Titre: Title:	A fundamental approach to the problem of domain decomposition in structured grid generation
Auteur: Author:	Pasquale Piperni
Date:	2003
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Piperni, P. (2003). A fundamental approach to the problem of domain decomposition in structured grid generation [Thèse de doctorat, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/7168/

In compliance with the
Canadian Privacy Legislation
some supporting forms
may have been removed from
this dissertation.

While these forms may be included
in the document page count,
their removal does not represent
any loss of content from the dissertation.

UNIVERSITÉ DE MONTRÉAL

A FUNDAMENTAL APPROACH TO THE PROBLEM OF DOMAIN DECOMPOSITION IN STRUCTURED GRID GENERATION

PASQUALE PIPERNI
DÉPARTEMENT DE GÉNIE MÉCANIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION
DU DIPLÔME DE PHILOSOPHIÆ DOCTOR (Ph.D.)
(GÉNIE MÉCANIQUE)
Septembre 2003

Your file Votre référence
ISBN: 0-612-86448-0
Our file Notre référence
ISBN: 0-612-86448-0

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée:

A FUNDAMENTAL APPROACH TO THE PROBLEM OF DOMAIN DECOMPOSITION IN STRUCTURED GRID GENERATION

présentée par: PIPERNI Pasquale
en vue de l'obtention du diplôme de: Philosophiæ Doctor
a été dûment acceptée par le jury d'examen constitué de:

M. GUIBAULT François, Ph.D., président
M. CAMARERO Ricardo, Ph.D., membre et directeur de recherche
M. DOMPIERRE Julien, Ph.D., membre
M. BEALE Steven, Ph.D., membre

Dedicated to Loretta, Andria and Elisia

Acknowledgements

I wish to thank my thesis supervisor, Dr. Ricardo Camarero, for his constant support, patience, and timely advice throughout the course of this work. I also want to thank the other members of the jury, Dr. François Guilbault, Dr. Julien Dompierre, and Dr. Steve Beale, for their many helpful comments and suggestions.

I want to thank Dr. Fassi Kafyeke and Dr. Farzad Mokhtarian, of the Advanced Aerodynamics Department at Bombardier Aerospace, for providing longstanding support for this work, which was funded in part by the Strategic Technology Budget of Bombardier Aerospace, and in part by the Defense Industrial Research (DIR) Program of the Canadian Department of National Defense. I also want to express my gratitude to Ms. Josée Boudreau, also of the Advanced Aerodynamics Department, for her valuable contribution over many years to the successful application of the MBGRID program at Bombardier Aerospace.

On a more personal level, I'm deeply indebted to my wife, Loretta, for her constant devotion and energy, and for often taking on more than her fair share while I completed this work. Finally, a very heartfelt "thank you" goes to my wonderful daughters, Andria and Elisia, for giving me a sense of balance, and for constantly inspiring me with their smiles and filling my days with genuine love and affection.

Résumé

Une nouvelle méthode est présentée pour l'automatisation du processus de génération de maillage dans des domaines non simplement connexes. Dans cette méthode, le problème de décomposition du domaine est posé sous forme de problème aux conditions limites dans lequel la topologie du maillage est définie par le choix des conditions limites sur les frontières du domaine. L'automatisation du processus de décomposition est accomplie par la transformation de l'espace physique à un espace topologique, où le problème de décomposition est sujet à une solution rigoureuse. Une fois que le domaine est décomposé dans l'espace topologique, le maillage est généré dans l'espace physique par la solution d'équations différentielles partielles non linéaires et elliptiques, qui tiennent compte de la courbure de l'espace physique. La forme des surfaces de décompositions est obtenue directement de la solution de l'opérateur différentiel. Ce dernier est résolu dans tout le domaine non simplement connexe, par l'intermédiaire de sous-domaines chevauchants et dans lesquels seuls les points sur les frontières sont fixés à l'avance.

Il est démontré que la modélisation de la courbure du domaine est un élément essentiel d'une approche fondamentale de décomposition du domaine. Dans un espace courbé, la courbure des surfaces de décompositions doit suivre la courbure de l'espace pour produire un maillage de haute qualité. Puisque la décomposition du domaine est définie dans l'espace topologique, la courbure de l'espace physique doit être réinjectée dans le système par l'entremise d'un opérateur différentiel approprié. Un nouvel opérateur a été dérivé et est présenté pour ce besoin. La dérivation de cet opérateur

est entièrement générale et peut être appliquée à des domaines bidimensionnels ou tridimensionnels de formes arbitraires.

Le mariage d'un opérateur différentiel qui modélise la courbure de l'espace physique avec une méthode de décomposition de l'espace topologique permet la génération automatique de maillages structurés dans des domaines non simplement connexes. Cette approche peut être appliquée à des géométries convexes ou non convexes, et à des topologies complexes.

La thèse est divisée en quatre chapitres. Le premier chapitre donne un sommaire des développements principaux dans les méthodes de décomposition d'un domaine depuis les vingts dernières années. Dans le deuxième chapitre, le fondement des idées qui même au développement d'une nouvelle méthode pour la décomposition d'un domaine est présenté. La dérivation d'un opérateur différentiel qui tient compte de la courbure de l'espace est ensuite présentée au chapitre 3, suivi par les applications au chapitre 4.

Abstract

A new approach is presented for the automation of structured grid generation in multiply-connected domains. In this approach, the domain decomposition problem is cast as a classical boundary value problem in which the mesh topology is defined through the imposition of appropriate boundary conditions on the domain boundaries. The automation of the domain decomposition process is achieved by transferring it from the physical space to the topological space, where it is amenable to a rigorous solution. Once the domain is decomposed in the topological space, the mesh is generated in the physical space via the solution of a non-linear elliptic partial differential operator which takes into account the curvature of the physical space. The forms of the decomposition surfaces are obtained as part of the solution of the differential operator. The latter is solved iteratively in a system of overlapping sub-domains in which the decomposition surfaces are left floating, and in which only the shape of the domain boundaries and the point distribution thereon influence the form of the final mesh.

It is shown that the proper representation of domain curvature is an essential element to the success of the domain decomposition strategy. In any curved space, the curvature of the decomposition surfaces must closely mirror the curvature of the space in order to yield a high quality mesh. Since the decomposition of the multiply-connected domain is done in the topological space, the curvature of the physical space must be re-injected into the system through the solution of an appropriate differential operator. A new mathematical formulation is derived for this purpose and takes the form

of a new forcing function in the elliptic grid generation equations. This new curvature term is completely general and can be applied to both two- and three-dimensional domains of arbitrary shape.

The combination of the new grid generation equations and the domain decomposition strategy provides a methodology for generating structured meshes in multiply-connected domains without the requirement of a manual decomposition of the space. This method can be applied to both convex and non-convex geometries, and can be used to generate meshes of any topology for which the curvilinear coordinates are continuous.

The presentation of the subject matter is divided into four chapters. In Chapter 1, an overview of the various domain decomposition techniques developed over the last two decades is provided. In Chapter 2, the basic principles leading to the development of the new domain decomposition approach are presented, followed in Chapter 3 by the derivation the new grid generation equations. The viability of the overall method is then demonstrated in Chapter 4 with applications in two and three dimensions.

Condensé en Français

Introduction

En dépit des nombreuses années de recherche qui ont étées dévouées à l'automatisation du processus de génération de maillages structurés, le niveau d'effort requis pour mailler des domaines complexes est encore considéré comme étant trop élevé (Park and Lee, 1999). Ceci n'est pas surprenant étant donné le caractère fondamental du problème. Dans la science des algorithmes, le problème de décomposer une région bidimensionnelle non simplement connexe en quadrilatères convexes a été classé comme étant un problème "NP-Dur" (Lubiw, 1985). Les problèmes faisant partie de cette classe sont considérés comme étant essentiellement insoluble. Conséquement, les solutions de ce type de problème sont nécessairement de caractère approximatif ou doivent employer des méthodes heuristiques. Ceci n'est pas le cas pour les maillages non structurés, puisque des algorithmes rigoureux pour décomposer un espace en triangles (simplexes) sont connus depuis plus de soixante ans (O'Rourke, 1993).

Par ailleurs, le problème de décomposition pour un maillage structuré est davantage compliqué par le fait que la topologie de décomposition n'est généralement pas unique. Il peut y exister plusieurs décompositions valides pour un espace donné. Ceci implique qu'un algorithme de décomposition doit pouvoir "choisir" une topologie parmi plusieurs, et satisfaire les exigences d'utilisateurs divers dont les besoins peuvent être très différents d'un problème à l'autre.

Donc, les problèmes fondamentaux auquels il faut s'addresser pour développer une solution rigoureuse au problème de décomposition d'un domaine sont les suivants:

1. la décomposition d'un espace non simplement connexe n'est pas unique
2. le problème de décomposition d'un domaine est NP-Dur
3. la solution doit être applicable à des domaines de complexité arbitraire.

Sélection de topologie: un problème aux conditions limites

Les lignes d'un maillage structuré peuvent être interprétées comme étant des lignes d'iso-potentiels ou des lignes de courant. Dans cette optique, il est facile de voir qu'un maillage structuré peut être obtenu par la solution d'un système d'équations différentielles partielles. Ceci implique que le problème de génération de maillage se ramène essentiellement à un problème aux conditions limites.

Dans un espace tridimensionnel, si (ξ, η, ζ) représentent les coordonnées curvilignes et $\mathbf{r} = \mathbf{r}(\xi, \eta, \zeta)$ les coordonnées physiques du maillage, alors le problème aux conditions limites consiste à prescrire les valeurs de (ξ, η, ζ) sur les frontières et à résoudre les fonctions $\mathbf{r} = \mathbf{r}(\xi, \eta, \zeta)$. Puisque des conditions limites différentes mènent à des topologies différentes, il est clair que la sélection de la topologie d'un maillage est aussi un problème aux conditions limites.

Donc, le choix d'une topologie ne doit pas être le résultat d'une méthode de décomposition mais plutôt un choix fait par l'utilisateur par l'entremise des conditions limites.

La notion d'un problème aux conditions limites et la solution d'une équation différentielle ne présentent aucune nouveauté dans le domaine de génération de maillage, puisque cette notion est implicite dans les méthodes classiques telles que les transformations conformes. Toutefois, l'objectif ici est de généraliser cette approche pour un

domaine tridimensionnel non simplement connexe, et de générer un maillage dans ce domaine sans le besoin d'une décomposition manuelle de l'espace.

Pour un domaine simplement connexe, la génération d'un maillage structuré implique la transformation de l'espace physique vers un espace topologique qui est toujours en forme de rectangle. Par contre, si le domaine est non simplement connexe, l'espace physique se transforme en un espace topologique qui prend la forme d'une région polygonale "*rectiligne*". (Une région polygonale est un polygone avec un ou plusieurs "trous" internes; une région polygonale rectiligne en est une dont les côtés sont soit verticaux soit horizontaux). En plus, la *forme* de cette région polygonale dépend des conditions limites de la transformation. Ceci est une propriété importante des espaces non simplement connexes, puisque cela implique que des conditions limites uniques mènent à un espace topologique unique, qui, une fois décomposé, mène à une topologie de décomposition unique pour l'espace physique.

Décomposition de l'espace topologique

Tel que mentionné ci-haut, le problème de décomposer une région polygonale bidimensionnelle non simplement connexe en quadrilatères convexes a été classé comme problème NP-Dur. Par contre, il a aussi été démontré que si la région polygonale en question est *rectiligne*, un algorithme rigoureux existe pour sa décomposition (Lipski et al., 1979). Ceci est un résultat important puisque l'espace topologique est toujours rectiligne.

Donc, l'automatisation du processus de décomposition ne peut être réalisé que si la décomposition est faite dans l'espace topologique et non dans l'espace physique. Alors il n'est pas surprenant que les méthodes de décomposition existantes les plus efficaces ont toutes utilisé cette approche d'une manière ou autre (Shaw and Weatherill, 1992; Allwright, 1988; Park and Lee, 1999; Dannenhoffer III, 1996).

Toutefois, aucune de ces méthodes aborde le problème de la transformation inverse,

c'est-à-dire le problème de transformer la décomposition topologique en une décomposition physique de l'espace. Dans la plupart des cas, ce processus nécessite une interface graphique et l'intervention de l'utilisateur, ce qui implique en fait que ces méthodes ne sont pas entièrement automatisées.

D'un point de vue fondamental, un système de coordonnées optimales dans un espace généralement courbé doit être obtenu par l'entremise d'une solution d'un opérateur différentiel qui tient compte de la courbure de l'espace. La forme des surfaces de décomposition doit, idéalement, être le résultat d'une telle solution et non une forme arbitraire imposée par un utilisateur à partir d'une interface graphique.

On peut donc conclure qu'une méthode formelle pour automatiser la décomposition d'un espace physique doit, d'une part, incorporer un algorithme rigoureux pour la décomposition de l'espace topologique, et d'autre part pouvoir projeter la décomposition de l'espace topologique dans l'espace physique sans intervention de la part de l'utilisateur.

Algorithmes de décomposition

L'algorithme développé par Lipski et al. (1979) donne la décomposition d'une région polygonale rectiligne bidimensionnelle en rectangles. Cet algorithme garantit que la décomposition donne le nombre minimum de rectangles qui couvrent la région. Les rectangles générés par cet algorithme sont disjoints, et il sont designés rectangles de décomposition.

Un deuxième ensemble de rectangles, également important dans le contexte présent, ont été définis par Lodi et al. (1979). Ce deuxième ensemble comprend les rectangles designés "primes". Cet ensemble contient tous les rectangles distincts de grandeur maximale qui peuvent être inclus dans le polygone. Contrairement aux rectangles de décomposition, les rectangles primes peuvent se chevaucher.

L'importance des rectangles primes dans le contexte présent est qu'ils forment un

ensemble de zones chevauchantes qui peut être utilisé pour optimiser le maillage. Ceci fait en sorte que le maillage final ne dépend que des points sur les frontières (internes et externes) du domaine, et que les surfaces de décomposition demeurent flottantes durant l'optimisation.

Les algorithmes de Lipski et Lodi s'appliquent seulement à des domaines bidimensionnels. Donc, pour la décomposition d'un domaine tridimensionnel, il est nécessaire de subdiviser le domaine en une série d'extrusions bidimensionnelles, et ensuite d'appliquer les algorithmes de décomposition à chaque zone extrudée. Puisque tout espace topologique peut être subdivisé en un nombre fini d'extrusions bidimensionnelles, cette approche peut être appliquée à des espaces topologiques tridimensionnels de formes arbitraires.

Projection de la décomposition topologique à l'espace physique

Une fois que l'espace topologique est décomposé, l'étape suivante est de projeter cette décomposition dans l'espace physique. Puisque l'espace topologique est démuni de courbure, sa décomposition ne donne aucune information sur les formes des surfaces de décomposition dans l'espace physique. Il est évident que la courbure des surfaces de décomposition doit suivre la courbure de l'espace physique, et donc l'opérateur différentiel qui est utilisé pour générer le maillage dans l'espace physique doit tenir compte de la courbure de ce dernier.

L'opérateur différentiel le plus souvent utilisé pour générer les maillages structurés est basé sur l'équation de Laplace (Thompson et al., 1985) avec des transformations spéciales pour contrôler les distributions de points et l'orthogonalité du maillage (Spekreijse, 1995, 1999). Cet opérateur peut être utilisé pour contrôler la plupart des caractéristiques d'un maillage. Par contre, il est reconnu que les systèmes basés sur le Laplacien ont tendance à diffuser la courbure du maillage, ce qui engendre des expansions (ou contractions) du maillage dans les zones courbées. Pour cette raison, un nouvel opérateur a été dérivé pour incorporer correctement la courbure de l'espace.

Il est démontré que ce nouvel opérateur differentiel retient tous les avantages de l'équation de Laplace sans la diffusion de la courbure. La dérivation du nouvel opérateur est complètement générale et s'applique pour des maillages bidimensionnels et tridimensionnels.

Applications

La nouvelle méthode de décomposition du domaine est appliquée à plusieurs cas bidimensionnels et tridimensionnels qui démontrent le fonctionnement de la méthode. Les cas bidimensionnels comprennent entre autres, un échangeur de chaleur et un système hyper-sustentateur à trois éléments. En trois dimensions, des maillages sont générés pour une configuration d'avion en forme d'aile-fuselage et pour un avion complet. Dans tous les cas, le domaine est décomposé automatiquement à partir des conditions limites appliquées sur les frontières.

Conclusions

Une nouvelle méthode est présentée pour l'automatisation du processus de génération de maillage dans des domaines non simplement connexes. Dans cette méthode, le problème de décomposition du domaine est posé sous la forme d'un problème aux conditions limites dans lequel la topologie du maillage est définie par le choix des conditions limites sur les frontières du domaine. L'automatisation du processus de décomposition est réalisée par la transformation de l'espace physique vers un espace topologique, où le problème de décomposition est le résultat d'une solution formelle. Une fois que le domaine est décomposé dans l'espace topologique, le maillage est généré dans l'espace physique par la solution d'équations différentielles partielles non linéaires et elliptiques, qui tiennent compte de la courbure de l'espace physique. La forme des surfaces de décomposition est obtenue directement de la solution de l'opérateur différentiel. Ce dernier est résolu dans tout le domaine non simplement connexe, par l'intermédiaire de sous-domaines chevauchants et pour lesquels seuls les

points sur les frontières sont fixés à l'avance.

Il a été démontré que la modélisation de la courbure du domaine est un élément essentiel d'une approche fondamentale de décomposition du domaine. Dans un espace courbé, la courbure des surfaces de décomposition doit suivre la courbure de l'espace pour produire un maillage de haute qualité. Puisque la décomposition du domaine est définie dans l'espace topologique, la courbure de l'espace physique doit être réinjectée dans le système par l'entremise d'un opérateur differentiel approprié. Un nouvel opérateur a été dérivé à cette fin. La dérivation de cet opérateur est entièrement générale et peut être appliqué à des domaines bidimensionnels ou tridimensionnels de formes arbitraires.

Le mariage d'un opérateur différentiel qui modélise la courbure de l'espace physique avec une méthode de décomposition de l'espace topologique permet la génération automatique de maillages structurés dans des domaines non simplement connexes. Cet approche peut être appliquée à des géométries convexes ou non convexes, et à des topologies complexes.

Table of Contents

List of Figures

Introduction

The field of numerical grid generation came into being over the last four decades, in parallel with the broader field of computational fluid dynamics. The earliest algorithms were developed by Winslow (1966), Barfield (1970), Chu (1971), Godunov and Prokopov (1972), and Amsden and Hirt (1973). These early methods involved the mapping of a two-dimensional physical region onto a regular computational domain through the numerical solution of Laplace's equation. Thompson et al. (1974) later extended this approach to multiply-connected regions containing any number of arbitrarily shaped bodies. The latter work, although limited to two-dimensional applications at the time, effectively demonstrated that it was possible to numerically generate body-fitted coordinate systems about geometries of arbitrary complexity. It was also about this time that the first finite difference solutions of the transonic potential equation were obtained on simple meshes in two-dimensions (Murman and Cole, 1971) and in three-dimensions (Jameson, 1974). As computational algorithms matured and computing power increased, the field of numerical grid generation grew in scope and by the late 1970's numerous approaches were developed for different applications. The first dedicated conferences on the subject were held in 1980 (Smith, 1980) and 1982 (Thompson, 1982). The proceedings from the latter conference provided a comprehensive survey of the field, covering virtually all methods of grid generation known at the time, including algebraic and partial differential equation methods, as well as conformal mapping, variational, adaptive, and singularity methods. Multi-block methods, which were first investigated by Ruppert and Lee (1980), were also discussed in this reference. Thus, by the early 1980's, the theoretical foundations for

numerical grid generation were in place.

From the mid to late 1980's, further developments in structured grid generation focused on improving the robustness, versatility and efficiency of the various methods, and on their application to increasingly complex three-dimensional configurations. This period also saw the development of various automation techniques in the area of domain decomposition, and the proliferation of multi-block methods in general. At the same time, unstructured grid generation methods were emerging as alternatives to structured methods. By the 1990's, structured and unstructured grids were being generated fairly routinely on complex three-dimensional geometries. The last decade also saw the development of workstation-based grid generation "systems", complete with CAD tools and graphical user interfaces.

Today the field of grid generation is quite mature, with countless industrial applications in aerodynamics, thermodynamics, and hydrodynamics, among others. At the same time, the complexity of the meshing tasks continue to increase, often exposing the limits of the technology. In high Reynolds-number fluid dynamics, such as the aerodynamic flow over commercial transport aircraft, the increasingly routine solutions of the Navier-Stokes equations on complex geometry are placing high demands on both the quality and efficiency of grid generation.

In structured grid generation, the automation of the domain decomposition process is still an active area of investigation. This part of grid generation is still considered a bottleneck, and often limits the range of application of structured flow solvers. Despite the wealth of decomposition methods that have been developed over the years, a fundamental solution to this problem has yet to be presented, and this is the motivation for the present work.

The objective of this thesis is to formulate a new approach to domain decomposition in structured grid generation. The approach taken is to develop a methodology from first principles and to expose the fundamental nature of the problem, without relying on heuristic algorithms or on a "bag of tricks" approach.

The environment for the implementation, validation, and application of this work is the Advanced Aerodynamics Department of Bombardier Aerospace, where the work was carried out. The developments presented here represent an enhancement to the grid generation capability previously developed at Bombardier Aerospace, and some of the applications included herein are drawn from this industrial context.

In order to place the work in its proper context, an overview of the various domain decomposition techniques developed over the last two decades is provided in Chapter 1. A description of the new domain decomposition methodology is then presented in Chapters 2 and 3, followed by applications in Chapter 4.

Chapter 1

Overview of Domain Decomposition Techniques

1.1 Background

The most common approach to structured grid generation is to subdivide the domain into smaller and simpler regions called "blocks", in a process known as multi-block grid generation. The development of this approach was motivated by the need to simplify the task of meshing complex geometries in order to make the problem more manageable. However, the segmentation of the physical space into simpler regions also gives rise to the additional task of domain decomposition and connectivity generation between the various regions of the domain.

In the generation of a multi-block mesh, various choices can be made with respect to the mesh topology. In the physical space, there are essentially five levels of grid continuity that can be imposed across block interfaces. These are:

1. complete point and slope continuity,

2. point continuity,

3. partial (1 to n) point continuity,

4. no point continuity (on contiguous, non-overlapping boundaries),

5. no continuity with overlapping boundaries.

For continuity types 1), 2) and 3), an additional variation involves whether the curvilinear coordinate species are continuous across block interfaces. For instance, a curvilinear coordinate may change species or direction across a block interface, even though the mesh is continuous in the physical space. In general, the less continuity (whether physical or topological) is imposed at the block boundaries, the greater the freedom in the domain decomposition, and the easier it is to generate the mesh. However, lesser degrees of mesh continuity also place greater demands on the flow solver.

The most accurate block interface type, from the point of view of the flow solver, is one that maintains complete grid continuity across block boundaries. In this case, each boundary point on a block face will have an identical image on the neighboring block face, and the slope of the grid line transverse to the block face is continuous across the face. Complete continuity at the interfaces implies that no special boundary treatment is required in the flow solver, although it is generally more difficult to generate the grid.

A more easily implemented interface condition is point continuity only. In this approach, each block can be generated independently, as long as the grid points on the interface are common. This relaxed requirement reduces the demands on the grid generator, but can also introduce solution errors, depending on the type of flow solver used, due to slope discontinuities at the interface points. A less restrictive but related interface condition is one in which there is a "1 to n" matching of the points at the interface. This would occur for instance when a fine mesh is desired in one block, but only a coarse mesh is needed in the adjacent block, in which only one out of every n points match on the interface boundary (Raj et al., 1987). Methods that maintain no point continuity at all on a shared interface have similar characteristics, except

that a more general interpolation algorithm is required from the flow solver. Both approaches require the use of conservative interpolation techniques.

The most general block interface type is one that allows the mesh in different blocks to overlap. In this type of method, individual grid blocks are generated for each geometric component, with component blocks overlapping. The result is usually a mesh having mixed component topologies, with the possibility that some blocks will be completely embedded in others. This type of approach is sometimes referred to as a "chimera", "overset" or "mesh embedding" technique. Steger et al. (1985) is one of the early examples of this method. Naturally, this type of block interface is the most complicated from the point of view of the flow solver. The major difficulty lies in automating the interpolation algorithm for a generally overlapping mesh, which is by no means a trivial matter. Problems may occur if strong flow gradients appear in regions of overlap. Essentially, this type of grid generation approach transfers much of the modeling complexity to the flow solver. The major advantage, of course, is that grids can be generated relatively quickly for complex configurations. This technique is also well suited to geometries which include moving components, such as stores separating from a military aircraft.

Numerous types of domain decomposition methods have been developed over the years to generate structured meshes in multiply-connected domains. Part of the reason for this is that, as discussed above, there are many types of structured meshes that one can generate, with varying degrees of continuity. Generally, the lesser the degree of continuity in the overall mesh, the simpler the domain decomposition task. Fully continuous meshes are the most challenging to generate in terms of the domain decomposition effort involved. However, they generally provide a more accurate discretization of the domain. It is this type of fully continuous mesh that is of interest in the present work, and it is thus domain decomposition methods pertaining to this type of mesh that are reviewed herein.

1.2 Literature Survey

1.2.1 Early Methods: Manual Domain Decomposition

The early multi-block codes that were developed were essentially programs that required manual input of every edge describing the domain (Ruppert and Lee, 1980; Manhardt and Baker, 1982; Roberts, 1982; Ghia et al., 1983; Sorenson and Steger, 1983; Weatherill and Forsey, 1984). For each block, the point distributions had to be specified on the edges of each curved surface forming a boundary of the block. Once this was done for all surfaces bounding the block, the three-dimensional mesh within the block was generated using the surface grids as boundary conditions. Typically, the input files consisted of the geometry curves with the associated point distributions, plus topological information such as a table describing the block interface connections and boundary conditions. However, the time required to generate meshes around complex geometries with these codes could be of the order of months. Early examples of such codes are described in references Thompson (1987) and Sorenson (1988).

1.2.2 Methods Hardwired for Specific Configurations

One of the early approaches developed to automate the grid generation process was to design domain decomposition procedures for specific configuration types and hardwire them into a dedicated program. This type of approach was first used to generate single-block grids around simple geometries, such as isolated wings (Jameson, 1974), wing-fuselage combinations (Caughey and Jameson, 1980), and wing-fuselage-tail combinations (Baker, 1991), among others. Since these grid generation procedures were applied to specific configurations, using predetermined topologies, the entire process could be automated relatively easily. The early single-block methods, however, which typically employed conformal mapping techniques, provided limited control

over the mesh characteristics.

The idea of a configuration-specific grid generation tool was also implemented using a multi-block approach (Schuster and Atta, 1989; Schuster, 1992). In this approach, a fixed set of grid topologies were programmed to mesh a finite set of aircraft component geometries. In this case automation is achieved at the expense of versatility, since the method can only be applied to a limited set of geometry types.

1.2.3 Feature-Associated/Component Adaptive Methods

One of the first grid generation methods that incorporated some real measure of automation in its domain decomposition approach was that of Shaw et al. (1988) and Shaw and Weatherill (1992). In this method, each component of the geometry is represented schematically as a bounded planar region in a Cartesian coordinate system. Once input by the user, the code automatically decomposes the Cartesian domain into a number of sub-domains. The automation here is achieved by considering the overall domain to be composed of a matrix of blocks, the topology of which is determined by the assigned integer limits of the planar regions input by the user. In this manner, the total number of blocks, the block numbering and block-face neighbor information is generated automatically by the program.

Once the domain has been decomposed, the grid density is then fixed by specifying the number of points across each block row in the three coordinate directions. This procedure is limited to a H-H overall topology around the geometry, since a Cartesian coordinate system is chosen at the outset. The quality of the final mesh is then optimized by defining control surfaces inside the domain, and optimizing the latter interactively using an elliptic solver.

1.2.4 Hyper-Cube Concepts

Allwright (1988) developed the concept of a wireframe representation of the geometry in which the main component of the representation was the so-called "hyper-cube". This wireframe representation of the geometry was built manually using a graphical interface. In this approach, every component of the geometry in the physical space was represented schematically as a hyper-cube in a parameter space. The decomposition of the domain was performed in the parameter space and the connectivity information between the various geometric components was automatically generated by the program. This concept was recently developed further by Park and Lee (1999). In the latter work, the geometry associated with each hyper-cube in parameter space was represented by a non-uniform rational B-Spline (NURBS) volume, which mapped the parametric space back into the physical space. However, this method cannot be applied to strongly non-convex geometric shapes without first dividing them into a set of convex shape elements.

1.2.5 Knowledge-Based Methods

Given the inherent difficulty of the domain decomposition problem in multi-block grid generation, it was inevitable that someone attempt to use the principles of Artificial Intelligence (AI) or so-called Knowledge-Based systems to tackle the problem. This route was investigated early on by Andrews (1987, 1988, 1990) for two-dimensional applications.

The main difficulty in this approach is that not all problems are considered tractable for the application of knowledge-based systems. As discussed in Andrews (1988), problems that are amenable to a knowledge-based approach normally have the following characteristics:

- The problem has no closed form solution
- Expertise is required to solve the problem
- An expert can solve the problem fairly quickly
- The skill can easily be taught to non-experts
- Solution of the problem does not involve visualization
- Experts agree on how to solve the problem

Clearly, the problem of domain decomposition has some of these characteristics, but not all. Notably, the art of domain decomposition is not easily taught, and it has a very strong perceptual element to it. Furthermore, while there are experts in the field, they by no means agree on what constitutes the optimal domain decomposition nor on what technique is best suited to solve the problem.

Nevertheless, in order to get around these difficulties, Andrews designed an interactive knowledge-based system, where some of the information (notably the information requiring perception) is supplied by the user. In addition, he developed a model (a set of rules) and a corresponding language to describe the fundamentals of flow field domain decomposition in two-dimensions.

Andrews' solution to the problem of the lack of expert consensus in domain decomposition was to establish a series of zoning "archetypes", which could be tuned to reflect a user's bias. Each archetype was defined as a collection of parameters which described a user bias. The archetypes could then be tuned by assigning qualitative weights to each parameter.

In this way, Andrews developed a semi-automated zoning procedure consisting of a sequence of zoning actions applied to zoning objects. It is in the determination of this sequence of actions that the zoning expertise of the user came into play. This

expertise would then be stored in the knowledge base in the form of "sub-plans", which described the sequences of zoning actions applied by the user.

Andrews applied his method to fairly simple two-dimensional gridding problems, but they nevertheless demonstrated that different decompositions could be obtained by specifying different weights to the tuning parameters in the knowledge base. No attempt was made to extend this approach to three-dimensional applications.

1.2.6 Methods Employing Decomposition Templates

A method for achieving a partial automation of the domain decomposition problem is to define a library of adaptable and re-usable decomposition templates (Piperni, 1994; Piperni and Boudreau, 2003). In this approach, the templates describe the decomposition topology in a wireframe structure to which a geometry can be "attached". This is an extension of the idea of configuration-specific decompositions. However, in this approach, the decomposition templates can be adapted or modified to suit any new configuration. The key to this approach is that the topological information pertaining to the mesh, i.e. index limits and directions, mesh spacing distributions, etc., are separated from the geometric information and stored as mesh attributes. In this way, the geometry may be modified or replaced without impacting or altering the topological information. This type of approach can therefore be considered as "semi-automated", in the sense that the initial decomposition of a new configuration is performed manually, but subsequent decompositions are automated.

The method of decomposition templates described above was implemented at Bombardier Aerospace in a program called MBGRID (Piperni and Boudreau, 2003). The MBGRID program is the software environment in which the new domain decomposition methodology presented herein is implemented.

1.2.7 Singularity Methods

An elegant approach to domain decomposition is to trace the streamlines and equipotential lines from a linear potential solution obtained from a singularity method (Klevenhusen, 1982; King and Williams, 1988; Shima, 1980; Nelson et al., 1991). The advantage of this approach is the fact that panel methods can provide flow solutions around complex geometries without having to discretize the space around them. Once such a solution is obtained, the computational domain can be subdivided into simply-connected, topologically rectangular blocks by tracing the appropriate streamlines and equipotential lines.

In Nelson et al. (1991)'s method, a H-type grid is generated around a multi-element high-lift system, with the stagnation streamlines and trailing edge streamlines dividing each element. The number of blocks and their connectivity is determined automatically without user intervention. The point distribution on the elements is also determined automatically, using the gradients from the potential flow solution as weighting functions. Once the points have been distributed around the airfoil elements, the connectivity matrix is used to transfer the resulting number of points to adjacent blocks until the entire grid is specified.

This method is very powerful and works very well in two-dimensions. The major advantage of this method in comparison to all others is the fact that flow information is used to perform the domain decomposition. This will tend to result in a better discretization of the wakes, since the latter are obtained from the solution.

There are disadvantages, however. The application of this method to three-dimensional geometries with lifting surfaces would require the addition of wake panels and a special treatment of the discontinuity across the wakes when tracing streamlines. Furthermore, the streamlines obtained from the flow solution over three-dimensional lifting surfaces may not yield a suitable grid due to the vorticity present in the flow (e.g. the presence of wing tip vortices).

A technique for avoiding the above problems in three-dimensions is to use singularity methods to model an electrostatic field between the geometry and the far field (Vassberg, 1999, 2000; Spekreijse and Kok, 2000). In this approach, the Laplace equation is solved in the region between the geometry and the far-field boundary by assuming that both boundaries are perfectly conducting closed surfaces set at different constant potential values. The mesh is then generated by marching along the electrostatic field from each mesh point on the geometry to the far-field boundary. This approach results in a O-type mesh topology around the configuration. The main limitation of this technique is that it cannot be applied to strongly non-convex geometries.

1.2.8 Domain Decomposition Using Unstructured Methods

Some researchers (Bergman, 1990; Cordova, 1992) have employed unstructured mesh generation techniques to perform the domain decomposition required to generate multi-block structured meshes. In this approach, a constrained Delauney triangulation is used in a first step to decompose the domain into triangles. In a second step, the triangles are transformed into quadrilaterals by the removal of the appropriate edges. The result is an unstructured mesh consisting of quadrilaterals.

In the work of Bergman (1990), large triangles were generated, and each of these was then subdivided into three quadrilaterals that were used as blocks. However, this technique apparently resulted in very skewed blocks in some cases.

In order to avoid the generation of skewed blocks, Schonfeld and Weinerfelt (1991) developed an alternative method whereby the domain is decomposed directly into unstructured quadrilaterals using an advancing-front technique (Lo, 1985). Once the quadrilaterals are constructed in this manner, a structured grid is generated inside each quadrilateral forming a block. This approach was successfully applied to a two-dimensional multi-element high-lift system.

The major issue with the use of unstructured methods to perform domain decompo-

sitions for structured meshes is the lack of robustness of the methods and the limited control they provide over the quality of the decomposition (Stokes et al., 2000).

1.2.9 Interactive Methods

The most common approach to domain decomposition in modern grid generation software can best be described as semi-automated interactive methods. The natural evolution of grid generation methods to interactive environments was fuelled by the rapid advances in workstation technology in the last decade. Since domain decomposition is inherently a visual task, the porting of grid generation codes to workstations with graphical tools was a significant step forward in the development of this technology.

The development of interactive grid generation platforms fostered the widespread proliferation of grid generation packages. In the late 1980's and 1990's, numerous grid generation methods were developed in university, industry, and government research establishments in North America (Remotigue et al., 1992; Steinbrenner and Chawner, 1992; Sorenson and McCann, 1992; Soni, 1991), in Europe (Seibert, 1988; Herrman, 1991; Dener and Hirsch, 1991; Akdag and Wulf, 1992) and in Japan (Fujitani and Himeno, 1991). Making the grid generation or domain decomposition process interactive, however, did not in itself automate the process, but rather, it made the grid generation work less error prone and more efficient.

In terms of the development of new grid generation methodologies that were a direct result of interactive environments, the most significant has been the development of interactive "replay" capabilities that can be used to automate the domain decomposition. In this type of approach, the commands executed in a previous interactive session are stored in a "journal" file, which can be replayed to mesh topologically similar configurations. However, user intervention is usually required during the replay session, to adapt the previously stored commands to the new configuration. This type

of capability has been implemented in the ICEM-CFD grid generation code (Bertin et al., 1992; Akdag and Wulf, 1992; Wulf and Akdag, 1995), among others.

Another interactive approach that enabled some measure of automation is the so-called grid wrapping or "warping" method developed by Dannenhoffer III (1991, 1993). In this approach, a Cartesian background grid is used as a starting point in the grid generation process. The background grid is used as a topological representation of the configuration to be gridded. Initially, the geometry is represented as rectangular shapes superposed onto the background grid. This superposition, which Dannenhoffer calls a configuration "abstraction", is then analyzed by the program to generate topological information about the configuration. This information includes the number of components, their approximate shapes, and their relative sizes and locations.

Once the background grid has been set up in this manner, the task of generating the mesh involves interactively warping the rectangular mesh onto the geometry. The warping of the background grid is performed by successively "dragging" selected background grid points onto the geometry and "attaching" them there. Once the two end points of a background grid line have been attached to the end points of a curve on the geometry, all the intervening points on the background grid line are mapped to equally spaced points on the configuration curve. The remaining background grid points are then smoothed into place using a local Laplacian solver. This process is then repeated until the entire background grid has been mapped onto the geometry. In most cases, the connectivity information of the resulting multi-block grid is deduced automatically by the program.

Dannenhoffer also extended this method to three-dimensions, through the use of predefined "templates", which consist of simple rectangular shapes used as starting points for the three-dimensional background grid (Dannenhoffer III, 1995, 1996).

Chapter 2

Development of a New Domain Decomposition Approach

2.1 Introduction

Despite the wealth of methods that have been developed over the years to automate the domain decomposition process, it is still considered a bottleneck in structured grid generation (Park and Lee, 1999). This is not surprising when one considers the underlying nature of the problem. In the field of computational geometry, the problem of decomposing a two-dimensional multiply-connected polygonal region into convex quadrilaterals has been proven to belong to the class of "NP-Hard" problems (Lubiw, 1985). Problems belonging to this class or to the more general class of "NP-Complete" problems are considered to be essentially intractable[1]. Consequently, solutions to these problems typically involve some form of approximation or employ heuristic algorithms. In contrast, rigorous algorithms to decompose multiply-connected domains

[1]In the field of algorithms, a problem is deemed "intractable" if it can be proven that no deterministic algorithm of polynomial time complexity exists to solve the problem (Garey and Johnson, 1979). (Polynomial time complexity here implies that the number of operations to execute the algorithm is proportional to a polynomial function of n, where n represents the size of the problem at hand).

into triangles (in two dimensions) or tetrahedra (in three dimensions) (e.g. Delaunay triangulation) have been known for over sixty years (O'Rourke, 1993).

The problem of domain decomposition is further complicated by the fact that the decomposition topology for a multiply-connected physical space is generally non-unique. There may be numerous, equally valid decompositions of a given space. This implies that a decomposition algorithm must somehow "select" one of the possible topologies. This topology selection capability must satisfy varying user requirements, and must be applicable to arbitrarily complex domains.

Hence, in order to develop a rigorous solution to the domain decomposition problem, the following three fundamental issues must be addressed:

1. The decomposition of a given space is non-unique
2. The domain decomposition problem is NP-Hard
3. The solution must be shown to be applicable to arbitrarily complex domains

Each of the above issues is addressed in turn in the following sections.

2.2 Topology Selection as a Boundary Value Problem

The problem of generating a structured coordinate system is akin to generating a set of intersecting equi-potential and/or stream surfaces which are aligned with the boundaries of the domain. Since equi-potential or stream surfaces can be generated through the solution of partial differential equations, in a fundamental way so should the coordinate system. Thus the generation of a structured mesh can be posed as a classical boundary value problem involving the solution of partial differential equations.

In a three-dimensional space, if we denote the three families of mesh surfaces by the curvilinear coordinates (ξ, η, ζ), and the physical coordinate system by $\mathbf{r} = \mathbf{r}(\xi, \eta, \zeta)$, then the boundary value problem consists of prescribing the values of (ξ, η, ζ) on the boundaries and finding the functions $\mathbf{r} = \mathbf{r}(\xi, \eta, \zeta)$. Since different boundary conditions lead to different mesh topologies, it follows that the selection of the mesh topology is itself a boundary value problem.

Thus the topology selection should not be the result of a domain decomposition algorithm but rather a direct prescription from the user via the boundary conditions. This is a natural way for the user to select the preferred topology from amongst the many possibilities.

In the discussions that follow, the common practice of representing the curvilinear coordinates (ξ, η, ζ) by their corresponding mesh index values, (i, j, k), is followed.

Figures 2.1 to 2.4 illustrate the process of topology selection via the choice of boundary conditions. In these figures, the values of the mesh indices are prescribed on the boundaries of a simple two-dimensional multiply-connected domain. In figure 2.1, the inner boundary of the domain is mapped onto a square in the topological (i, j) space, whereas in figure 2.3 the inner boundary of the space is mapped onto a slit. Figures 2.2 and 2.4 show the two meshes resulting from this simple change of boundary conditions.

The notion of a boundary value problem and the solution of partial differential equations is nothing new in mesh generation, as these are implicit in such basic methods as conformal mapping. The issue here is the generalization of this approach to a three-dimensional, multiply-connected domain of arbitrary complexity, and the generation of a mesh in such a domain without the requirement for a manual decomposition of the space into simply-connected or otherwise simpler sub-domains.

In a simply-connected domain, the generation of the structured coordinate system involves the mapping of the physical space onto a rectangular topological space. In

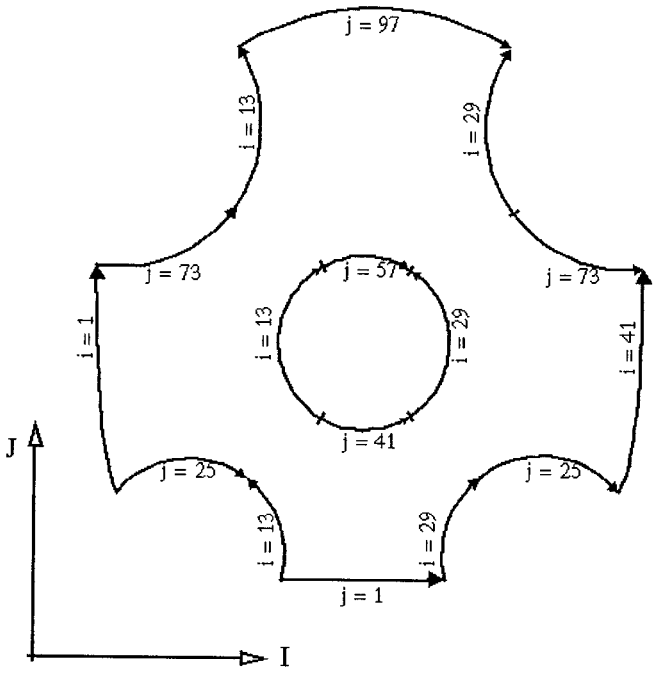

Figure 2.1: Boundary conditions for multiply-connected domain with inner boundary mapped onto a square.

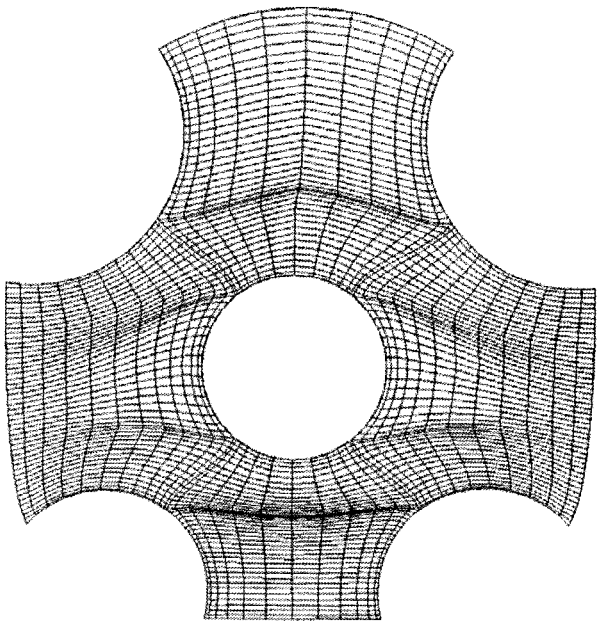

Figure 2.2: Mesh resulting from boundary conditions shown in Fig. 2.1.

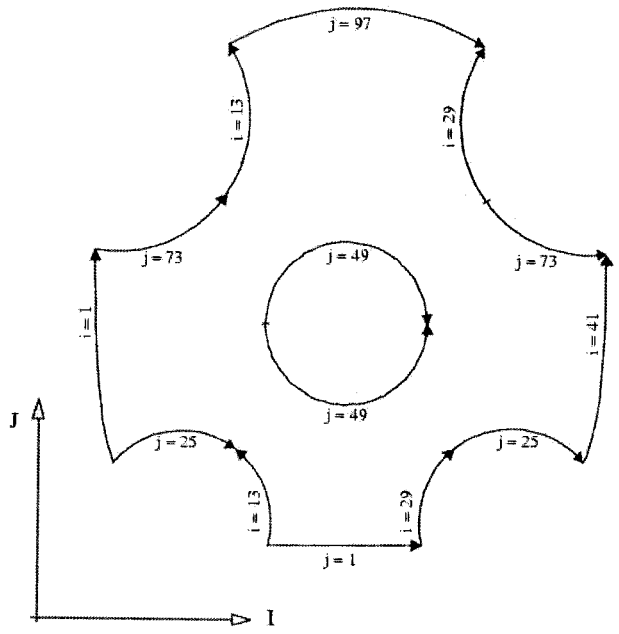

Figure 2.3: Boundary conditions for multiply-connected domain with inner boundary mapped onto a slit.

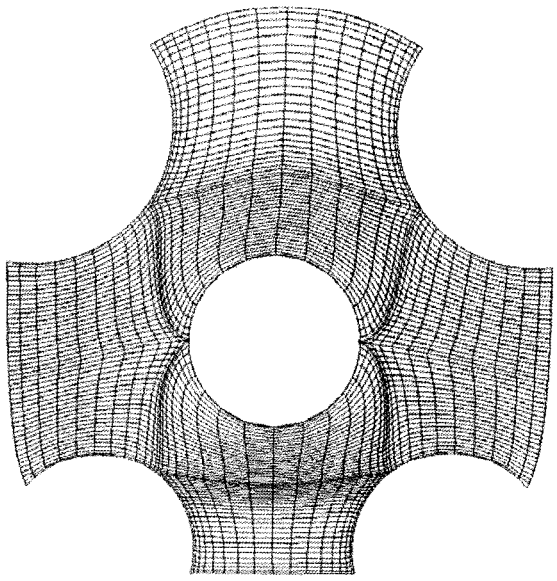

Figure 2.4: Mesh resulting from boundary conditions shown in Fig. 2.3.

contrast, the generation of a mesh in a multiply-connected domain involves the mapping of the physical space onto a rectilinear polygonal region. (A polygonal region here is defined as a polygon with one or more interior "holes"; a rectilinear polygonal region is one whose sides are either vertical or horizontal.) This is illustrated in figures 2.5 and 2.6 below. Figure 2.5 shows the polygonal region in (i, j) space associated with the boundary conditions shown in figure 2.1, and figure 2.6 shows the polygonal region in (i, j) space associated with the boundary conditions shown in figure 2.3. The important observation here is that although the physical space is the same in both cases, the polygonal regions, i.e. the topological spaces are different, and their form depends on the boundary conditions. This is an important property of multiply-connected spaces, since it suggests that a unique set of boundary conditions leads to a unique topological space which, if decomposed, leads to a unique decomposition topology for the physical space.

2.3 Specification of Boundary Conditions

In the previous section, the process of topology selection through the imposition of boundary conditions was illustrated by specifying the values of the mesh indices on the various parts of the domain boundary. Figure 2.7 provides another example of this process. A rectilinear polygonal region is obtained in the (i, j) space corresponding to the values of the mesh indices on the boundaries. The resulting mesh is shown in figure 2.8.

The specification of the mesh indices as boundary conditions is a very basic way to select the topology. However, in practice, specifying the mesh indices as boundary conditions has the disadvantage of linking the mesh topology directly to the mesh density. Fundamentally, the topology of a mesh is independent of its density. For instance, once a topology has been defined, the mesh density can be doubled without altering the topology.

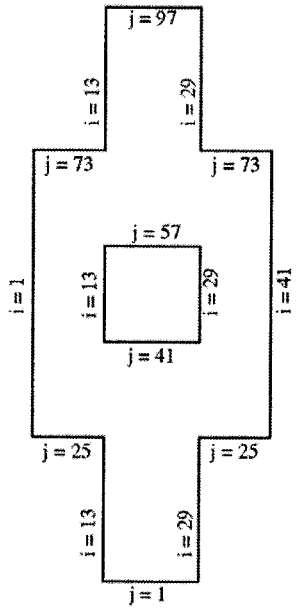

Figure 2.5: Rectilinear polygonal region in topological space resulting from boundary conditions shown in Fig. 2.1.

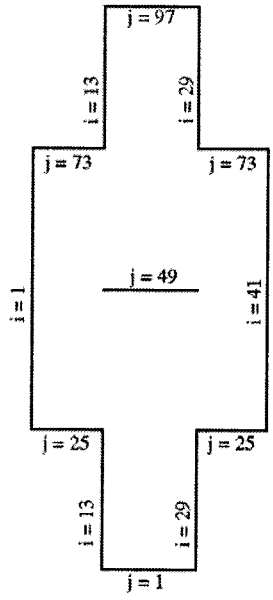

Figure 2.6: Rectilinear polygonal region in topological space resulting from boundary conditions shown in Fig. 2.3.

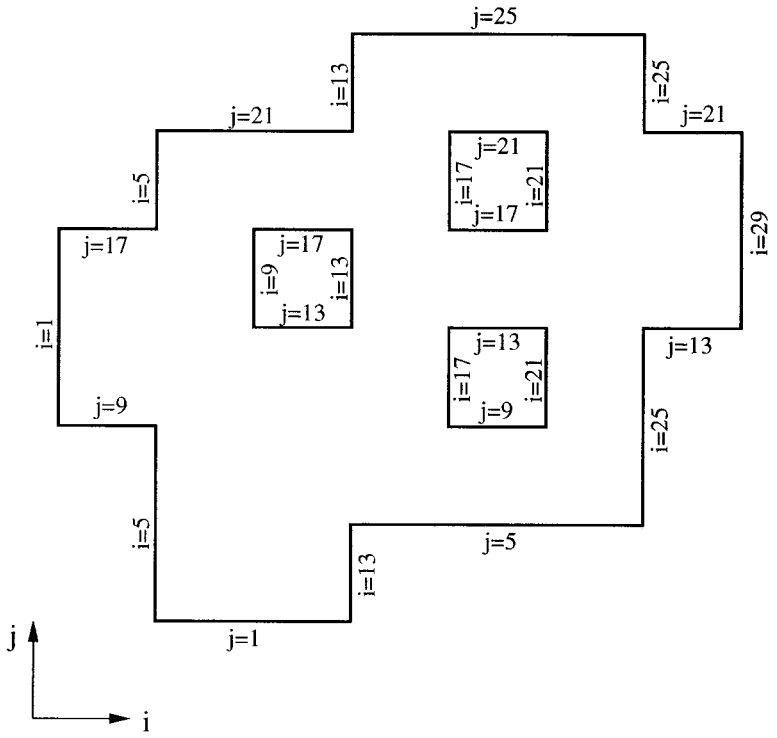

Figure 2.7: Mesh indices as boundary conditions for multiply-connected domain.

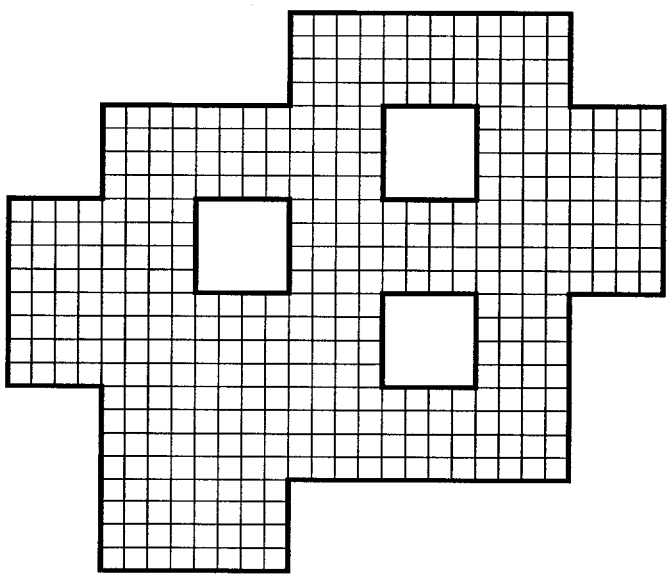

Figure 2.8: Mesh resulting from boundary conditions shown in Fig. 2.7.

The implication of the foregoing is that it is not the absolute values of the mesh indices that actually determine the topology of the mesh, but rather the *relative* values of the indices from one domain boundary to the next. For instance, in figure 2.7, the only significant information pertaining to the two $i = 5$ boundaries is that they lie to the right of the $i = 1$ boundary, to the left of the $i = 9$ boundary, and that they are aligned with each other. The two $i = 5$ boundaries can be changed to any i value greater than 1 or less than 9 without altering the basic layout of the topological space.

This property of topological spaces can be used to simplify the task of prescribing boundary conditions by dissociating the topology generation process from the mesh density distribution. The separation of topology and mesh density can be achieved by introducing the concept of "rows" into the mesh structure, as is done in the MBGRID program at Bombardier Aerospace (Piperni and Boudreau, 2003).

The concept of rows is illustrated in figure 2.9. In this figure, the topological space of figure 2.7 is divided into seven rows in the i-direction, and six rows in the j-direction. The row structure in the i-direction is determined by the number of $i = constant$ boundaries with distinct i values, and the row structure in the j-direction is determined by the number of $j = constant$ boundaries with distinct j values. In this approach, the mesh density in each row can be defined by specifying global mesh index parameters using the following format:

IMAX = 5-5-5-5-5-5-5
JMAX = 5-5-5-5-5-5

The above descriptors tell the program that the mesh has seven rows in the i-direction, and each row has a density of 5 nodes per row (the mesh density in each row is an independent variable). Similarly, the mesh has six rows in the j-direction, wherein each row also has a mesh density of 5.

The concept of row subdivision in each coordinate direction dissociates the mesh topology from the mesh density, and also simplifies the task of topology selection

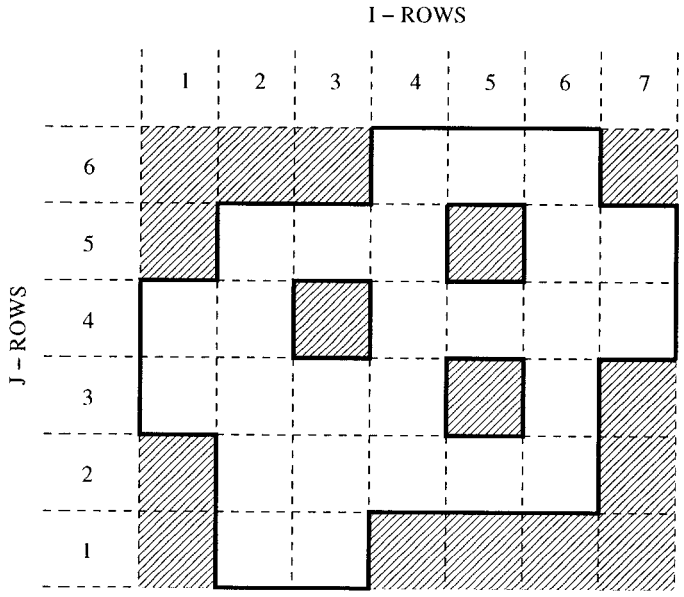

Figure 2.9: Definition of row structure in topological space.

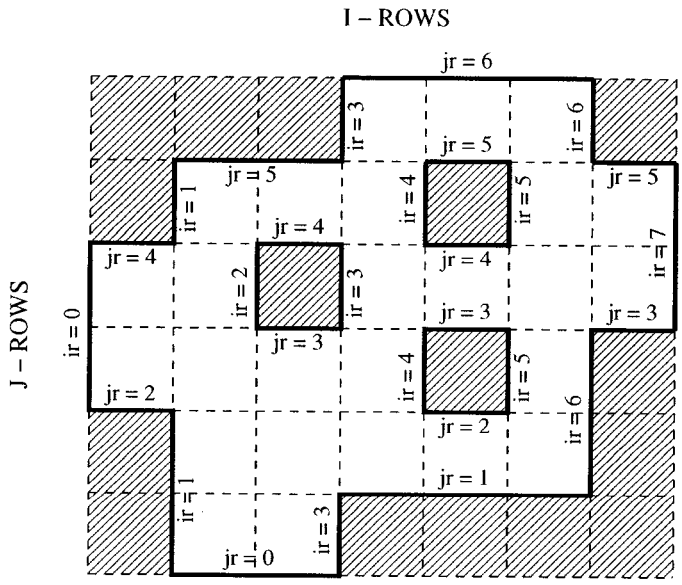

Figure 2.10: Row indices as boundary conditions for multiply-connected domain.

through the specification of boundary conditions. As shown in figure 2.10, once the row structure has been defined, the boundary conditions defining the topology can be specified in terms of the row indices rather than the mesh indices. In this figure, the "ir" and "jr" values denote the row values in the i and j directions, respectively, ascribed to the various boundaries.

The row structure of a topological space is an intrinsic property of the space that defines its organization and layout. Furthermore, when the topology is defined in this manner, the local mesh densities can be modified through the global index parameters without changing the boundary conditions defining the topology.

Another important advantage of the row structure is that it allows the specification of discontinuous[2] mesh indices in a multiply-connected domain, as shown in figures 2.11 and 2.12. Here the mesh density in the i-direction (in the middle portion of the domain) can be specified as a local edge parameter (overriding the global index parameters) without changing the topology of the mesh or the associated boundary conditions.

2.4 Decomposition of the Topological Space

As stated in Section 2.1, the problem of decomposing a two-dimensional multiply-connected polygonal region into convex quadrilaterals has been proven to belong to the class of "NP-Hard" problems (Lubiw, 1985). However, it has also been shown that if the polygonal region is rectilinear, then a rigorous algorithm does exist for its decomposition (Lipski et al., 1979). This is an important result since, as stated earlier, the polygonal region representing the topological space is indeed rectilinear.

Thus the automation of the decomposition process can only be achieved if it is per-

[2]The discontinuity here pertains only to the numerical value of the indices. In the current method, the species and direction of the indices along a coordinate line must remain continuous throughout the domain.

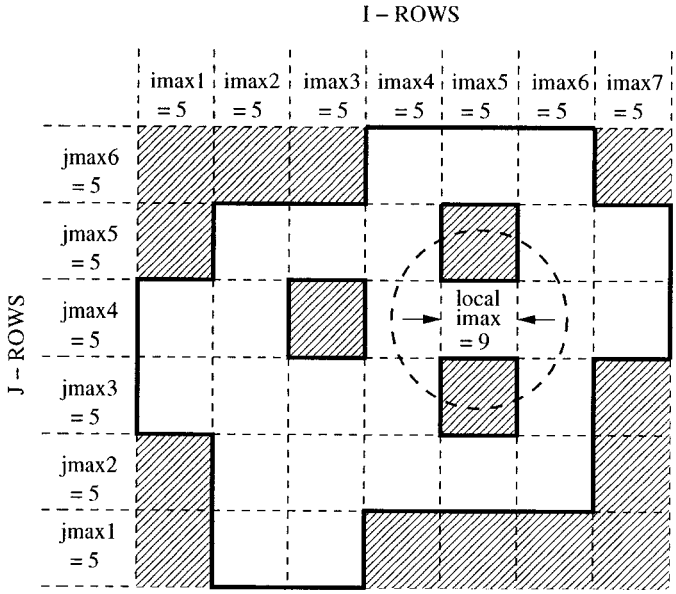

Figure 2.11: Row + local densities defining mesh shown in Fig. 2.12.

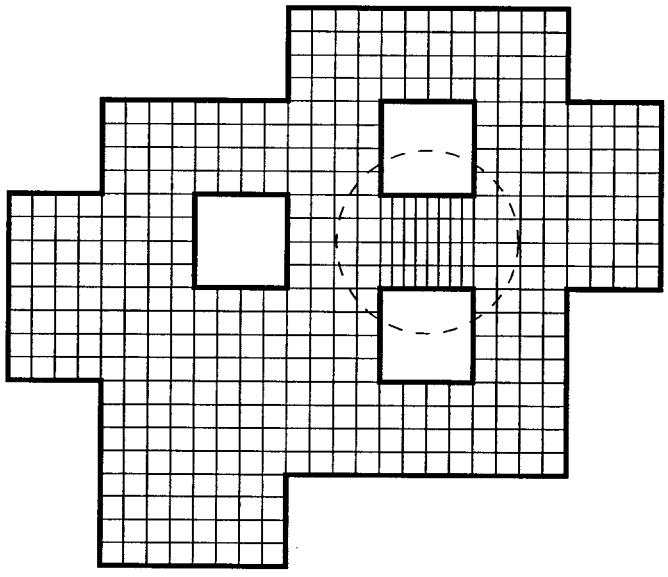

Figure 2.12: Multiply-connected domain with discontinuous mesh indices.

formed in the topological space. It is not surprising then that the more successful domain decomposition methodologies have all used this approach in one form or another (Shaw and Weatherill, 1992; Allwright, 1988; Park and Lee, 1999; Dannenhoffer III, 1996)

However, none of the above methodologies address the important issue of mapping the decomposed topological space back into the physical space. In most cases (Shaw and Weatherill, 1992; Allwright, 1988; Dannenhoffer III, 1996) a graphical user interface is required for the mesh optimization step, which implies that the methods are not fully automated.

The point that is raised here is that fundamentally, the generation of an optimal structured coordinate system in a multiply-connected curved space should be obtained through the solution of a suitable differential operator which takes into account the curvature of the space and in which only the conditions on the boundaries of the domain are prescribed. The forms of the decomposition surfaces in the physical space should be a result of the solution of the appropriate operator, not an artificial boundary condition imposed in the field via a graphical user interface.

Thus a rigorous approach to automate the domain decomposition process must involve on the one hand a rigorous algorithm to decompose the topological space, and on the other a methodology for generating the optimal mapping of the decomposed topological space into the physical space without user intervention.

2.5 Decomposition Algorithms

The algorithm developed by Lipski et al. (1979) performs the decomposition of a two-dimensional rectilinear polygon into a set of rectangles. The application of Lipski's work, which stems from the sphere of computer science, was the optimization of two-dimensional data organization in computers. Lipski's algorithm guarantees that the decomposition will yield the minimum number of rectangles that will cover the entire polygonal region. Furthermore, this algorithm will compute the set of N rectangles in at most $O(E^3)$ operations, where E is the number of edges describing the polygonal region.

Lipski's algorithm defines a set of disjoint (i.e. non-overlapping) rectangles which covers the entire area inside the polygon. These rectangles delineate the decomposition of the domain, and are referred to as the "decomposition" rectangles.

A second set of rectangles, equally important in the present context, was defined by Lodi et al. (1979) in a companion paper to Lipski et al. (1979). This second set of rectangles consists of what are called "prime" rectangles. The set of prime rectangles contains all the distinct rectangles of maximum size which can be included in the polygon. Unlike the decomposition rectangles, the prime rectangles are allowed to overlap.

These two types of rectangles are illustrated in figures 2.13 to 2.16. Figures 2.13 and 2.15 show the decomposition rectangles (delineated by dashed lines) corresponding to the polygons in figures 2.5 and 2.6, respectively. Figures 2.14 and 2.16 show the corresponding set of prime rectangles. (Note that some or all of the decomposition rectangles may also be prime rectangles.)

A detailed description of the algorithms of Lipski et al. (1979) and Lodi et al. (1979) is provided in Appendix A.

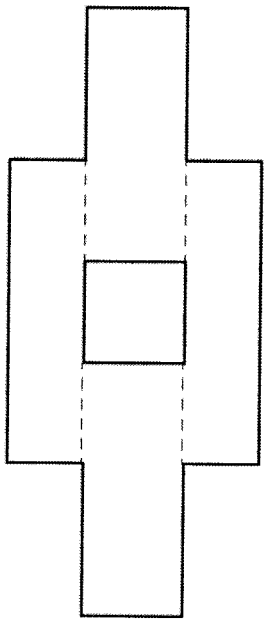

Figure 2.13: Decomposition rectangles associated with polygon in Fig. 2.5.

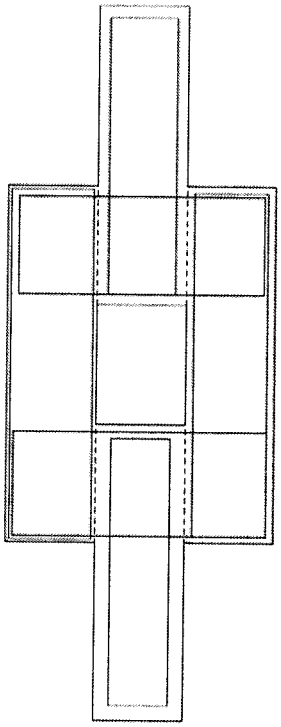

Figure 2.14: Prime rectangles associated with polygon in Fig. 2.13.

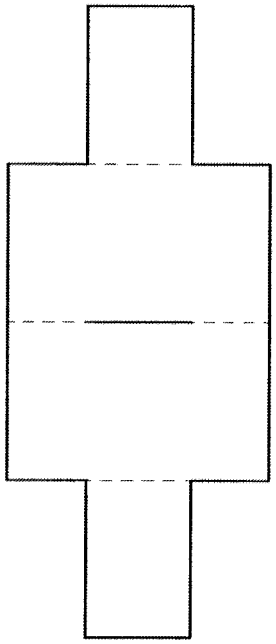

Figure 2.15: Decomposition rectangles associated with polygon in Fig. 2.6.

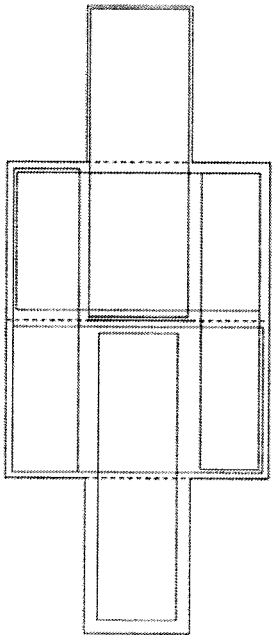

Figure 2.16: Prime rectangles associated with polygon in Fig. 2.15.

2.6 Prime Rectangles and the Optimization of the Mesh

The relevance of the prime rectangles in the current context is that they can be used to provide a set of overlapping zones in which a global optimization of the mesh can be performed. In order to ensure that the final mesh is dependent only on the inner and outer boundaries of the domain and the point distribution thereon, the internal decomposition boundaries must be left free-floating in the optimization process. Since by definition the prime rectangles covering a given area of the polygonal region are the rectangles of maximum size for that area, they will delineate the contact zones between all opposing boundaries (inner or outer) of the domain (as shown figures 2.14 and 2.16). Thus, this set of overlapping rectangles provides the most effective means for communicating all the boundary information into the interior of the space.

2.7 Mapping Back to the Physical Space

Since the topological space is devoid of curvature, it is not surprising that its decomposition can easily be obtained. The more difficult step however is to map this decomposition back into the curved physical space. The success of the decomposition procedure ultimately depends on the success of this last step.

Based on the foregoing considerations, the following procedure is proposed for the automatic decomposition of a multiply-connected space:

1. Construct the physical space by defining the geometry boundaries and the far-field boundaries;

2. Select the decomposition topology by specifying the boundary conditions on the geometry and far-field boundaries;

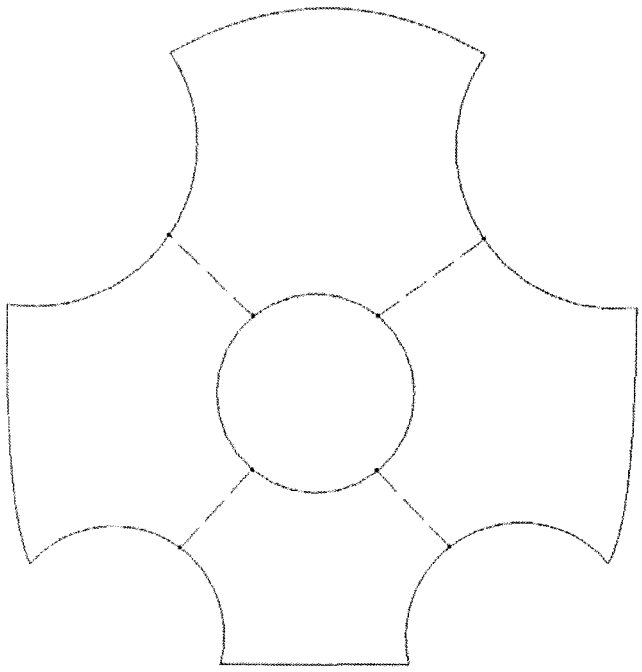

Figure 2.17: Lines of decomposition in figure 2.13 mapped to physical space.

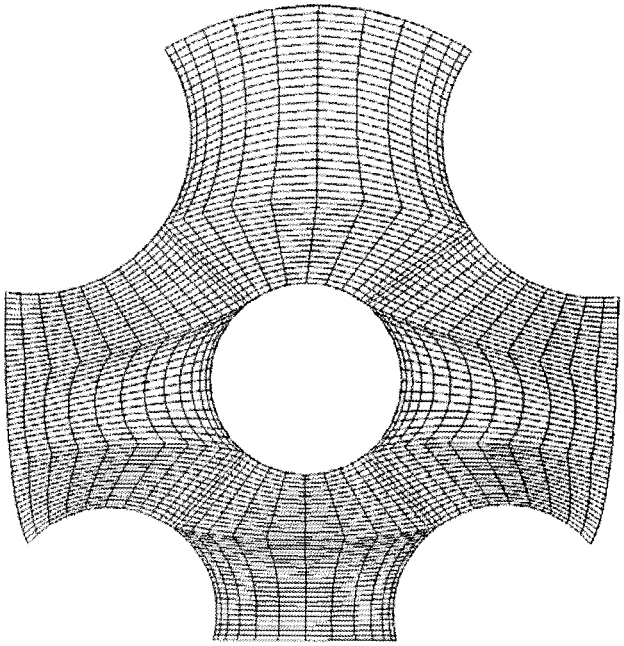

Figure 2.18: Mesh initialized in each decomposition "rectangle" shown in figure 2.17 using transfinite interpolation.

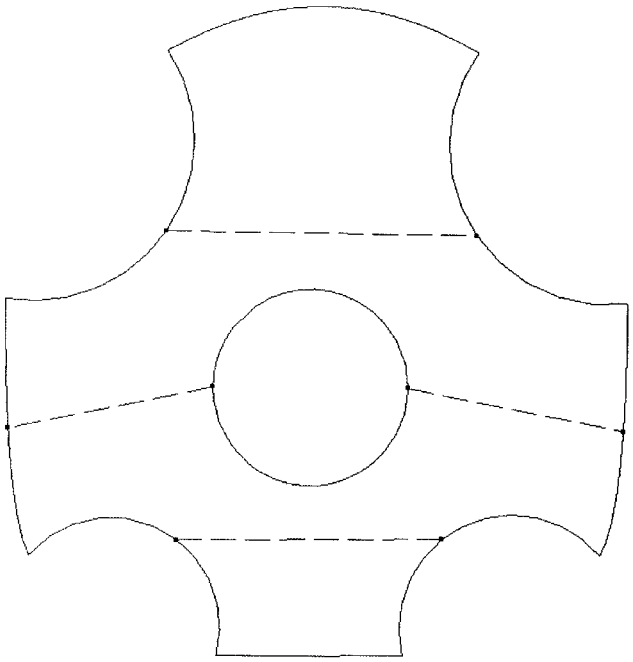

Figure 2.19: Lines of decomposition in figure 2.15 mapped to physical space.

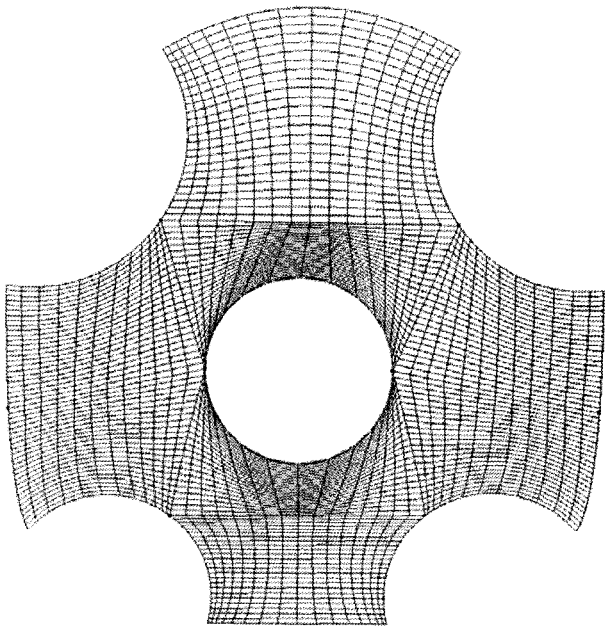

Figure 2.20: Mesh initialized in each decomposition "rectangle" shown in figure 2.19 using transfinite interpolation.

3. Construct the rectilinear polygon, (i.e. the shape of the computational space) associated with the prescribed boundary conditions. Decompose the rectilinear polygon into a set of rectangles using the algorithm from Lipski et al.;

4. Map the corners of the rectangles generated in the computational space back into the physical space using the one-to-one correspondence between physical and computational spaces;

5. Define the initial domain decomposition in the physical space by joining the corners of the decomposition rectangles using straight line connectors;

6. Define an initial algebraic mesh (using transfinite interpolation; see Thompson et al. (1985)) in the physical space by generating a mesh in each decomposition rectangle;

7. Define the complete set of prime rectangles associated with the decomposition polygon using the algorithm from Lodi et al.;

8. Optimize the decomposition in the physical space by successively optimizing the overlapping prime rectangles in an iterative loop.

The first three steps in the process outlined above were illustrated in figures 2.1, 2.3, 2.5, 2.6, 2.13, and 2.15. Steps 4 to 6 are illustrated in figures 2.17 to 2.20. The prime rectangles shown in figures 2.14 and 2.16 (step 7) are then used in step 8 to obtain the optimized mesh (figures 2.2 and 2.4)

2.8 Domain Decomposition of Three-Dimensional Domains

The concept of topology selection based on the specification of boundary conditions is directly applicable to three-dimensional domains. Figure 2.22 gives an example of a topological space generated for the aircraft configuration shown in figure 2.21. As shown, the curved surfaces in the physical (x, y, z) space become planar surfaces in the topological (ir, jr, kr) space[3]. The main aircraft features, i.e. the wing plane, the fuselage-mounted nacelles, and the T-tail configuration, are recognizable in the topological space.

The extension of the domain decomposition methodology to three-dimensional topological spaces requires the extension of the algorithms from Lipski et al. (1979) and Lodi et al. (1979) to three dimensions. The approach developed here to extend these algorithms to three-dimensional space is based on a simple method of extrusion. The main steps in this approach are the following:

1. In each of the three coordinate directions, the three-dimensional topological space (e.g. figure 2.22) is subdivided into a set of extruded two-dimensional domains. The number of extrusions in the ir-direction is determined by the number of $ir = constant$ boundaries of distinct value in this direction (figure 2.23). Every $ir = constant$ boundary in the topological space delineates an extrusion boundary, where each extrusion is treated as a two-dimensional domain in which only $jr = constant$ and $kr = constant$ boundaries are present. The same reasoning applies to the extrusions in the jr-direction (figure 2.24) and kr- direction (figure 2.25). This leads to three sets of extruded two-dimensional domains, where each set covers the entire three-dimensional domain;

2. In each extruded two-dimensional domain, in each of the three coordinate di-

[3]Here the (ir, jr, kr) denote the row indices in the (i, j, k) directions, respectively (see Section 2.3).

rections, Lipski and Lodi's algorithms are used to obtain the complete set of two-dimensional decomposition and prime rectangles, respectively;

3. Each decomposition rectangle obtained as part of an extrusion is given a third dimension (in the direction of the extrusion) equal to the width of the extrusion. Rectangles in adjacent extrusions that share a common face are merged. The extrusion direction (ir, jr, or kr) yielding the minimum number of decomposition rectangles is selected. The decomposition rectangles obtained from the extrusions in the other two directions are discarded;

4. Each prime rectangle obtained as part of an extrusion is given a third dimension (in the direction of the extrusion) representing the maximum width that can be ascribed to that prime rectangle without causing it to cross a domain boundary. All the prime rectangles obtained from the three sets of extrusions are assembled into a three-dimensional system. Redundant prime rectangles, or prime rectangles wholly contained in other prime rectangles, are then discarded.

As indicated above, the procedure for defining extruded decomposition rectangles differs from that used to define the extruded prime rectangles. The difference lies in the definition of the third dimension of each rectangle. The reason for this difference is that the decomposition rectangles must form a *non-overlapping* set, and thus the only way they can be extended beyond the width of their respective extrusions is by merging them with adjacent rectangles. In contrast, the prime rectangles form an overlapping set and each prime rectangle must be of maximum size in the region it covers. Thus, each prime rectangle is extended fore and aft in the extruded direction until both extremities touch a domain boundary.

It is important to note that the method of extrusion described above will not necessarily yield the minimum number of decomposition rectangles covering the three-dimensional space, although in most cases the number will be close to the minimum[4].

[4]The desirability of a minimum number of blocks in a multi-block mesh depends on the computing platform used for the flow solver. For applications on a massively parallel machine, a large number of small blocks may be desired.

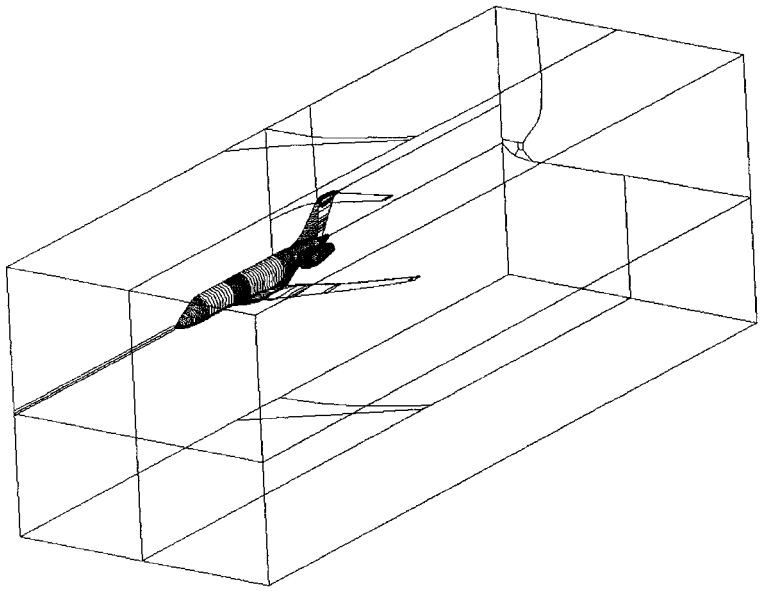

Figure 2.21: Aircraft configuration geometry with far-field boundaries.

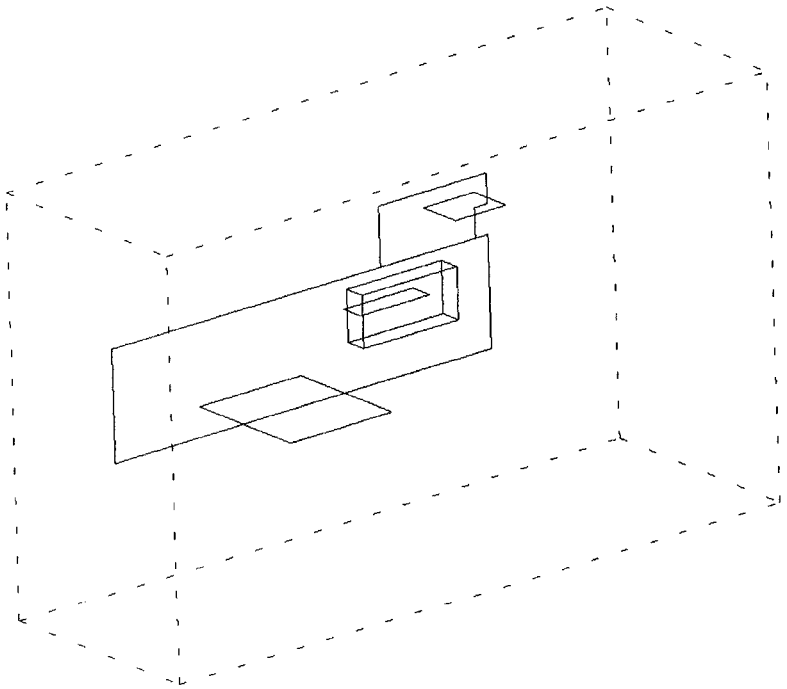

Figure 2.22: Three-dimensional topological space defined for the aircraft configuration shown in figure 2.21.

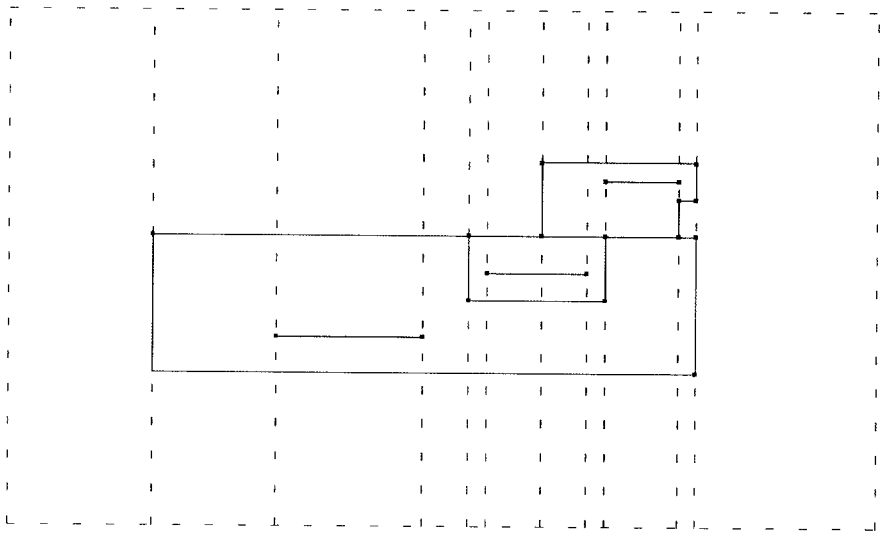

Figure 2.23: Side view of three-dimensional topological space showing extrusions in streamwise ir-direction.

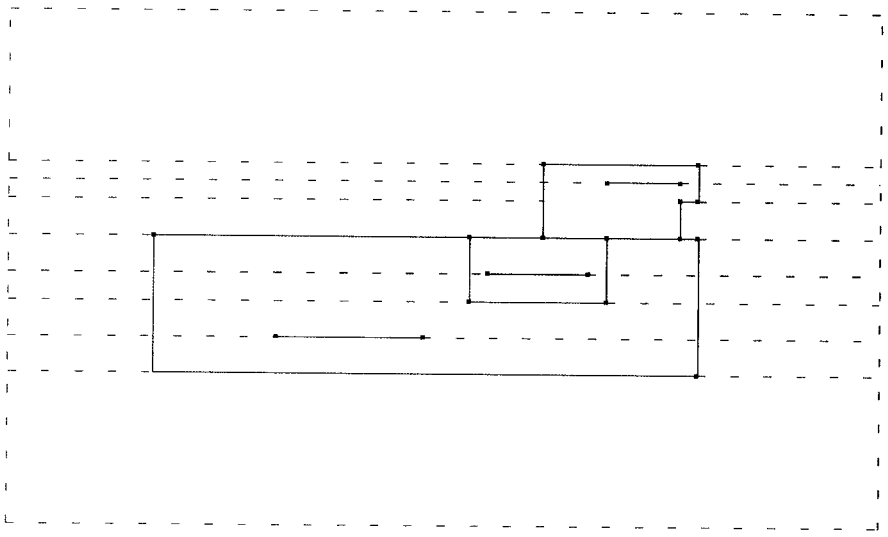

Figure 2.24: Side view of three-dimensional topological space showing extrusions in vertical jr-direction.

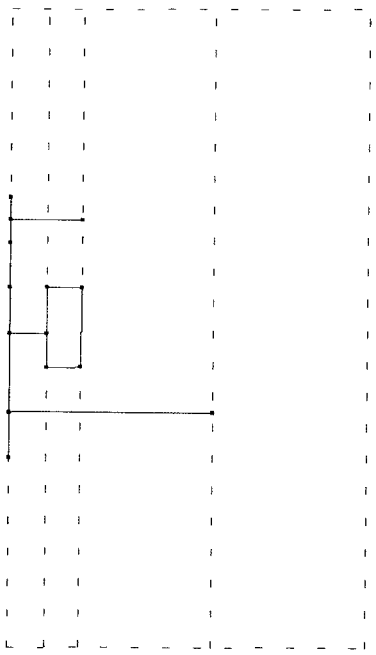

Figure 2.25: Front view of three-dimensional topological space showing extrusions in wing spanwise kr-direction.

In addition, the number of steps involved in the decomposition process may be somewhat greater than the theoretical minimum[5]. However, these elegant features of the Lipski and Lodi algorithms are not essential requirements in grid generation. Since any three-dimensional topological space can be subdivided into a finite set of extrusions, the method described here can be applied to three-dimensional topological spaces of arbitrary complexity.

Once the topological space has been decomposed, the initialization and optimization of the mesh in three dimensions follows steps 4 to 8 outlined in Section 2.7 for two-dimensional domains. In both two and three dimensions, the set of decomposition rectangles are used to define the zones or blocks of the mesh and the corresponding connectivity information, and the set of overlapping prime rectangles are used to perform the global optimization of the mesh.

2.9 Domains of Arbitrary Complexity

In the approach presented here, the decomposition of the topological space can be done for any level of complexity, since the only effect of increasing the complexity is to increase the total number of rectangles. In this sense the approach is both modular and scalable, and provides a framework for attacking domain decomposition problems of arbitrary complexity.

However, as noted earlier, the key to the success of the present method is the quality of the optimization step, i.e. the mapping of the topological space back into the physical space. Thus, a robust mesh optimization program, capable of handling general three-dimensional domains with arbitrary point distributions on all boundaries, is required. Any limitation of the mesh optimization algorithm may translate into a limitation in the automation of the domain decomposition process. In this sense, the present

[5]The computing resources required to perform the decomposition of a three-dimensional topological space into rectangles are negligible.

method transfers the difficulty of the domain decomposition problem to the science of mesh optimization.

An additional consideration is the fact that the quality of the mesh also depends in part on the choice of boundary conditions (i.e. the selection of topology) and the associated boundary point distributions. In three dimensions, not all boundary conditions will lead to valid topologies, and even when they do, the topology and the point distributions on the boundaries must be chosen in a sensible manner to produce a good mesh. As the complexity of the domain increases, the sheer number of boundary conditions may at some point make the process tedious.

2.9.1 Embedded Regions

A further consideration is the situation in which embedded meshes are desired, such as the case of an O-mesh embedded inside a H-mesh. In the present method, since the topological space is generated solely by the specification of boundary conditions, the species and direction of the curvilinear coordinates (ξ, η, ζ) must be continuous everywhere in the interior of the space. If an embedded mesh creates direction changes in the curvilinear coordinates, the inner and outer regions must be defined as separate domains in order to maintain the continuity of the curvilinear coordinates in each.

Fundamentally, each embedded region represents a separate boundary value problem, since the boundary between the inner and outer regions can be of arbitrary shape, size, and location, and the topology of the embedded region is non-unique and must be selected in the same manner as the outer region, i.e. via boundary conditions. Thus, in the present method, the perimeter of an embedded mesh must be provided as an input, along with the boundary conditions defining its internal topology. However, each region can be a *multiply-connected* domain in its own right.

Since embedded meshes are a common occurrence in structured grid generation, they frequently incur additional domain decomposition work. Further automation in this

area can be achieved, but only if simplifying assumptions are made with respect to the size, form and topology of the embedded region. For instance, a common practice is to assume that a region embedded around a configuration component will have the same form as the component but offset by a distance prescribed by the user. This leads to an embedded region in the form of an "O" topology. These types of assumptions eliminate the need for the specification of a separate set of boundary conditions to define the embedded mesh, and in this manner special algorithms can be developed to automate the process.

The investigation of the various automation techniques that have been developed for the generation of embedded meshes is beyond the scope of this work. However, in the MBGRID program developed at Bombardier Aerospace, which is the software environment the present method is implemented in, the generation of inner mesh layers around geometric components is handled with semi-automated "skin growth" tools (Piperni and Boudreau, 2003). Once a skin layer (if required) is built around a given configuration, the decomposition of the multiply-connected space outside the skin can then be automated with the present method.

2.10 Curvature of the Physical Space

In the above discussions, the three key issues in the development of a rigorous approach to the domain decomposition problem, as spelled out at the end of Section 2.1, have been addressed. On the first issue, i.e. the non-uniqueness of the decomposition topology, it was shown that if the problem of domain decomposition is treated as a boundary value problem, then the topology can be *selected* through the imposition of boundary conditions. On the second issue of the classification of the decomposition problem as "NP-Hard", it was shown that if the decomposition process is transferred from the physical space to the topological space, then it is no longer NP-Hard and is therefore amenable to a rigorous solution. Lastly, on the issue of domain complexity,

it was indicated that the approach can be applied to domains of arbitrary complexity since the algorithm used to decompose the topological space is scalable and can produce any number of decomposition rectangles.

However, if the domain decomposition is performed in the topological space, the decomposition process will not yield any information with regard to what forms the decomposition surfaces should take in the physical space. Clearly, the curvature of the decomposition surfaces must mirror the curvature of the physical space, and the generation of these surfaces through the solution of a differential operator is a natural approach to domain decomposition.

The most commonly used operator to generate structured meshes is the Laplace equation (Thompson et al., 1985), with special transformations designed to control the mesh clustering and orthogonality (Spekreijse, 1995, 1999). The solution of this operator can be used to control most characteristics of a given mesh. However, one notable shortcoming of a Laplacian-based system is that it will inherently smooth out the mesh curvature away from the boundaries, thereby altering the mesh clustering near curved boundaries. This is not necessarily a problem when the mesh properties can be controlled through the manual decomposition of the space. However, if one wishes to control the mesh properties in a complex multiply-connected space with only the boundary point distribution, a differential operator that alters the mesh clustering cannot be used. Formulations designed to counter this property of the Laplacian with boundary control functions (Sorenson and Steger, 1983; Thompson et al., 1985) provide only a local and approximate improvement of this behavior.

However, an important property of the Laplacian, and one that must be retained for the purposes of grid generation, is that it exhibits an extremum principle[6] in the field. This property of the Laplacian guarantees a one-to-one mapping of the topological space onto the physical space.

Hence, the operator that is required in the context of the domain decomposition ap-

[6]An extremum principle implies that extrema of the solution cannot occur within the field.

proach presented here is one that retains all the properties of the Laplacian-based systems *except* the smoothing of curvature. Such an operator solved in a unit circle with simple Dirichlet conditions must yield a mesh consisting of equally spaced concentric circles when the boundary points are equally spaced. Furthermore, the mesh lines generated between two boundaries that have an identical curvature distribution must also have the same curvature distribution (e.g. the mesh lines between two identical sine waves must also be identical sine waves).

The derivation of such an operator is an essential element to the development of the domain decomposition approach presented here. The derivation of this operator is presented in the next Chapter.

Chapter 3

Derivation of Curvature Operator

Elliptic grid generation equations based on the Laplacian operator have the well known property of clustering the mesh near convex boundaries and declustering it near concave boundaries. The result of this effect is that the mesh spacing near curved boundaries may not reflect the spacing prescribed by the user. An example of this effect is shown in figure 3.1, which depicts the solution of the Laplace equation (i.e. potential flow) over a circular cylinder. As shown, the curvature effect is due to the expansion of the streamlines in concave areas (where the flow decelerates), or to the contraction of streamlines over convex boundaries (where the flow accelerates).

In grid generation, this property of the Laplacian may cause unacceptably large cell sizes in concave corners. As a result, it is often necessary for the user to activate local control functions which yield the desired spacing on the boundary. For complex three-dimensional meshes, this process can be quite time consuming since it requires the user to visually inspect the susceptible areas of the mesh and then adjust the control functions to produce the desired result.

Thus, a Laplacian-based operator is a problem not only from the point of view of the proper modeling of space curvature for the purposes of domain decomposition, but also from the point of view of the automation of the mesh optimization process.

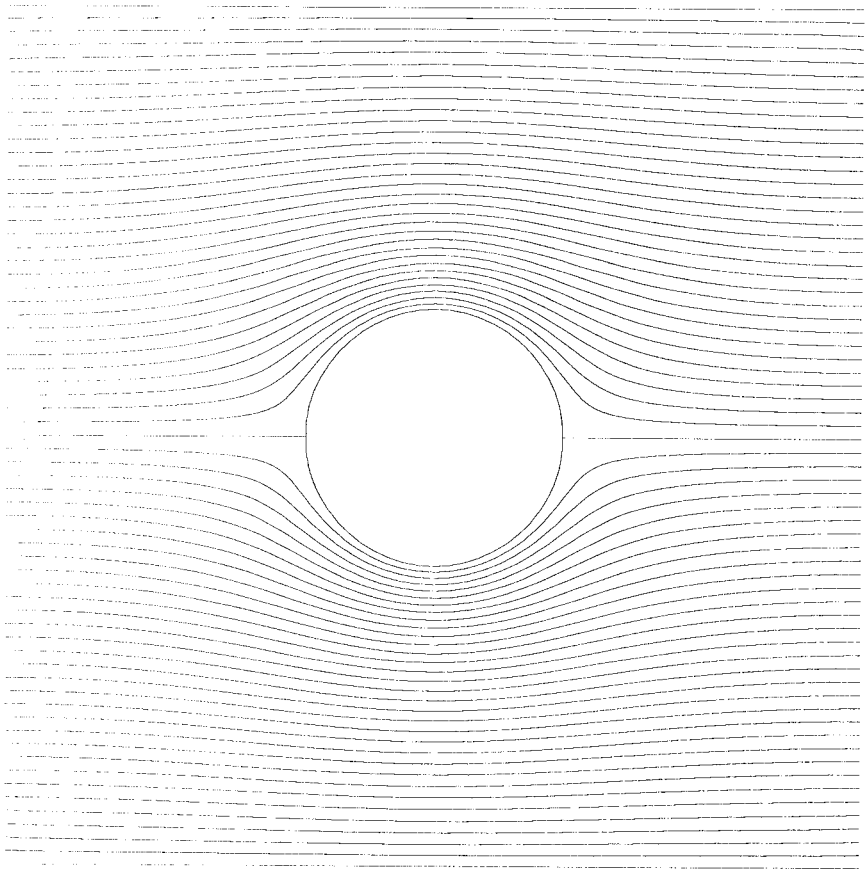

Figure 3.1: Laplace solution over a circular cylinder.

3.1 Current Approaches to Control Curvature Effects

Most of the research that has gone into the development of the grid generating equations has focused on the control of the mesh spacing and orthogonality, with relatively little effort devoted to the control of the mesh curvature.

The method most commonly used to counter this curvature effect is the approach developed by Thompson et al. (1985). In this approach, a curvature control function

is defined by interpolating the point distribution and the radius of curvature from the boundaries. However, this curvature control function is not exact since its form assumes that the mesh is orthogonal at the boundaries. Furthermore, this method is subject to the approximations inherent to the process of interpolating the curvature terms.

Another common method of controlling curvature is through the imposition of mesh spacing and orthogonality at the boundaries (Sorenson and Steger, 1983). However, this type of method can only exercise local control on the mesh near boundaries, and as a result it may be very ineffective in cases where the mesh spacing normal to the boundary is very small, as it is in Navier-Stokes meshes. Furthermore, even for Euler-type meshes, this method cannot be used to control the overall curvature away from the boundaries.

An alternative approach, proposed by Takahashi and Eiseman (1994), involves the re-distribution of grid points by a process termed "uniformalization". Here the functions controlling the mesh spacing are evaluated numerically by equi-distributing points along each coordinate line and solving for the control functions using the generating equations. However, no attempt has been made to apply this method to a mesh with variable spacing.

Perhaps the most accurate technique developed to date to control mesh curvature is due to Sethian (1994). In this approach, the mesh is generated through the solution of a Hamilton-Jacobi-type equation for a propagating level set function. The generating equation is hyperbolic, however, and it therefore cannot be applied to a generally closed domain.

Hence, it is clear that an exact mathematical formulation to properly model the space curvature has yet to be developed for the general case.

3.2 Curvature Operator For a Uniform Mesh

The starting point for the derivation of the curvature operator is the following mathematical statement of the problem. Let ξ^i denote the curvilinear coordinates, and $\partial\xi^i/\partial n$ the spatial derivative of ξ^i in the direction normal to a $\xi^i = constant$ surface (or curve, in two dimensions). Then, if the tendency of the Laplace equation to expand or contract the mesh in curved regions is to be removed, the curvilinear coordinates must satisfy the following equations:

$$\frac{\partial^2 \xi^i}{\partial n^2} = 0; \qquad i = 1, 2, 3. \tag{3.1}$$

The above equations state that the second spatial derivative normal to the $\xi^i = constant$ surfaces must be zero everywhere in the physical space. Thus, for a mesh with uniform spacing on the boundaries, the coordinates will not be allowed to expand or contract, i.e. they will be equally spaced regardless of the curvature of the domain.

The above equations can be expanded as follows. From tensor algebra (Warsi, 1981) we have the following expression for the derivative normal to a coordinate surface on which ξ^i is constant:

$$\frac{\partial \xi^i}{\partial n} = \frac{1}{\sqrt{g^{ii}}} g^{ij} \frac{\partial \xi^i}{\partial \xi^j}. \tag{3.2}$$

In the above expression, the g^{ij} are the contravariant metric tensor components, and the summation convention is used on the repeated upper and lower dummy indices. The second derivative normal to ξ^i is therefore given by

$$\frac{\partial^2 \xi^i}{\partial n^2} = \frac{1}{\sqrt{g^{ii}}} g^{ij} \frac{\partial}{\partial \xi^j} \Big(\frac{1}{\sqrt{g^{ii}}} g^{ik} \frac{\partial \xi^i}{\partial \xi^k} \Big), \tag{3.3}$$

$$= \frac{1}{\sqrt{g^{ii}}} g^{ij} \frac{\partial}{\partial \xi^j} (\sqrt{g^{ii}}), \tag{3.4}$$

$$= \frac{1}{2g^{ii}} g^{ij} \frac{\partial g^{ii}}{\partial \xi^j}. \tag{3.5}$$

The above relation can be simplified further if we use the following expression for the derivative of the metric tensor (Warsi, 1981),

$$\frac{\partial g^{ii}}{\partial \xi^j} = -2\Gamma^i_{jk} g^{ik}, \tag{3.6}$$

where the Γ^i_{jk} are the Christoffel symbols of the second kind. If we now substitute equation (3.6) into equation (3.5), we obtain

$$\frac{\partial^2 \xi^i}{\partial n^2} = -\Gamma^i_{jk} g^{ij} g^{ik} / g^{ii}. \tag{3.7}$$

Hence, substituting equation (3.7) into (3.1), it is clear that the equations we must solve in order to generate a mesh with no adverse curvature effects are the following:

$$-\Gamma^i_{jk} g^{ij} g^{ik} / g^{ii} = 0; \qquad i = 1, 2, 3. \tag{3.8}$$

The above equations bear an interesting similarity to the Laplacian operator applied to the curvilinear coordinates:

$$\nabla^2 \xi^i = -\Gamma^i_{jk} g^{jk} = 0. \tag{3.9}$$

This similarity can be attributed to the fact that equation (3.1) represents a one-dimensional Laplacian taken in a local direction which varies in space, i.e. the direction normal to the $\xi^i = constant$ surface.

3.3 Equivalent Poisson System

Equations (3.8) represent a new set of fundamental equations which retain all the properties of the Laplacian without the latter's adverse curvature effects. One way to demonstrate this is to recast the new equations in terms of an equivalent Poisson system, and to show that in the absence of curvature the Laplace operator is recovered.

The Poisson equation form is also convenient because it allows the curvature control properties of the new equations to be isolated and identified as control functions, and the latter can then be combined with the functions controlling the mesh stretching and orthogonality to produce a unified mesh optimization system.

The equivalent Poisson form of the equations is written as

$$\nabla^2 \xi^i = C^i; \qquad i = 1, 2, 3 \tag{3.10}$$

where C^i represents the curvature control function. An expression for C^i can be obtained by relating equations (3.8) and (3.10) as follows:

$$\nabla^2 \xi^i - C^i = -\Gamma^i_{jk} g^{ij} g^{ik} / g^{ii} = 0. \tag{3.11}$$

Substituting equation (3.9) into the above yields an expression for the curvature control function:

$$C^i = -\Gamma^i_{jk} g^{jk} + \Gamma^i_{jk} g^{ij} g^{ik} / g^{ii}, \tag{3.12}$$

$$= -\Gamma^i_{jk} (g^{ii} g^{jk} - g^{ij} g^{ik}) / g^{ii}. \tag{3.13}$$

The expression on the right-hand side of equation (3.13) can be simplified further by noting that, in the summation over j and k, all terms for which either j or k is to equal i vanish. Furthermore, it turns out that all the non-vanishing terms are of a form which can be expressed more simply in terms of the covariant metric tensor components, i.e. they are all of the form

$$g^{rs} g^{mn} - g^{rn} g^{ms} = g_{ij} / g, \tag{3.14}$$

where (i, r, m) and (j, s, n) are cyclic, and where g_{ij} and g are the components and determinant of the covariant metric tensor, respectively.

Hence, substituting relations (3.14) into equation (3.13) and simplifying, we obtain the following expression for the curvature control function:

$$C^i = -\frac{1}{g g^{ii}} (\Gamma^i_{jj} g_{kk} - 2\Gamma^i_{jk} g_{jk} + \Gamma^i_{kk} g_{jj}), \tag{3.15}$$

where (i, j, k) are cyclic (no sum on j and k). A physical interpretation of the above expression can easily be made by noting that, for a $\xi^i = \textit{constant}$ surface, the sum of the local principal curvatures of the surface is given by Warsi (1981):

$$K_1^{(i)} + K_2^{(i)} = \frac{1}{g (g^{ii})^{3/2}} (\Gamma^i_{jj} g_{kk} - 2\Gamma^i_{jk} g_{jk} + \Gamma^i_{kk} g_{jj}), \tag{3.16}$$

where (i, j, k) are cyclic (no sum on j and k). Hence, the final expression for the

curvature control function can now be obtained by substituting (3.16) into (3.15):

$$C^i = -(K_1^{(i)} + K_2^{(i)})\sqrt{g^{ii}}. \tag{3.17}$$

Substituting the above expression into equation (3.10) then gives

$$\nabla^2 \xi^i = -(K_1^{(i)} + K_2^{(i)})\sqrt{g^{ii}}; \qquad i = 1, 2, 3. \tag{3.18}$$

The above Poisson equation yields the intuitive result that the curvature control function is proportional to the local curvature of the coordinate surface (or curve in two-dimensions). Hence, in the absence of curvature, equation (3.18) reduces to the Laplace equation. It can therefore be concluded that the new system of equations do retain all the properties of the Laplacian other than the curvature effect. And since the only effect of the curvature control function is to ensure that the grid lines are equally spaced, we can expect equation (3.18) to exhibit the same extremum principle as the Laplace equation and thereby guarantee a one-to-one mapping between the physical space and the topological space.

3.4 General Curvature Operator

The expression for the curvature operator given by equation (3.18) is applicable only to meshes with uniform spacing. Hence, in its present form it is not useful since most meshes of interest have a non-uniform spacing.

In the general case of a non-uniform mesh, the curvilinear coordinates will not satisfy equation (3.1), but rather an equation of the form

$$\frac{\partial^2 \xi^i}{\partial n^2} = f^i; \qquad i = 1, 2, 3 \tag{3.19}$$

where f^i is a forcing function representing the prescribed spacing distribution. However, rather than deriving an expression for f^i, a simpler approach can be taken. In the implementation of the elliptic equations developed by Spekreijse (1995), the functions controlling the mesh spacing, denoted here as s^i $(i = 1, 2, 3)$, are made to satisfy the Laplace operator in the physical space, i.e.

$$\nabla^2 s^i = 0; \qquad i = 1, 2, 3. \tag{3.20}$$

The above equations lead to a Poisson system in terms of the curvilinear coordinates (Spekreijse, 1995). However, since the functions s^i controlling the mesh spacing satisfy the Laplace operator, the resulting mesh will exhibit the curvature properties of the latter.

Therefore, in order to preclude undesirable curvature effects, the s^i functions must satisfy equation (3.1) rather than the Laplacian:

$$\frac{\partial^2 s^i}{\partial n^2} = 0; \qquad i = 1, 2, 3. \tag{3.21}$$

Alternatively, the s^i functions can satisfy the equivalent Poisson system:

$$\nabla^2 s^i = \overline{C}^i; \qquad i = 1, 2, 3 \tag{3.22}$$

where

$$\overline{C}^i = -(\overline{K}_1^{(i)} + \overline{K}_2^{(i)})\sqrt{\overline{g}^{ii}}. \tag{3.23}$$

The bar over the terms in equations (3.22) and (3.23) denotes that they are functions of s^i, not functions of the curvilinear coordinates ξ^i.

A general form for the curvature operator can then be derived as follows. Expressing the Laplacian of s^i in equation (3.22) as a function of the curvilinear coordinates, we obtain

$$\nabla^2 s^i = g^{jk}\frac{\partial^2 s^i}{\partial\xi^j\partial\xi^k} + \nabla^2\xi^l\frac{\partial s^i}{\partial\xi^l}, \tag{3.24}$$

$$= \overline{C}^i. \tag{3.25}$$

Isolating $\nabla^2\xi^l$ in the above equation then leads to

$$\nabla^2\xi^l = R^l - g^{jk}\frac{\partial\xi^l}{\partial s^i}\frac{\partial^2 s^i}{\partial\xi^j\partial\xi^k}; \qquad l = 1,2,3 \tag{3.26}$$

where

$$R^l = \overline{C}^i\frac{\partial\xi^l}{\partial s^i}. \tag{3.27}$$

In equation (3.26) above, the second term on the right-hand side is the function controlling the mesh spacing, as derived by Spekreijse (1995). Equation (3.27) gives the general form of the curvature control function, R^l, for a mesh with non-uniform spacing. For a mesh with uniform spacing, equation (3.26) reduces to equation (3.18).

The mesh generation equations are then built by using the identity $\nabla^2\mathbf{r} = 0$ (where $\mathbf{r} = [x\ y\ z]^T$) and expressing it in terms of the curvilinear coordinates (ξ, η, ζ):

$$\nabla^2\mathbf{r} = g^{jk}\frac{\partial^2\mathbf{r}}{\partial\xi^j\partial\xi^k} + \nabla^2\xi^l\frac{\partial\mathbf{r}}{\partial\xi^l} = 0. \tag{3.28}$$

The above system of equations is solved to yield the mesh coordinates, with the $\nabla^2\xi^l$ term given by equation (3.26).

3.5 Analytical Solutions of The Curvature Operator

3.5.1 Polar Coordinates

It is well known that the Laplace equation cannot be used to generate a polar coordinate system with uniform spacing. The reason for this is that the Laplace operator generates a potential flow solution that is irrotational, and hence the streamlines flowing along a circular arc must contract (or expand) to remain irrotational. This can also be seen directly from the Laplace equation written in polar coordinates:

$$\nabla^2\psi = \frac{\partial^2\psi}{\partial r^2} + \frac{1}{r}\frac{\partial\psi}{\partial r} + \frac{1}{r^2}\frac{\partial^2\psi}{\partial\theta^2} = 0. \tag{3.29}$$

If we evaluate the Laplacian of r and θ using the above expression, we obtain

$$\nabla^2 r = \frac{1}{r}, \tag{3.30}$$

$$\nabla^2\theta = 0. \tag{3.31}$$

In contrast, it can be shown that a polar coordinate system is an analytical solution of the curvature operator. In order to do this, we write equations (3.10) explicitly using relation (3.15) as follows:

$$\nabla^2\xi^1 = -\frac{1}{gg^{11}}(\Gamma^1_{22}g_{33} - 2\Gamma^1_{23}g_{23} + \Gamma^1_{33}g_{22}), \tag{3.32}$$

$$\nabla^2\xi^2 = -\frac{1}{gg^{22}}(\Gamma^2_{33}g_{11} - 2\Gamma^2_{31}g_{31} + \Gamma^2_{11}g_{33}), \tag{3.33}$$

$$\nabla^2\xi^3 = -\frac{1}{gg^{33}}(\Gamma^3_{11}g_{22} - 2\Gamma^3_{12}g_{12} + \Gamma^3_{22}g_{11}). \tag{3.34}$$

The evaluation of the right-hand side of the above equations is done using the following well known relations (Warsi, 1981):

$$g_{ij} = \mathbf{r}_{\xi^i} \cdot \mathbf{r}_{\xi^j}, \tag{3.35}$$

$$g = \det |g_{ij}|, \tag{3.36}$$

$$g^{ij} = (g_{rs}g_{mn} - g_{rn}g_{ms})/g, \tag{3.37}$$

$$\Gamma^i_{jk} = g^{il}\mathbf{r}_{\xi^l} \cdot \mathbf{r}_{\xi^j\xi^k}, \tag{3.38}$$

where (i, r, m) and (j, s, n) are cyclic, and where $\mathbf{r} = [x\ y\ z]^T$.

For a polar coordinate system we have

$$x = r\cos\theta, \tag{3.39}$$

$$y = r\sin\theta. \tag{3.40}$$

If we define the curvilinear coordinates as $\xi^1 = r$ and $\xi^2 = \theta$, then it can be shown that relations (3.39) and (3.40) lead to the following expressions for the metric tensor components and determinant:

$$g_{11} = 1/g^{11} = 1, \tag{3.41}$$

$$g_{22} = 1/g^{22} = r^2, \tag{3.42}$$

$$g_{33} = 1/g^{33} = 1, \tag{3.43}$$

$$g_{12} = g_{23} = g_{13} = 0, \tag{3.44}$$

$$g = r^2. \tag{3.45}$$

It can also be shown that the only non-zero Christoffel symbol of the transformation is

$$\Gamma^1_{22} = -r. \tag{3.46}$$

If we now substitute the above relations into equations (3.32) and (3.33), we recover equations (3.30) and (3.31), respectively. Thus, the polar coordinate system is an analytical solution of the curvature operator.

3.5.2 Spherical Coordinates

For a spherical coordinate system we have

$$x = r\sin\theta\cos\phi, \tag{3.47}$$

$$y = r\sin\theta\sin\phi, \tag{3.48}$$

$$z = r\cos\phi, \tag{3.49}$$

and the Laplace equation expressed in spherical coordinates is written as

$$\nabla^2\psi = \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial\psi}{\partial r}\right) + \frac{1}{r^2\sin\theta}\frac{\partial}{\partial\theta}\left(\sin\theta\frac{\partial\psi}{\partial\theta}\right) + \frac{1}{r^2\sin^2\theta}\frac{\partial^2\psi}{\partial\phi^2} = 0. \tag{3.50}$$

As was the case for polar coordinates, spherical coordinates cannot be generated directly with the Laplace equation. If we evaluate the Laplacian of r, θ, and ϕ using the above expression, it can be shown that the spherical coordinates satisfy the following equations:

$$\nabla^2 r = \frac{2}{r}, \tag{3.51}$$

$$\nabla^2\theta = \frac{1}{r^2\tan\theta}, \tag{3.52}$$

$$\nabla^2\phi = 0. \tag{3.53}$$

If we define the curvilinear coordinates as $\xi^1 = r$, $\xi^2 = \theta$ and $\xi^3 = \phi$, then relations (3.47) to (3.49) lead to the following expressions for the metric tensor components and determinant:

$$g_{11} = 1/g^{11} = 1, \tag{3.54}$$

$$g_{22} = 1/g^{22} = r^2, \tag{3.55}$$

$$g_{33} = 1/g^{33} = r^2 \sin^2\theta, \tag{3.56}$$

$$g_{12} = g_{23} = g_{13} = 0, \tag{3.57}$$

$$g = r^4 \sin^2\theta. \tag{3.58}$$

It can also be shown that, of the required Christoffel symbols, the only non-zero values are the following:

$$\Gamma^1_{22} = -r, \tag{3.59}$$

$$\Gamma^1_{33} = -r \sin^2\theta, \tag{3.60}$$

$$\Gamma^2_{33} = -\cos\theta \sin\theta. \tag{3.61}$$

If we now substitute the above relations into equations (3.32) to (3.34), we recover equations (3.51) to (3.53), respectively. Thus the spherical coordinate system is an analytical solution of the curvature operator.

3.6 Applications of The Curvature Operator

The grid generation equations developed herein have been implemented in a program called EGRID, an elliptic smoother for three-dimensional multi-block meshes developed at Bombardier Aerospace (Piperni et al., 1992; Piperni, 1998).

The first application is a simple domain on which the boundary points are equally spaced (figures 3.2 and 3.3). This example includes fairly pronounced concave corners which illustrate the behavior of the Laplacian-based operator in comparison to the new operator. The meshes shown in figures 3.2 and 3.3 were generated without and with curvature control, respectively. It can be seen in figure 3.3 that the curvature effects have been eliminated in the entire mesh, and the discretization of the concave areas is greatly improved when curvature effects are taken into account. Note that orthogonality conditions were not applied in this case, to better illustrate the basic properties of the new equations.

The second application is a mesh over a cylinder (figures 3.4 and 3.5). In this case, orthogonality conditions were applied on the boundaries (via Neumann boundary conditions (Spekreijse, 1995)). The purpose of this test case is to demonstrate that the curvature effects can be properly controlled in conjunction with the application of orthogonality conditions on the boundaries. As shown, the mesh generated with the curvature control terms yielded the desired cell sizes near the concave corners at the base of the cylinder (i.e. cell sizes that reflect the specified cell spacing along the boundaries). In contrast, the mesh generated without curvature control (figure 3.4) would have required user intervention to obtain the same result.

The next two-dimensional application illustrates the overall robustness of the new equations. A complex domain was defined to test the code (figures 3.6 and 3.7). Again, figure 3.6 shows a mesh generated without curvature control, while figure 3.7 shows a mesh generated with curvature control. In both cases, the meshes were generated automatically. As shown, the mesh in figure 3.7 is of higher quality since

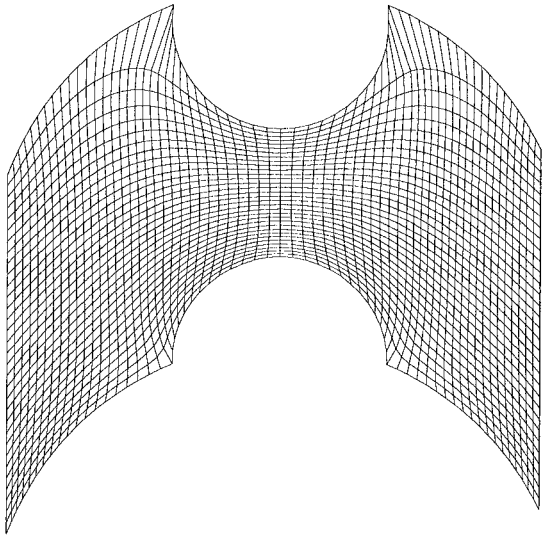

Figure 3.2: Grid generated without curvature control.

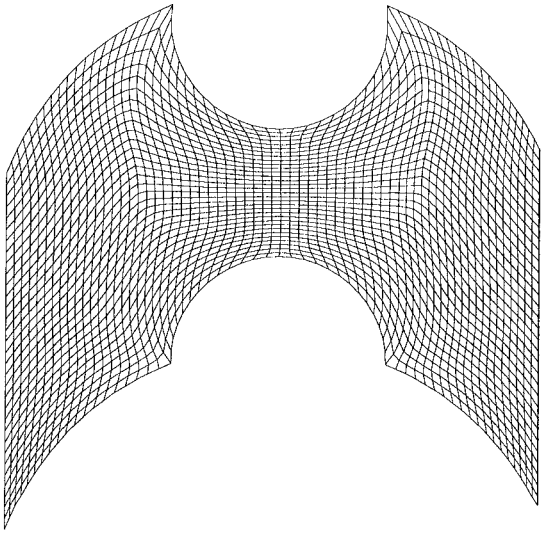

Figure 3.3: Grid generated with curvature control.

the curvature control term ensured that the mesh adhered closely to the boundary. In contrast, the discretization of the concavity in the upper left-hand boundary in figure 3.6 is clearly inadequate. This example indicates that the present grid generation equations with the new curvature control term provide a robust, automatic grid quality optimization method.

The new equations were also applied successfully to three-dimensional meshes. Figures 3.8 and 3.9 provide two views of a multi-block Euler mesh generated around Bombardier's Challenger CL-601 aircraft. The implementation of the curvature control term in three-dimensions did not pose any special problems, since the derivation of the equations was completely general. Figure 3.9 provides a front view of the mesh intersecting the forward fuselage. As shown, the mesh spacing in the concave corner at the top of the fuselage is well behaved. This mesh was generated without user tuning.

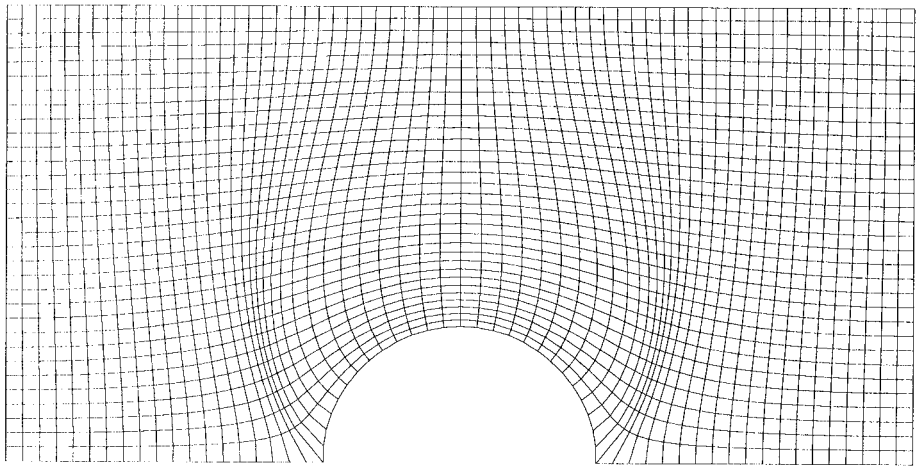

Figure 3.4: Grid generated without curvature control.

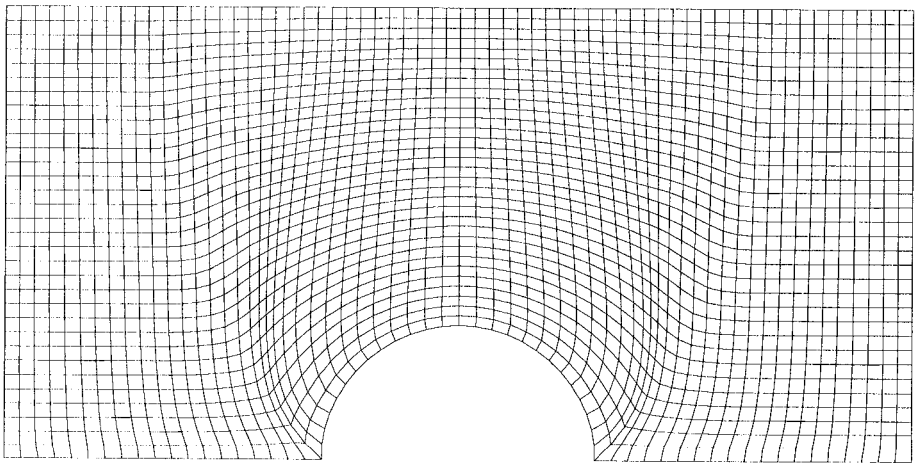

Figure 3.5: Grid generated with curvature control.

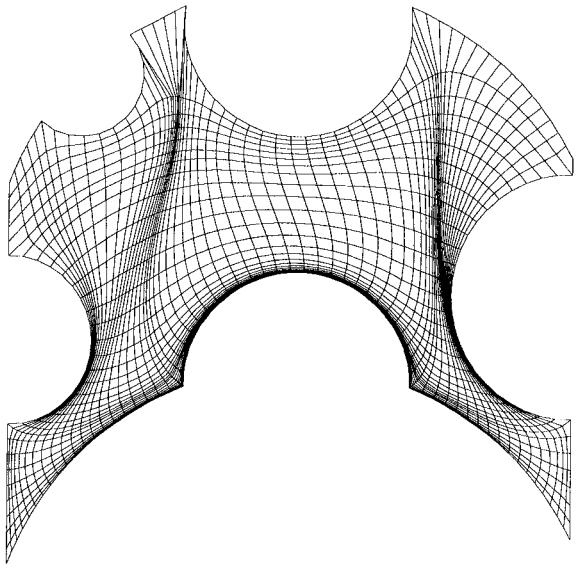

Figure 3.6: Grid generated without curvature control.

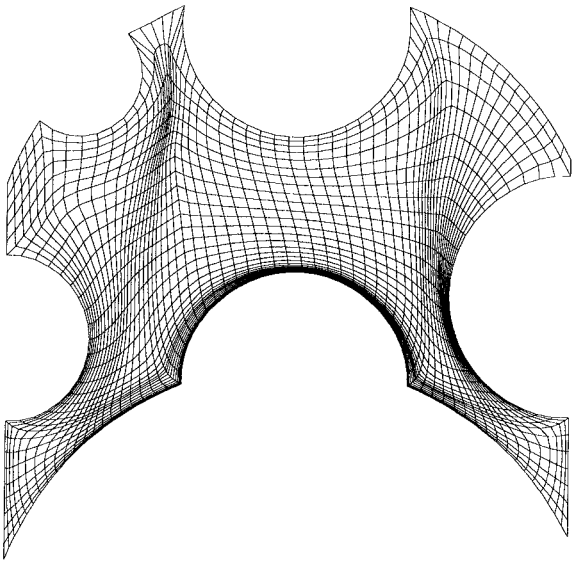

Figure 3.7: Grid generated with curvature control.

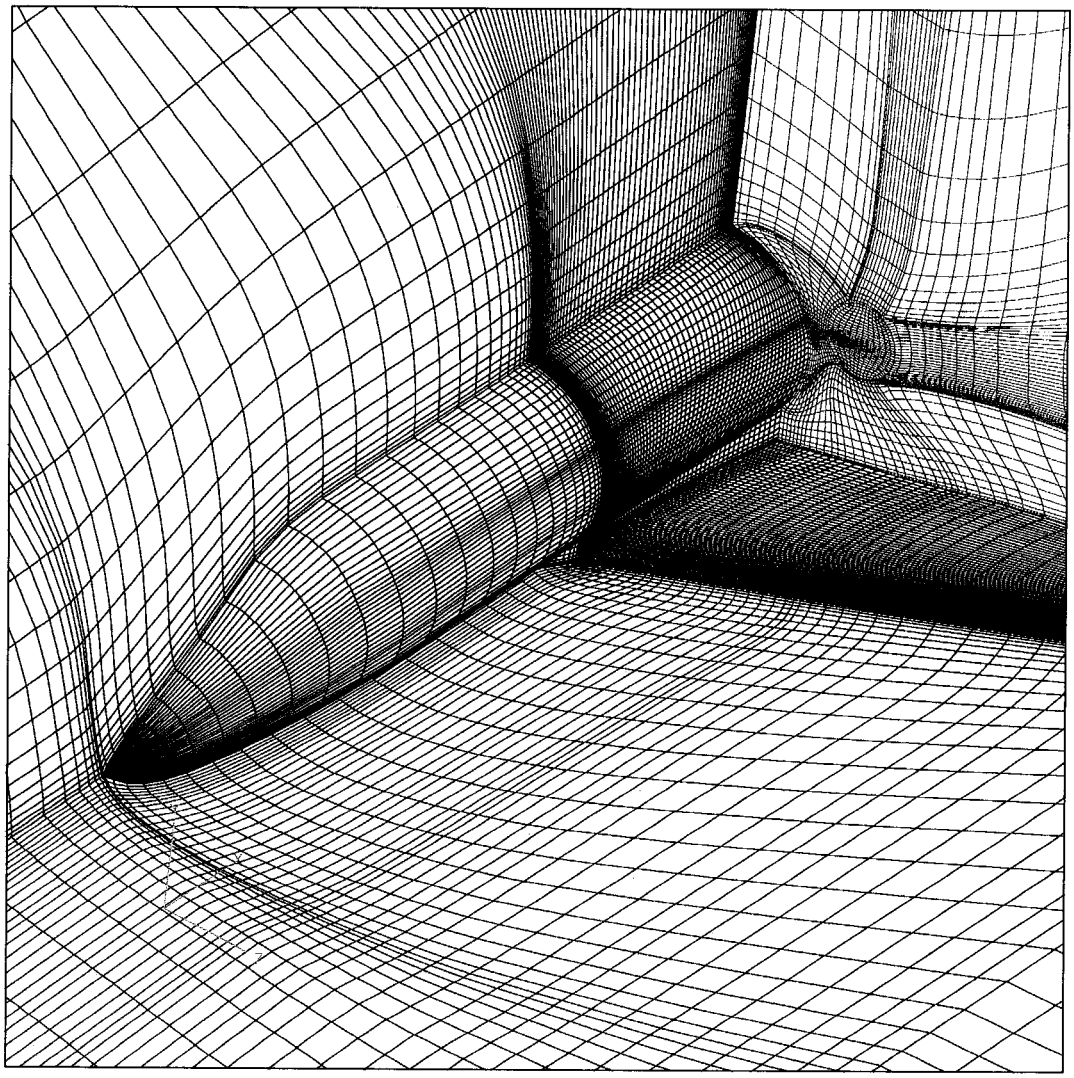

Figure 3.8: Multi-Block Euler mesh on forward part of Challenger CL-601 aircraft.

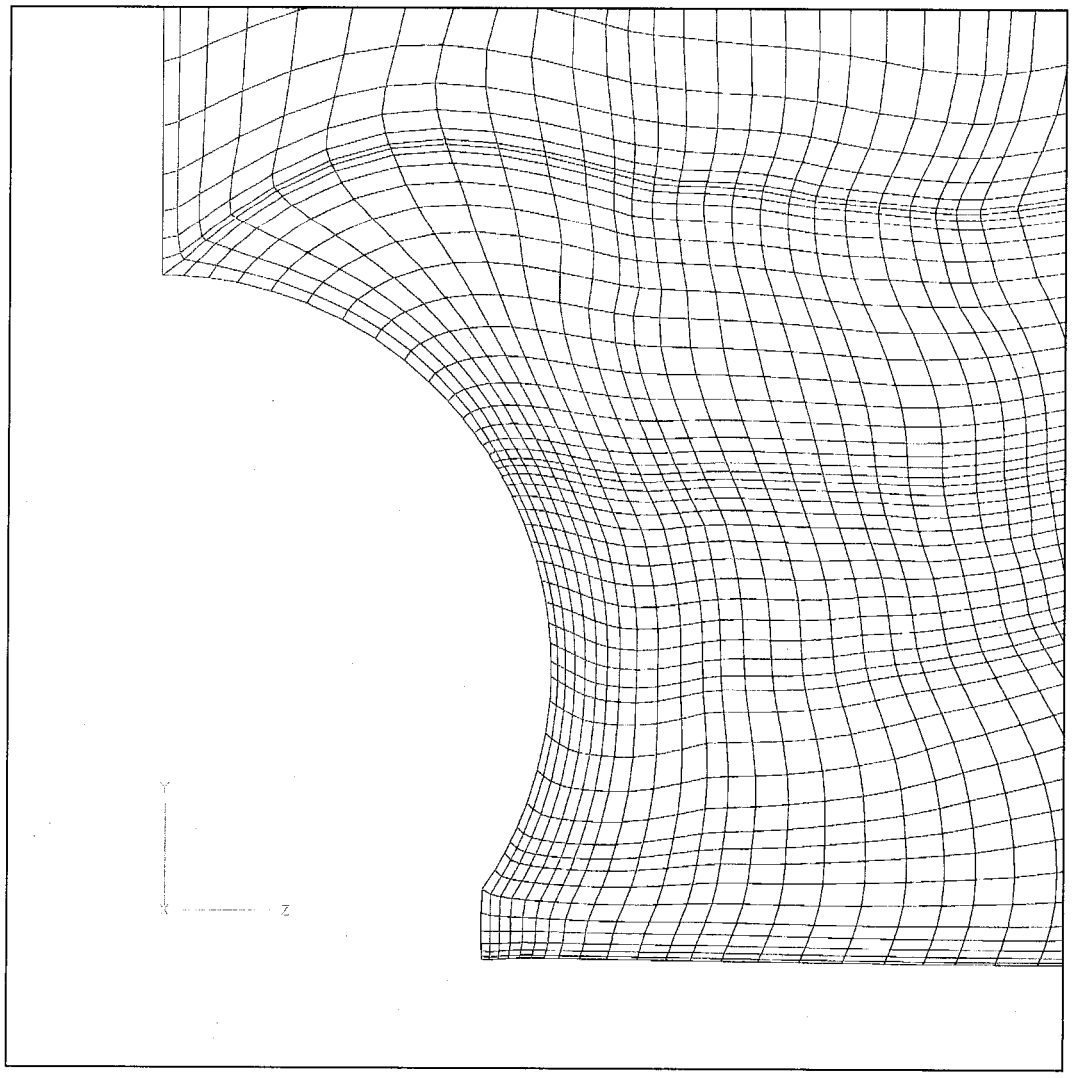

Figure 3.9: Front view of mesh intersecting fuselage.

Chapter 4

Applications of New Domain Decomposition Methodology

The domain decomposition methodology described herein was implemented in MB-GRID, a CATIA-based grid generation software developed at Bombardier Aerospace (Piperni et al., 1992; Piperni, 1994, 1998; Piperni and Boudreau, 2003). The grid generation equations derived in Chapter 3 were implemented in a program called EGRID, an elliptic smoother for three-dimensional multi-block meshes also developed at Bombardier Aerospace (Piperni et al., 1992; Piperni, 1998).

In EGRID, the grid generation equations were discretized using second order, central finite differences, solved in an alternating direction, line implicit (adi) scheme. The over-relaxation parameter used in the solution of the discretized equations ranged between 1.0 and 1.3. In addition, it was found necessary to under-relax the curvature control term by a factor ranging between 0.05 and 0.2.

The first applications of the new domain decomposition methodology were the two simple meshes discussed in Chapter 2 (figures 2.2 and 2.4). Recall that these meshes were generated by applying the boundary conditions shown in figures 2.1 and 2.3, respectively. The meshes were initialized by mapping the decomposition lines shown

in figures 2.13 and 2.15 to the physical space (figures 2.17 and 2.19, respectively), and by generating a mesh inside each decomposition rectangle using transfinite interpolation. The prime rectangles shown in figures 2.14 and 2.16 were then used to perform the optimization of the mesh. Five cycles of optimization were performed to obtain the meshes, where each cycle involved 50 smoothing iterations in each prime rectangle.

An example of a more complex domain is shown in figure 4.1, which represents a simple two-dimensional heat exchanger model. The rectilinear polygon and the decomposition rectangles associated with this domain are shown in figure 4.2, the prime rectangles are shown in figure 4.3, and the final mesh is shown in figure 4.4. The only work required to generate this mesh was the prescription of the boundary conditions and spacing distributions on the inner and outer boundaries of the domain. As before, the optimization of the mesh was achieved in five cycles of optimization, where each cycle involved 50 smoothing iterations in each prime rectangle.

The next application demonstrates that the above procedure can be applied to real grid generation problems. Figures 4.5 to 4.10 illustrate the various steps in the automatic meshing of a three-element high-lift device. The topology chosen for this application is a C-type mesh wrapping around the main element (figure 4.5). Figure 4.6 provides a closer view of the geometry and the wake line (the latter was constructed manually since it lies on a domain boundary). Figure 4.7 shows the polygon associated with this topology. In this figure, the lower boundary of the domain corresponds to the main element and wake upper and lower surfaces, beginning from the wake lower surface at the far-field, wrapping around the main element and onward to the upper wake surface at the far-field. The two slits in this figure correspond topologically to the flap (forward slit) and slat (aft slit) elements. There are two decomposition rectangles and five prime rectangles in this topology, as shown in figures 4.7 and 4.8, respectively. The initial algebraic mesh is shown in figure 4.9. As shown, this initial mesh is very badly conditioned, and the optimization process would not recover from this initial mesh without making use of the prime rectangles. The fi-

nal optimized mesh is shown in figure 4.10. Again, five cycles of optimization were performed to optimize the mesh, but in this case 200 smoothing iterations per prime rectangle were performed in each cycle.

The interesting aspect of this application is the curvature of the domain, and the quality of the resulting decomposition. The forms of the decomposition curves, which are a by-product of the optimization process, are delineated by the dark bands of high mesh density. This illustrates the effectiveness of the present approach in producing a high quality decomposition in a complex domain without user intervention. Figures 4.11 to 4.14 further illustrate the importance of using the curvature operator for this test case. The mesh optimized with the curvature operator, shown in figures 4.12 and 4.14, clearly follows the domain curvature and reflects the mesh clustering imposed on the boundaries. The same mesh optimized with a Laplacian-based operator, shown in figures 4.11 and 4.13, does not exhibit these characteristics.

The first three-dimensional application of the above procedure is the wing-fuselage configuration shown in figure 4.15. In this case, the boundary conditions applied to the various boundaries were chosen to define a H-H topology around the configuration. Again, the mesh point distributions are controlled only from the boundary conditions on the geometry and far-field boundaries.

Figure 4.16 shows the decomposition rectangles obtained for this configuration. As shown, only two rectangular regions are sufficient to cover the entire domain. The lower rectangular region represents the zone below the wing in the topological space, and the upper rectangular region represents the zone above the wing. In this case the two decomposition rectangles are also part of the set of prime rectangles. There are also three additional prime rectangles, as shown in figures 4.17 and 4.18. These three additional rectangles correspond to the zones forward, aft, and outboard of the wing in the topological space.

In this case, 10 cycles of optimization were performed to achieve proper convergence, where each cycle involved 200 smoothing iterations in each of the five prime rectan-

gles. Unlike the two-dimensional test cases described above, which were optimized directly on the CATIA workstation, this mesh (which consisted of 688,000 points) was optimized in batch mode by running EGRID on a NEC SX-6, 8-CPU machine available at Bombardier Aerospace. The resulting mesh is shown in figures 4.19 to 4.24. Figure 4.19 shows the point distribution prescribed on the far-field boundaries in order to control the mesh in the interior, and figure 4.20 gives an isometric view of the resulting mesh. Figures 4.21 to 4.23 provide front views of the grid at various stations intersecting the wing and fuselage. Figure 4.24 gives a side view of a grid surface intersecting the wing at about the mid-span position. These figures illustrate the quality and control of the mesh that was achieved by holding fixed only the points on the geometry and far-field boundaries. It can also be seen in figures 4.21 to 4.23 that the curvature operator properly discretized the concave corners at the top and bottom of the fuselage surface.

The second three-dimensional application is the full aircraft configuration shown in figure 2.21. The boundary conditions chosen for this mesh resulted in the topological space shown in figure 2.22. This topological space was subdivided into three sets of two-dimensional extrusions (one in each coordinate direction). The extrusions in the (ir, jr, kr) directions[1] are shown in figures 2.23, 2.24 and 2.25, respectively. As shown, there are 11 extrusions in the streamwise ir-direction, 9 in the vertical jr-direction, and 4 in the wing spanwise kr-direction. This process yielded a total of 9 extruded decomposition rectangles and 26 extruded prime rectangles.

As in the previous test case, 10 cycles of optimization were performed to smooth the mesh, but in this case 500 smoothing iterations per cycle were performed in each of the 26 extruded prime rectangles. Various views of the resulting mesh are shown in figures 4.25 to 4.29. Figure 4.25 shows the point distribution required on the far-field boundaries to control the mesh in the interior. Figure 4.26 gives a isometric view of the mesh, while figures 4.27 to 4.29 provide closer views to illustrate the result of the

[1]Recall that the (ir, jr, kr) denote the row indices in the (i, j, k) directions, respectively (see Section 2.3).

optimization. This mesh consisted of 2.2 million points.

Unlike the previous test cases, the convergence of the optimization process in this case was relatively difficult to achieve. The convergence was found to be very sensitive to the point distribution on the boundaries, and a number of adjustments had to be made before a satisfactory level of convergence was obtained.

This test case highlighted the fact that the present domain decomposition approach relies on the robustness of the mesh optimization algorithm to achieve success. Furthermore, since the entire mesh is controlled only by the point distributions on the geometry and far-field boundaries, these must be chosen carefully to obtain the desired mesh.

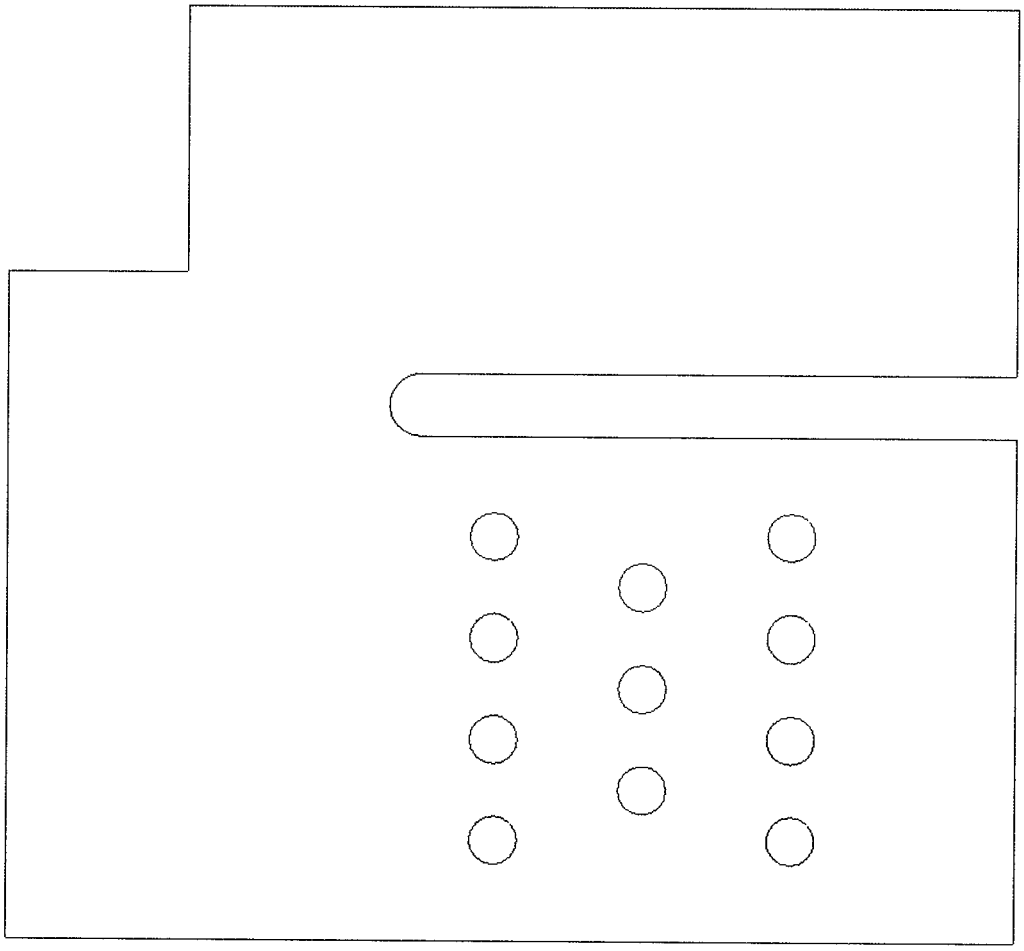

Figure 4.1: Simple two-dimensional heat exchanger model.

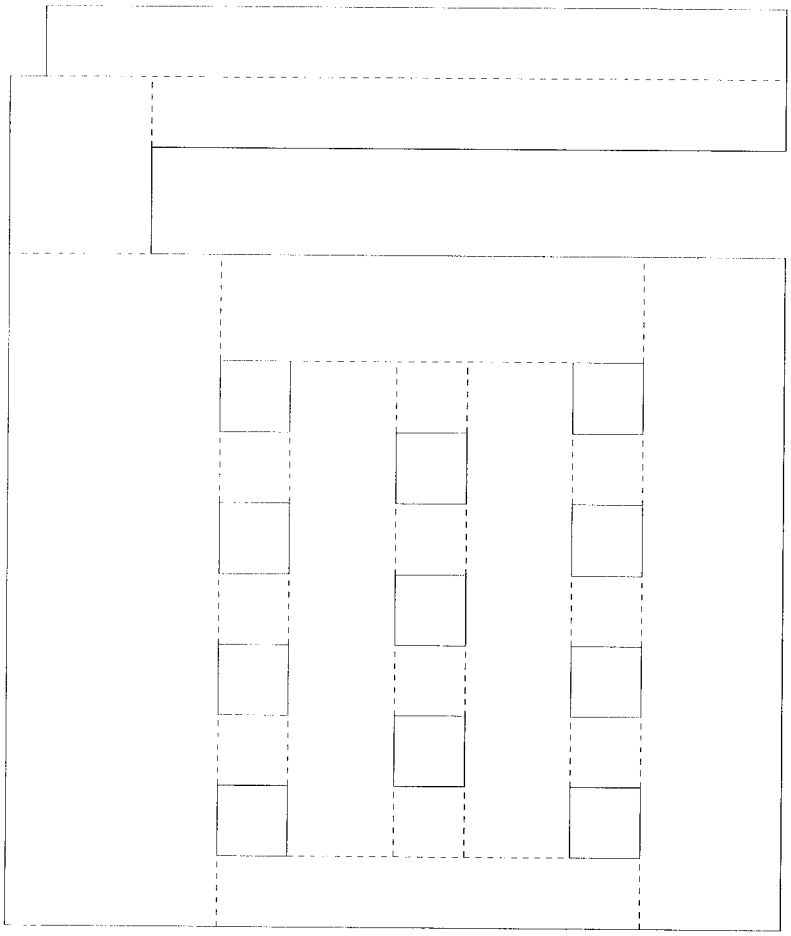

Figure 4.2: Rectilinear polygonal region (solid lines) with decomposition rectangles (dashed lines) for heat exchanger model.

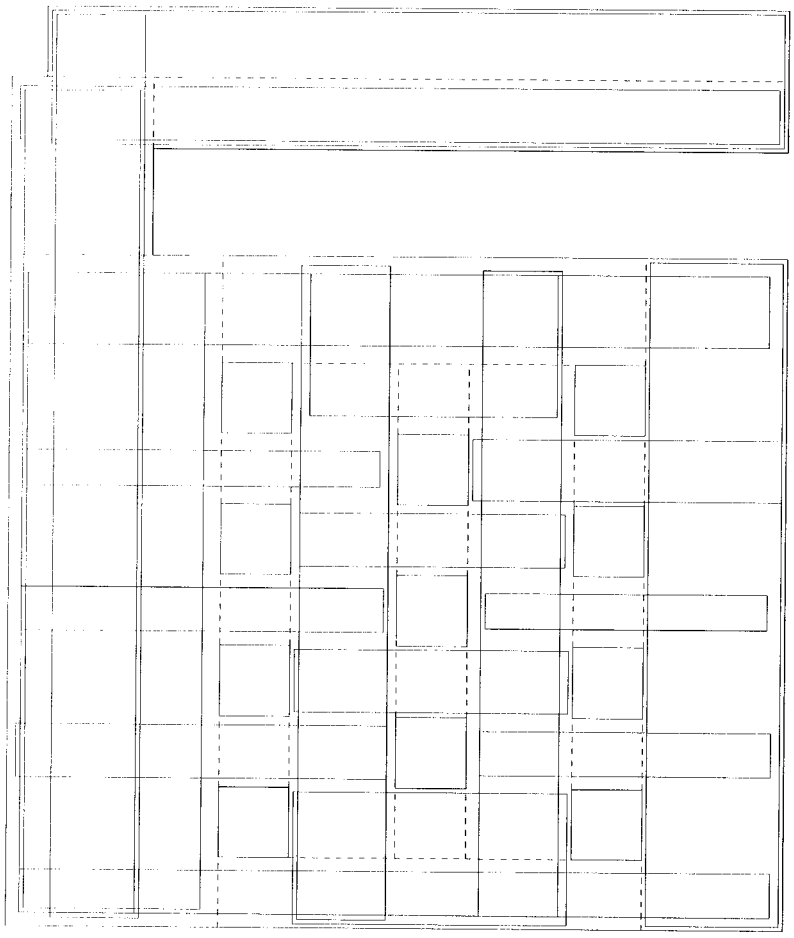

Figure 4.3: Rectilinear polygonal region for heat exchanger model, showing prime rectangles.

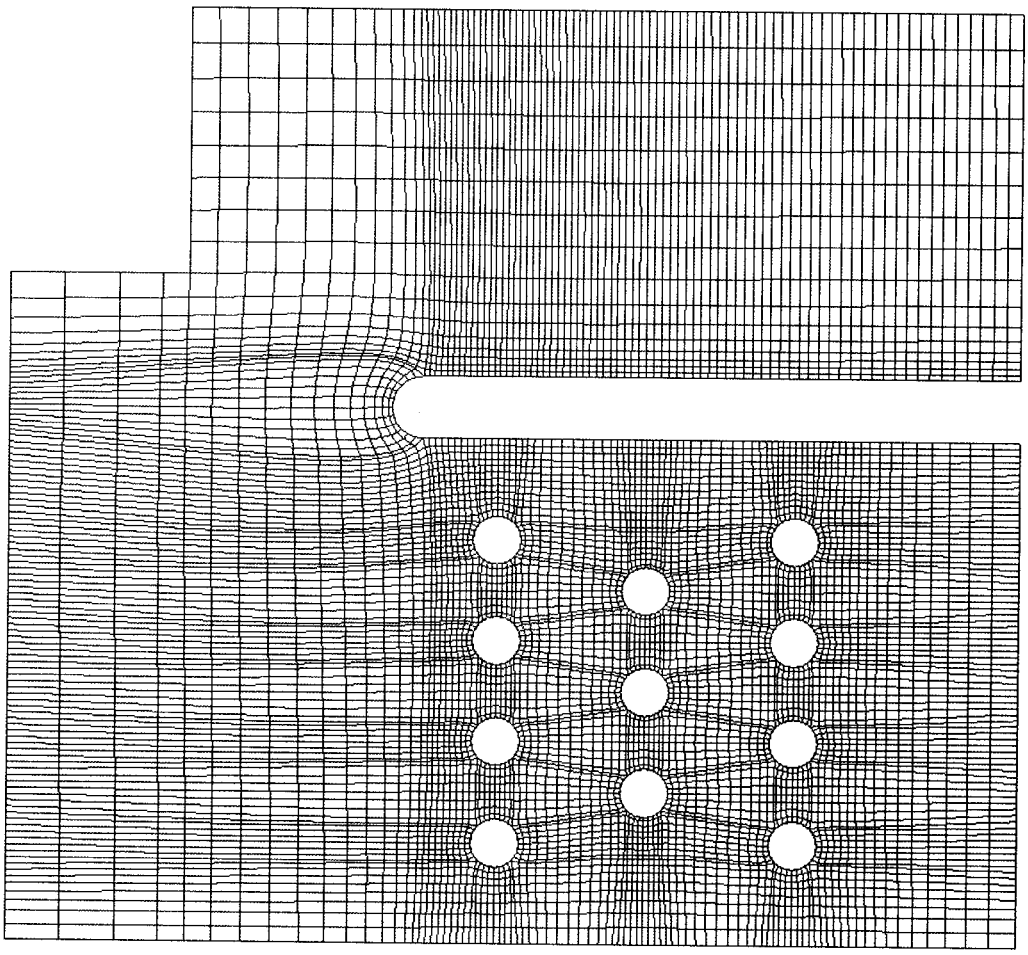

Figure 4.4: Optimized mesh for heat exchanger model.

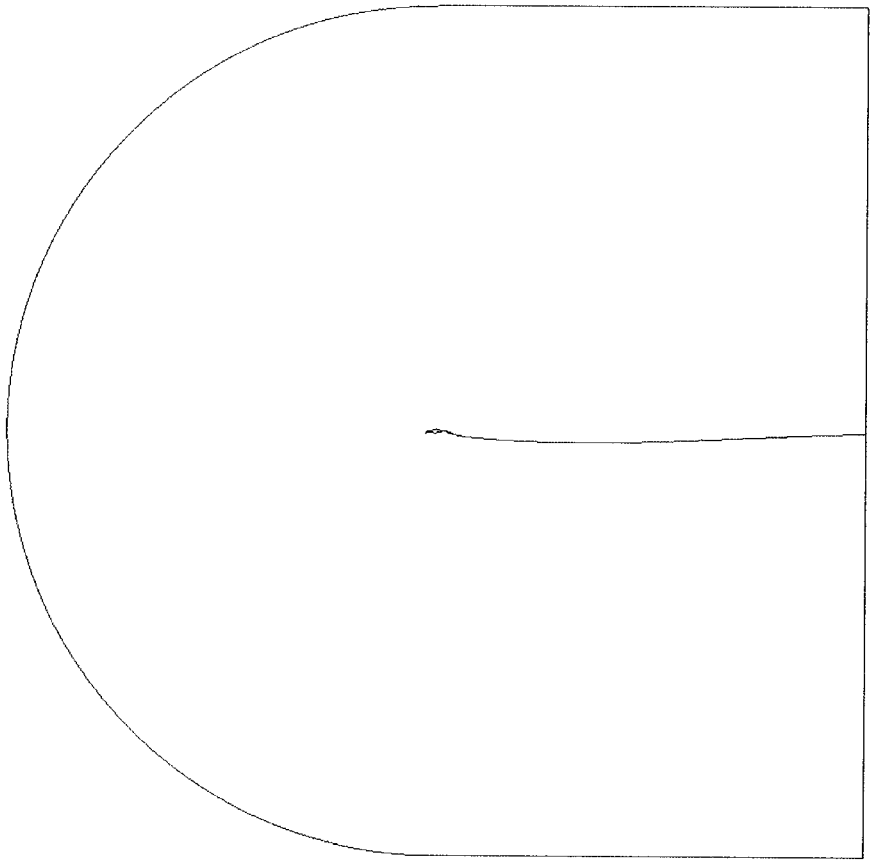

Figure 4.5: Far-field boundary and wake line for C-mesh around multi-element high-lift system.

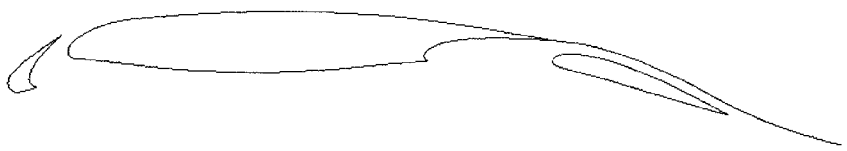

Figure 4.6: Close-up of multi-element airfoil geometry and wake definition.

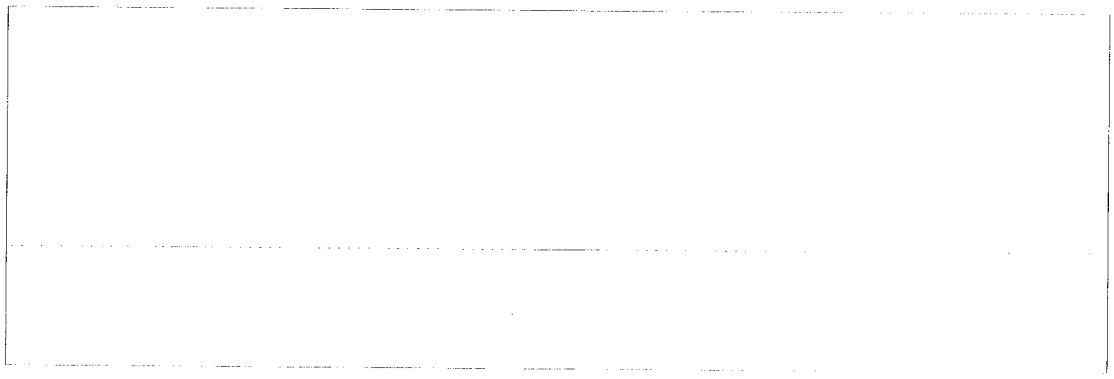

Figure 4.7: Polygonal region for multi-element topology (with decomposition rectangles shown in dashed lines).

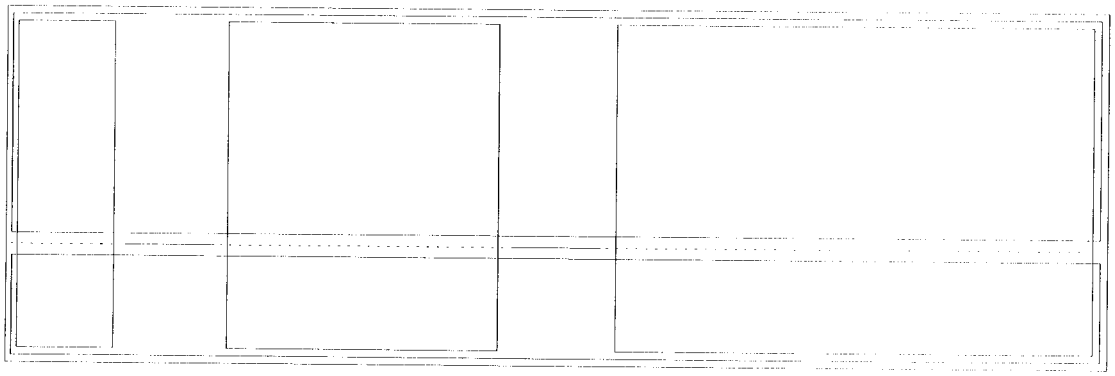

Figure 4.8: Polygonal region for multi-element topology, showing the five prime rectangles.

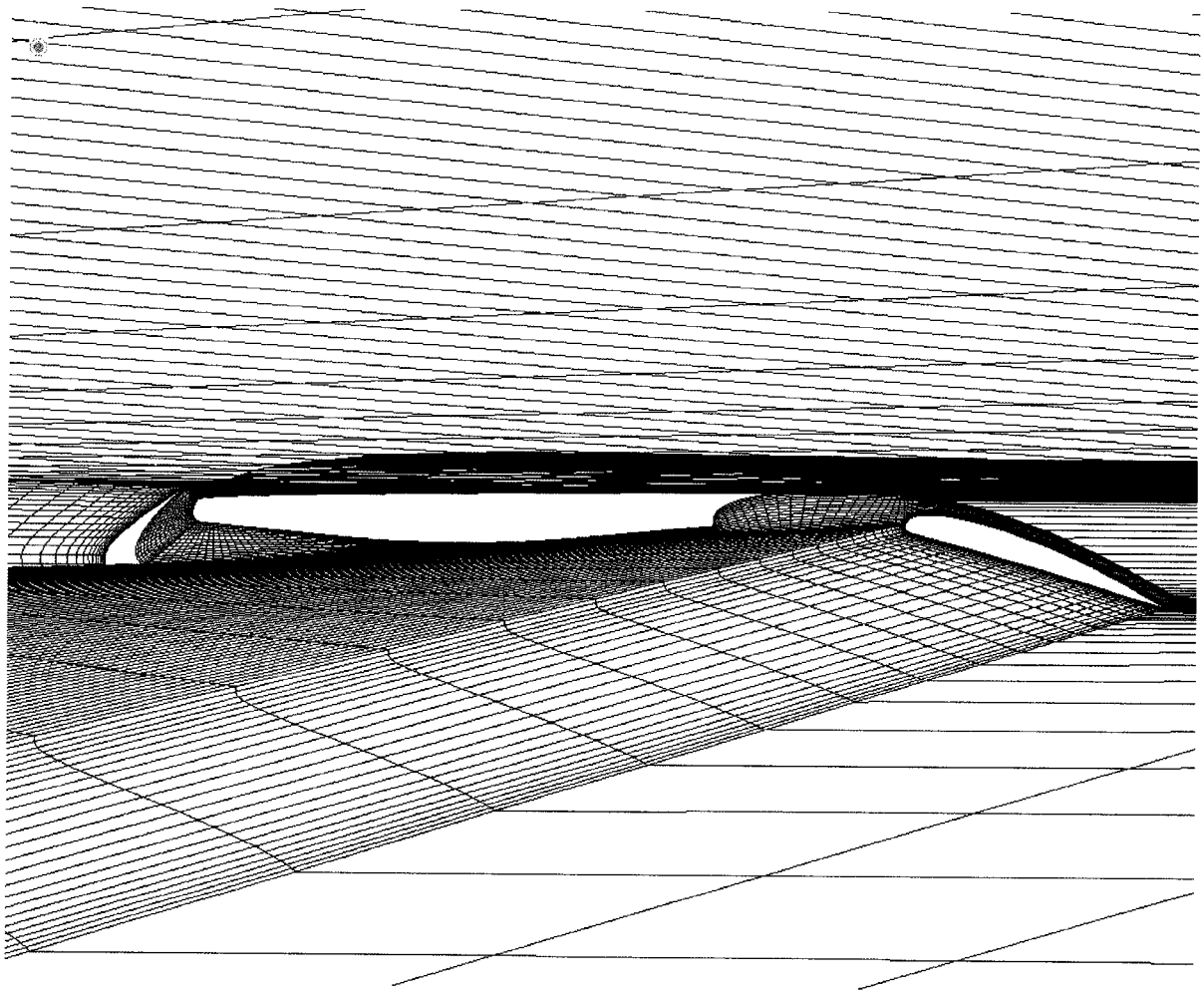

Figure 4.9: Initial algebraic mesh around high-lift system.

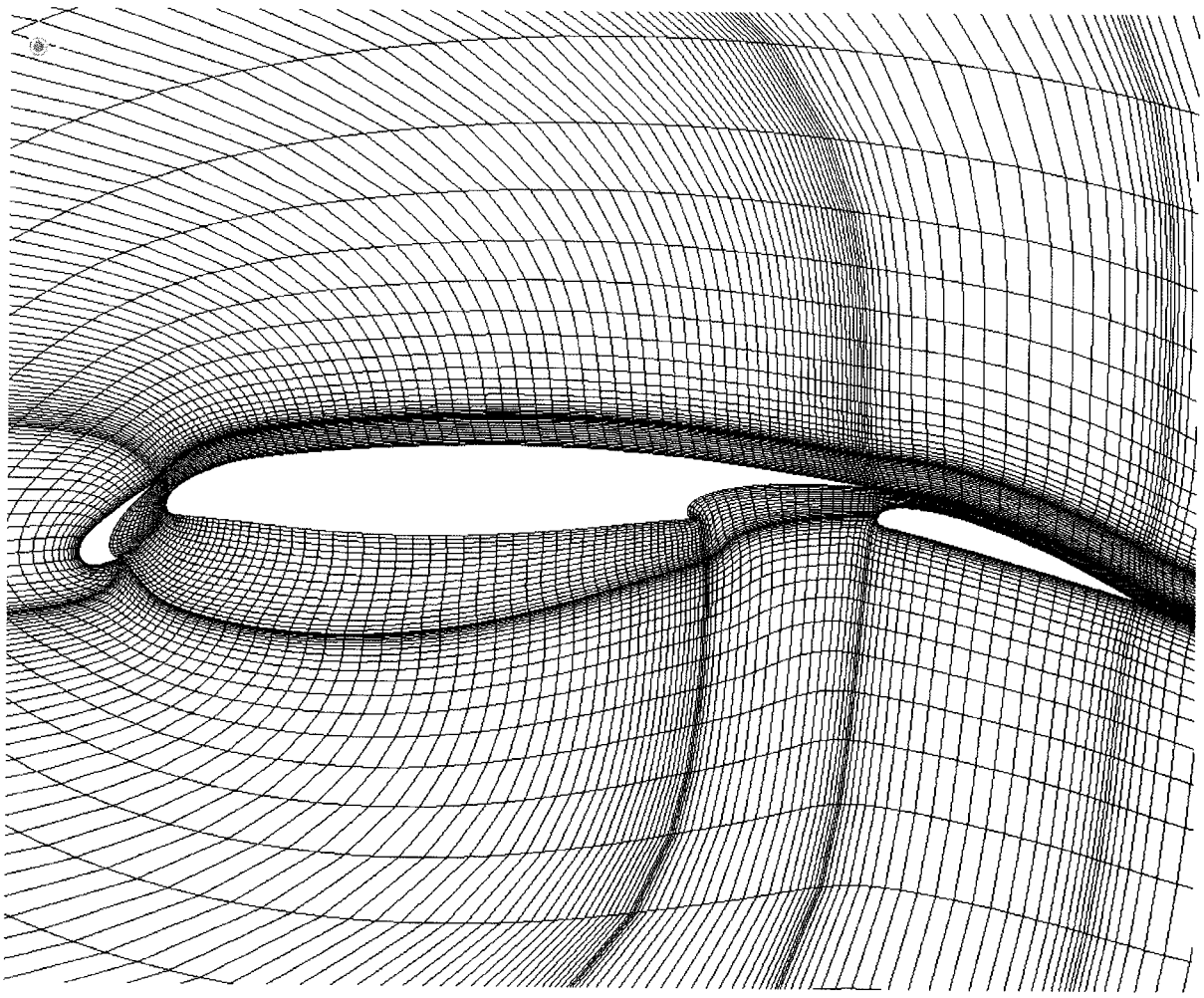

Figure 4.10: Mesh optimized with curvature operator around high-lift system.

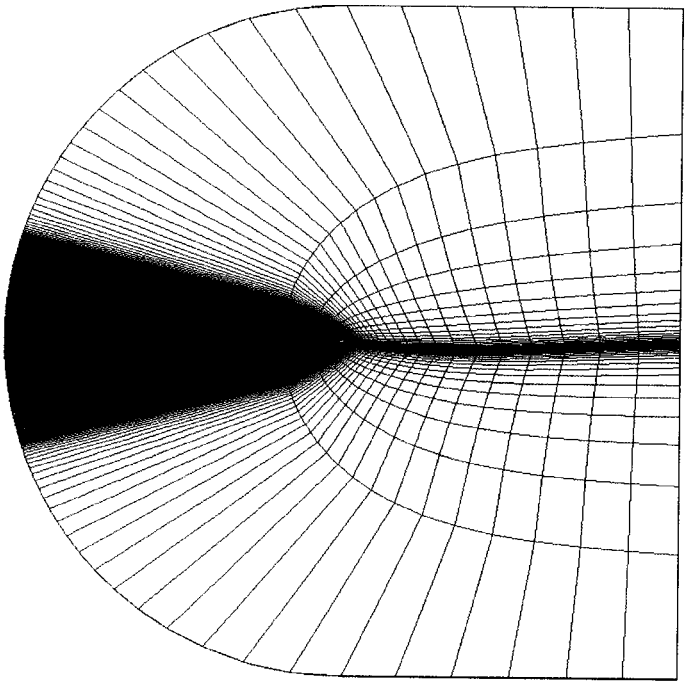

Figure 4.11: Mesh optimized with Laplacian-based operator.

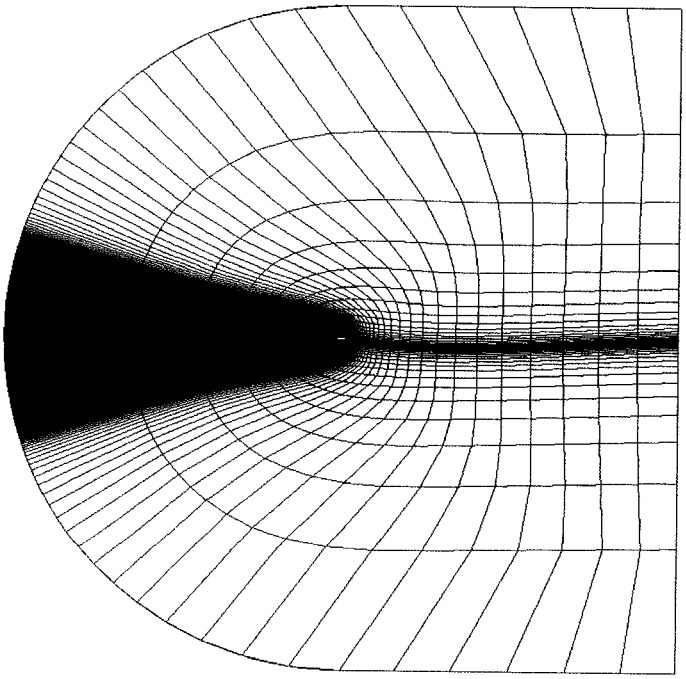

Figure 4.12: Mesh optimized with curvature operator.

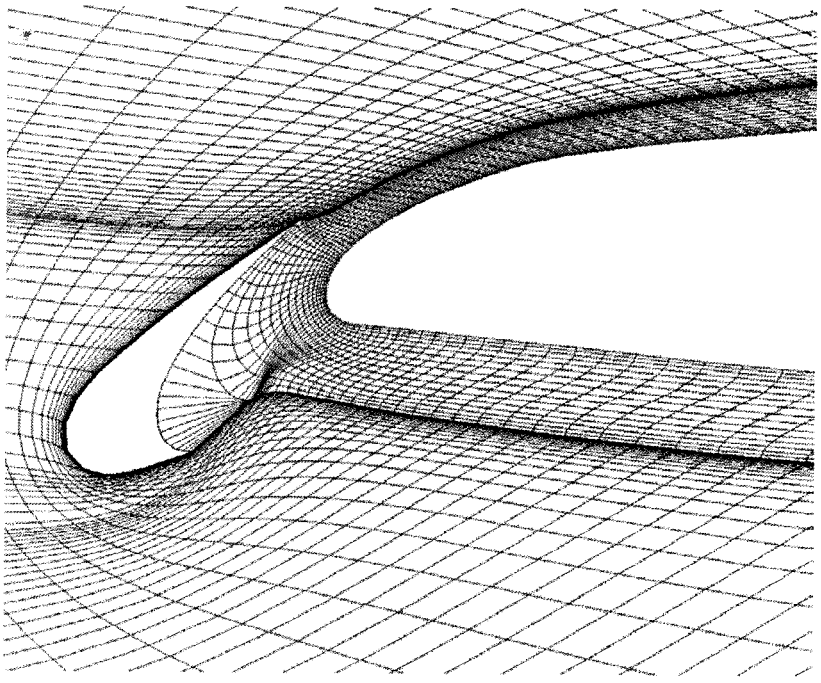

Figure 4.13: Mesh optimized with Laplacian-based operator.

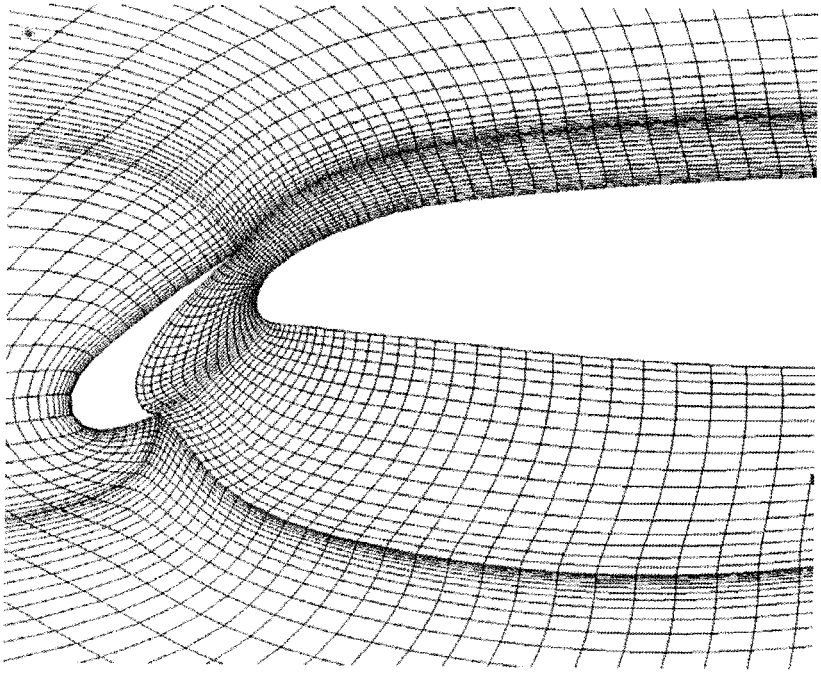

Figure 4.14: Mesh optimized with curvature operator.

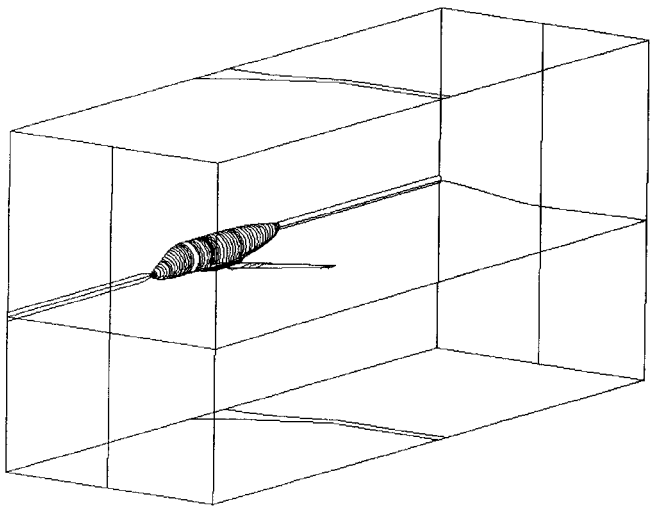

Figure 4.15: Physical space around a wing-fuselage configuration.

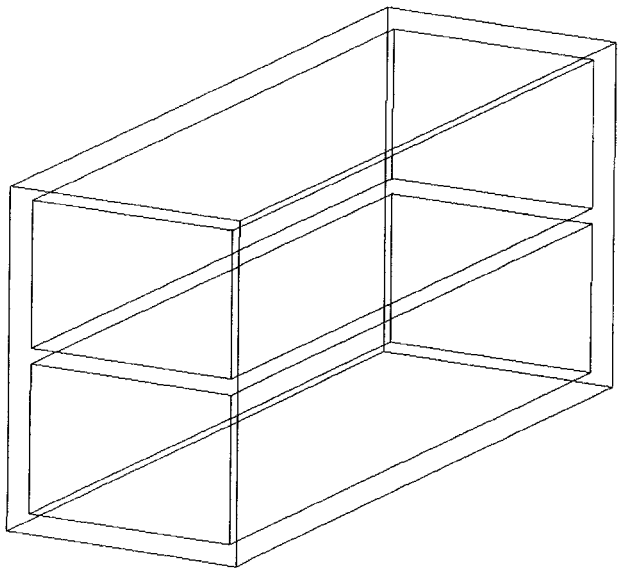

Figure 4.16: Decomposition and prime rectangles for H-H mesh around wing-fuselage configuration; rectangles represent zones above and below wing in topological space.

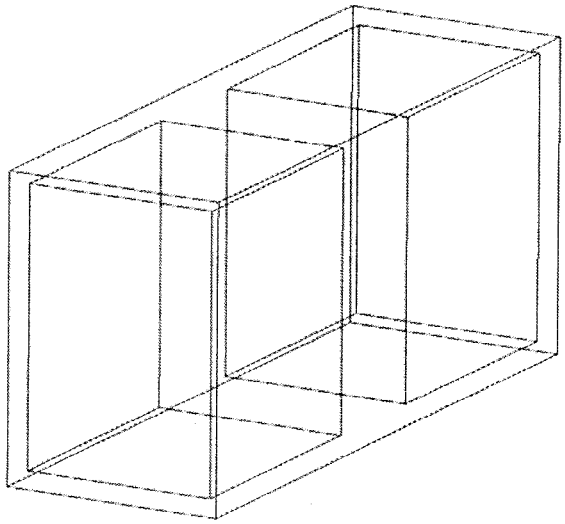

Figure 4.17: Prime rectangles forward and aft of wing in topological space.

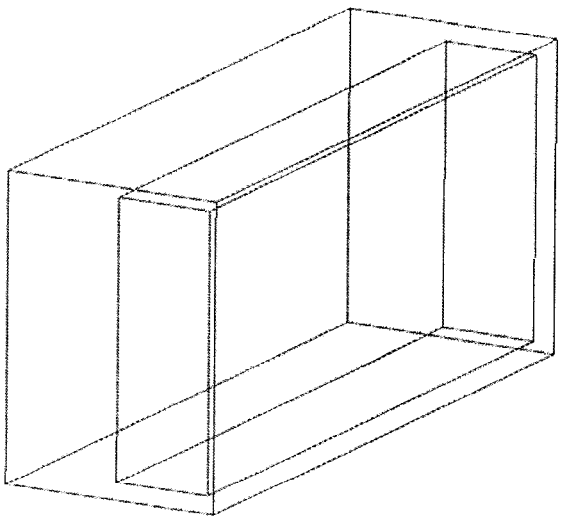

Figure 4.18: Prime rectangle outboard of wing in topological space.

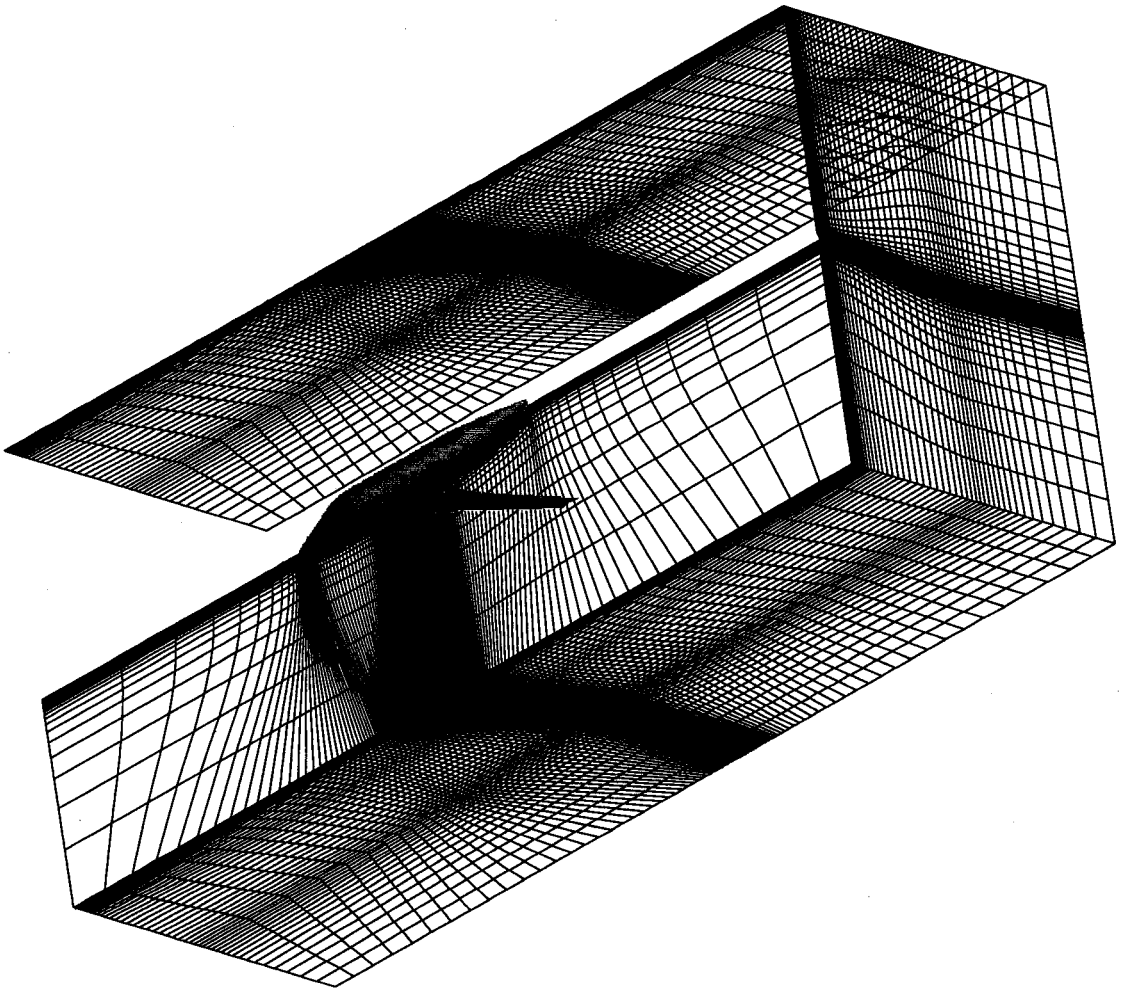

Figure 4.19: Point distribution on far-field boundaries for wing-fuselage mesh.

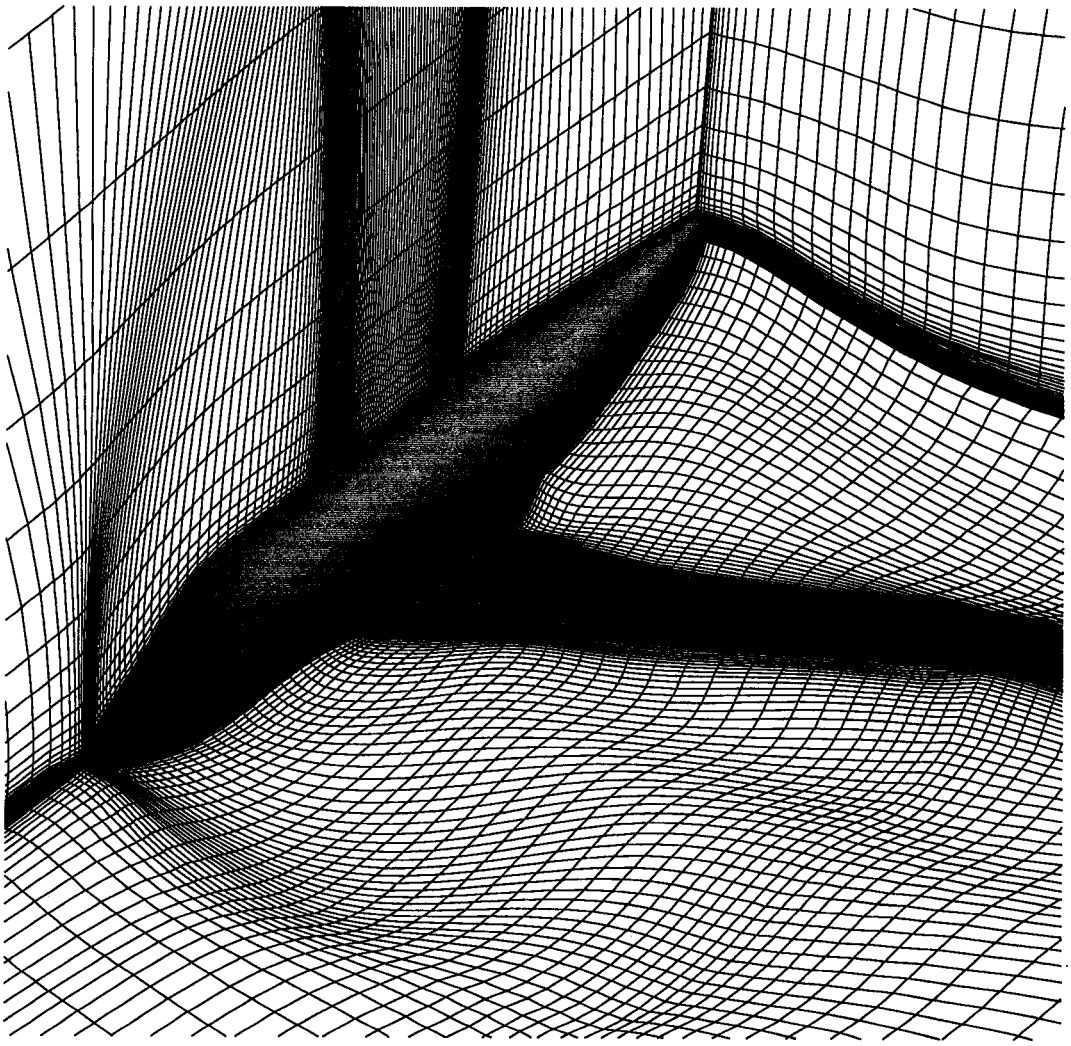

Figure 4.20: Optimized mesh for wing-fuselage configuration.

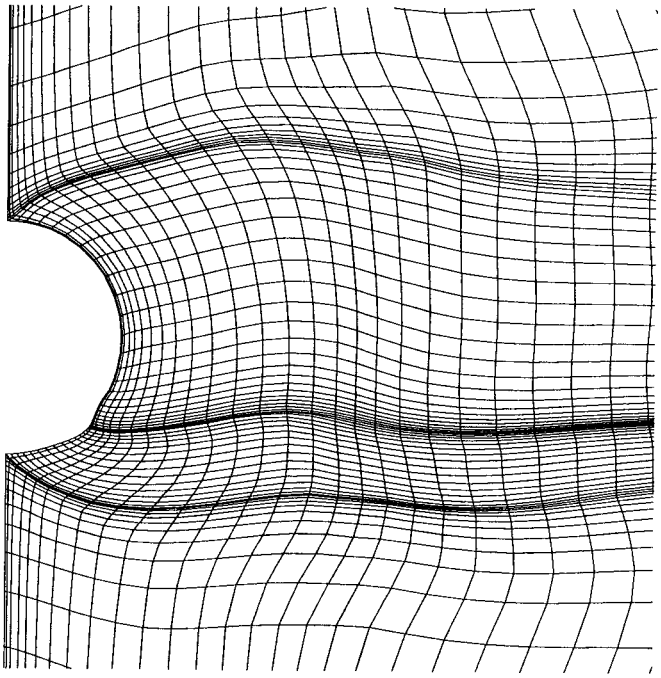

Figure 4.21: Front view of a grid plane intersecting fuselage forward of wing.

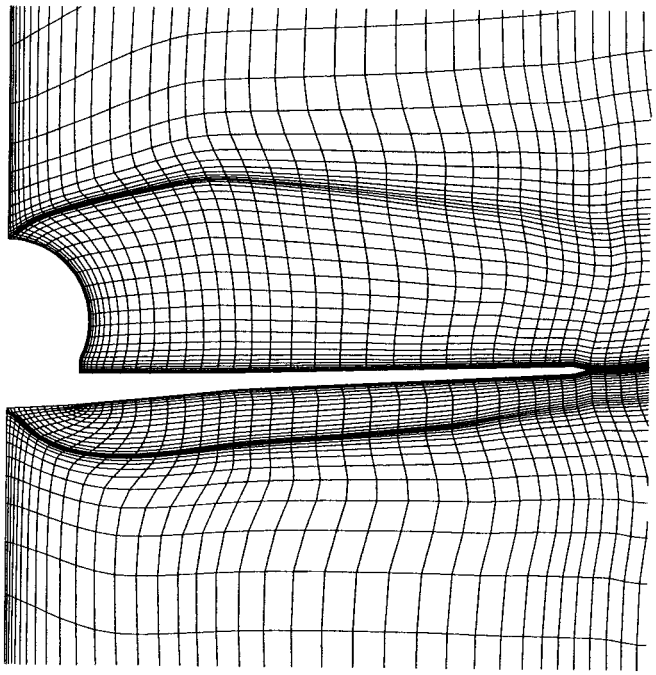

Figure 4.22: Front view of a grid plane intersecting wing and fuselage.

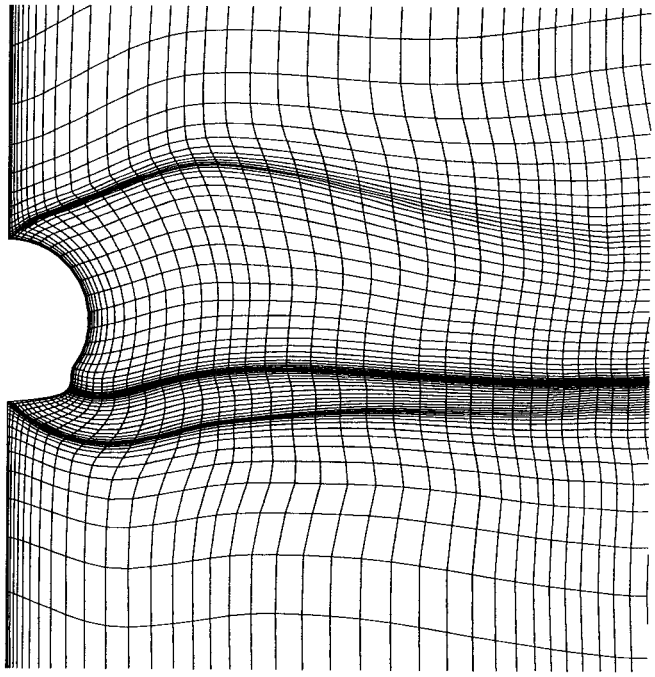

Figure 4.23: Front view of a grid plane intersecting fuselage aft of wing.

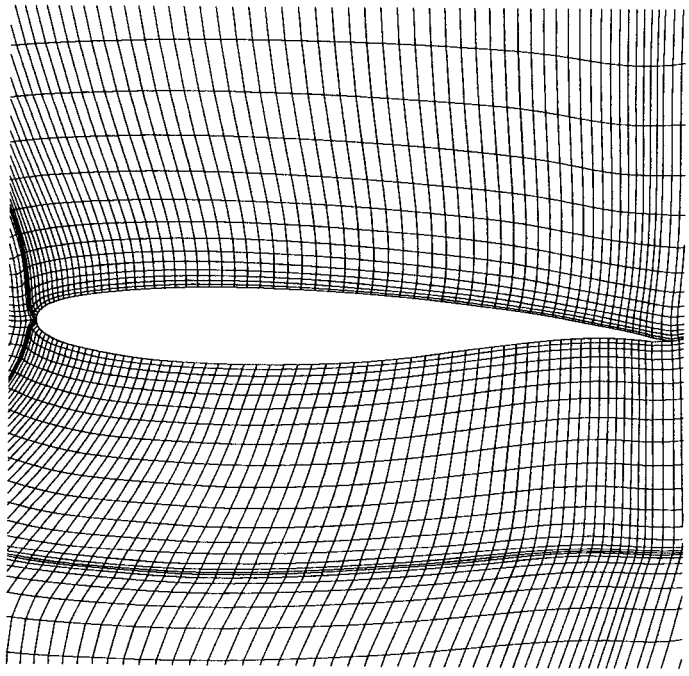

Figure 4.24: Side view of a grid plane intersecting wing.

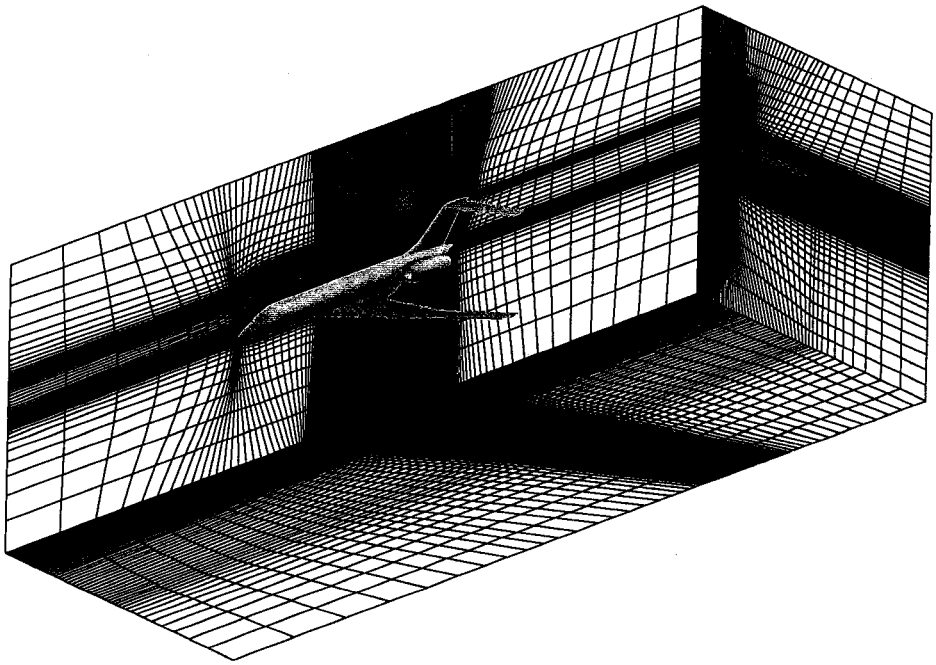

Figure 4.25: Full aircraft configuration showing mesh on far-field boundaries.

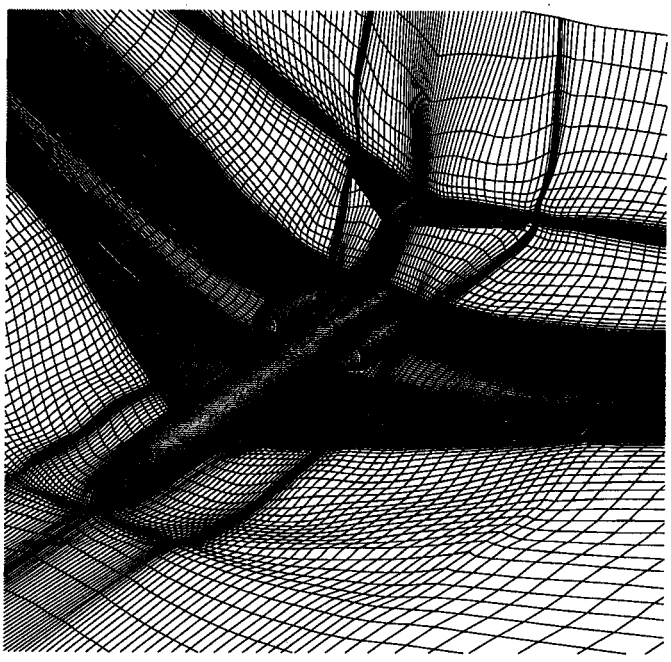

Figure 4.26: Isometric view of optimized full aircraft mesh.

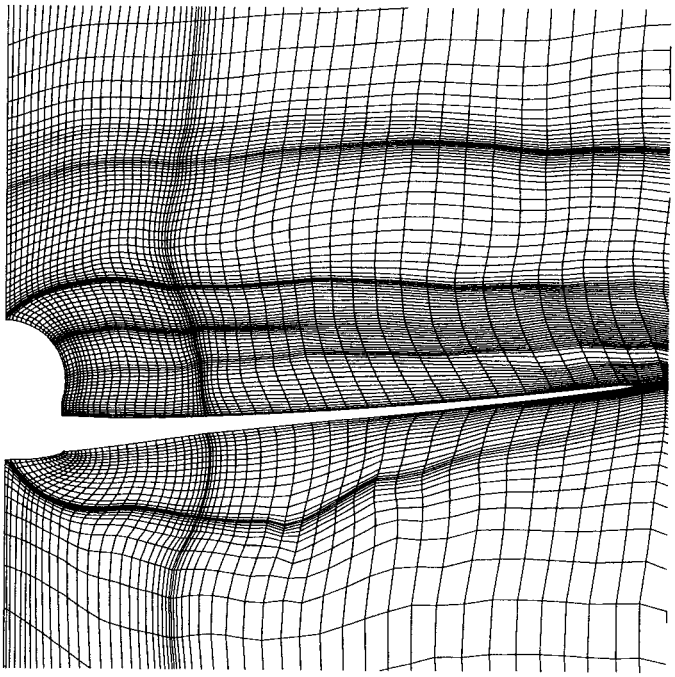

Figure 4.27: Front view of mesh intersecting wing and fuselage.

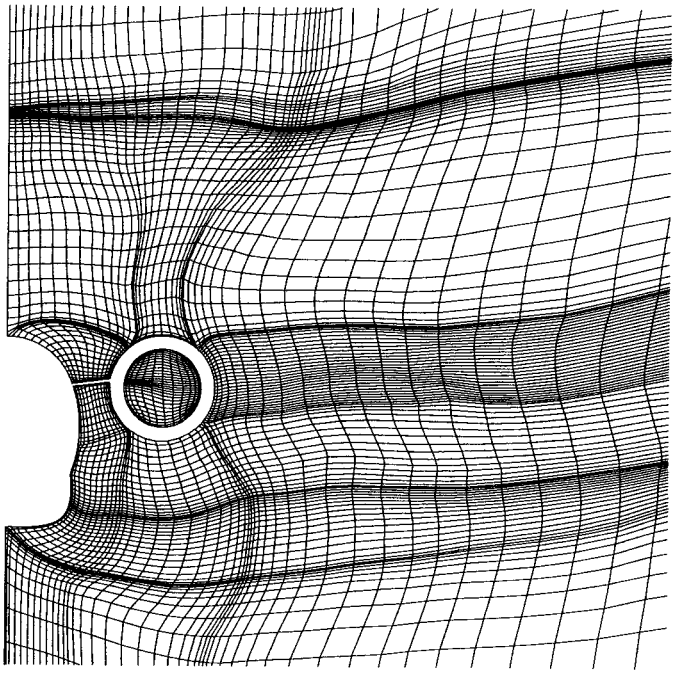

Figure 4.28: Front view of mesh intersecting fuselage and nacelle, midway between the wing and tail surfaces.

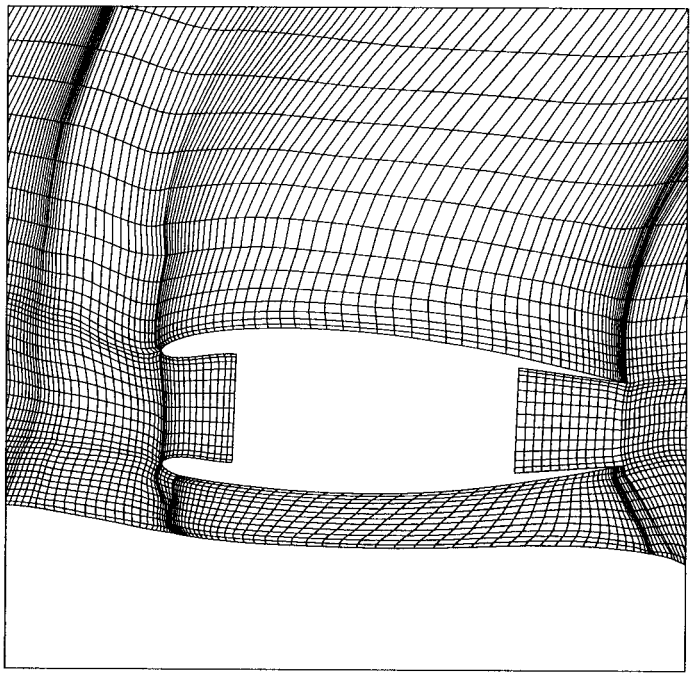

Figure 4.29: Top view showing local mesh around nacelle.

Conclusion

A new approach was presented for the automation of structured grid generation in multiply-connected domains. In this approach, the domain decomposition problem is cast as a classical boundary value problem in which the mesh topology is defined through the imposition of appropriate boundary conditions on the domain boundaries. The automation of the domain decomposition process is achieved by transferring it from the physical space to the topological space, where it is amenable to a rigorous solution. Once the domain is decomposed in the topological space, the mesh is generated in the physical space via the solution of a non-linear elliptic partial differential operator which takes into account the curvature of the physical space. The forms of the decomposition surfaces are obtained as part of the solution of the differential operator. The latter is solved iteratively in a system of overlapping sub-domains in which the decomposition surfaces are left floating, and in which only the shape of the domain boundaries and the point distribution thereon influence the form of the final mesh.

It was shown that the proper representation of domain curvature is an essential element to the success of the domain decomposition strategy. In any curved space, the curvature of the decomposition surfaces must mirror the curvature of the space in order to yield a high quality mesh. Since the decomposition of the multiply-connected domain is done in the topological space, the curvature of the physical space must be re-injected into the system through the solution of an appropriate differential operator. A new mathematical formulation was derived for this purpose and takes the form

of a new forcing function in the elliptic grid generation equations. This new curvature term is completely general and can be applied to both two- and three-dimensional domains of arbitrary shape.

The combination of the new grid generation equations and the domain decomposition strategy provides a methodology for generating structured meshes in multiply-connected domains without the requirement of a manual decomposition of the space. This method can be applied to both convex and non-convex geometries, and can be used to generate meshes of any topology for which the curvilinear coordinates are continuous.

It is important to note, however, that the quality of the final mesh often depends on the choice of boundary conditions (i.e. the selection of topology) and the associated boundary point distribution. In three-dimensions, not all boundary conditions will lead to valid topologies, and even when they do, the topology and the point distribution on the boundaries must be chosen in a sensible manner to produce a good mesh. As the complexity of the domain increases, the sheer number of boundary conditions may at some point make this process tedious.

A further consideration is the situation in which embedded meshes are required, such as the case of an O-mesh embedded inside a H-mesh to model the viscous layer around a geometry. Generally, each embedded region represents a separate boundary value problem, since the boundary between the inner and outer regions can be of arbitrary shape, size, and location, and the topology of the embedded region is non-unique and must be selected in the same manner as the outer region, i.e. via boundary conditions. In the present method, since no simplifying assumptions are made with respect to the topology of the embedded region, its perimeter must be provided as an input, along with the boundary conditions defining its internal topology. However, each region can be a *multiply-connected* domain in its own right.

The most important element to the success of the present method is the quality of the mesh produced in the optimization step, i.e. the mapping of the decomposed

topological space back into the physical space. Thus, a robust mesh optimization program, capable of handling general three-dimensional domains with arbitrary point distributions on all boundaries, is required. Any limitation of the mesh optimization program may translate into a limitation in the automation of the domain decomposition process. However, this does not represent an inherent limitation to the method. The continuing development of the technology in this area will allow the application of this methodology to increasingly complex domains. In this sense, the present method transfers the difficulty of the domain decomposition problem to the science of mesh optimization.

References

AKDAG, V. and WULF, A. (1992). Integrated geometry and grid generation system for complex configurations. In Smith, R., editor, *Proceedings from the NASA Workshop on Software Systems for Surface Modeling and Grid Generation*, Hampton, Virginia. NASA Langley Research Center.

ALLWRIGHT, S. E. (1988). Techniques in domain decomposition and surface grid generation. In *Numerical Grid Generation in Computational Fluid Dynamics '88.* Pineridge Press Limited.

AMSDEN, A. A. and HIRT, C. W. (1973). A simple scheme for generating general curvilinear grids. *Journal of Computational Physics*, 11:348.

ANDREWS, A. E. (1987). Progress and challenges in the application of artificial intelligence to computational fluid dynamics. *AIAA Journal*, 26(1).

ANDREWS, A. E. (1988). Knowledge-based flow field zoning. In *Numerical Grid Generation in Computational Fluid Dynamics '88.* Pineridge Press Limited.

ANDREWS, A. E. (1990). Automated domain decomposition for computational fluid dynamics. *Journal of Computers and Fluids*, 18(4):329–346.

BAKER, T. (1991). Single block mesh generation for a fuselage plus two lifting surfaces. In *Numerical Grid Generation in Computational Fluid Dynamics.* Elsevier Science Publishers, B.V., North-Holland.

BARFIELD, W. D. (1970). An optimal mesh generator for Lagrangian hydrodynamic calculations in two space dimensions. *Journal of Computational Physics*, 6:417.

BERGMAN, M. (1990). *Development of Numerical Techniques for Inviscid Hypersonic Flows Around Re-entry Vehicles.* PhD thesis, INP Toulouse.

BERTIN, D., LORDON, J., and MOREAUX, V. (1992). A new automatic grid generation environment for CFD applications. In *AIAA Paper 92-2720-CP.*

CAUGHEY, D. A. and JAMESON, A. (1980). Progress in finite-volume calculations for wing-fuselage combinations. *AIAA Journal*, 18(11).

CHU, W. H. (1971). Development of a general finite difference approximation for a general domain. *Journal of Computational Physics*, 8:392.

CORDOVA, J. Q. (1992). Toward a theory of automated elliptic mesh generation. In Smith, R., editor, *NASA Workshop on Software Systems for Surface Modeling and Grid Generation*, Hampton, Virginia. NASA.

DANNENHOFFER III, J. F. (1991). Computer-aided block-structuring through the use of optimization and expert system techniques. In *AIAA-91-1565.*

DANNENHOFFER III, J. F. (1993). A new method for creating grid abstractions for complex configurations. In *AIAA - 31st Aerospace Sciences Meeting & Exibit*, number AIAA-93-0428.

DANNENHOFFER III, J. F. (1995). Automatic blocking for complex three-dimensional configurations. In *Surface Modeling, Grid Generation, and Related Issues in Computational Fluid Dynamics*, number N96-28723 10-02, pages 123–142. NASA Lewis Research Center.

DANNENHOFFER III, J. F. (1996). Automatic generation of block structures - progress and challenges. In *5th International Conference on Numerical Grid Generation in Computational Field Simulations*, pages 403–412. Mississippi State University.

DENER, C. and HIRSCH, C. (1991). IIGG - an advanced interactive grid generation system. In *Numerical Grid Generation in Computational Fluid Dynamics*, North-Holland. Elsevier Science Publishers, B.V.

FUJITANI, K. and HIMENO, R. (1991). A CAD data oriented grid generation system and its application to automotive aerodynamics. In *Numerical Grid Generation in Computational Fluid Dynamics*. Elsevier Science Publishers, B.V., North-Holland.

GAREY, M. R. and JOHNSON, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completenesss*. W.H. Freeman and Company, New York.

GHIA, U., GHIA, K. N., and RAMAMURTI, R. (1983). Hybrid c-h grids for turbomachine cascades. In *Advances in Grid Generation*, Houston. ASME Fluids Engineering Conference.

GODUNOV, S. K. and PROKOPOV, G. P. (1972). The use of moving meshes in gas-dynamical computations. *USSR Comp. Math. Phys.*, 12:182.

HERRMAN, U. (1991). IMESH - an interactive mesh generation package for graphics super workstations. In *Numerical Grid Generation in Computational Fluid Dynamics*, North-Holland. Elsevier Science Publishers, B.V.

JAMESON, A. (1974). Iterative solution of transonic flows over airfoils and wings, including flows at mach 1. *Comm. Pure. Appl. Math.*, 27:283–309.

KING, D. A. and WILLIAMS, B. R. (1988). Developments in computational methods for high-lift aerodynamics. *Aeronautical Journal*, 92:265–288.

KLEVENHUSEN, K. D. (1982). 2D elliptic grid generation using a singularity method and its application to transonic interference flows. In Thompson, J., editor, *Numerical Grid Generation*. North Holland.

LIPSKI, W., LODI, E., LUCCIO, F., MUGNAI, C., and PAGLI, L. (1979). On two-dimensional data organization II. *Annales Societis Mathematicae Polonae, Series IV: Fundamentae Informaticae II*, pages 245–260.

LO, S. H. (1985). A new mesh generation scheme for arbitrary planar domains. *International Journal for Numerical Methods in Engineering*, 21:1403–1426.

LODI, E., LUCCIO, F., MUGNAI, C., and PAGLI, L. (1979). On two-dimensional data organization I. *Annales Societis Mathematicae Polonae, Series IV: Fundamentae Informaticae II*, pages 211–226.

LUBIW, A. (1985). Decomposing polygonal regions into convex quadrilaterals. In *1st Symposium on Computational Geometry*. ACM.

MANHARDT, P. D. and BAKER, A. J. (1982). Automated three-dimensional grid refinement on a minicomputer. In Thompson, J., editor, *Numerical Grid Generation*. North Holland.

MURMAN, E. M. and COLE, J. D. (1971). Calculation of plane steady transonic flows. *AIAA Journal*, 9(1):114–121.

NELSON, T. E., ZINGG, D. W., and JOHNSTON, G. W. (1991). Automated grid generation for high-lift configurations. In *3rd Canadian Symposium on Aerodynamics*, Toronto.

O'ROURKE, J. (1993). *Computational Geometry in C*. Cambridge University Press.

PARK, S. and LEE, K. (1999). A new approach to automated multi-block decomposition for grid generation: A hypercube++ approach. In Thompson, J., Soni, B., and Weatherill, N., editors, *Handbook of Grid Generation*. CRC Press.

PIPERNI, P. (1994). Multi-block grid generation with CAD-based domain decomposition techniques. In *Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*, pages 109–121, Swansea, U.K. Pineridge Press Ltd.

PIPERNI, P. (1998). New elliptic grid generation equations with exact curvature control. In *6th International Conference on Numerical Grid Generation in Computational Field Simulations*, pages 267–276, Greenwich, U.K. International Society of Grid Generation.

PIPERNI, P. and BOUDREAU, J. (2003). The evolution of structured grid generation at Bombardier Aerospace. In *Proceedings of 9th Aerodynamics Symposium of the Canadian Aeronautics and Space Institute*, Montreal, Canada.

PIPERNI, P., MOKHTARIAN, F., and KAFYEKE, F. (1992). Multi-block grid generation for complete aircraft configurations. *Canadian Aeronautics and Space Journal*, 38(4).

RAJ, R., BRENNAN, J. E., KEEN, J. M., MANI, K. K., and OLLING, C. R. (1987). Three-dimensional Euler aerodynamic method (team), vol.1: Computational method. Technical Report TR-87-3074, Air Force Wright Aero. Lab., Wright-Patterson AFB, OH.

REMOTIGUE, M. G., HART, E. T., and STOKES, M. L. (1992). Eagleview: A surface and grid generation program and its data management. In Smith, R., editor, *Proceedings from the NASA Workshop on Software Systems for Surface Modeling and Grid Generation*, Hampton, Virginia. Langley Research Center.

ROBERTS, A. (1982). Automatic topology generation and generalized B-spline mapping. In Thompson, J., editor, *Numerical Grid Generation.* North Holland.

RUPPERT, P. E. and LEE, K. D. (1980). Transonic flow computations using grid systems with block structure. In *7th International Conference on Numerical Methods in Fluid Dynamics*, pages 266–271, New-York. Springer-Verlag.

SCHONFELD, T. and WEINERFELT, P. (1991). The automatic generation of quadrilateral multi-block grids by the advancing front technique. In *Numerical Grid Generation in Computational Fluid Dynamics*, North-Holland. Elsevier Science Publishers, B.V.

SCHUSTER, D. M. (1992). Batch mode grid generation: An endangered species? In Smith, R., editor, *Proceedings from the NASA Workshop on Software Systems for Surface Modeling and Grid Generation*, Hampton Virginia. Langley Research Center.

SCHUSTER, D. M. and ATTA, E. H. (1989). Flight loads prediction methods for fighter aircraft, volume II - complete aircraft mesh program (CAMP) User's guide. Technical Report WRDC-TR-89-3104, WRDC.

SEIBERT, W. (1988). A graphic-interactive program-system to generate composite grids for general configurations. In *Numerical Grid Generation in Computational Fluid Dynamics '88*. Pineridge Press Limited.

SETHIAN, J. A. (1994). Curvature flow and entropy conditions applied to grid generation. *Journal of Computational Physics*, 115:440–454.

SHAW, J. A., GEORGALA, J. M., and WEATHERILL, N. P. (1988). The construction of component-adaptive grids for aerodynamic geometries. In *Numerical Grid Generation in Computational Fluid Dynamics '88*. Pineridge Press Limited.

SHAW, J. A. and WEATHERILL, N. P. (1992). Automatic topology generation for multi-block grids. *App. Mathematics and Computation*, 52:355–388.

SHIMA, E. (1980). Numerical analysis of multiple element high lift devices by Navier-Stokes equation using implicit TVD finite volume method. In *AIAA Paper 88-2574-CP*.

SMITH, R. E. (1980). Numerical grid generation techniques. Technical Report CP2166, NASA.

SONI, B. K. (1991). TIGER: Turbomachinery interactive grid generation. In *Numerical Grid Generation in Computational Fluid Dynamics*, North-Holland. Elsevier Science Publishers, B.V.

SORENSON, R. L. (1988). Three-dimensional zonal grids about arbitrary shapes by Poisson's equation. In Sengupta, Hauser, J., Eiseman, P., and Thompson, J., editors, *Numerical Grid Generation in Computational Fluid Dynamics '88*, Swansea, UK. Pineridge.

SORENSON, R. L. and McCANN, K. (1992). Grapevine: Grids about anything by Poisson's equation in a visually interactive networking environment. In Smith, R., editor, *Proceedings from the NASA Workshop on Software Systems for Surface Modeling and Grid Generation*, Hampton, Virginia. Langley Research Center.

SORENSON, R. L. and STEGER, J. L. (1983). Grid generation in three dimensions by Poisson equations with control of cell size and skewness at boundary surfaces. In *Advances in Grid Generation*, Houston. ASME Fluids Engineering Conference.

SPEKREIJSE, S. P. (1995). Elliptic grid generation based on Laplace equations and algebraic transformations. *Journal of Computational Physics*, 118:38–61.

SPEKREIJSE, S. P. (1999). Elliptic generation systems. In Thompson, J., Soni, B., and Weatherill, N., editors, *Handbook of Grid Generation*. CRC Press.

SPEKREIJSE, S. P. and KOK, J. C. (2000). Semi-automatic domain decomposition based on potential theory. In *7th International Conference on Numerical Grid Generation in Computational Field Simulations*, pages 135–144, Whistler, Canada.

STEGER, J. L., DOUGHERTY, F. C., and BENEK, J. A. (1985). A chimera grid scheme. In Ghia, K. and Ghia, U., editors, *Advances in Grid Generation*, volume 5. American Society of Mechanical Engineers FED.

STEINBRENNER, J. P. and CHAWNER, J. R. (1992). Recent enhancements to the GRIDGEN structured grid generation system. In Smith, R., editor, *Proceedings from the NASA Workshop on Software Systems for Surface Modeling and Grid Generation*, Hampton, Virginia. Langley Research Center.

STOKES, S., LEATHAM, M., and SHAW, J. A. (2000). Three dimensional hybrid mesh generation for viscous flows using the advancing-layer approach with automatic layer topology modification. In *7th International Conference on Numerical Grid Generation in Computational Field Simulations*, pages 233–242, Whistler, Canada.

TAKAHASHI, S. and EISEMAN, P. R. (1994). Adaptive grid movement with respect to boundary curvature. In *Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*, pages 563–572, Swansea, U.K. Pineridge Press Ltd.

THOMPSON, J. F. (1982). *Numerical Grid Generation.* North Holland.

THOMPSON, J. F. (1987). A composite grid generation code for general 3D regions. In *AIAA Paper 87-0275.*

THOMPSON, J. F., THAMES, F. C., and MASTIN, C. W. (1974). Automatic numerical generation of body-fitted curvilinear coordinate system for field containing any number of arbitrary two-dimensional bodies. *Journal of Computational Physics*, 15:299–319.

THOMPSON, J. F., WARSI, Z. U. A., and MASTIN, C. W. (1985). *Numerical Grid Generation, Foundations and Applications.* North Holland, New York.

VASSBERG, J. C. (1999). Multi-block mesh extrusion driven by a globally-elliptic system. In *5th U.S. National Congress of Computational Mechanics, 2nd Symposium on Trends in Unstructured Mesh Generation*, Bolder, CO. University of Colorado.

VASSBERG, J. C. (2000). Further study of globally elliptic meshing. In *7th International Conference on Numerical Grid Generation in Computational Field Simulations*, pages 101–110, Whistler, Canada.

WARSI, Z. U. A. (1981). Tensors and differential geometry applied to analytic and numerical coordinate generation. Technical Report Report MSSU-EIRS-ASE-81-1, Mississippi State University.

WEATHERILL, N. P. and FORSEY, C. R. (1984). Grid generation and flow calculations for complex aircraft geometries using a multi-block scheme. In *AIAA Paper 84-1665.*

WINSLOW, A. M. (1966). Numerical solution of the quasi-linear Poisson equation in a non-uniform triangular mesh. *Journal of Computational Physics*, 2(2):149.

WULF, A. and AKDAG, V. (1995). Tuned grid generation with ICEM CFD. In *NASA-CP-3291, Workshop Proceedings: Surface Modeling, Grid Generation and Related Issues in Computational Fluid Dynamics.*

Appendix A

Decomposition Algorithms

A.1 Lipski Algorithm

The algorithm developed by LIPSKI et al. (1979) performs the decomposition of a two-dimensional rectilinear polygonal region into a set of rectangles. Lipski's algorithm defines a set of disjoint (i.e. non-overlapping) rectangles which covers the entire area inside the polygon.

As shown in figure A.1, a rectilinear polygonal region is composed of an external contour and possibly one or more internal contours. The spaces enclosed by the internal contours, if present, are termed the *holes* of the polygonal region. The external and internal contours consist of *edges* and *vertices*, where the edges are either vertical or horizontal.

Two edges of the contour are said to be *aligned* if they lie on the same straight line, and can be connected by a line. A vertex is described as either *concave* or *convex*, where the contour is concave or convex, respectively.

There are two essential elements to Lipski's algorithm. The first element, which

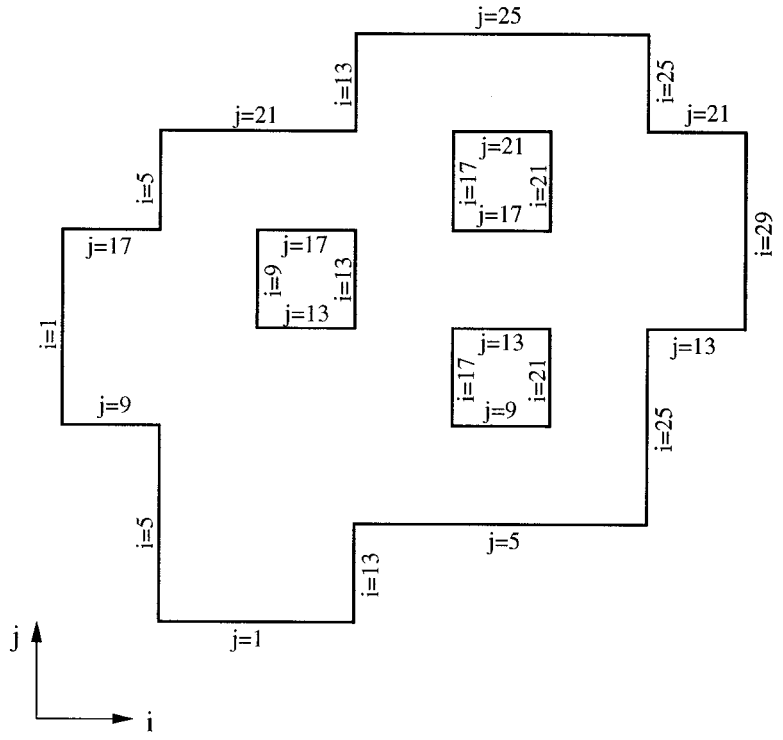

Figure A.1: Mesh indices as boundary conditions for multiply-connected domain.

pertains to polygonal regions *without* aligned edges, relates the minimum number N of rectangles covering the region to the number of holes and concave vertices. Letting *ho* and *cv* denote the number of holes and concave vertices in the polygonal region, respectively, it can be shown that

$$N = cv - ho + 1 \tag{A.1}$$

Thus, for a polygon with no aligned edges, the number of rectangles in the decomposition is proportional to the number of concave vertices in the contour. Based on the above relation, the polygon can be decomposed by simply drawing one *internal* line in the domain for every concave vertex. Each internal line emanates from a concave vertex, either vertically or horizontally, and terminates on an adjacent part of the contour or on another internal line, whichever is nearest. The resulting set of internal lines delineates the decomposition of the polygon into the minimum number

of rectangles. Note that there may be several distinct decompositions that yield the minimum number of rectangles.

The second element of Lipski's algorithm deals with the presence of aligned edges in the polygon. Lipski showed that when aligned edges are present, the minimum number of rectangles can be obtained only if all the aligned edges are connected first (with internal lines), before decomposing the rest of the domain. There are two criteria to observe when connecting the aligned edges. Firstly, the connecting lines must not touch or cross one another. Secondly, the aligned edges must be connected in such a manner as to produce the *maximum* number of connected *couples* of aligned edges.

For polygons with aligned edges, the minimum number of rectangles is given by the following relation. Letting S denote the maximum number of connected couples of aligned edges, it can be shown that

$$N = cv - ho - S + 1 \tag{A.2}$$

Based on the above principles, the implementation of the Lipski algorithm is done in the four steps described below.

A.1.1 Ordering of Edges

The rectilinear polygonal region is defined by an equal number of vertical and horizontal edges. The edges are input in a list where each edge is defined by three integers. For vertical ($i = constant$) edges, the first number denotes the i value of the edge, and the other two specify the lower and upper j limits of the edge. For horizontal ($j = constant$) edges, the first number denotes the j value of the edge, and the other two specify the lower and upper i limits of the edge.

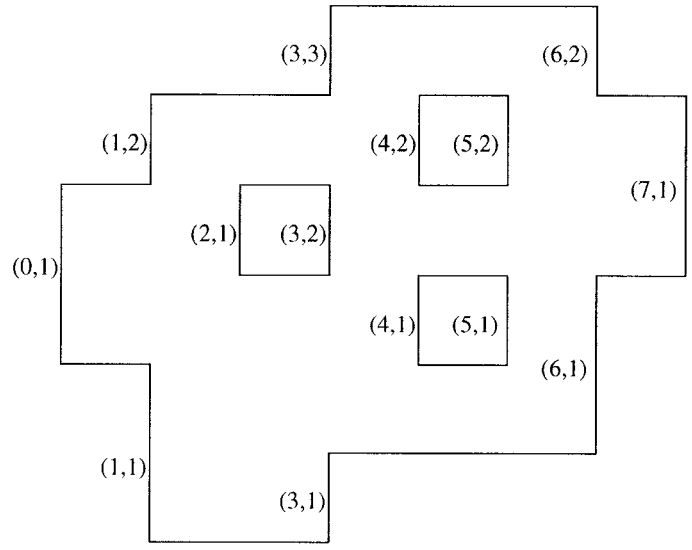

Figure A.2: Ordering of vertical edges in topological space.

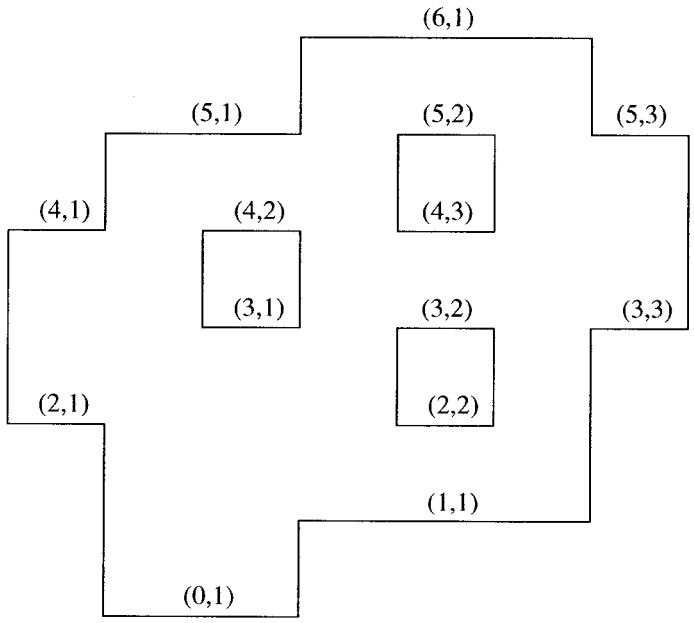

Figure A.3: Ordering of horizontal edges in topological space.

Regardless of the order of the input edges, the edges are re-ordered to facilitate the searching process and minimize the number of operations. As shown in figure A.2, the vertical edges are sorted in $i = constant$ rows from left to right. In each i-row, the edges are sorted from bottom to top. Thus, a vertical edge denoted by the address (3,2) is the second edge from the bottom in the third i-row.

Similarly, the horizontal edges are sorted in $j = constant$ rows from bottom to top, as shown in figure A.3. In each j-row, the edges are sorted from left to right. Thus, an horizontal edge denoted by the address (3,2) is the second edge from the left in the third j-row.

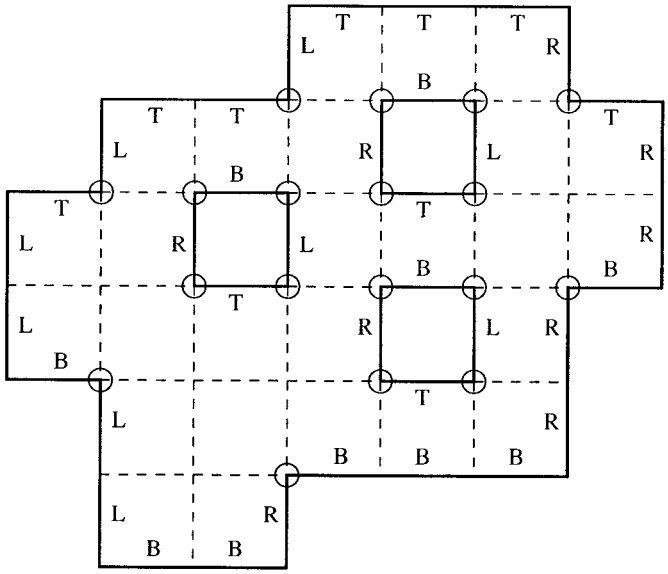

Figure A.4: Identification of left, right, bottom, and top boundaries of domain. The encircled vertices are identified as concave, the remaining vertices are convex.

A.1.2 Identification of Edges and Vertices

Every vertical edge is identified as either a Left (L) edge or a Right (R) edge. A Left edge is one for which the interior of the domain is to the right, and a Right edge is one for which the interior of the domain is to the left.

Similarly, every horizontal edge is identified as either a Bottom (B) edge or a Top (T) edge. A Bottom edge is one for which the interior of the domain lies above, and a Top edge is one for which the interior of the domain lies below.

The identification of the edges is done using the row structure of the polygonal region, as shown in figure A.4. The vertical edges are identified using the j rows. In each j row, the vertical edges are scanned from left to right. The first edge in the row is identified as Left (L), and the subsequent edges are tagged Right (R) or Left (L) in an alternating fashion. (Note that a vertical edge may span more than one j row, in which case it is repeatedly tagged with the same label.)

The horizontal edges are identified using the i rows. In each i row, the horizontal edges are scanned from bottom to top. The first edge in the row is identified as Bottom (B), and the subsequent edges are tagged Top (T) or Bottom (B) in an alternating fashion. (As before, an horizontal edge may span more than one i row, in which case it is repeatedly tagged with the same label.)

When all the edges have been identified in this manner, the vertices are then tagged as either concave or convex. The vertices of interest in this decomposition process are the concave vertices, since all internal decomposition lines must emanate from a concave vertex.

The vertex type is defined by the intersection of a vertical and a horizontal edge. There are four distinct intersections defining a concave vertex. These are the following:

1. The top vertex of a Left edge is coincident with the right vertex of a Bottom edge;

2. The top vertex of a Right edge is coincident with the left vertex of a Bottom edge;

3. The bottom vertex of a Left edge is coincident with the right vertex of a Top edge;

4. The bottom vertex of a Right edge is coincident with the left vertex of a Top edge.

The concave vertices detected in this manner have been encircled in figure A.4. The remaining vertices are convex.

A.1.3 Connecting Couples of Aligned Edges

As stated earlier, in order to minimize the number of decomposition rectangles, the maximum number of couples of aligned edges must be connected before decomposing the rest of the domain.

Two edges of the contour are said to be *aligned* if they lie on the same straight line, and can be connected by a internal line. The line connecting a couple of aligned edges must emanate from a concave vertex of one of the two edges and terminate on a concave vertex of the other edge. In addition, the connecting line cannot cross any part of the domain boundary.

An internal line connecting a couple of aligned edges is called a *heavy* line. In order to minimize the number of decomposition rectangles, a maximum number of heavy lines must be created. However, the heavy lines cannot touch or intersect one another.

The first step in the creation of the heavy lines is to connect all couples of aligned

edges, as shown in figure A.5. Note that all heavy lines connect concave vertices only, and do not cross any part of the domain boundary.

The second step is to delete a minimum number of heavy lines such that the remaining heavy lines do not intersect one another. This is done as follows:

1. For every heavy line, record the number of intersections it creates with other heavy lines;
2. Identify the heavy line with the greatest number of intersections and delete it. If two or more heavy lines have the same number of intersections, delete only one of them (chosen at random);
3. Repeat steps 1 and 2 until there are no intersections left.

The above process was applied to the set of heavy lines shown in figure A.5, and a reduced set of heavy lines was obtained, shown in figure A.6. Note that the set of heavy lines created in this manner is generally non-unique for a given polygonal figure.

A.1.4 Decomposition into Disjoint Rectangles

Once the final set of heavy lines has been created, all concave vertices that have been connected by a heavy line are marked. If all the concave vertices of the polygon are marked in this way, the decomposition process is complete. However, if there are concave vertices that remain unmarked, additional internal lines must be created to complete the decomposition.

For each unmarked concave vertex, one additional internal line must be created. This internal line must emanate from the concave vertex either vertically or horizontally[1],

[1]The direction can be chosen randomly.

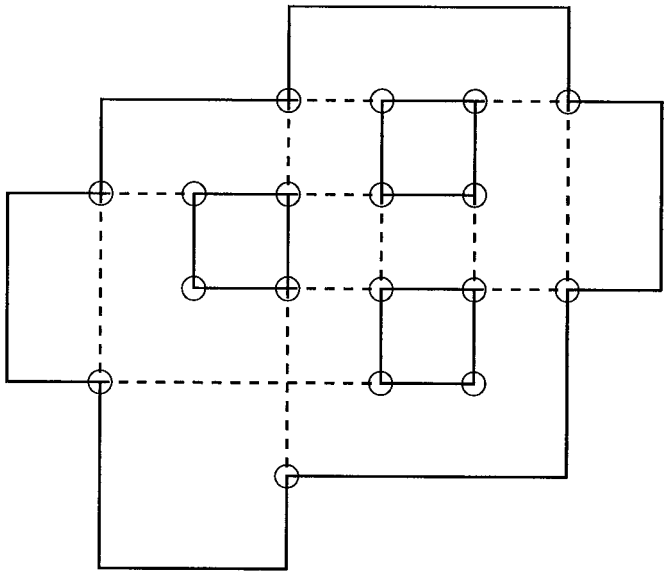

Figure A.5: Internal lines connecting all couples of aligned edges.

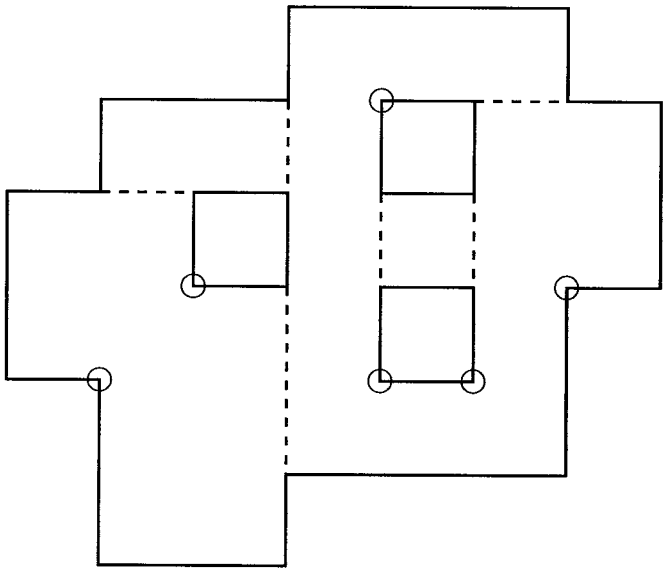

Figure A.6: Non-incident internal lines connecting couples of aligned edges.

and terminate on an adjacent part of the contour or on another existing internal line, whichever is nearest. The additional internal lines so created are then added to the set of heavy lines.

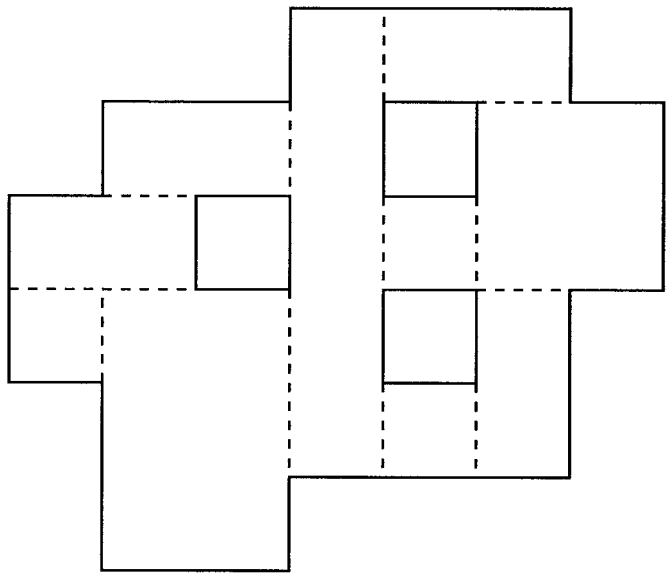

Figure A.7: Topological space decomposed into a set of disjoint rectangles using Lipski's algorithm.

The resulting set of internal lines delineates the decomposition of the polygon into the minimum number of rectangles, as shown in figure A.7. Note again that there may be several distinct decompositions that yield the minimum number of rectangles.

The decomposition rectangles shown in figure A.7 are used to define a multi-block structure for the mesh, and to generate all the relevant connectivity information automatically.

In addition, the internal lines generated as part of the decomposition process are projected into the physical space as straight lines to initialize the physical mesh. Once these lines are drawn, the mesh inside each rectangle is initialized using transfinite interpolation.

This initial mesh is then optimized globally using a system of overlapping rectangles called the prime rectangles. In this global optimization, the internal lines of the decomposition are left floating and their final form will depend only on the shape of the domain boundary and the point distributions thereon.

The algorithm used to define the set of prime rectangles is described below.

A.2 Definition of Prime Rectangles

A prime rectangle of a polygonal region is defined as a rectangle of maximal size for a given area of the region. Figure A.8 provides examples of prime rectangles. Thus, by definition, a part of each of the four sides of a prime rectangle must touch a domain boundary (inner or outer). If this were not the case, then the rectangle could be expanded on one or more sides and hence would not be of maximal size.

As stated earlier, some or all of the decomposition rectangles (obtained using Lipski's algorithm) may also be prime rectangles. Hence, the first step in the process of defining the set of prime rectangles is to identify the subset of decomposition rectangles that are also prime rectangles. This is done by discarding the decomposition rectangles for which at least one side does not touch a domain boundary, i.e. is defined entirely by an internal line.

Since the set of decomposition rectangles covers the entire domain, any additional rectangle must either cross an internal line or be smaller than, and fully contained in, one of the existing decomposition rectangles. However, the smaller rectangles are not prime rectangles since by definition, a prime rectangle is of maximal size in a given area. Thus, all prime rectangles distinct from the decomposition rectangles must cross an internal line.

Thus, in order to find the remaining prime rectangles, the following procedure (derived

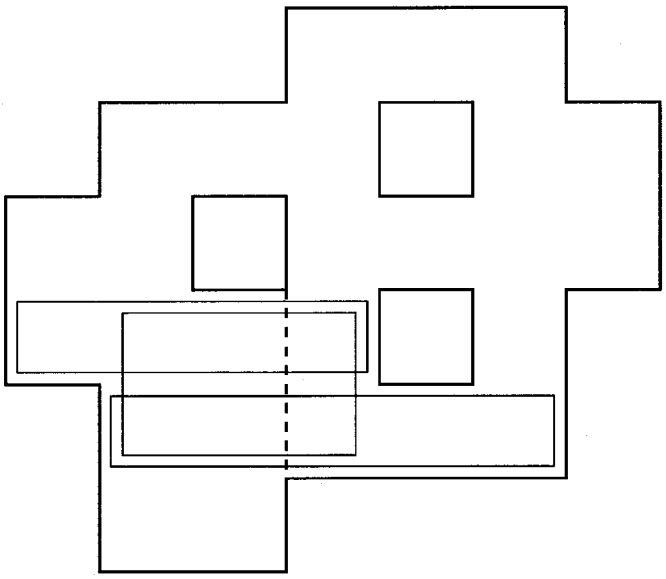

Figure A.8: Definition of prime rectangles crossing a vertical internal line.

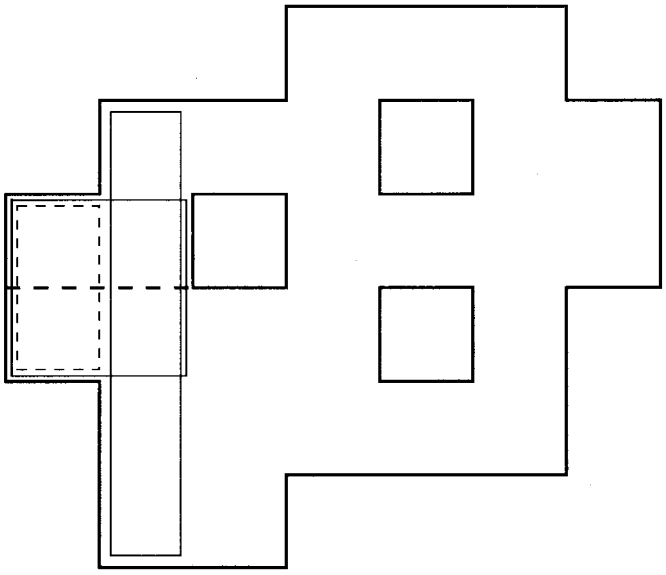

Figure A.9: Definition of candidate prime rectangles crossing an horizontal internal line.

from LODI et al. (1979)[2], algorithm A3, page 222) is used:

1. For each *vertical* internal line of the decomposed polygonal region, create all the rectangles of distinct heights and vertical position that cross the internal line *and* whose left and right-hand sides touch a part of the domain boundary (see figure A.8);

2. For each *horizontal* internal line of the decomposed polygonal region, create all the rectangles of distinct widths and horizontal position that cross the internal line *and* whose bottom and top sides touch a part of the domain boundary (see figure A.9);

3. Assemble the previously obtained decomposition rectangles and the newly formed rectangles into one expanded set of candidate prime rectangles;

4. Discard any candidate rectangle that is wholly contained in another rectangle.

The rectangle shown in dashed lines in figure A.9 is an example of a candidate rectangle that is wholly contained in another rectangle, and is thus not part of the set of prime rectangles.

The complete set of over-lapping prime rectangles is shown in figure A.10. Unlike the set of disjoint decomposition rectangles, the complete set of prime rectangles is unique for a given rectilinear polygonal region.

For a discussion of the theorems and proofs supporting the above decomposition algorithms, the reader is referred to LIPSKI et al. (1979) and LODI et al. (1979).

[2]Once the set of disjoint decomposition rectangles has been defined, only a simplified version of Lodi's algorithm is needed to find the remaining prime rectangles.

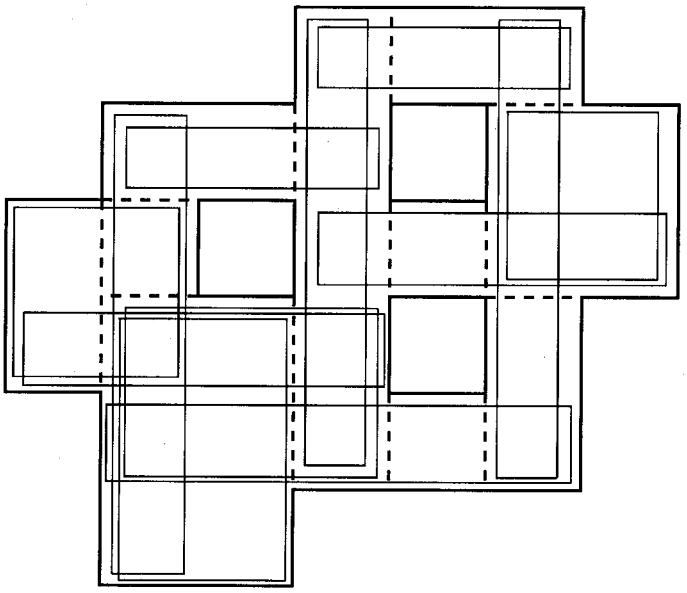

Figure A.10: Complete set of prime rectangles.