect to the matching orientation.

▷ For block 2: block dimension in i–direction is extended to $I + 1$, and grid points for $I + 1$ plane are copied by the $J - 1$ plane of block 1 with respect to the matching orientation.

This extension of block boundaries is termed *block overlap*. In generation of block overlap, a special memory storage is allocated for changing matching orientation and data exchange among blocks. The memory is termed *intermediate memory*.

Using the above example, the method developed for overlap generation is explained in the following:

▷ **Read the block topology file**. The information about face connectivity and matching orientation is obtained from a block topology file.

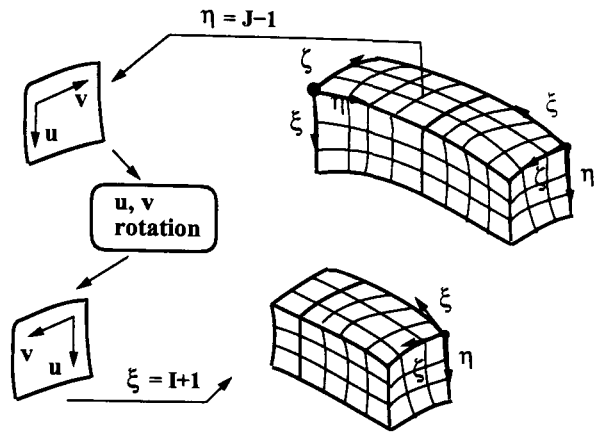


Figure 9.20: **A block interface is overlapped by extending a plane from a neighboring block.**

- ▷ **Allocate intermediate storage.** Intermediate storage is allocated for every face. The intermediate storage of a face contains information about block number, face number, and the size of the two–dimensional array.

- ▷ **Export an overlap plane to the intermediate storage.** Consider face 5 of block 2. The plane $\eta = J - 1$ is exported to the intermediate storage that is addressed for face 4 of block 1, where the points are saved in sequence matched to plane $\xi = I$ of block 1.

- ▷ **Import an overlap plane from the intermediate storage.** Consider face 4 of block 1. To extend its boundary on face 4, the data can be obtained from the intermediate storage, addressed by face 4 of block 1, directly, without regarding its face matching.

9.2.1.2 Face Overlap

The faces on a physical boundary are connected to each other through their edges. Grid points along a common edge can have a face or an edge constraint. In case (a), grid points along the common edge have two degrees of freedom. Movement of these points necessitates a data coupling between two neighboring faces. This coupling is built by generating face overlap.

In a face connectivity file, the information about face connectivity and its matching orientation is given. Based on this information, the face overlap will be generated. Since the face boundaries are extended through a face overlap, the grid points along common edges are converted into the interior of faces. Fig. 9.21 shows an example of face overlap. Circles denote points on a common edge. The face boundary of block 2 is extended by face overlap, and its new boundary is described by points denoted by hollow squares. An arbitrary point, denoted by a hollow circle, is coupled by points marked by a cross.

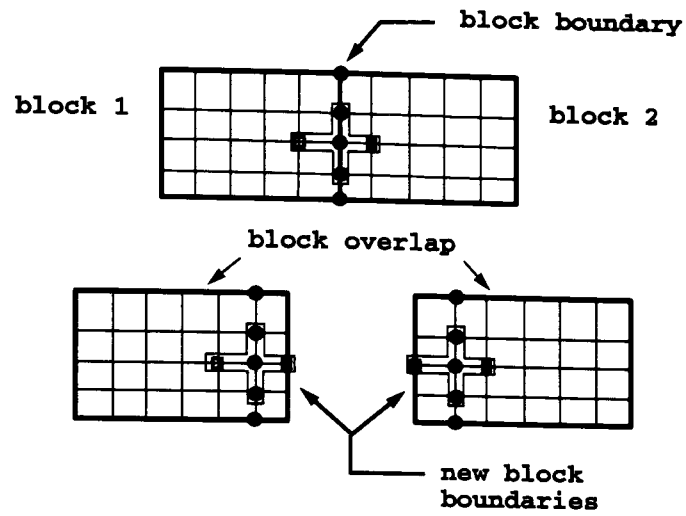


Figure 9.21: **Data coupling on block boundary is realized by face overlap.**

9.2.1.3 Example of C^1–Continuity on Block Interfaces

The attempt to giving a two–dimensional example is to examine whether the problem of the C^1–discontinuity on block interfaces can be solved using the method for data exchange on block interfaces.

The complete grid of the example in section 9.1.3.7 (see Fig. 9.8, page 143) is divided into four blocks, which have different dimensions and their own orientations of local coordinate systems, as shown in Fig. 9.22. The orientations of the local coordinate systems are ordered as given in Table 9.11.

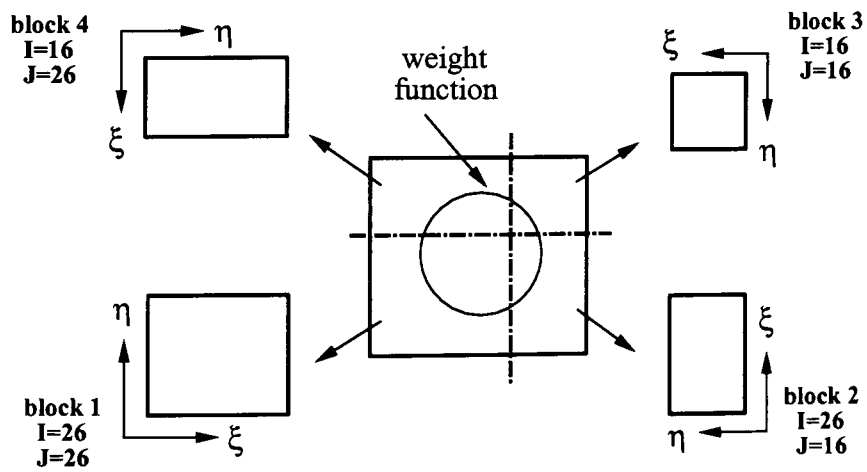


Figure 9.22: **Two–dimensional 4 blocks test case for active grid adaptation.**

Based on a block connectivity file, generated by the algorithm explained in

Table 9.11: Block dimensions and coordinate systems.

block #	*I*	*J*	*ξ direction*	*η direction*
1	26	26	east	north
2	26	16	north	west
3	16	26	south	south
4	16	16	west	east

sections 5.1.2 (page 49) and 5.1.3 (page 52), the information about neighboring blocks as well as the matching orientation is used for overlapping block interfaces. Block extension through overlap rows or columns ensures the continuity of weight functions on block interfaces. This guarantees C^1–continuity for grid lines during an adaptation process, as shown in Fig. 9.23.

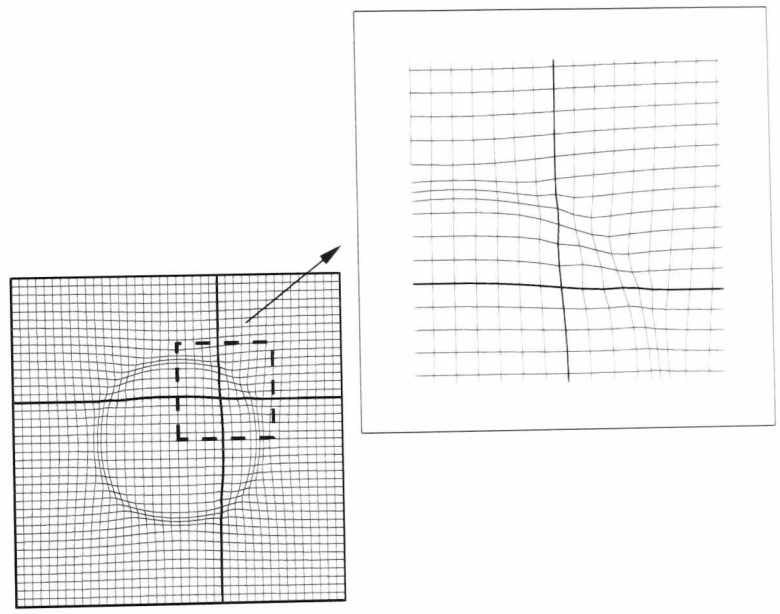


Figure 9.23: **Using P–Q control functions [8], [36], a four block grid is adapted by elliptic grid generation. Since the strategy for multiblock grid adaptation is implemented in this test case, C^1–continuity of grid lines across block boundaries is guaranteed.**

9.2.2 Design of an Adaptation Procedure

In [9], two major problems are addressed. First, when adaptation is employed on one grid block, all neighboring blocks have to be adapted in order to preserve continuity across block interfaces. This can be expensive if adaptation is only necessary for some blocks, but all neighboring blocks have to be adapted to

ensure connectivity. Second, the adaptation of one block according to a solution may be impeded by the constraint of boundary point connectivity. Adjacent blocks may have very different point density requirements.

In this thesis, adaptive computation is performed successively from one block to another. Due to the complexity of the grid topology, a multiblock grid can not be merged into a super–block. Therefore, block interfaces are coupled by overlap rows or columns. This coupling provides the information of flow variables and grid point positions of the neighboring blocks, needed for C^1–continuity of grid lines across block interfaces. Using this type of data coupling, the grid adaptation process can be performed in sequence. The core strategy for adapting a multiblock grid is explained as follows:

- ▷ **1. Adapt grid points with three degrees of freedom**. Internal points are adapted. During adaptation of the internal points, external points are considered as *fixed points*, providing the necessary *boundary conditions*, e.g., weights, arc lengths etc.. After this, all points with three degrees of freedom are frozen.

- ▷ **2. Adapt grid points with two degrees of freedom**. External points with a face constraint are adapted subject to the geometric surface of the physical boundary. It is supposed that points along edges are *fixed*, providing the *boundary conditions* for the computation. After this, all points with two degrees of freedom are frozen.

- ▷ **3. Adapt grid points with one degree of freedom**. External points with edge constraints are adapted along a parametric curve. Points with vertex constraint are fixed, and provide *boundary conditions*. After this, all points with one degrees of freedom are frozen and the adaptation process is either finished or continued at step 1.

Fig. 9.24 depicts the strategy for grid adaptation based on the principle *reduction of degrees of freedom*. However, implementation of this strategy requires a strict logic. The implemented procedure designed contains the following steps.

1. Generation of grid topology. Suppose that a three–block grid has to be adapted, as shown in Fig. 9.25. The requirements for grid topology generation are as follows:

- ▷ **(a) Local coordinate system**. All blocks should have the right–handed local coordinate systems. The identification of local coordinate system is found in section 5.1.1 (page 46).

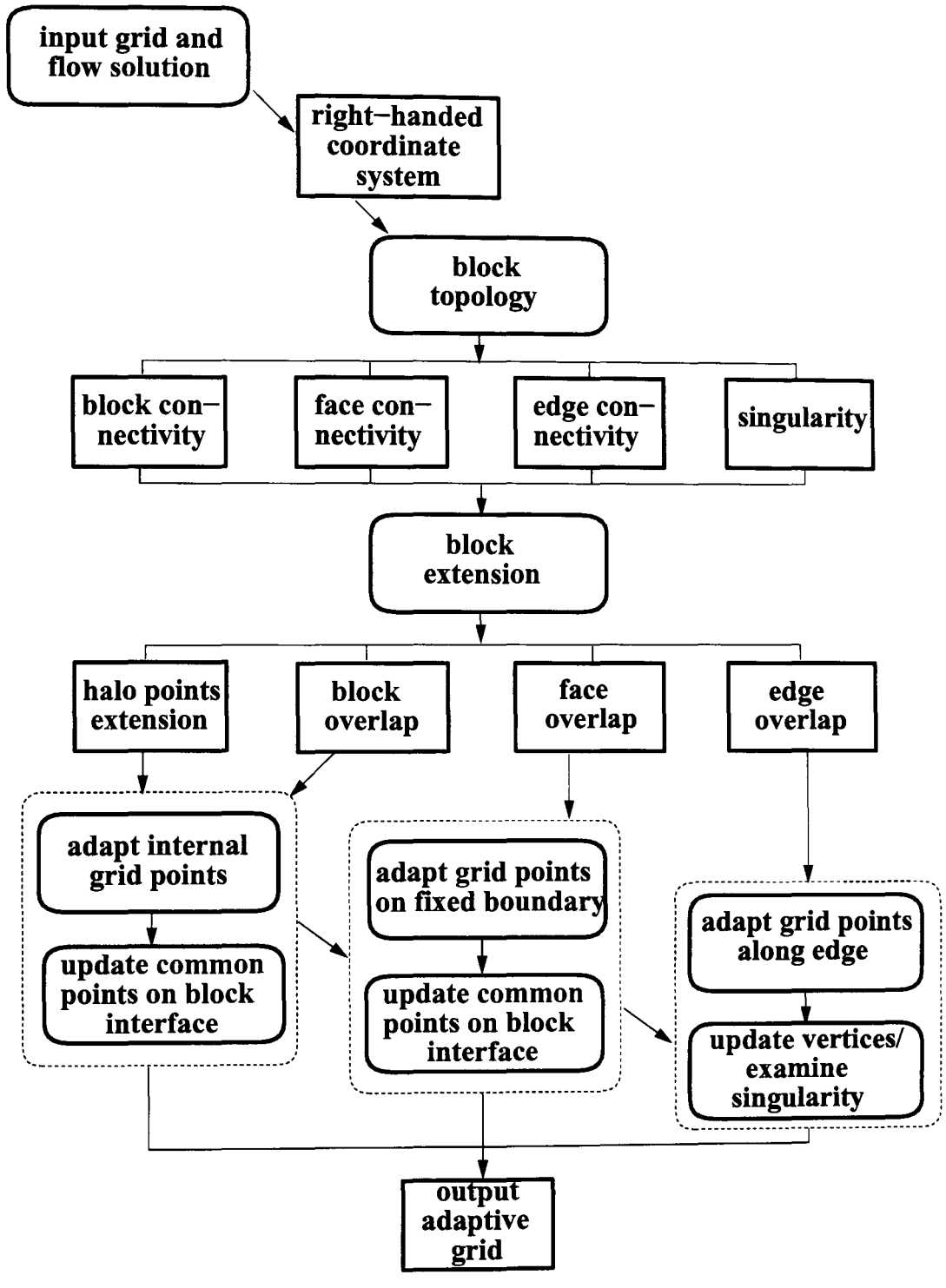


Figure 9.24: The flow chart diagram shows the process of adapting grid points with different degrees of freedom. The rectangles with rounded off vertices denote processes, which the rectangles are grid data.

- ▷ **(b) Block connectivity**. Neighboring relations among blocks are determined by comparing positions of grid points on block boundaries of different blocks. The method is found in section 5.1.2 (page 49).
- ▷ **(c) Matching orientation**. Grid point coordinates of neighboring blocks must be matched, such that blocks will correctly overlap. The method is found in section 5.1.3 (page 52).
- ▷ **(d) Geometry features on physical boundaries**. Based on these features, degrees of freedom of grid points on physical boundaries are determined. The method is found in section 5.3 (page 53).

2. Extension of block boundaries. Block boundaries are extended, such that the grid points of a block boundary become the interior grid points of a solution domain, as shown in Fig. 9.25.

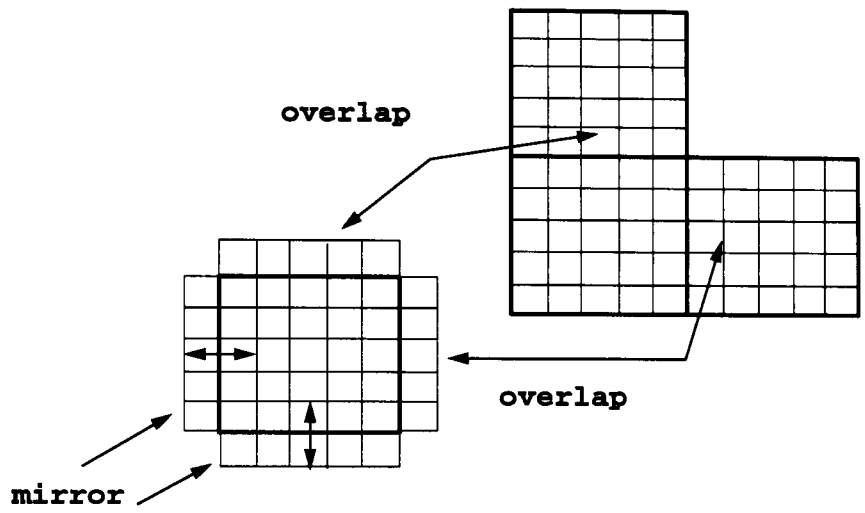


Figure 9.25: **Boundaries are extended by overlap. Grid points on block boundaries are interior points of the neighboring solution domain.**

- ▷ **(a) Overlap row or column**. In case that neighboring blocks exist, overlap rows are generated.

3. Weight function. To generate a weight function, scalar quantities of a flow solution are chosen as weight variables.

- ▷ **(a) Norm of weight variables**. Weight variables are normalized by Eq. (9.4).
- ▷ **(b) Generation of weight functions**. A weight function may be generated using high order filter.
- ▷ **(c) Half–point weights**. Weight functions are interpolated onto the midpoints of grid line segments. For a grid point, the weights can be generated by Eq. (9.46).

After block extension and weighting grid line segments, a block has the form.

4. Parameter coordinate system. Adaptation computation is performed in parameter space. The transformation of physical coordinates into parameter space is given by Eq. (9.1).

5. Grid adaptation. Grid points with three degrees of freedom are adapted successively by solution of objective function with grid quality control.

- ▷ **(a) Governing equation**. The governing equation of grid adaptation is formulated in the form of a stencil equation for an interior grid point.
- ▷ **(b) Adapt interior grid points**. Computation of new grid coordinates in parameter space using an algorithm for grid adaptation.
- ▷ **(c) Update grid points on block interfaces**. The process of updating grid points on block interfaces ensure C^1–continuity across block interfaces. A grid point will be updated using its two neighboring points, shown in Fig. 9.26.

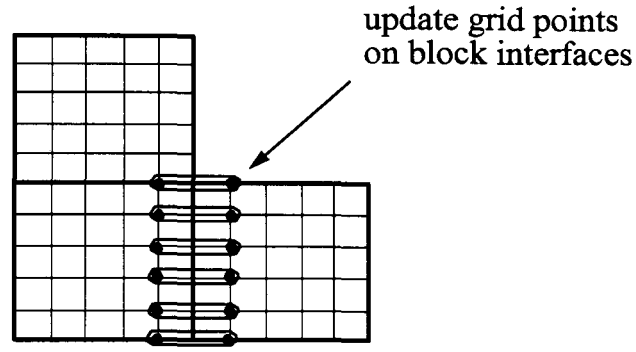


Figure 9.26: **Grid points on a block boundary are updated using one neighboring point.**

- ▷ **(d) Adapt grid points on a fixed boundary**. In the previous step (b), grid points at a fixed boundary are treated as interior grid points. These grid points are adapted with respect to their geometric constraints, and the new adapted positions overwrite their positions from step (b), shown in Fig. 9.27.

6. Generation of physical coordinates. Using the differential relations between parameter and physical space, the physical coordinates as well as new weight functions are computed.

7. Update singularity. Singularity points are not movable. They remain at their original positions, as shown in Fig. 9.27.

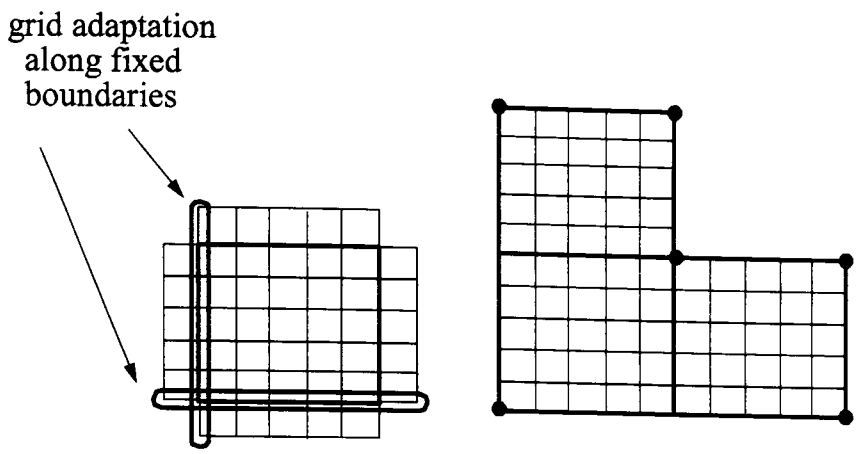


Figure 9.27: **Grid points on block boundaries are adapted with respect to geometric constraints, while vertices remain at their original positions.**

9.2.3 Results of Multiblock Grid Adaptation

The strategy and adaptation algorithm is implemented and tested by different examples. It should be noted that the major objective of the test example focuses on the efficiency of a strategy for *multiblock grid adaptation with reduction of degree of freedom*, while the algorithm for block adaptation is secondary. The adaptation tests should provide the following results.

First, the objective functions with smoothness and cell aspect ratio control should constrain the movement of grid points within the restricted region. Furthermore, the control functions are not only penalty functions, but their contribution should improve the robustness of an adaptation algorithm as well as high grid quality.

Second, a very important step to ensuring a successful multiblock grid adaptation is to generate data coupling on block interfaces. The method used for generating data coupling is the block overlap technique [36], [41] and [99]. The test cases should show C^1–continuity of grid lines across block interfaces.

Finally, grid points on a physical boundary have geometric constraints. These must be identified and specified in grid topology files. The test cases should show the effectiveness of grid point movement with constraints.

Requirements for grids and weight functions of test cases are as follows:

- ▷ **Cartesian grid**. A Cartesian multiblock grid is needed to test the sym-

metry property of an adaptive grid in case of a axisymmetric distribution of weight functions.

- ▷ **Curvilinear grid**. A curvilinear multiblock grid is needed to test the efficiency of the algorithm in case of non–uniform grid spacings and non–symmetric weight function.

- ▷ **Large number of blocks/complex block topology**. A grid with a large number of blocks and topology complexity is needed to test block overlap, as well as the grid adaptation strategy.

9.2.3.1 Three–Dimensional Example 1

The solution domain of this example is represented by a cube in range $x, y, z \in [-20, 20]$, meshed by $41 \times 41 \times 41$ points. Grid lines are uniformly and orthogonally distributed with constant increments Δx, Δy, and $\Delta z = 1$ in ξ–, η–, and ζ–directions. An analytical weight variable is generated using the following tangent hyperbolic equation

$$w(x, y, z) = tanh\left[1 - \left(\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2}\right)\right], \quad -20 \leq x, y, z \leq 20 \qquad (9.56)$$

where a, b and c denote half–axes of the ellipsoid. The complete solution domain is decomposed into eight blocks with same dimensions, i.e., each block has $21 \times 21 \times 21$ points. Four test cases are designed to examine the adaptation strategy and the algorithm described above. The example contains the following serial tests:

- ▷ **7–block case**. Three faces are on the physical boundary, on which gradients of weight variable can be calculated. Both face and edge constraints have to be regarded during grid adaptation.

- ▷ **6–block case**. Four faces are on the physical boundary, on which gradients of weight variable can be calculated. Both face and edge constraints have to be regarded during grid adaptation.

- ▷ **5–block case**. Five faces are on the physical boundary, on which gradients of weight variable can be calculated. Both face and edge constraints have to be regarded during grid adaptation.

- ▷ **4–block case**. Four faces are on the physical boundary, on which gradients of weight variable can be calculated. Only face constraints have to be regarded during grid adaptation, as shown in Fig. 9.28.

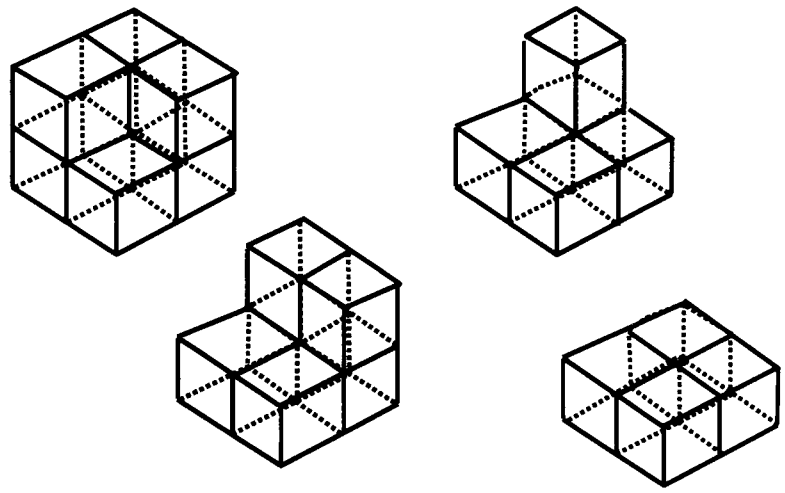


Figure 9.28: **The serial test consists of four cases with different block numbers: (a) 7–block case, (b) 6–block case, (c) 5–block case, and (d) 4–block case.**

The goal of the serial tests is to prove that the algorithm is capable of identifying fixed boundaries and sharp edges, and adapting grid points with respect to their geometric constraints. Block number should not influence the adaptive grid quality.

Using the same half–axes for the ellipsoids, weight variables are generated. The stencil objective functions of the first type with smoothness control are written for all interior grid points. For an interior grid point, the stencil equation describes a seven–point system, where six neighboring grid points of the central point are considered as stationary. Since these functions are quadratic, the Newton method is used for the solution of the system of equations. At each grid point, the search direction and step are locally determined. The results in one step are shown Figs. 9.29 to 9.32.

The change of block numbers aims at detecting the errors in the algorithm implementation, e.g., logical or computational errors. In four cases, grid points on fixed boundaries as well as on edges are clustered along the circle described by gradients of weight variables. There is no jump or slope discontinuity of grid lines on block interfaces. Moreover, the symmetry of weight variables is represented by grid point clustering.

Comparing the test examples in section 9.1.3.7 (page 139), it can be seen that the grid density in regions of clustering in this case is higher. The reason is that the contributions of six neighboring grid points on their central point is not contained in this solution method.

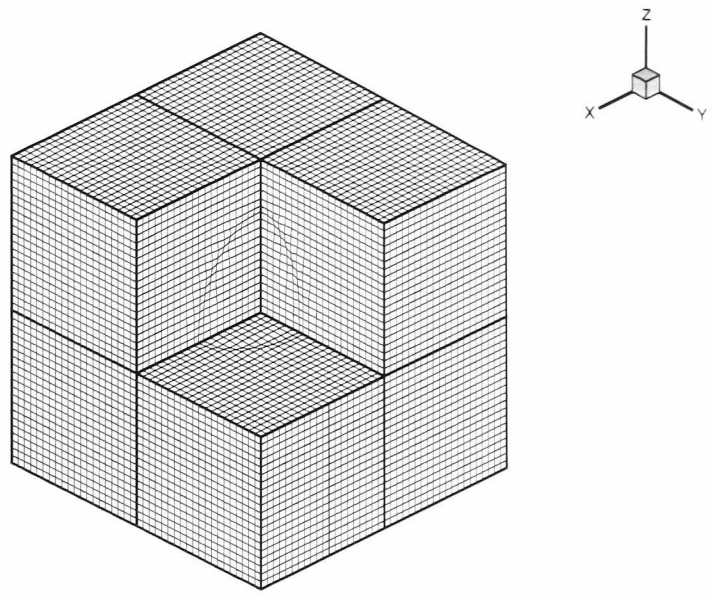


Figure 9.29: **7–block test case of the first three–dimensional example.**

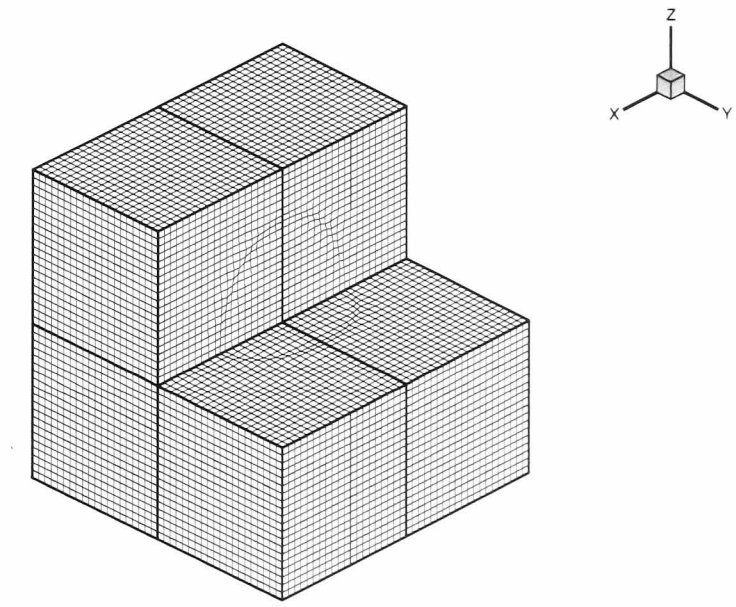


Figure 9.30: **6–block test case of the first three–dimensional example.**

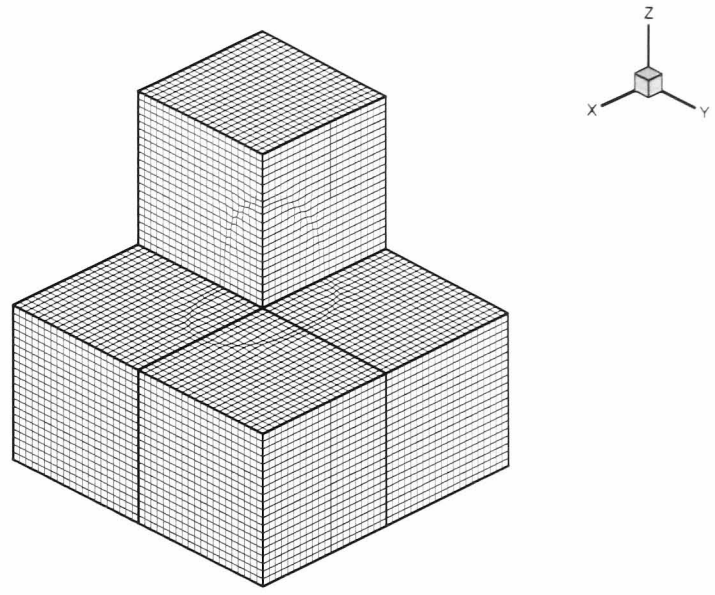


Figure 9.31: **5–block test case of the first three–dimensional example.**

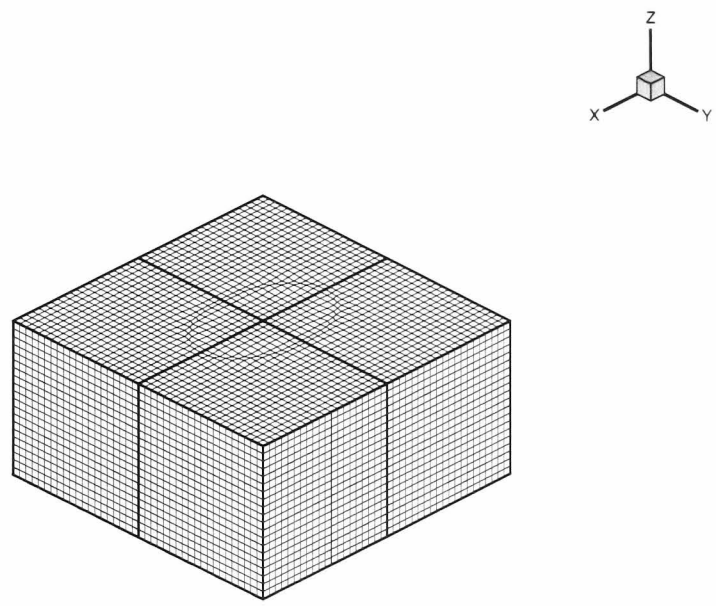


Figure 9.32: **4–block test case of the first three–dimensional example.**

In the second test, different values of the half–axes are chosen. The eight–block grid is adapted by the objective function with smoothness control. The contour of weight variables is represented by grid point clustering symmetrically in three orthogonal planes, as shown in Fig. 9.33.

9.2.3.2 Three–Dimensional Example 2

Solution domain of the example is a part of hollow sphere. The grid consists of eight blocks with the dimension $15 \times 15 \times 15$ grid points in ξ–, η–, and ζ–directions.

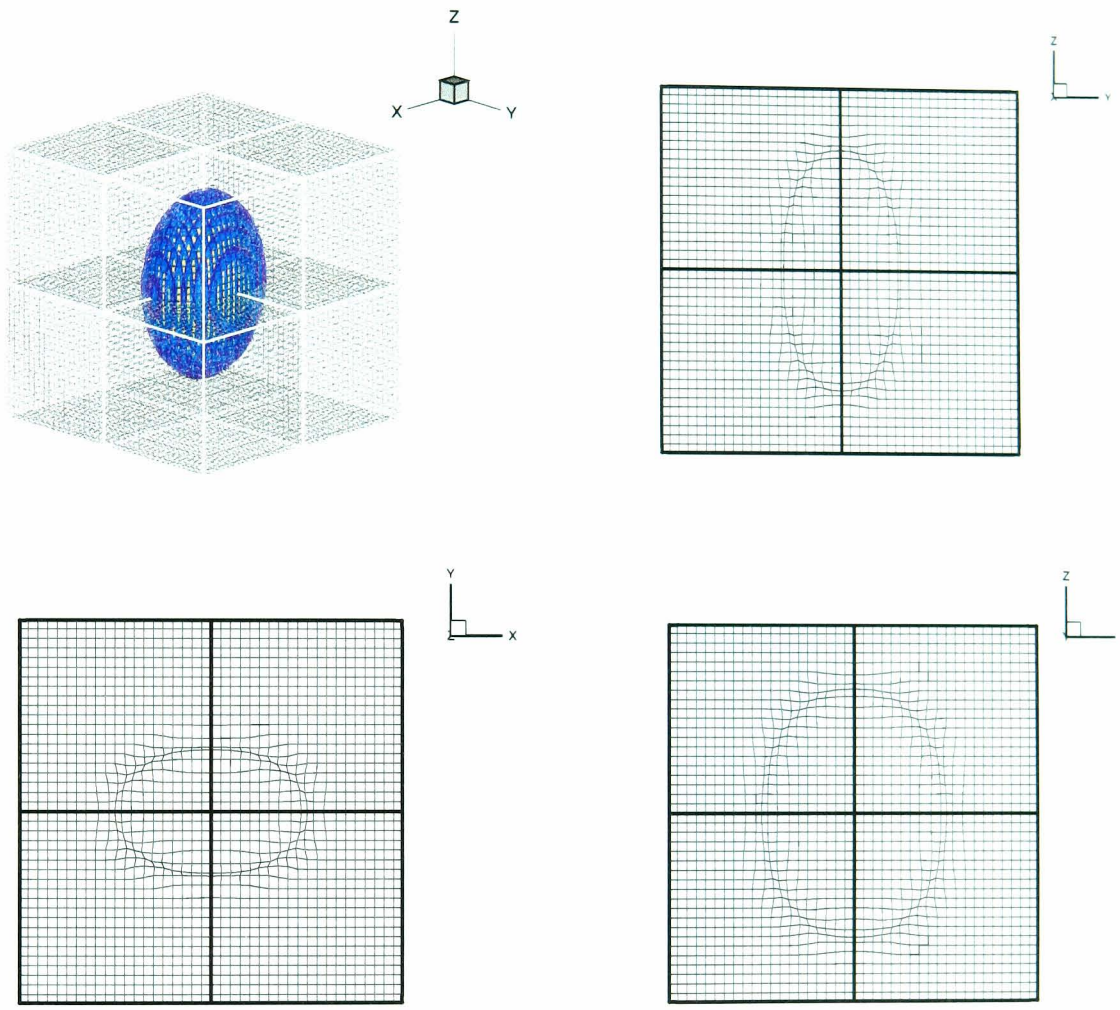


Figure 9.33: **The second test using different half–axes. The solution domain is represented by eight blocks. The result is visualized in four different views.**

respectively. Three grid points on outer boundary are arbitrarily chosen. They are used for generation of analytical weight variables. Using the following analytic equation

$$\frac{x}{a} + \frac{y}{b} + \frac{z}{c} = 1 \tag{9.57}$$

a, b, and c are determined by three chosen points. Using the following tangent hyperbolic equation

$$w(\xi, \eta, \zeta) = k \cdot tanh \left[1 - \left(\frac{\xi}{a} + \frac{\eta}{b} + \frac{\zeta}{c} \right) \right] \tag{9.58}$$

analytical weight variables are generated on a plane in computational space, where the coefficient k is employed to vary the magnitudes of weight variable. In physical space, weight variables describe a curved surface, as shown in Fig. 9.34.

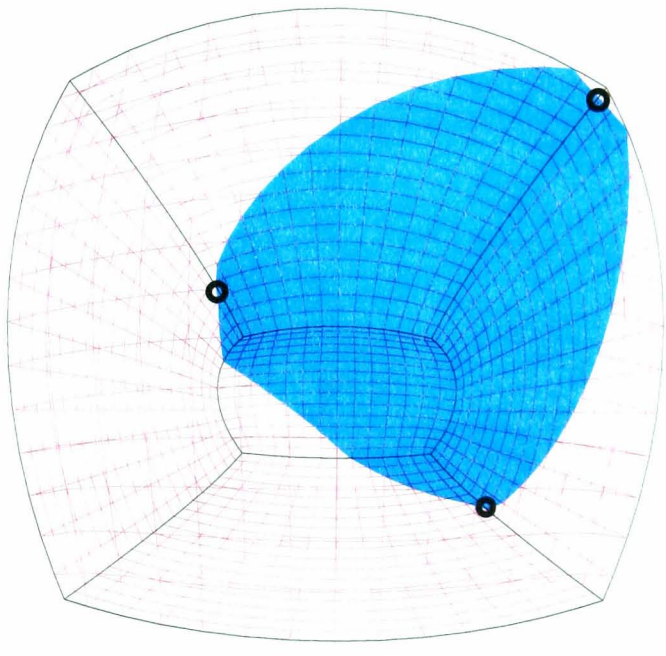


Figure 9.34: **A segment of a sphere is chosen as the solution domain of the present example. From three points, marked by bold circles, a cut plane is built. Weight variables are put onto the cut plane in computational space.**

The grid is divided into eight blocks with same dimensions, $8 \times 8 \times 8$, in each direction. The block topology is shown in Fig. 9.35.

To automatically identify sharp edges on a physical boundary, the measure for a sharp edge is specified by the maximal intersection angle $\theta_{max} \leq 10^o$ (see

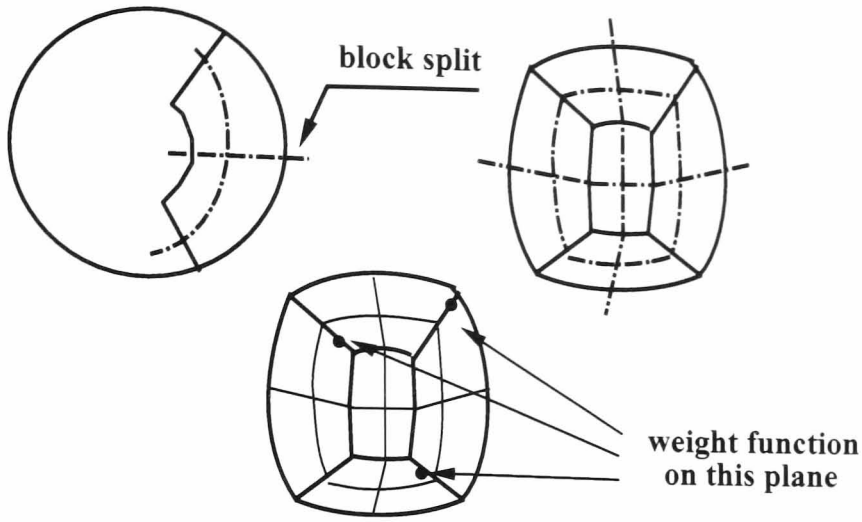


Figure 9.35: **The solution domain, a part of a hollow sphere, consists of eight blocks. Weight variables describe a curved surface in seven blocks.**

section 5.3.1, page 53).

The objective function with smoothness control is minimized by the Newton method in one search step. The result is shown in Fig. 9.36. Grid points on physical boundary are clustered corresponding to the gradients of weight variables. The sharp edges of solution domain are retained unchanged after grid adaptation.

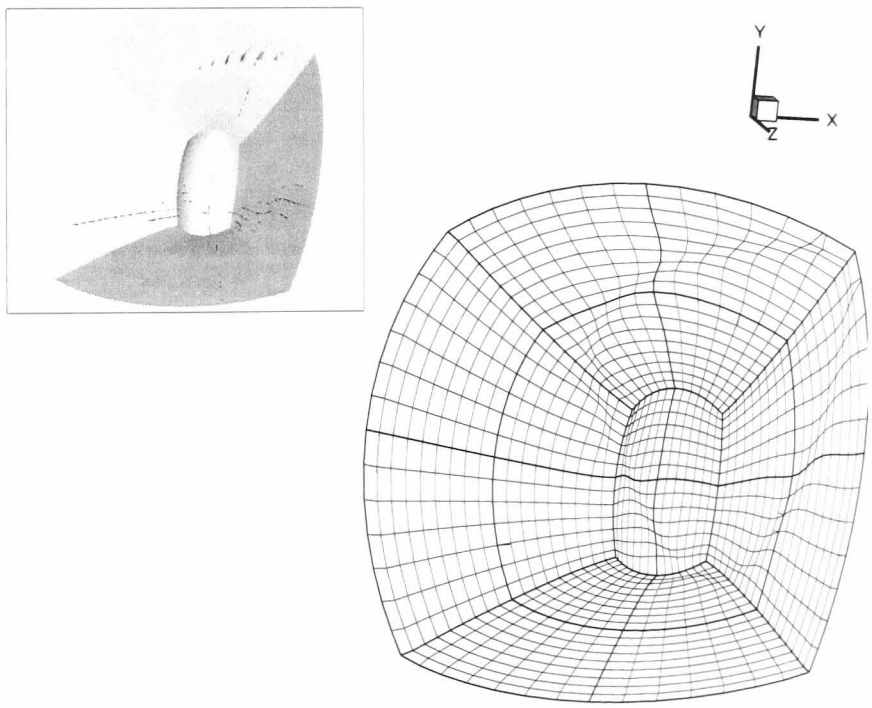


Figure 9.36: **The adaptive grid is visualized in shaded view in order to depict grid quality on the physical boundary.**

The methods for moving grid points on a fixed boundary or along an edge approximate the new positions either using partial differential equations or a

spline function. The new positions are, however, not exactly on the geometric surface. The tolerances between the geometric surface and adaptive points can be seen in the shaded view of Fig. 9.36.

9.2.3.3 Three–Dimensional Example 3

A generic wing placed in a large box is chosen as the next example. The trailing edge of the wing is rounded off, because the treatment of the trailing edge using a redistribution scheme is a special task, and it is beyond the intention of the present work. The outer boundary of the solution domain is represented by a large box.

The C–H–type topology is used for blocking the entire solution domain. The internal C–type block loop builds a *boundary layer topology*, wrapped by a H–type block topology. At the two–dimensional level, the grid consists of 16 blocks. This block topology is extended into the third dimension. In the third dimension, the grid is split into two parts, that is, the volume grid has 32 blocks in total, as shown in Fig. 9.37.

Three points on physical boundary are arbitrarily chosen to generate analytical weight variables. Using the same method as in the previous example, three parameters a, b, and c are determined by Eq. (9.57). Weight variables are generated in physical space, i.e., instead of using computational coordinates, physical coordinates are used in Eq. (9.58).

Block number is increased by splitting the volume grid into two parts in the third dimension. The topological complexity is increased by using a C–H block structure. It requires that block number and topological complexity should not play a role in grid adaptation.

The result is visualized in Fig. 9.38. Although the analytical weight variables lie on a plane that intersects both windward and leeward sides, as well as the outer boundary, C^1–continuity on the fixed boundary, which requires face overlap, as well as C^1–continuity across block interfaces, which requires block overlap, are retained.

After an adaptation process, the common vertices may have different positions

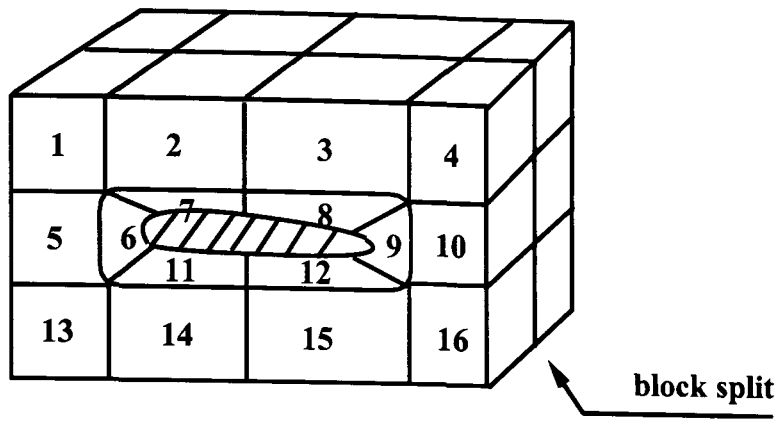


Figure 9.37: **The grid of the test example is generated using a two–dimensional C–H topology with 16 blocks. The grid is split in the third dimension. The block number is doubled, while block topology is unchanged in this dimension.**

in space. Calculating the mean values of their positions, the common vertices are moved to the position resulting from these mean values. In this example, the weight function goes through the common vertices. The result shows that the position of the common vertices is unique, as shown in Fig. 9.39.

9.3 Chapter Summary

The main contribution of the present chapter is that the method for grid adaptation is developed. The core of this method is that the grid point redistribution is formulated in the form of an objective function with control terms. The control terms are constructed using the measures for grid quality, such as orthogonality, cell aspect ratio, and cell smoothness. Different iterative methods for solution of objective functions for single–block grid adaptation are presented.

Furthermore, the strategy for extending the single–block case to the multiblock case is described. In order to adapt multiblock grids with complex geometry and block topology, grid points are classified according to their degrees of freedom of movement. In grid adaptation, they are treated sequentially by decreasing order of degrees of freedom. The sequence of grid adaptation follows the principle

- ▷ **Adapt interior grid points of all blocks**;
- ▷ **Update all grid points on block interfaces**;
- ▷ **Adapt external grid points with face constraint**;

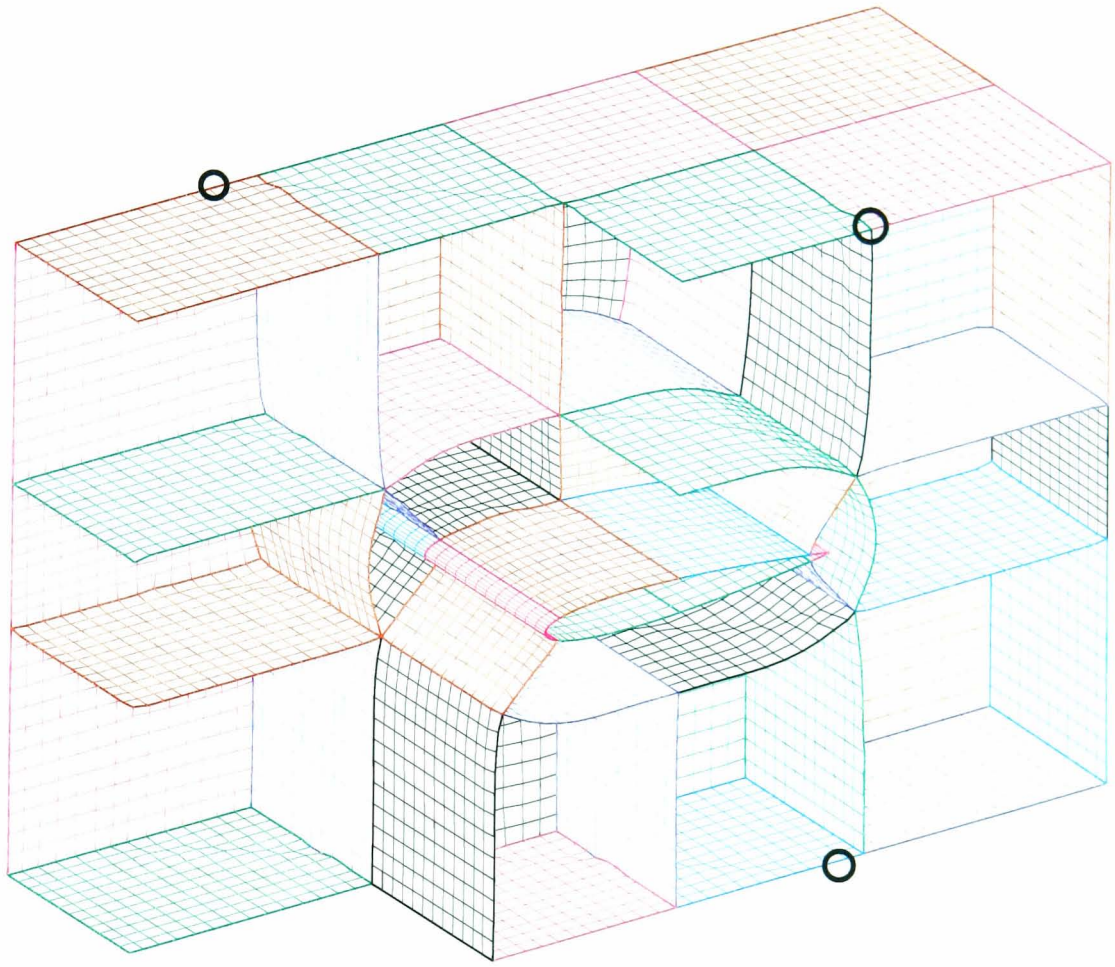


Figure 9.38: **Weight variables lie on a plane obtained by three arbitrarily chosen points. This plane intersects both the windward and leeward sides of the wing, as well as the outer boundary of the solution domain. The adaptation result shows C^1–continuity of grid lines across these boundaries.**

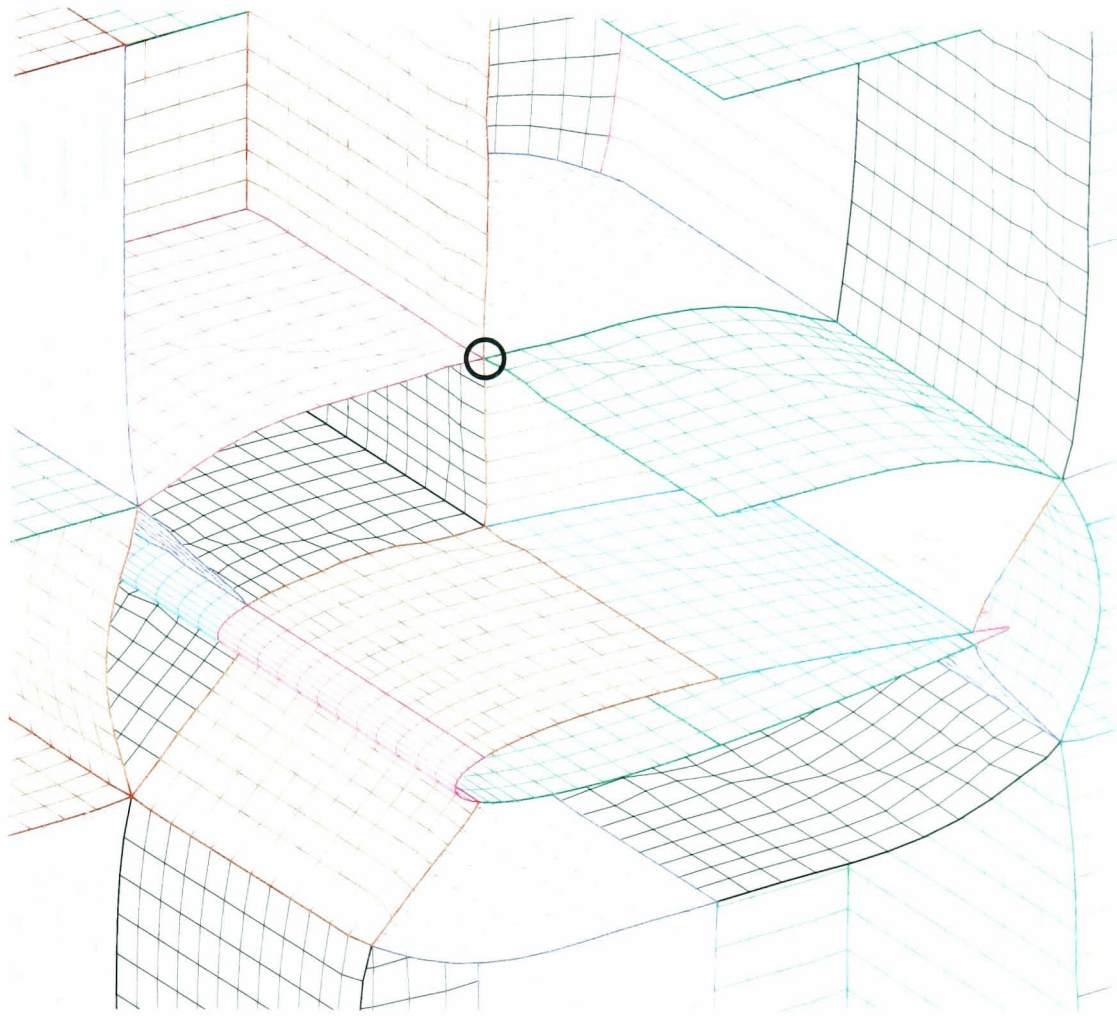


Figure 9.39: **A close up of the example of Fig. 9.38. The weight function goes through a common vertex. It can be seen that the adaptation at this point is undisturbed.**

- ▷ **Adapt external grid points with edge constraints;**
- ▷ **Update all common vertices.**

Chapter 10

Grid Adaptation Using Smart Cell Strategy

The essential core of a *smart topology*, presented in Chapter 8, is to control the local grid density using some special block structures. However, grid adaptation using the smart topology is usually performed during a grid generation process, and it is solution–independent.

The combination of a smart topology with cell refinement results in the concept of a smart cell adaptation [73]. This concept takes the advantage of a refinement scheme, namely, keeping an initial grid points fixed, and introducing new cells into an existing grid, but preserves a one–to–more connectivity among cells.

The method is initially developed for structured Cartesian grids and can easily be extended to body fitted grids in multiblock mode.

10.1 Concept of Smart Cells

The idea of a smart cell adaptation is based on the principle of cell splitting. In comparison with a traditional refinement scheme, new cells have a one–to–one connectivity on cell boundaries to each other.

10.1.1 Definition of a Smart Cell

Supposed that a two–dimensional structured grid is used as the initial grid. It is represented by

$$P = \{p_{i,j}(\mathbf{x}) \mid i = 1, \ldots, I; j = 1, \ldots, J\} \tag{10.1}$$

Grid points are ordered in a natural sequence shown in Fig. 10.1

$$P = \{p_{1,1}(\mathbf{x}), \ldots, p_{I,1}(\mathbf{x}), p_{1,2}(\mathbf{x}), \ldots, p_{I,1}(\mathbf{x}), \ldots, p_{I,J}(\mathbf{x})\}$$

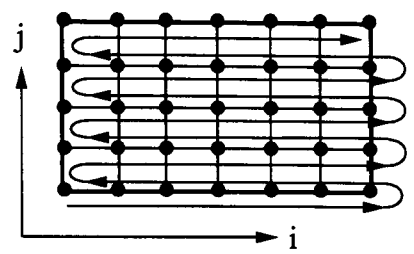


Figure 10.1: **Grid points of a structured mesh are ordered in a natural sequence indicated by arrows.**

Cells of the grid are denoted by

$$C_{i+\frac{1}{2},j+\frac{1}{2}} = \{p_{i,j}(\mathbf{x}), p_{i+1,j}(\mathbf{x}), p_{i,j+1}(\mathbf{x}), p_{i+1,j+1}(\mathbf{x})\} \tag{10.2}$$

where the $i = 1, \ldots, I-1$ and $j = 1, \ldots, J-1$, and smart cells are indexed by mid–points of the initial grid.

An arbitrary cell is chosen for describing the concept of smart cell, as shown in Fig. 10.2.

Definition 10.1: Smart cell. A smart cell consists of five small cells with a defined cell connectivity. Four cells are placed around the boundary of an initial cell. Each of them has a common side connected to four sides of the initial cell. The boundary of the central cell is connected to the four cells. The five small cells are termed *child–cells*.

In the following, four points of the central child–cell are denoted by $q_1(\mathbf{x})$, $q_2(\mathbf{x})$, $q_3(\mathbf{x})$ and $q_4(\mathbf{x})$, while the boundary of the smart cell is given by the four points $Q_1(\mathbf{x})$, $Q_2(\mathbf{x})$, $Q_3(\mathbf{x})$ and $Q_4(\mathbf{x})$, respectively, as shown in Fig. 10.2.

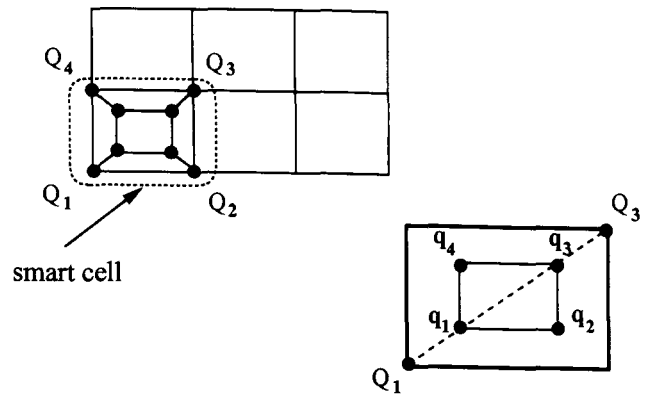


Figure 10.2: **In two–dimensional cases, a smart cell consists of five child cells. They have the defined cell connectivity as depicted.**

10.1.2 Generation of a Smart Cell

In order to avoid skewness of child–cells, the points $q_1(\mathbf{x})$ and $q_3(\mathbf{x})$, as well as $q_2(\mathbf{x})$ and $q_4(\mathbf{x})$ are generated over the diagonals $\overline{Q_1Q_3}$ and $\overline{Q_2Q_4}$, respectively. The ratio of the diagonal lengths of the initial and the central cells λ is a measure for the sizes of the five child cells. It is expressed by

$$\lambda = \frac{|q_3(\mathbf{x}) - q_1(\mathbf{x})|}{|Q_3(\mathbf{x}) - Q_1(\mathbf{x})|} \tag{10.3}$$

The size of child cells can be varied by specifying a λ value ($0 < \lambda < 1$). The points $q_1(\mathbf{x})$ and $q_3(\mathbf{x})$ are obtained by

$$\begin{aligned} q_1(\mathbf{x}) &= Q_1(\mathbf{x}) + \frac{1-\lambda}{2}[Q_3(\mathbf{x}) - Q_1(\mathbf{x})] \\ q_3(\mathbf{x}) &= Q_3(\mathbf{x}) + \frac{1-\lambda}{2}[Q_1(\mathbf{x}) - Q_3(\mathbf{x})] \end{aligned} \tag{10.4}$$

The points $q_1(\mathbf{x})$ and $q_3(\mathbf{x})$ are obtained in the similar manner.

10.2 Grid Adaptation Using Smart Cells

This type of cell refinement is applied in Cartesian grids. Computation is performed without coordinate transformation. In grid generation process, the main computational demands are the determination of flow features and the generation of smart cells.

10.2.1 Determination of Flow Features

Gradients or curvatures of flow variables are used for determination of flow features. Flow variables, chosen for this purpose, are normalized using Eq. (9.4) in section 9.1.2.1 (page 124). Instead of using weight functions, gradients of flow variables are used as measures for cell adaptation. Since smart cells are generated based on Cartesian grids, there is no need for coordinate transformation. Change of flow variables along cell boundaries are computed by

$$\begin{aligned}
\frac{\partial \phi_{j_{min}}}{\partial x_{i,j}} &= \frac{\phi_{i+1,j} - \phi_{i,j}}{x_{i+1,j} - x_{i,j}} \\
\frac{\partial \phi_{j_{max}}}{\partial x_{i,j}} &= \frac{\phi_{i+1,j+1} - \phi_{i,j+1}}{x_{i+1,j+1} - x_{i,j+1}} \\
\frac{\partial \phi_{i_{min}}}{\partial y_{i,j}} &= \frac{\phi_{i,j+1} - \phi_{i,j}}{y_{i,j+1} - y_{i,j}} \\
\frac{\partial \phi_{i_{max}}}{\partial y_{i,j}} &= \frac{\phi_{i+1,j+1} - \phi_{i+1,j}}{y_{i+1,j+1} - y_{i+1,j}}
\end{aligned} \tag{10.5}$$

where the subscripts j_{min}, j_{max}, i_{min} and i_{max} denote four boundaries of a cell. This can be seen that any small change of flow variables can be captured.

An immediate determination of flow features is to compare the flow variables normalized over two diagonals of a cell. It is expressed by

$$\begin{aligned}
\sigma_1 &= |\phi_{i,j} - \phi_{i+1,j+1}| \quad or \\
\sigma_2 &= |\phi_{i+1,j} - \phi_{i,j+1}|
\end{aligned} \tag{10.6}$$

where σ_i is specified as a measure for generating smart cells. In cases when $\sigma_1 > 0$ or $\sigma_2 > 0$, smart cells will be generated.

10.2.2 Adaptation Process

The smart cell scheme does not require data coupling on a block interface. Therefore, grid adaptation can be performed block–by–block. The adaptation process contains the following steps:

- ▷ **Step 1. Read a grid and flow variables**. Cartesian and block structured grids are read. Scalar quantities ϕ from flow solutions are chosen as flow variables.
- ▷ **Step 2. Normalize flow variables**. Flow variables are normalized using Eq. (9.4), so that they are dimensionless for computation.

- ▷ **Step 3. Determine the flow feature**. Changes of flow variables along cell boundaries are computed, e.g., using Eq. (10.5).
- ▷ **Step 4. Smart cell generation**. Comparing gradient values with the measure for smart cell generation specified, cells will be generated using Eq. (10.4).
- ▷ **Step 5. Smart cell data**. Nodes of the smart cell and its link relations are generated and saved in a list.

10.3 Results of Grid Adaptation Using Smart Cells

Shapes of smart cells considerably depend on those of their parent–cells. Therefore, three Cartesian grids with uniform grid line distribution are generated as test examples. The first and the third examples are two– and three–dimensional single block grids. Flow variables of both examples are generated analytically. The second example is selected from a real flow simulation.

10.3.1 Test Case 1: A Circle Weight Function

A single block grid is generated with 41×41 points in i– and j–directions, respectively. Solution domain is in the range $x, y \in [-20,\ 20]$. Grid lines are uniformly generated with the constant increments $\Delta x = 1$ and $\Delta y = 1$. Analytical flow variables are generated using the following equation

$$\phi_{i,j} = 1 + tanh\left[1 - \left(\frac{i}{10}\right)^2 - \left(\frac{j}{10}\right)^2\right] \tag{10.7}$$

where $i, j = -20, \ldots, 20$. The flow variables describe a circle in the solution domain.

Flow features are determined by Eq. (10.6), i.e., differences of flow variables over cell diagonals are computed, and they are compared with a σ value specified as the measure for smart cell adaptation. Sizes of smart cells are scaled by $\lambda = 0.3$. As result, 81 smart cells are generated during the adaptation process, as shown in Fig. 10.3.

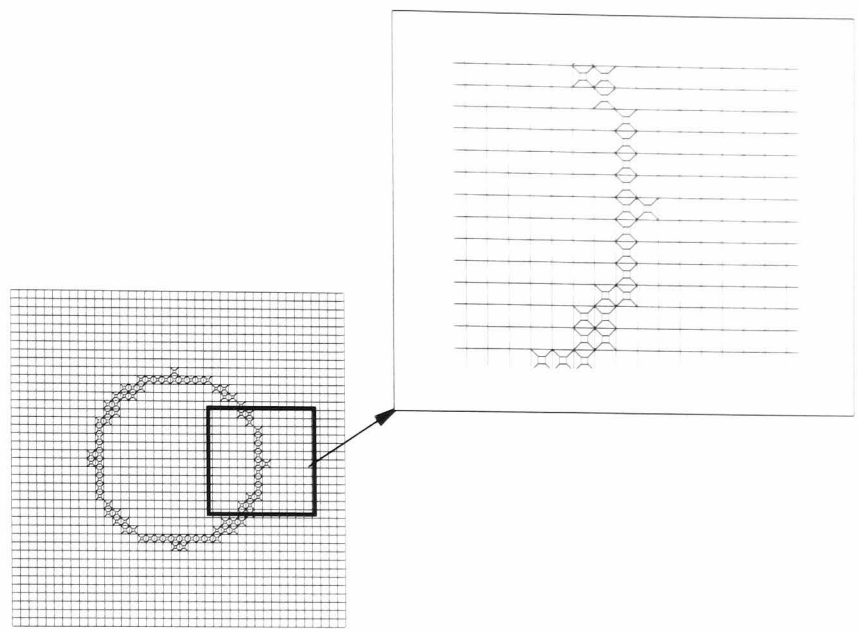


Figure 10.3: **First test example of smart cell adaptation. Analytical flow variables describe a circle in the solution domain.** 81 **smart cells are generated. The cells, within which large changes of flow variables are computed, are adapted by generated smart cells.**

10.3.2 Test Case 2: Multi–Block Forward Facing Step

Solution domain of the forward facing step is meshed by three blocks with 31×11, 31×41 and 121×41 grid points, respectively. Using the free stream condition $M_\infty = 3$, the flow is simulated.

The objective of the test is twofold. Firstly, the test case is chosen from a real flow simulation. The main interest focuses on shock wave reflection in the channel. Smart cells should be generated to increase grid density in region, where the shock wave is detected. Secondly, there is no process for block overlap, i.e., data coupling on block interfaces does not exist. The test should prove that there is no abrupt zone of smart cells on block interface can be retained using this scheme.

Using Eq. (10.6), the differences of normed flow variables over both diagonals of a cell are taken as the measure for generating smart cells. Restricting this measure within a small range, the grid is adapted, as shown in Fig. 10.4. There is no additional computation for identifying the position of shock wave or other flow phenomena. Besides the shock reflection, in other regions, where large flow gradients exist, smart cells are generated.

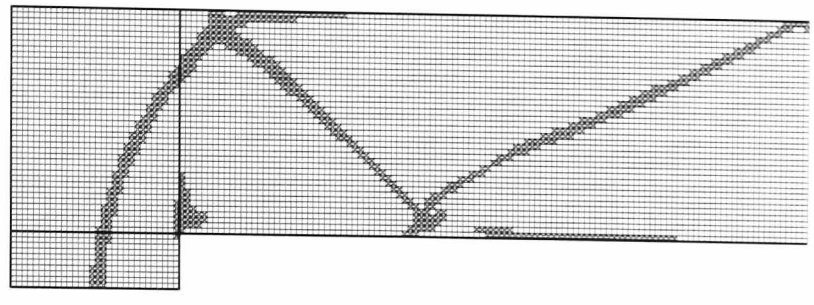


Figure 10.4: **Shock reflection of a forward facing step as adapted using the smart cell scheme.**

Fig. 10.5 shows smart cells on block interfaces. It is obvious that block boundary does not interfere in the process of smart cell generation.

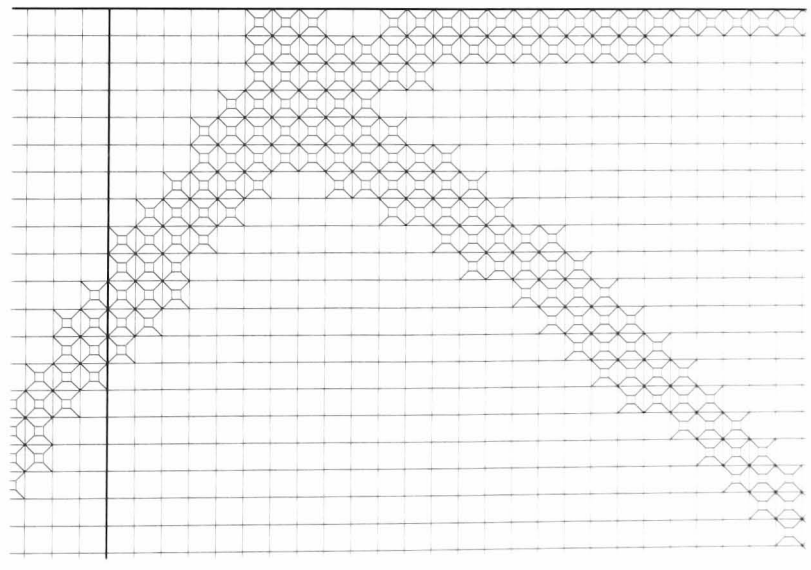


Figure 10.5: **Close up of smart cell adaptation of example** *forward facing step*.

10.3.3 Test Case 3: A Sphere in a Box

The grid is generated with $41 \times 41 \times 41$ grid points in the i, j and k–directions, respectively. The solution domain is in the range $x, y, z \in [-20, 20]$. Grid lines are uniformly generated with constant increments $\Delta x = 1$, $\Delta y = 1$ and $\Delta z = 1$, respectively. Similar to the first example, flow variables are generated using analytical equation

$$\phi_{i,j,k} = 1 + tanh\left[1 - \left(\frac{i}{10}\right)^2 - \left(\frac{j}{10}\right)^2 - \left(\frac{k}{10}\right)^2\right] \tag{10.8}$$

where $i, j, k = -20, \ldots, 20$. Flow variables represent a sphere in the solution domain. Flow features are computed by comparing the change of flow variables σ over four diagonals. Scaling the cell size with $\lambda = 0.3$, and restricting σ within an adequate range, smart cells are generated, as shown in Fig. 10.6.

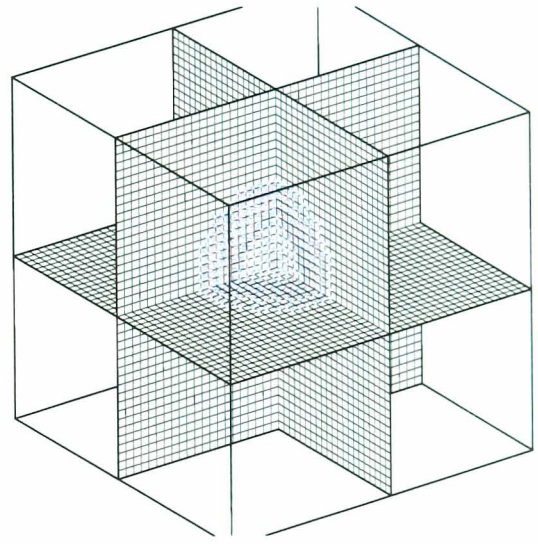


Figure 10.6: **A three–dimensional example of grid generation using smart cells.**

10.4 Discussion

Although the smart cell approach is exemplified using a Cartesian type mesh, the approach can be easily extended to non–uniform meshes in Cartesian or multiblock body fitted meshes.

However, the limitation of this scheme is that the quality of smart cells depends considerably on the shapes of the initial cells. Therefore, it is proposed that its application should be restricted to Cartesian grids with uniform grid line

distribution.

10.5 Chapter Summary

The main contribution of the present chapter is that a new concept of smart cell adaptation is developed based on the idea of cell refinement with smart topology, namely block refinement. A smart cell consists of five cells, and they are generated with an initial cell. Five cells have defined connectivity relations. The measure for a smart cell generation is gradients of flow variables along a cell boundary. Different from a traditional refinement scheme, the smart cell refinement keeps a one–to–one connectivity at cell boundaries.

Chapter 11

Conclusions and Future Work

The present thesis develops a strategy for object–oriented grid generation and its application to complex geometries. Two major topics are addressed in this thesis:

- ▷ the method for grid generation for highly complex geometries,
- ▷ the method for multiblock grid adaptation.

The objective of this thesis is to answer some general questions in grid generation. A summary of the thesis is given below.

11.1 Surface Description for Grid Generation

Data generation for surfaces used to define the boundary of a meshing domain is termed *surface description* in this thesis. The first question arising in dealing with creating a surface description is:

- *Which properties must a geometric configuration have, such that a meshing domain can be defined for it?*

In order to define the meshing domain, a geometric configuration must have several properties. In this thesis, these properties are summarized as *rigidity of the meshing object*, *boundary determinism of its surface* and *homogeneity of surface description* (see section 2.1.1 and [78]).

A geometric configuration may have a complex form with certain shape features. In most cases, the complete configuration is decomposed into a set of sub–components. Their boundaries will be described analytically or discretely in

a piecewise manner. The second question arising in dealing with decomposing a complex configuration is:

- *What are the measures for the decomposition of a complex geometric configuration?*

A complex configuration is decomposed with respect to the following points:

- ▷ *engineering features*, and
- ▷ *shape features.*

Decomposition of a complex geometric configuration into a set of components aims at reducing the geometric complexity of a configuration, and generating its surface geometry data in a simplified manner.

11.2 Grid Construction Rules

The principle of the object–oriented method for grid generation can be summarized as:

- ▷ *Domain decomposition for local wireframe building followed by integration of all local wireframe models.*

The essential core of this method is to divide a complete meshing task of highly geometric complexity into a set of sub–tasks. Their block topologies are built at a lower level of geometric and topological complexity. The question arising in block topology building for complex geometries is

- *Is it possible to establish a set of grid construction rules, such that a structured grid will be efficiently generated?*

Based on the experience in practical grid generation, several general methods are summarized as *Grid Construction Rules.* They deal with the tasks such as domain decomposition, surface description, topology building. A very complex example of an *Ariane 5 launcher* is given in order to explain the strategy of object–oriented grid generation. More examples are presented in [103] and [104].

11.3 Grid Adaptation

Instead of a homogeneous grid line distribution, grid density should be generated in a heterogenous manner in order to save memory. The question arising in dealing with grid adaptation is:

- *How does one generate a grid whose grid density is globally heterogenous?*

A heterogeneous distribution of grid points can be achieved in two ways. Firstly, local grid density is controlled by block topology, termed *passive grid adaptation.* Secondly, grid density is changed with respect to flow features, termed *active grid adaptation.*

11.3.1 Passive Grid Adaptation

Using three forms of block topologies, local grid density can be improved when it is required. These forms of grid line control do not depend upon flow solutions, and are therefore termed *passive grid adaptation.*

Boundary layer topology. A boundary layer topology builds a closed loop of blocks, which can provide a one–dimensional enrichment for grid lines on body surfaces. In three–dimensional cases of complex geometries, this type of grid provides the advantage of *one block topology for generating both Euler and Navier–Stokes grids.*

Block encapsulation. The boundary of a region in solution domain is fixed on a closed internal surface. At both sides of the closed surface, different grid spacings are possible. A local clustering of grid points can be encapsulated within this region.

Smart topology. A set of blocks is added to an existing grid through local refinement of the existing grid topology. The block refinement yields high local grid density.

The passive grid adaptation is solution–independent. However, its generation is manual. Future research topics are:

▷ **1. Automatic generation of a boundary layer topology.**

On a fixed boundary, block topology is two–dimensional (see *grid construction rule 3* in section 6.3.3.1). Extending this block topology in the third dimension. a boundary layer topology can be obtained. This principle can be implemented for an automatic generation of a boundary layer topology.

▷ **2. Automatic generation of smart topology, i.e., automatic block refinement.**

A possible extension of smart topology is to automatically refine blocks with respect to flow features. For instance, gradients of flow variables on block boundaries can be considered as measures for block refinement [31].

11.3.2 Active Grid Adaptation

In the present thesis, solution–based grid adaptation is termed *active grid adaptation.* The method for adapting a grid is to formulate the relations between grid spacings and weight functions in the form of objective functions. The measures for grid quality, such as cell smoothness, cell aspect ratio, and grid line orthogonality are used as penalty functions incorporated in the objective functions. An iterative method is used for solution of the objective functions.

In applying an algorithm in multiblock grid adaptation, the first question arising in dealing with C^1–continuity on block interfaces is:

- *How can one ensure C^1–continuity on block interfaces?*

Instead of merging blocks together, a *block overlap* technique is employed to retain C^1–continuity of grid lines on block interfaces. This type of block coupling necessitates data coupling among neighboring blocks.

The second question arising in dealing with active grid adaptation is:

- *How can one move grid points on a fixed boundary without destroying its geometry?*

Adaptation of these points is performed on two–dimensional parametric surfaces or along one–dimensional parametric curves. which describe the surface geometry

or sharp edge of the fixed boundary.

In a multiblock problem, adaptation of grid points with different geometric constraints requires more complex algorithms.

The third question dealing with multiblock grid adaptation is:

- *How can one design a procedure for adapting grid points with various degrees of freedom?*

The strategy for adapting a multiblock grid follows the principle:

▷ *reduction of degrees of freedom.*

Firstly, internal grid points, which have three degrees of freedom, are adapted. Then grid points on physical boundaries, which have reduced degrees of freedom are adapted.

11.3.3 Needs for Improvement of Adaptation Algorithm

In the test examples, the iteration number for minimization of the objective functions is manually specified. In practical application, the computation should be automatically terminated, when a global termination criterion for the search process is met.

The first unsolved question is

- *How can one define a global termination criterion for a grid adaptation process?*

In addition, the values of penalty functions are proportionally increased with iteration numbers, such that search steps become smaller, when the minimum of an objective function will be reached.

The second unsolved question is

- *How can one find a relation to increase penalty functions, such that a search step will vanish at the minimum of an objective function?*

In two–dimensional test cases, the control functions for orthogonality and aspect ratio are considered as the functions of the independent variables θ, s_1, and s_2 in the form $f(\mathbf{u}, \theta, s_1, s_2)$. A more precise description of orthogonality and aspect ratio is to use the independent variables in the physical space. Future work is to implement measures for cell orthogonality and aspect ratio in the form of $f(\mathbf{x})$.

11.4 Grid Adaptation Using Smart Cells

The concept of smart cell is applied in adapting Cartesian grids both in two– and three–dimensional cases. A problem with C^1–continuity on block interfaces does not exist. Since cell boundaries are connected in a one–to–one manner, it is possible to implement this scheme in a flow solver for dynamic grid adaptation.

In test examples, there is a large proportion of sheared cells. Their shapes and reduced alignment with flow directions may have an influence on the accuracy of flow simulations [62]. Future work will focus on the improvement of the adaptation algorithm in two aspects.

1. Reduction of shear cells. In two–dimensional cases, a smart cell will be generated within an initial cell. Four child–cells may be sheared. In order to reduce shear, smart cells are generated in the following manner:

- ▷ **Target the initial cells to be adapted**. Suppose that the cells to be adapted are determined, and they are targeted, as shown in Fig. 11.1(a).
- ▷ **Merge the targeted cells**. The neighboring relations among smart cells will be determined, such that they can be merged together, as shown in Fig. 11.1(b).
- ▷ **Generate smart cells**. Smart cells are linked to their neighboring smart cells or connected to the boundaries of their initial cells, as shown in Fig. 11.1(c).

2. Restriction of cell increment. A maximal percentage of cell size increment is specified. During grid adaptation, smart cells can be added to the solution domain as well as deleted from the solution domain. Smart cells will be generated interactively, so that the user can choose the region to be adapted in accordance with flow development, and specify the amount of cell increment.

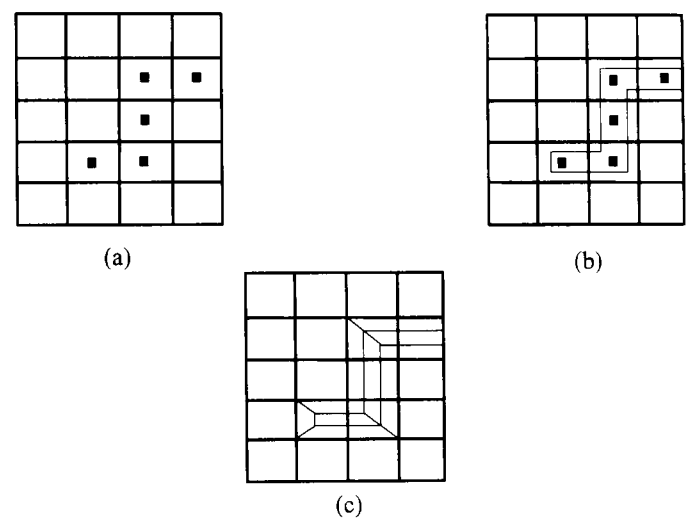


Figure 11.1: **A modification of smart cell concept.**

Bibliography

[1] Anderson, D. A.: Adaptive Grid Methods for Partial Differential Equations, *Advances in Grid Generation*, pp. 1–15, edited by K. N. Ghia, U. Ghia, Applied Mechanics, Bioengineering, and Fluid Engineering Conference, Houston, Texas, 1983.

[2] Baker, T. J.: Delaunay–Voronoi Methods, *Handbook of Grid Generation*, (16) 1–11, edited by J.F. Thompson, B.K. Soni, N.P. Weatherill, CRC Press Inc., 1999.

[3] Bennett, C. R., Patel, M. K., Zhao, J., Hu, L.: Grid Adaption Strategy Using Neural Network, *Numerical Grid Generation in Computational Fluid Simulations*, pp. 869–878, edited by M. Cross, B. K. Soni, J. F. Thompson, J. Häuser, P. R. Eiseman, Proceedings of the 6th International Conference, University of Greenwich, London, 1998.

[4] Bennett, C. R.: Dynamic Grid Adaptation Using the LPE Equation, *Ph.D. thesis, School of Mathematics, Statistics and Scientific Computing, University of Greenwich, London*, March 1999.

[5] Benson, R. A.: Development of a Time–Accurate Solution Algorithm Coupled to a Dynamic Solution–Adaptive Grid Algorithm with Applications to Generic Inlet/Diffuser Configurations, *Ph.D. Thesis, Department of Mechanical and Aerospace Engineering, North Carolina State University*, 1994.

[6] Berge, C.: *Topological Spaces.* Dover Publications, Inc. Mineola, New York, 1997.

[7] Bertin, D., Bohbot, J., Magnin, H.: Multiblock Meshing Refinement for Navier–Stokes Calculation, *Numerical Grid Generation in Computational Fluid Simulations*, pp. 131–140, edited by M. Cross, B. K. Soni, J. F. Thompson, J. Häuser, P. R. Eiseman, Proceedings of the 6th International Conference, University of Greenwich, London, 1998.

[8] Bockeli, M. J., Eiseman, P. R.: A Time–Accurate Adaptive Grid Method and the Numerical Simulation of a Shock–Vortex Interaction, *NASA Technical Paper 2998*, 1990.

[9] Bond, R. B., McRae, D. S.: A Conservative Interblock Communication Algorithm for Dynamically Discontinuous Block Boundary Grids, *Numerical Grid Generation in Computational Field Simulations*, pp. 111–120, edited by B. Soni, J. Häsuer, J. Thompson, P. Eiseman, Proceedings of the 7th

International Conference on Numerical Grid Generation, Whistler, Canada, September, 2000.

[10] Braaten, M. E., Connell, S. D.: Three–Dimensional Unstructured Adaptive Multigrid Scheme for the Navier–Stokes Equations. *AIAA Journal*, Vol. 34, No. 2, pp. 281–290, 1996.

[11] Brackbill, J. U., Saltzman, J. S.: Adaptive Zoning for Singular Problems in Two Dimensions, *Journal of Computational Physics*, Vol. 46, pp. 342–368, 1982.

[12] Castillo, J. E.: Discrete Variational Grid Generation, *Mathematical Aspects of Numerical Grid Generation*, edited by J.E. Castillo, Frontiers in Applied Mathematics 8, SIAM, Philadelphia, 1991.

[13] Catherall, D.: An Assessment of the Advantages of Adptivity on Structured Grids, *Numerical Grid Generation in Computational Field Simulations*, pp. 539–550, edited by N. P. Weatherill, P. R. Eiseman, J. Häuser, J. F. Thompson, Proceedings of the 4th International Conference, Swansea, Wales, 1994.

[14] Catherall, D.: Adaptivity via Mesh Movement with Three–Dimensional Block–Structured Grids, *Numerical Grid Generation in Computational Field Simulations*, pp. 57–66, edited by B. K. Soni, J. F. Thompson, J. Häuser, P. R. Eiseman, Proceedings of the 5th International Conference, Mississippi State University, 1996.

[15] Chan, W. M.: Hyperbolic Methods for Surface and Field Grid Generation, *Handbook of Grid Generation*, (5) 1–26, edited by J. F. Thompson, B. K. Soni, N. P. Weatherill, CRC Press Inc., 1999.

[16] Charakhch'yan, A. A., Ivanenko, S. A.: A Variational Form of the Winslow Grid Generator, *Journal of Computational Physics*, Vol. 136, pp. 385–398, 1997.

[17] Chawner, J., Steinbrenner, J., Wyman, N.: Hybrid Grid Generation for Complex Geometries Using GridGen, *Numerical Grid Generation in Computational Field Simulations*, pp. 417–426, edited by B. K. Soni, J. F. Thompson, J. Häuser, P. R. Eiseman, Proceedings of the 7th International Conference on Numerical Grid Generation, Whistler, Canada, September, 2000.

[18] Coirier, W. J., Powell, K. G.: Solution–Adaptive Cartesian Cell Approach for Viscous and Inviscid Flows, *AIAA Journal*, Vol. 34, No. 5, pp. 938–945, 1996.

[19] Dannehoffer, J. F. III: Grid Adaptation for Complex Two–Dimensional Transsonic Flows, *Ph.D. Thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology*, 1987.

[20] Dannehoffer, J. F. III: A Block–Structuring Technique for General Geometries, *AIAA–91–0145*, 1991.

[21] Dassault Systemes: *Mofit CATIA Solution User's Guide*, Dassault Systemes, 1997.

[22] Deutsche Gesellschaft für Luft– und Raumfahrt: Ariane 5: Mehr Leistung für neue Märkte, *Luft– und Raumfahrt*, J. 19, H. 4, pp. 22–24, 1998.

[23] Deutsche Gesellschaft für Luft– und Raumfahrt: Neue Ziele für Europas Raumfahrt, *Luft– und Raumfahrt*, J. 20, H. 3, p. 14, 1999.

[24] Dong, Z. M.: Design for Automated Manufacturing, *Concurrent Engineering*, pp. 207–233, edited by A. Kusiak, John Wiley & Sons Inc., 1992.

[25] Dwyer, H. A.: Grid Adaption for Problems with Separation, Cell Reynolds Number, Shock–Boundary Layer Interaction, and Accuracy, *AIAA Paper 83–0449*, presented at AIAA 21st Aerospace Sciences Meeting, Reno, Nevada, January, 1983.

[26] Eca, L.: Orthogonal Generation Systems, *Handbook of Grid Generation*, (7) 1–25, edited by J. F. Thompson, B. K. Soni, N. P. Weatherill, CRC Press Inc., 1999.

[27] Eiseman, P. R.: Orthogonal Grid Generation, *Numerical Grid Generation*, edited by J. F. Thompson, pp. 193–226, North–Holland, New York, 1982.

[28] Eiseman, P. R.: Adaptive Grid Generation, *Computer Methods in Applied Mechanics and Engineering*, Vol. 64, pp. 321–376, 1987.

[29] Eiseman, P. R., Cheng, Z. M.: *GridPro/az3000, User's Guide and Reference Manual*, PDC, 300 Hamilton Ave, Suite 409, White Plains, NY 10601, 1996.

[30] Eiseman, P. R.: Enrichment of Grid Lines via Block Localization, *Private Communication*, July, 1998.

[31] Eiseman, P. R.: Encapsulation of Partial Meshing Domains via Block Topologies, *Private Communication*, October, 1999.

[32] Fletcher, R., Reeves, C. M.: Function Minimization by Conjugate Gradients, *The Computer Journal*, Vol. 7, pp. 149–154, 1964–1965.

[33] Gaither, A.: An Efficient Block Detection Algorithm for Structured Grid Generation, *Numerical Grid Generation in Computational Field Simulation*, pp. 443–450, edited by B. K. Soni, J. F. Thompson, J. Häuser, P. R. Eiseman, Proceedings of the 5th International Conference, Mississippi State University, 1996.

[34] Goldberg, U., Peroomian, O., Chakravathy, S., Sekar, B.: Validation of CFD++ Code Capability for Supersonic Combuster Flowfields, *AIAA Paper No. 97–3271*, Seattle 1997

[35] Häuser, J., Paap, H.–J.: Boundary Fitted Conformed Coordinate Systems for Selected Two Dimensional Fluid Flow Problems, Part I, II, *Journal of Numerical Methods for Fluids*, Vol. 6, pp. 507–539, 1986.

[36] Häuser, J., Muylaert, J., Paap, H.–G., Spel, M., Eiseman, P. R.: Grid Generation for Spaceplanes, *International Space Course, TU München*, Germany, 1993.

[37] Häuser, J., Xia, Y., Spel, M., Paap, H.–G., Eiseman, P. R., Cheng, Z.–M.: Grid Generation for Microaerodynamics Simulations of the Cassini–Huygens Space Probe, *Numerical Grid Generation in Computational Field Simulation*, pp. 107–116, edited by B. K. Soni, J. F. Thompson, J. Häuser, P. R. Eiseman, Proceedings of the 5th International Conference, Mississippi State University, 1996.

[38] Häuser, J., Xia, Y., Muylaert, J., Paap, H.–J.: Euler and N–S Grid Generation for Halis Configuration with Body Flap, *Numerical Grid Generation in Computational Fluid Simulations*, pp. 887–900, edited by B. K. Soni, J. F. Thompson, J. Häuser, P. R. Eiseman, Proceedings of the 5th International Conference, Mississippi State University, 1996.

[39] Häuser, J., Xia, Y., Eiseman, P. R., Cheng, Z.–M.: Topology Databases Construction for Automatic Generation of Aircraft and Spacecraft, *DGLR–Jahrbuch 1997(II)*, pp. 1187–1191, DGLR, October, 1997, GW ISSN 0070–408, 1997.

[40] Häuser, Xia, Y., Spel, M.: Application of 3D Clamp Techniques, Grid Generation for Complex Geometries, *Proceedings of the 15th IMACS World Congress on Scientific Computation, Modeling and Applied Mathematics*, pp. 167–171, Berlin, Germany, 1997.

[41] Häuser, J., Eiseman, P. R., Xia, Y., Cheng, Z.–M.: Parallel Multiblock Structured Grids, *Handbook of Grid Generation*, (12) 1–26, edited by J. F. Thompson, B.K. Soni, N. P. Weatherill, CRC Press Inc., 1999.

[42] Hagmeijer, R.: Anisotropic Grid Adaptation Based on Diffusion Equations, *Numerical Grid Generation in Computational Field Simulations*, pp. 489–500, edited by N. P. Weatherill, P. R. Eiseman, J. Häuser, J. F. Thompson, Proceedings of the 4th International Conference, Swansea, Wales, 1994.

[43] Hagmeijer, R.: Adaptation of Structured Grids Based on Weighted Least Squares Formulations, *Ph.D Thesis, Faculty of Aerospace Engineering, Delft University of Technology*, 1997.

[44] Hosaka, M.: *Modeling of Curves and Surface in CAD/CAM*, Springer–Verlag Berlin Heidelberg, 1992.

[45] Ivanenko, S. A.: Adaptive–Harmonic Grid Generator, *Numerical Grid Generation in Computational Field Simulations*, pp. 166–176, edited by B. K. Soni, J. F. Thompson, J. Häuser, P. R. Eiseman, Proceedings of the 5th International Conference, Mississippi State University, 1996.

[46] Ivanenko, S. A.: Harmonic Mapping, *Handbook of Grid Generation*, edited by J. F. Thompson, B. K. Soni, N. P. Weatherill, CRC Press Inc., 1999.

[47] Ives, D., Miller, R., Siddons, W., and van Dyke, K.: Grid Generation – A View from the Trenches, *NASA Conference Publication 3291*, Proceedings of the Surface Modeling, Grid Generation and Related Issues in Computational Fluid Dynamics Workshop, NASA Lewis Research Center, Cleveland, OH, May 1995.

[48] Ives, D.: Unstructured Boundary Layer Grid Generation *Numerical Grid Generation in Computational Field Simulations*, pp. 13–25, edited by B. K. Soni, J. F. Thompson, J. Häuser, P. R. Eiseman, Proceedings of the 7th International Conference on Numerical Grid Generation, Whistler, Canada, September, 2000.

[49] Jacquotte, O.–P.: Recent Progress on Mesh Optimization, *Numerical Grid Generation in Computational Field Simulations*, pp. 581–596 edited by A. S. Arcilla, P. R. Eiseman, J. Häuser, J. F. Thompson, Proceedings of the 3th International Conference on Numerical Grid Generation, 1991.

[50] Jacquotte, O.–P., Coussement, G., Desbois, F., Gaillet, C.: Contribution to the Development of a Multiblock Grid Optimization and Adaption Code, *Multiblock Grid Generation*, pp. 225–262, Vieweg, 1993.

[51] Jacquotte, O.–P.: Grid Optimization Methods for Quality Improvement and Adaptation, *Handbook of Grid Generation*, (33) 1–33, edited by J. F. Thompson, B. K. Soni, N. P. Weatherill, CRC Press Inc., 1999.

[52] Jones, J. W., Weatherill, N.: A Parallel Environment for Large Scale Computational Engineering Simulations, *Numerical Grid Generation in Computational Field Simulations*, pp. 923–932, edited by B. K. Soni, J. F. Thompson, J. Häuser, P. R. Eiseman, Proceedings of the 7th International Conference on Numerical Grid Generation, Whistler, Canada, September, 2000.

[53] Kania, L.: Three–Dimensional Adaptive Grid Generation with Applications in Nonlinear Fluid Dynamics, *AIAA 92–0661*, 1992.

[54] Khamayseh, A., Kuprat, A.: Surface Grid Generation Systems, *Handbook of Grid Generation*, (9) 1–29, edited by J. F. Thompson, B. K. Soni, N. P. Weatherill, CRC Press Inc., 1999.

[55] Kroll, E.: Modeling and Reasoning for Computer–Based Assembly Planning, *Concurrent Engineering*, pp. 177–205, edited by A. Kusiak, John Wiley & Sons Inc., 1992.

[56] Kusiak, A.: *Intelligent Manufacturing Systems*, Prentice–Hall, Inc., 1990.

[57] Laflin, K. R., McRae, D. S.: Solution–Dependent Grid Quality, *Numerical Grid Generation in Computational Fluid Simulations*, pp. 119–130, edited by M. Cross, B. K. Soni, J. F. Thompson, J. Häuser, P. R. Eiseman, Proceedings of the 6th International Conference, University of Greenwich, London, 1998.

[58] Liseikin, V. D.: *Grid Generation Methods*, Spring–Verlag, 1999.

[59] Löhner, R., Cebral, J. R.: Parallel Advanced Front Grid Generation, Proceedings of the 8th International Meshing Roundtable, pp. 67–74, October, 1999.

[60] Lu, N., Eiseman, P. R.: A Grid Quality Manipulation System, *Numerical Grid Generation in Computational Fluids and Related Fields*, pp. 607–616, Proceedings of the 3th International Conference on Numerical Grid Generation in Computational Fluid Dynamics, edited by A. S. Arcilla, J. Häuser, P. R. Eiseman, J. F. Thompson, June, 1991.

[61] Marchant, M. J., Weatherill, N. P.: Unstructured Grid Generation for Viscous Flow Simulations, *Numerical Grid Generation in Computational Fluids and Related Fields*, pp. 151–162. Proceedings of the 4th International Conference on Numerical Grid Generation, edited by N. P. Weatherill, P. R. Eiseman, J. Häuser, J. F. Thompson, April, 1994.

[62] McRae, D. S.: A Dynamic Solution Adaptive Mesh Algorithm for Structured Meshes, *Lectures on Computational Error Analysis*, North Carolina State University, Raleigh, NC, USA, 1994.

[63] McRae, D. S.: r–Refinement Grid Adaptation Issues, *Numerical Grid Generation in Computational Fluid Simulations*, pp. 33–52, edited by M. Cross, B. K. Soni, J. F. Thompson, J. Häuser, P. R. Eiseman, Proceedings of the 6th International Conference, University of Greenwich, London, 1998.

[64] McRae, D. S.: Influence of Adapted Cell Shapes on Flow Simulations, *Private Communication*, September, 2000.

[65] Mortenson, M. E.: *Geometry Modelling*, second edition, John Wiley & Sons, Inc., 1997.

[66] Morris, M. R.: Two Dimensional Multiblock Grid Optimization by Variational Techniques, *Multiblock Grid Generation*, pp. 6–18, edited by N. P. Weatherill, M. J. Marchant, D. A. King, 1993.

[67] NASA Geometry Data Exchange Specification for CFD (NASA IGES), Ames Research Center, *NASA Reference Publication 1338*, 1994.

[68] Noble, S. S., Cordova, J. Q.: Blocking Algorithms for Structured Mesh Generation, *AIAA 92–0659*, 1992.

[69] Papalambros, P. Y., Wilde, D. J.: *Principles of Optimal Design: Modeling and Computation*, Cambridge University Press, 1988.

[70] Park, S., Lee, K.: A New Approach to Automated Multiblock Decomposition for Grid Generation: A Hypercube++ Approach, *Handbook of Grid Generation*, (10) 1–22, edited by J. F. Thompson, B. K. Soni, N. P. Weatherill, CRC Press Inc., 1999.

[71] Patel, M. K., Pericleous, K. A., Baldwin, S.: The Development of a Structured Mesh Grid–Adaptation Technique for Resolving Shock Discontinuities in Upwind Navier–Stokes Codes, *Journal of Numerical Mathematics in Fluids*, Vol. 20, pp. 1179–1197, 1995.

[72] Patel, M. K.: An Interactive Tool for Grid Quality Control and Cell Repair, *Private Communication*, September, 2000.

[73] Patel, M. K.: A Possible Extension of Smart Topology into Cell Adaptation, *Private Communication*, September, 2000.

[74] Pattan, B.: Satellite Systems: Principles and Technologies, Van Nostrand Reinhold, New York, 1993.

[75] Peraire, J., Peiró, J., Morgan, K., 1999: Advancing Front Grid Generation. *Handbook of Grid Generation*, (17) 1–22, edited by J. F. Thompson, B. K. Soni, N. P. Weatherill, CRC Press Inc., 1999.

[76] Ravishankar, L., Singh, K. D.: An Interactive Software for the Visualization and Analysis of 3D Multiblock Structured Grids and Flowfield. *Numerical Grid Generation in Computational Field Simulations*, pp. 839–850, edited by B. K. Soni, J. F. Thompson, J. Häuser, P. R. Eiseman, Proceedings of the 7th International Conference on Numerical Grid Generation, Whistler, Canada, September, 2000.

[77] Remotigue, M. G.: An Arbitrary Connected Structured Topology Model. *Numerical Grid Generation in Computational Field Simulations*, pp. 39–50, edited by B. K. Soni, J. F. Thompson, J. Häuser, P. R. Eiseman, Proceedings of the 7th International Conference on Numerical Grid Generation, Whistler, Canada, September, 2000.

[78] Requicha, A. G.: Representation for Rigid Solids, *Computing Surveys*, Vol. 12, No. 4, pp. 437–446, Dec. 1980.

[79] Rouse, F., Diwakar, P.: ICEM CFD Real Numerix^R MOM3D^{TM}: Meeting the Industrial Challenge of Anisotropic Mesh Adaptation, *Numerical Grid Generation in Computational Field Simulations*, pp. 851–860, edited by B. K. Soni, J. F. Thompson, J. Häuser, P. R. Eiseman, Proceedings of the 7th International Conference on Numerical Grid Generation, Whistler, Canada, September, 2000.

[80] Saltzman, J.: A Variational Method for Generating Multidimensional Adaptive Grids, *Mathematics and Computing, DOE/ER/03077–174*, Courant Mathematics and Computing Laboratory, New York University, 1982.

[81] Saunders, B., Wang, Q.: From 2D to 3D: Numerical Grid Generation and the Visualization of Complex Surfaces, *Numerical Grid Generation in Computational Field Simulations*, pp. 51–60, edited by B. K. Soni, J. F. Thompson, J. Häuser, P. R. Eiseman, Proceedings of the 7th International Conference on Numerical Grid Generation, Whistler, Canada, September, 2000.

[82] Shim, J., Chung, J., Lee, K. D.: A Grid Generation Strategy for CFD Analysis of Iced Airfoil Flows, *Numerical Grid Generation in Computational Field Simulations*, pp. 71–80, edited by B. K. Soni, J. F. Thompson, J. Häuser, P. R. Eiseman, Proceedings of the 7th International Conference on Numerical Grid Generation, Whistler, Canada, September, 2000.

[83] Shirsat, A., Gupta, S., Shevare, G.: Generation of Multi–Block Topology for Discretisation of Three–Dimensional Domains, *Computers and Graphics*, Vol. 23, pp. 45–57, 1999.

[84] Smith, R. E.: Transfinite Interpolation (TFI) Generation Systems, *Handbook of Grid Generation*, (3) 1–14, edited by J. F. Thompson, B. K. Soni, N. P. Weatherill, CRC Press Inc., 1999.

[85] Soni, B. K., Weatherill, N. P.: Geometry–Grid Generation. *Computer Science and Engineering Handbook*, pp. 791–819, edited by F. Kreith. CRC Press Inc., 1997.

[86] Spekreijse, S. P., 1999: Elliptic Generation Systems. *Handbook of Grid Generation*, (4) 1–49, edited by J. F. Thompson, B. K. Soni, N. P. Weatherill. CRC Press Inc., 1999.

[87] Steinbrenner, J. P., Chawner, J. R.: Incorporation of a Hierarchical Grid Component Structure into GRIDGEN. *AIAA–93–0429*. 31st Aerospace Sciences Meeting & Exhibit, Reno, NV, January 11–14, 1993.

[88] Thompson, J. F., Thames, F. C., Mastin, C. W.: Automatic Numerical Generation of Body Fitted Curvilinear Coordinate Systems for Field Containing Any Number of Arbitrary Two–Dimensional Bodies. *Journal of Computational Physics*, Vol. 15, pp. 299–319, 1974.

[89] Thompson, J. F., Warsi, Z. U. A., Mastin, C. W.: *Numerical Grid Generation: Foundations and Applications*, North Holland, 1985.

[90] Thompson, J. F.: A Reflection on Grid Generation in the 90s: Trends, Needs, and Influences, *Numerical Grid Generation in Computational Field Simulations*, pp. 1029–1110, edited by B. K. Soni, J. F. Thompson, J. Häuser, P. R. Eiseman, Proceedings of the 5th International Conference, Mississippi State University, 1996.

[91] Thompson, J. F., Weatherill, N. P.: Fundamental Concepts and Approaches. *Handbook of Grid Generation*, (1) 1–30, edited by J. F. Thompson, B. K. Soni, N. P. Weatherill, CRC Press Inc., 1999.

[92] Thornburg, H. J., Soni, B. K.: Weight Functions in Grid Adaptation. *Numerical Grid Generation in Computational Field Simulations*, pp. 551–562, edited by N.P. Weatherill, P.R. Eiseman, J. Häuser, Thompson, Proceedings of the 4th International Conference, Swansea, Wales, 1994.

[93] Uchitel, V., Harrand, V. J., Whitmire, J. B.: Automated, Parametric Geometry Modeling and Grid Generation for Complex Design Systems. *Numerical Grid Generation in Computational Field Simulations*, pp. 763–777, edited by B. K. Soni, J. F. Thompson, J. Häuser, P. R. Eiseman, Proceedings of the 7th International Conference on Numerical Grid Generation, Whistler, Canada, September, 2000.

[94] Ushakova, O. V.: Algorithm of Two–Dimensional Optimal Adaptive Grid Generation, *Numerical Grid Generation in Computational Field Simulations*, pp. 37–46, edited by B. K. Soni, J. F. Thompson, J. Häuser, P. R. Eiseman, Proceedings of the 5th International Conference, Mississippi State University, 1996.

[95] van Albada, G. D., van Leer, B., Roberts Jr., W. W.: A Comparative Study of Computational Methods in Cosmic Gas Dynamics. *Astronomy and Astrophysics*, Vol. 108, pp. 76–84, 1982.

[96] Venkatakrishnan, V.: Parallel Implicit Unstructured Grid Euler Solvers. *AIAA Journal*, Vol. 32, No. 10, pp. 1985–1991. 1994.

[97] Walshaw, C., Cross, M.: Load–Balancing for Parallel Adaptive Unstructured Grids, *Numerical Grid Generation in Computational Fluid Simulations*, pp. 781–792, edited by M. Cross, B. K. Soni, J. F. Thompson, J. Häuser, P. R. Eiseman, Proceedings of the 6th International Conference, University of Greenwich, London, 1998.

[98] Warsi, Z. U. A.: Fluid Dynamics: Theoretical and Computational Approaches, 2nd edition, CRC Press Inc., 1998.

[99] Weatherill, N. P.: An Introduction to Grid Generation Using the Multiblock Approach, *Multiblock Grid Generation*, pp. 6–18, edited by N. P. Weatherill, M. J. Marchant, D. A. King, 1993.

[100] Weatherill, N. P., Said, R., Morgan, K.: The Construction of Large Unstructured Grids by Parallel Delaunay Grid Generation. *Numerical Grid Generation in Computational Fluid Simulations*, pp. 53–78, edited by M. Cross, B. K. Soni, J. F. Thompson, J. Häuser, P. R. Eiseman, Proceedings of the 6th International Conference, University of Greenwich, London, 1998.

[101] Winslow, A. M.: Numerical Solution of the Quasilinear Poisson Equation in a Nonuniform Triangle Mesh, *Journal of Computational Physics*, Vol. 2, pp. 149–172, 1967.

[102] Wong, H., Xia, Y., Häuser, J.: Grid Topology Influence on X–38 Space Vehicle Simulations, Poster presentation at 6th International Conference on Numerical Grid Generation in Computational Field Simulation, University of Greenwich, July 1998.

[103] Xia, Y. Häuser, J., Muylaert, J., Spel, M., Walpot, L.: A General Grid Generation Strategy for Complex Aerospace Geometries, *Numerical Grid Generation in Computational Field Simulations*, pp. 145–160, edited by B. K. Soni, J. F. Thompson, J. Häuser, P. R. Eiseman, Proceedings of the 7th International Conference on Numerical Grid Generation, Whistler, Canada, September, 2000.

[104] Xia, Y. Häuser, J., Patel, M. K., Cross, M.: Feature–Oriented Grid Topology Design for Aerospace Configurations, to be published in *DGLR–Jahrbuch 2001*, September, 2001.

[105] Yagiu, T.: *Modeling Design Objects and Process*, Springer–Verlag Berlin Heidelberg 1991.

[106] Zhang, X. D., Trepanier, J.–Y., Reggis, M., Benmeddour, A.: Grid Influence on Upwind Schemes for the Euler and Navier–Stokes Equations. *AIAA Journal*, Vol. 34, No. 4, pp. 717–727, 1996.