TOWARDS A COMPREHENSIVE COMPUTATIONAL SIMULATION SYSTEM FOR TURBOMACHINERY

By

Ming–Hsin Shih

A Dissertation
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy
in the Department of Aerospace Engineering

Mississippi State, Mississippi

May 1994

TOWARDS A COMPREHENSIVE COMPUTATIONAL SIMULATION SYSTEM FOR TURBOMACHINERY

By

Ming–Hsin Shih

Approved:

[illegible]
Bharat K. Soni
Associate Professor of Aerospace Engineering
(Director of Dissertation)

C. Wayne Mastin
C. Wayne Mastin
Professor of Mathematics

[illegible]
Joe F. Thompson
Professor of Aerospace Engineering

[illegible]
Z.U.A. Warsi
Professor of Aerospace Engineering

[illegible]
David L. Whitfield
Professor of Aerospace Engineering

[illegible]
Kenneth R. Hall
Professor of Aerospace Engineering And Graduate Coordinator of the Department of Aerospace Engineering

[illegible]
Robert Altenkirch
Dean of the College of Engineering

Richard D Koshel
Richard D. Koshel
Dean of the Graduate School

Name: Ming–Hsin Shih

Date of Degree: May 14, 1994

Institution: Mississippi State University

Major Field: Aerospace Engineering

Major Professor: Dr. Bharat K. Soni

Title of Study: TOWARDS A COMPREHENSIVE COMPUTATIONAL SIMULATION SYSTEM FOR TURBOMACHINERY

Pages in Study: 177

Candidate for Degree of Doctor of Philosophy

The objective of this work is to develop algorithms associated with a comprehensive computational simulation system for turbomachinery flow fields. This development is accomplished in a modular fashion. These modules includes grid generation, visualization, network, simulation, toolbox, and flow modules. An interactive grid generation module is customized to facilitate the grid generation process associated with complicated turbomachinery configurations. With its user–friendly graphical user interface, the user may interactively manipulate the default settings to obtain a quality grid within a fraction of time that is usually required for building a grid about the same geometry with a general–purpose grid generation code. Non–Uniform Rational B–Spline formulations are utilized in the algorithm to maintain geometry fidelity while redistributing grid points on the solid surfaces. Bezier curve formulation is used to allow interactive construction of inner boundaries. It is also utilized to allow interactive point distribution. Cascade surfaces are transformed from three–dimensional surfaces of revolution into two–dimensional parametric planes for easy manipulation. Such a transformation allows these manipulated plane grids to be mapped to surfaces of revolution by any generatrix definition. A sophisticated visualization module is developed to allow visualization for both grid and flow solution, steady or unsteady. A network module is built to allow data transferring in the heterogeneous environment. A flow module is integrated into this system, using an existing turbomachinery flow code. A simulation module is developed to combine the network, flow, and visualization module to achieve near real–time flow simulation about turbomachinery geometries. A toolbox module is developed to support the overall task. A batch version of the grid generation module is developed to allow portability and has been extended to allow dynamic grid generation for pitch changing turbomachinery configurations. Various applications with different characteristics are presented to demonstrate the success of this system.

DEDICATION

In memory of

My dear Grandmother (1910–1990)

To my parents, brother, sister, and

To my wife, Mei–Lien

ACKNOWLEDGMENTS

There is no way that I can say enough "Thank Yous" to my major advisor, Dr. Bharat K. Soni, without whose guidance, advice, trust and endless support, I could never come this far. His optimistic attitude always inspired me and kept me from running out of steam. Still, I want to say, thank you, Dr. Soni. Additional thanks goes to his family for embracing my wife and me as their family members.

I would like to thank all my committee members: Dr. C. Wayne Mastin, Dr. Joe F. Thompson, Dr. Z.U.A. Warsi, and Dr. David L. Whitfield for their instruction and advice.

Special thanks goes to Dr. Dennis L. Huff of NASA Lewis Research Center, my technical monitor, for his enormous faith on TIGER and all the help, support and advice he provided over the years; Dr. Osamu Yamamoto, his tremendous patience on testing TIGER to help me debug the code, and provides me valuable advice and suggestions. He is also the one who built the grid for the turbofan; Mr. Timothy A. Beach, for sharing his precious experiences on turbomachinery grid generation; Dr. George P. Koomullil for his friendship and technical advice.

Thanks goes to Professor Charles Cliette, former department head of Aerospace Engineering, for his kind help; Dr. Keith K. Koenig, for his instruction and help; Dr. J. Mark Janus, for his advice and discussion on establishing

TIGER's compatibility with his TURBO code during the days, and his vicious dog–fight during the nights; Dr. Lafe K. Taylor, for enhancing TIGER's capability by bombarding TIGER's grids with his incompressible code; and to Dr. JenPing Chen and his wife, Arjene, for his advice on MSUTC code, and their wonderful friendship.

Thanks also goes to all the men and women who helped me on English writting. I also appreciate Dr. Jiann–Cherng Yang, for his endless discussions; Mr. Tzu–Yi Yu, for his help on NURBS and debugging; Dr. Hugh J. Thornburg, for his technical advice; Mr. Sujay Shaunak, for his help on programming and development of the network module; Mrs. Adenike Adebiyi, for her help on testing TIGER.

Finally, I would like to thank my grandmother in heaven for her endless love to her beloved grandson, "me", who did not see the completion of this work; to my parents, for their love, trust and support; to my brother and sister, for their love and support; and, to my beloved wife, Mei–Lien, without whose support and love, I could not see the end of this work.

Ming–Hsin Shih

Starkville, Mississippi

December, 1993

TABLE OF CONTENTS

CHAPTER Page

Page

LIST OF TABLES

LIST OF FIGURES

Figure | Page

Figure Page

NOMENCLATURE

ds	minimum spacing
G_r	geometry progression ratio
G_p	percentage for geometry progression distribution
g^{ij}	elements of the contravariant metric tensor
g_{ij}	elements of the convariant metric tensor
J	advance ratio
k, k_u, k_v	NURBS evaluation oerders
m', σ	parametric coordinates for cascade surface mapping
N_b	number of blades
N_ξ, N_η, N_ζ	number of points in ξ, η, and ζ directions
$\mathcal{N}_\xi, \mathcal{N}_\eta, \mathcal{N}_\zeta$	number of grid cells in ξ, η, andζ directions
$\mathcal{P}_i$	control functions
u	parametric coordinate
w_i, w_{ij}	weights
$\boldsymbol{x}, \boldsymbol{r}, \theta$	cylindrical coordinates
β	local blade angle
$\beta_{3/4}$	blade stagger angle
ξ, η, ζ	curvilinear coordinates

π	pi
Ω	ripple diffusion factor

Subscripts

ξ^i, ξ^j, ξ^k	partial differentiation

Superscripts

c	camber line
i, j, k	index
p	pressure side
s	suction side
T	transpose

Symbols

$\bullet$	dot product
$\| \ \|$	absolute value

CHAPTER I
INTRODUCTION

Fluid mechanics processes can be mathematically described by a set of nonlinear partial differential equations, known as the Navier–Stokes equations[1–3]. Unfortunately, this set of nonlinear equations is so complex that analytical solutions are not possible at the present time. Many approximate methods have been developed to solve this nonlinear system, such as expansion and perturbation methods, collocation and integral methods, and discrete methods like finite difference, finite volume and finite element methods[4–7]. Finite difference, finite volume, and finite element methods have all achieved substantial success in approximating these equation. Computational fluid dynamics (CFD) refers to the technologies centering around these numerical methods. CFD is among those disciplines which require powerful and expensive computer resources. These resources are needed for achieving detail and accurate numerical results that are useful for design process. It is therefore important to increase the efficiency of such effort for large applications.

A typical CFD application may be divided into three steps: grid generation (pre–processing), flow calculation (processing), and solution visualization (post–processing). These steps, however, are usually performed independently. This implies that there are iterative procedures between each steps. The purpose of this work is to develop algorithms associated with a comprehensive system that allows a user to perform a complete CFD application cycle associated with turbomachinery flow fields in a user–friendly environment.

There are several attempts to address such cross–displinary integration effort. Stokes and Huddleston[32] integrate an existing general purpose grid code EAGLEView and the PARC flow code[33] as well as the UBIFlow flow code[24] to form a two–dimensional numerical flow simulation system. Laurien et. al.[35] integrated graphics workstations and supercomputers to achieve interactive numerical flow simulation and grid adaption for hypersonic flow. The main motivation for the current work is to optimize the execution of these three steps for CFD applications of turbomachinery in view of efficiency and reduction of labor time.

However, despite that rapid increases in available computational power and algorithms have facilitated the analysis of very complex flow fields using CFD calculations, grid generation is often a limiting item in these applications. Traditional grid generation procedures have proven to be very time consuming for complex geometries. Advances in graphics workstations have provided the hardware capability for the execution of sophisticated, user–friendly and interactive grid generation software. This development has dramatically reduced the man–hours involved during the grid generation process for many engineering applications. GENIE^{++}[8–11], EAGLEView[11–13], NGP[14], ICEM[15], GRAPEVINE[16–19], and GRIDGEN[20–21] are examples of such developments. However, for applications involving complex turbomachinery flow fields, the grid generation process is extremely time consuming. Consequently, customized grid generation codes for turbomachinery configurations have been developed to address this problem. Some of the application–oriented codes are IGB[22,23], TIGG3D[24], and TIGER[25–29].

The IGB grid code[22,23] allows construction of three–dimensional grids for turbomachinery configurations by stacking a series of gridless surfaces between blade root and tip, which has been a very popular approach for many turbomachinery grid generation codes. It consists of three main modules which are executed independently, along with 4 more utility modules to assist the construction of the grids. The first module allows generation of the bounding surface grid with the input files which contain the geometry data and grid generation control parameters. The second module facilitates interactive manipulation of a series of two–dimensional cascade (blade–to–blade) surfaces and the complete three–dimensional grid is obtained by spline fitting. By setting the Bezier curves at different axial positions, the second module constructs the cascade surfaces patch by patch, which allows the user to design better grids. Overall, this code is fairly user–friendly, despite the fact that it requires multiple grid generation steps, and it requires the user to memorize a vast number of function keys, which can be a challenge for new users. It generates quality grids with H–type topology. Unfortunately, this code is constrained by its single block algorithm.

Unlike the IGB code, TIGG3D[24] is a multi–block grid generation code for turbomachinery configurations with multiple blade rows and multiple ducts. TIGG3D uses layers of pop–up menus for user dialogues required in the construction of grids. However, in this approach the layout of the menus is not clear. Its algorithm, which generates axisymmetric grids, has several severe drawbacks. It requires the blade surfaces to be defined along the same axial location for both suction side and pressure side. In other words, the first and the last axial points define the leading edge and the trailing edge on the grid. In general, they are not necessarily the true leading edge and trailing edge of the blade. Such restriction may offset the blade leading edge and trailing edge significantly, especially when the leading edge has a large radius, and the blades are set at low stagger angles. Because of the same algorithm restriction, its applications are limited to axial flow configurations only. TIGG3D does not have quality checking capability, and its graphical visualization capability is limited. Overall, TIGG3D is able to generate grids with sophisticated topology, with the aforestated limitations.

The previous version of the current work, TIGER[25–29] (Turbomachinery Interactive Grid genERation) was designed for use with the Mississippi State University TURBO turbomachinery flow code[30,31] and was intended to validate the concept of an integrated turbomachinery simulation system. Developed as a customized module in GENIE, TIGER utilized modified subroutines from GENIE and developed an algorithm that allows the user to generate single/two–block grids for axial propfan configurations, with the capability of handling an uneven blade count. This code requires very little user input from the screen through its home–brewed graphical user interface (GUI) using Graphics Library (GL). With its built–in automation algorithm, this code facilitates most of the grid based on minimum user–specified parameters with little user control over the resulting grid.

This simulation system is accomplished in modular fashion. These modules include grid generation, visualization, network, flow solver, simulation, and toolbox. The grid generation module is customized for general turbomachinery configurations, allowing grid construction in a timely fashion with desirable quality. The Visualization module developed for this system allows sophisticated visualization, static or dynamic. Network communication capability is developed for data transferring among different machines. A toolbox module is developed to support the overall applications. A flow module is established using an existing turbomachinery flow code.

The general discussion on turbomachinery design, possible configurations, major components and complexity challenges in grid generation is presented in Chapter II. The technical approaches to attack the aforestated challenges and the processes to improve grid quality are addressed in Chapter III. The development of the system and its algorithm are described in Chapter IV. The dynamic grid generation algorithm is discussed in Chapter V. The applications and results are presented in Chapter VI. And finally, in Chapter VII the research is concluded and future work addressed.

CHAPTER II

TURBOMACHINERY DESIGN AND BASIC TERMINOLOGY

The purpose of this chapter is to introduce the terminology, basic principles of design, and configurations associated with a typical turbomachinery system. This chapter also addresses the challenges in generating a grid for a turbomachinery flow domain.

The term turbomachinery is generally accepted as implying a class of rotating machines producing pressure or power, whose primary components are rotative. Aircraft propellers and compressors are examples of this type of machinery. Within the content of this work, the term *turbomachinery* is broadly extended to all the configurations that have the axisymmetry property. In this study, grid generation has been exercised on various configurations, including external, internal, and external–internal flow field configurations.

Coordinate Systems

Due to the axisymmetric characteristics of a turbomachine, it is a natural choice to use cylindrical coordinates instead of Cartesian coordinates for the grid generation algorithm. For a point in Cartesian coordinates $\boldsymbol{p(x,y,z)}$, it can be expressed as $\boldsymbol{p(r,\theta,z)}$, where $\boldsymbol{x}=\boldsymbol{r}\sin\theta$ and $\boldsymbol{y}=\boldsymbol{r}\cos\theta$. However, in this work, the cylindrical coordinates system is defined as $\boldsymbol{p(x,r,\theta)}$, where $\boldsymbol{y}=\boldsymbol{r}\sin\theta$ and $\boldsymbol{z}=\boldsymbol{r}\cos\theta$.

One advantage of using cylindrical coordinates is that a rotation about the rotation axis becomes a translation in the third coordinate, θ, which is ap–

parently less expensive in terms of computation time. Another advantage is that when doing transfinite interpolation (TFI) for a surface lying on a cylinder, this surface of revolution becomes a simple plane, with its second coordinate, r, being constant. The resulting surface is a perfect cylinder surface, no matter how the four boundaries are defined. If similar tasks are performed in Cartesian coordinates, the resulting surface will not always be a perfect cylinder surface.

Terminology

In this section, several terms[36,37] frequently used in turbomachinery design are to be defined and/or unified to avoid confusion. If there is more than one term representing the same definition, the one underlined is used for this work.

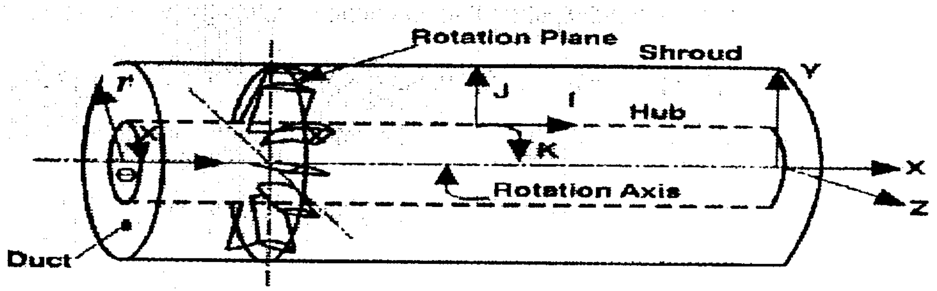


Figure 2.1 Schematic Presentation of a turbomachine

Rotation Axis: [Fig. 2.1]

The axis about which the machine is rotating. In this work, this axis is coincident with the X axis.

Rotation Plane: [Fig. 2.1]

The plane that is perpendicular to the rotation axis. Note that this plane is not unique.

Hub (Spinner): [Fig. 2.1]

The center portion of a turbomachinery where the roots of the blades are connected.

Wheel:

A wheel is referred to be the hub of an impeller.

Shroud (Casing): [Fig. 2.1]

An outer cover to a turbomachine.

Duct: [Fig. 2.1]

A duct is a tubular passage through which a fluid is conveyed.

Duct Lip:

The front end of the duct.

Suction Side (Camber Surface, Trailing Surface): [Fig. 2.2]

The suction side of a blade generates low pressure by having longer cambered length for an asymmetric airfoil, which induces high flow velocity.

Pressure Side (Face Surface, Leading Surface): [Fig. 2.2]

It is the opposite side of the suction surface. For an asymmetric air-foil, this is the side with high pressure (or low flow velocity).

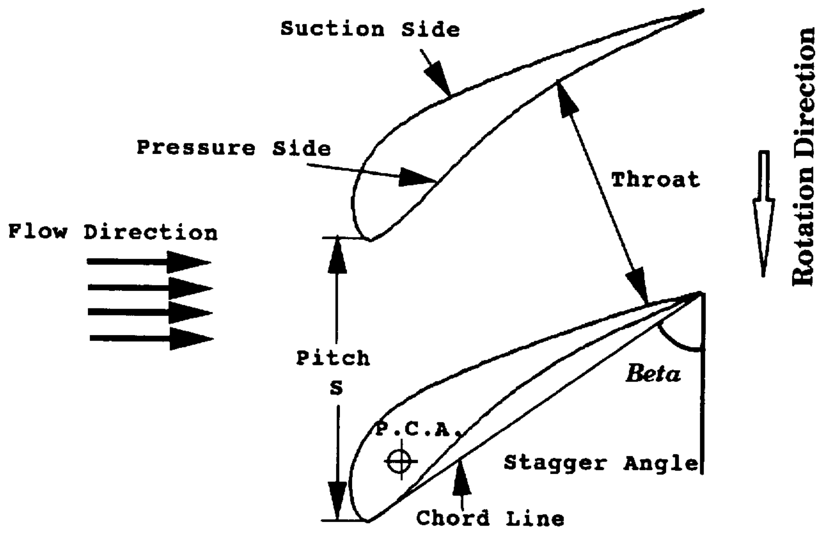


Figure 2.2 A Blade Passage

Chord Length: [Fig. 2.2]

It is considered as the distance between the blade leading edge to the trailing edge of an airfoil.

Stagger Angle (Pitch Angle, setting Angle): [Fig. 2.2]

The angle between the blade chord line with the rotation plane. For a three–dimensional blade, the stagger angle of the blade ($\beta_{3/4}$) is referred to the blade angle at 3/4 of the blade diameter.

Pitch Change Axis (Stack–Up Axis): [Fig. 2.2]

The axis along which blade profiles are stacked. When changing the blade stagger angle, this is the reference axis about which the blade are rotated.

Pitch (Spacing): [Fig. 2.2]

$S=r \int \theta \, d\theta$, i.e. circumferential distance between blades.

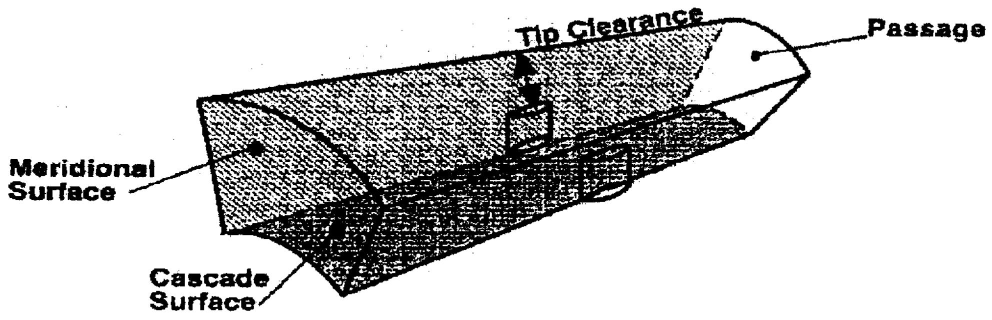


Figure 2.3 A Schematic Presentation of a Flow Passage

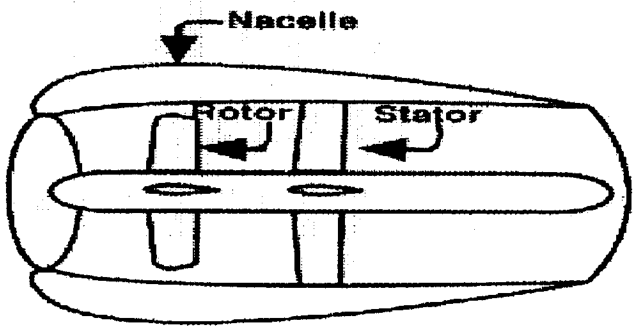


Figure 2.4 A Schematic Presentation of an Engine

<u>Meridional Surface</u>: [Fig. 2.3]

A surface in the X–R direction.

<u>Cascade Surface</u> (Blade–to–Blade Surface): [Fig. 2.3]

In this work, this term is referred to the tangential surface between the blades of the same blade row.

<u>Tip Clearance</u>: [Fig. 2.3]

The distance between the shroud and the rotor tip.

Passage (Channel): [Fig. 2.3]

A passage is a periodic volume from the inlet of the domain to the outlet of the domain, confined between two neighboring blades.

Nacelle: [Fig. 2.4]

A nacelle is a streamlined enclosure for housing the engine in an aircraft.

Possible Configurations

Turbomachinery configurations can be roughly divided into three categories, viz., radial–flow, axial–flow, and mixed–flow configurations, as shown in Figure 2.5. Examples for the first category are mainly the impellers, which

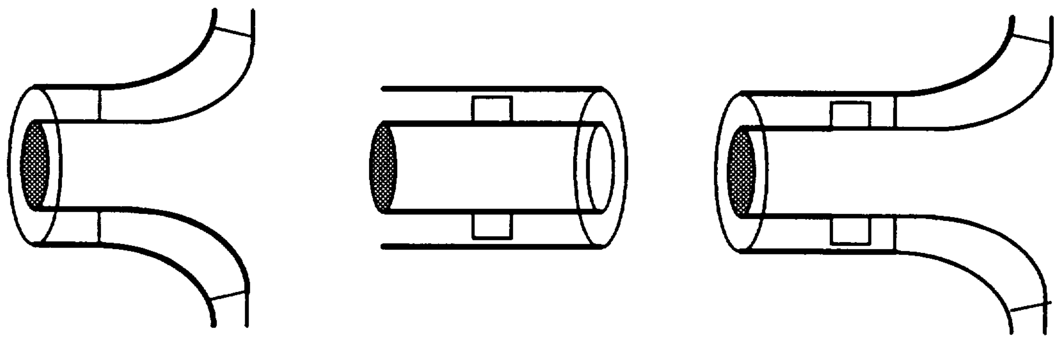

(a) Radial Flow (b) Axial Flow (c) Mixed–Flow

Figure 2.5 Possible Configurations

carry the flow either from the axial direction to the radial direction or vice versa. Examples for the second category are propellers for aircrafts or ships, and the propulsors of a torpedo. A mixed–flow configuration is the combination of both the radial–flow and axial–flow configurations.

Radial Flow Configuration

A centrifugal compressor has the advantages of light weight and a minimum number of parts. It is capable of producing a large pressure ratio for a single stage of compression, and is easily manufactured. However, its shortcomings are the relatively small flow capacity for a given frontal area and comparatively low efficiency presently obtainable in a high compression ratio stage. A centrifugal compressor is composed of three components: the inducer, the impeller, and the diffuser.

An inducer is the portion of the wheel or impeller near the entrance which serves to produce a solid body rotation of the fluid which is necessary to match the flow in the impeller. It may sometimes be built as part of the impeller and sometimes made as a separated part fastened to the wheel. When generating the grid for an inducer alone, the flow field may be considered as an axial flow field. The challenge for grid generation is that the function curve that defines the wheel or shroud may not be a single–value function, and therefore, the curve and surface spline routines must be general enough to handle the multiple–value functions.

Axial Flow Configuration

The design of propellers for marine vehicles usually focuses on the cavitation problem[38]. Cavitation can occur in many shapes, like bubble cavitation, sheet cavitation, and vortex cavitation. Cavitation, however, has generally a multitude of appearances. One of them is cloud cavitation, which, for example, occurs in a free shear layer. The cavitation problem influences propeller design significantly, and thus imposes different challenges for generating grids. A propeller designed to avoid sheet cavitation has thick, cambered profiles. This type of propeller usually does not create much of a problem for grid generation. However, a propeller designed to avoid tip vortex cavitation has a strongly reduced pitch at the tip to avoid tip vortex cavitation, which may cause a severe problem, like skewness of the cascade surface grid on the tip.

Cavitation problem does not exist in a compressible turbomachinery flow fields, such as the compressors and turbines in the axial jet engines. However, when the blade stagger angle is set to be very low, grid skewness is likely to be severe.

The basic principle of operation is the same for both the axial compressor and the centrifugal compressor, that is, kinetic energy is released into the air by the rotation of the blades, resulting in the conversion of kinetic energy to pressure rise[36,37]. For a typical axial engine compressor, the air enters axially into the inlet guide–vanes where it is turned through a certain angle to impinge on the first row of the stator. Each stage consists of two rows of blades, viz., rotor and stator, where the rotor blades increase the kinetic energy of the air flow and discharge it into the following row of stator blades to increase the pressure by diffusing. Most compressors have one to three rows of "straightener" or "diffuser" blades installed after the last stage to straighten and slow down the air prior to its entry into the combustion chambers. Sometimes these blades are designed to provide additional turbulence to the air flow for better air–fuel mixture in the combustion chamber, these diffuser blades are then called "mixer" blades. The advantage of the axial flow compressor is the high flow capacity for given diameter and the relatively high efficiency. However, present axial compressors have low pressure ratios produced by a single stage, this requiring the use of many stages and many blades to obtain the overall pressure ratio. For most of the compressor designs, the average pressure ratio per stage is only less than 1.20. It is then clear that a large number of stages is needed to achieve the required pressure rise. For example, if a total compression ratio of 3.7 is required, and the averaged pressure ratio per stage is only 1.14 for a certain compressor, then 10 stages of compression are needed to reach the required compression ratio. Typically, hundreds of blades are needed for such compressor. Therefore, the challenge in calculating the flow about this type of configuration becomes difficult and expensive if a large number of passages is needed to obtain the periodicity for time–accurate calculations.

Major Components of a Turbomachinery

There are several major components of a turbomachinery, they are: (*i*) the hub, (*ii*) the shroud/nacelle, and (*iii*) the blades. Both the hub and the shroud/nacelle surfaces are assumed to be surfaces of revolution. For most of the applications, this assumption is true. However, for applications where this assumption is no longer true for the entire geometry, but only for the rotative portion, then, only the rotative portion of the geometry is considered in this work. With this assumption, the geometry definitions for the hub and shroud/nacelle are simplified to be the definitions of two–dimensional meridional curves, which are functions of the form $r=f(\mathbf{x})$.

The blade geometry is the most important component of the entire turbomachinery design. In this work, a fin for a missile is also considered to be a "blade", as long as the fin maintains periodicity. A blade profile may be defined in various formats, including the IGES[39] format. In order to identify the data points on the blade surface, a blade may be defined with an array of (x_{ij}, r_j, θ^c_{ij}, $\Delta\theta_{ij}$), where i is the index for the axial direction, and j is the index for the radial direction, x is the axial coordinate, r is the radial coordinate, and θ^c

is the theta coordinate for the camber line, and $\Delta\theta$ is the offset θ for both the suction side and the pressure side. A blade profile may also be defined by ($\boldsymbol{x}_{ij}$, $\boldsymbol{r}_j$, θ^p_{ij}, θ^s_{ij}), where θ^p and θ^s are θ coordinates for pressure side and suction side of the blade, respectively. Another way is to define the blade by ($\boldsymbol{x}_{ij}$, $\boldsymbol{r}_{ij}$, θ^p_{ij}, θ^s_{ij}), which means that $\boldsymbol{r}$ may no longer be constant for each profile; or by ($\boldsymbol{x}^p_{ij}$, $\boldsymbol{r}^p_{ij}$, θ^p_{ij}, $\boldsymbol{x}^s_{ij}$, $\boldsymbol{r}^s_{ij}$, θ^s_{ij}). A blade in Cartesian coordinates may be defined as ($\boldsymbol{x}^p_{ij}$, $\boldsymbol{y}^p_{ij}$, $\boldsymbol{z}^p_{ij}$, $\boldsymbol{x}^s_{ij}$, $\boldsymbol{y}^s_{ij}$, $\boldsymbol{z}^s_{ij}$). In order to maintain flexibility, blade definition of the form ($\boldsymbol{x}^p_{ij}$, $\boldsymbol{r}^p_{ij}$, θ^p_{ij}, $\boldsymbol{x}^s_{ij}$, $\boldsymbol{r}^s_{ij}$, θ^s_{ij}) is used in this work. A built–in tool developed in this research allows the user to convert blade data defined with either of the associated formats into standard TIGER format.

Local Definitions of Blade Types

Based on its functionality, a blade may assume different names, such as rotor, stator, inducer, impeller, or diffuser. However, to simplify the grid generation process and to allow the algorithm to identify the blade type by its characteristics, this work categorizes blades as follows: (*i*) propfan, (*ii*) rotor, and (*iii*) stator.

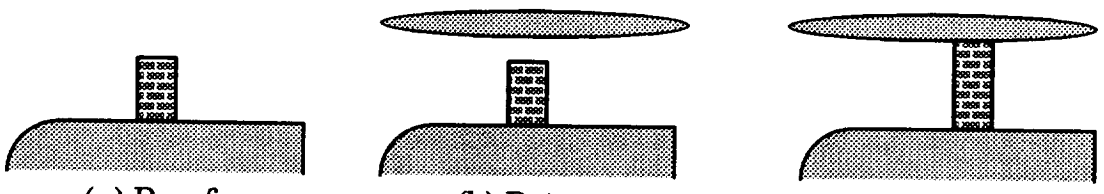

(a) Propfan (b) Rotor (c) Stator

Figure 2.6 Local Definitions for Blade Types

A propfan [Fig. 2.6(a)] in this work is considered to be blades for the external flow fields. Only the propfan's root intersects with the hub. Its tip is not confined by any geometry; therefore, radial grid lines (η–lines) can extend from the tip to the outer domain boundary. Propellers, fins of a missile, and unducted propfans are examples of this category. A rotor [Fig. 2.6(b)], however, is considered to be the type of blade whose root intersects with the hub, and its tip is not connected to the shroud or nacelle. Instead, there exists tip clearance. An impeller blade or a ducted propfan with tip clearance is considered to be in this category. Similar to a rotor, a stator [Fig. 2.6(c)] is the type of blade whose root and tip are both connected with the geometry surfaces, leaving no gap on both ends. With this local categorization, a rotor in a compressor stage becomes a stator if no tip clearance is considered.

With the recent development in CFD algorithms, flow calculations for turbomachinery geometries with tip clearances have been expanded to aid in the understanding of the detailed flow pattern near this region. The losses that arise from leakage through the clearances at the tips of unshrouded blades in multistage axial compressors are frequently quite large in comparison with other losses resulting from viscous effects. However, it is physically difficult to avoid large tip clearance in high speed compressors due to the fact that the rotor and the stator have different thermal expansion. For a contemporary ultra–high bypass ratio turbofan, the rotor tip is actually designed to be spherical so that its stagger angle may be changed freely without hitting the nacelle. The tip clearance problem is even more severe for this case. The losses resulting from the tip clearance flow are difficult to identify in a perfect fluid model. The roll–up vortex sheet is the principal mechanism for dissipation of kinetic energy of the jet through the clearance[36,37]. Prediction of the induced loss due to tip clearance for rotors, and more recently, the ducted prop–fans, is an important issue for today's CFD simulation for turbomachinery. The tip clearance gap δ is usually less than

1% of the blade diameter, it is a difficult region for structured grid generation. The grid lines in the gap must follow the overall grid trend, yet allow user control for the point distributions.

The blade stagger angle $\beta_{3/4}$ is another important factor that might pose a challenge to grid generation. When the blade is set to be the fairing position, i.e. $\beta_{3/4} = 90$ degree, the passage is most wide open. This position allows the structured grid to be generated in a better fashion. On the contrary, if $\beta_{3/4}$ decreases toward 0 degrees, the passage becomes narrow, and the grid line becomes skew.

CHAPTER III
GRID GENERATION APPROACHES

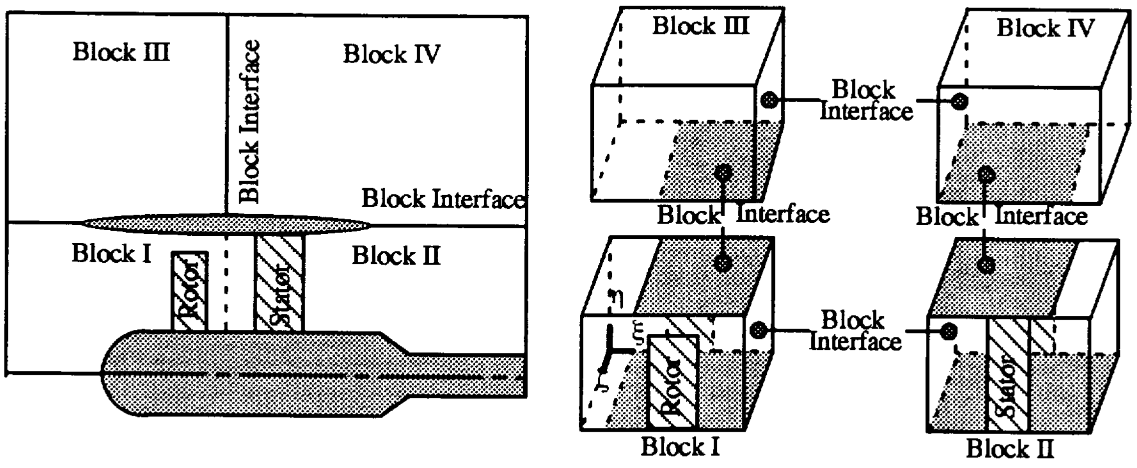
The numerical grid generation about a general turbomachinery configuration is a tedious and time consuming task. In order to facilitate the generation process and reduce man–hours required to produce a turbomachinery grid, customized procedures and methodologies are needed. These procedures must offer automization and user–friendly graphical interaction associated with labor–intensive operations. For a typical turbomachinery configuration, domain decomposition, the definition of solid components, and surface generation with proper distribution are extremely time–consuming. The methodologies and procedures associated with these operations have been optimized by developing customized algorithms and user–oriented graphical interface. These procedures and methodologies in five categories: (*i*) computational modeling, (*ii*) geometry modeling, (*iii*) user interactions, (*iv*) grid generation, and (*v*) quality check are discussed in this chapter. In computational modeling, an automatic block–construction scheme has been developed. In geometry modeling, Non–uniform Rational B–Spline (NURBS) formulations[40] are utilized for surface and curve grid point re–distribution[41]. Blade surface construction as well as tip clearance modeling are automized to reduce the labor involvement. Blade leading edge and trailing edge circle modeling algorithm for both two–dimensional and three–dimensional blade has been developed. Cascade surfaces are transformed into two–dimensional space, allowing easier grid construction and manipulation. Several new stretching functions are devel–

oped in the user–interaction steps. In grid generation process, weighted transfinite interpolation (WTFI)[10] approach is adopted, and an automatic sub–blocking algorithm is developed to enhance the robustness of algebraic grid generation system without increasing the required labor. Combination of elliptic smoothing and NURBS spline–fitting algorithm is developed to enhance the grid quality near the difficult regions on the cascade surfaces. These customizations offer time–efficient and user–friendly construction of grids.

Computational Modeling

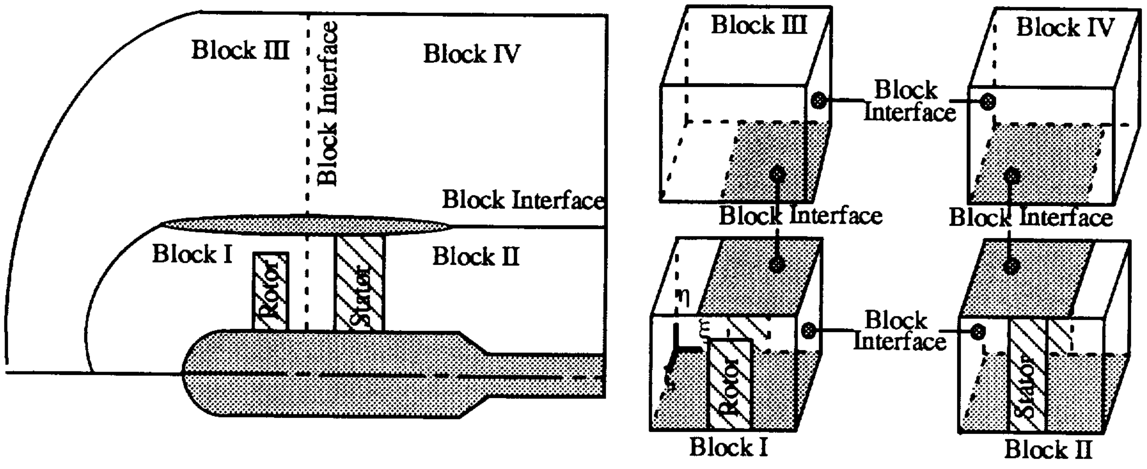
The numerical grid generation about a geometry requires a transformation from the physical domain to the computational domain that conforms to the boundaries of the flow region in such a way that boundary conditions can be accurately represented. This boundary–conforming curvilinear coordinate system facilitates the transformation of an arbitrarily–shaped physical domain into a rectangular computational domain, as shown in Figure 3.1. This rectangular computational domain, however, can be decomposed into several sub–domains or grid blocks to obtain grids with acceptable sizes for flow calculation. Each of these grid blocks can be further decomposed into sub–blocks to avoid grid construction difficulties near geometrically complicated regions.

In this work, a user can specify the topology for the domain transformation, i.e. H–type or C–type, and the entire physical domain is mapped to the computational domain automatically. As shown in Figure 3.1(a), an internal–external configuration is decomposed into four grid blocks. The notations ξ, η, and ζ represents the curvilinear coordinates in the axial, radial, and circumferential directions, respectively. The pressure side and suction side of each blade are transformed to be part of the ζ=1 and $\zeta=N_\zeta$ surfaces, as illustrated in



(a) Physical Domain (b) Computational Domain

Figure 3.1 H–Type Topology



(a) Physical Domain (b) Computational Domain

Figure 3.2 C–Type Topology

Figure 3.1(b). The nacelle is treated as a slat and becomes part of the common block interfaces (η=constant surfaces) as the shaded regions shown in Figure 3.1(b). Note that on the η=1 surface in Block–I, The un–shaded region represent the portion between the inlet and the hub nose. The same configuration with C–type transformation topology is shown in Figure 3.2(a). There is only a minor difference between Figure 3.1(b) and Figure 3.2(b). That is, the un–

shaded region disappeared in Figure 3.2(b) because the hub nose now resides at $\xi=1$.

In general, each blade row does not necessarily have the same blade count. Consequently, the circumferential spans of the passage associated with each blade row are not the same. In other words, they will not have the same circumferential span angle. The angle α associated with each block is determined by the number of blades (N_b) of the blade row contained within this block. That is,

$$\alpha = 2\pi / N_b \qquad (3.1)$$

If no blade is involved in a block, the blade angle is calculated by

$$\alpha = 2\pi \times \mathcal{N}_{\zeta} / \mathcal{N}_{\zeta-total} \qquad (3.2)$$

where

$\mathcal{N}_{\zeta}$ = Number of circumferential cells for this block passage,

$\mathcal{N}_{\zeta-total}$ = Total number of circumferential cells for the geometry.

Geometry Modeling

The process of grid generation includes generation of associated surface grids and generation of interior field points in the rectangular domain. Geometry modeling, the generation of surface grids associated with solid geometries, is one of the critical and labor–intensive steps of the overall grid generation process. Extreme care needs to be taken while manipulating the raw geometry data to obtain accurate geometry description on the resulting surface grid. In this work, NURBS definitions are utilized to process the geometry surfaces in order to maintain the geometry fidelity.

Geometry Definitions for Blade, Hub, and Shroud

A two–dimensional blade profile, as shown in Figure 2.2, can be defined in various ways. The simplest way of defining it would be describing the actual coordinates of discretized data points. Hamilton–Standard's blade definition is a more sophisticated approach[42]. For each cross–section, it requires the local offset angle (β) to the blade stagger angle, the distance between the chord line and the pitch change axis (D), the location of the leading edge with respect to the pitch change axis (Xle), the chord length (C), the distances of the pressure side and suction side to the chord length (Ys, Yp), the relative distance from the current location to the leading edge (Xb), etc.. A schematic description is shown in Figure 3.3. In this work, all the blade cross–sections are

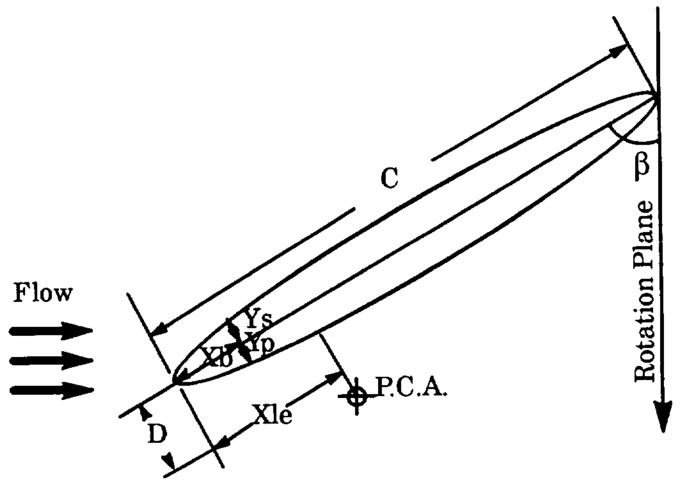


Figure 3.3 Blade Definition (Hamilton–Standard)

defined in discretized data form in cylindrical coordinate. Therefore, a user needs to convert the blade geometry data into the form ($\boldsymbol{x}^{p}{}_{ij}$, $\boldsymbol{r}^{p}{}_{ij}$, $\theta^{p}{}_{ij}$, $\boldsymbol{x}^{s}{}_{ij}$, $\boldsymbol{r}^{s}{}_{ij}$, $\theta^{s}{}_{ij}$) using the utility routines provided in this system. This standard format provides a user the flexibility for editing the data, if necessary.

As discussed in §2.4, the hub and shroud are assumed to be body of revolution, they are analog to generatrices which assume the form of g=($\mathbf{x}_i$,$\mathbf{r}_i$,*0*). Therefore, it is logical to define the hub and the shroud in the form of g=($\mathbf{x}_i$,$\mathbf{r}_i$). An example of the geometry definition is listed in Appendix A.

Blade Surface Construction

Geometry data obtained for blade surfaces are not necessarily well–connected to the hub and shroud. Therefore, there is a need to trim the blade surface with the hub surface and the shroud surface. A Newton–Raphson algorithm[12,13] is used to perform the surface–to–surface intersection. The intersection of a blade surface with a certain circumferential surface is decided by the algorithm based on the η–index information. After the intersection, each raw blade data point will be compared to the intersection curves to trim off the excess parts of the blade, as illustrated in Figure 3.4. The remaining data points are spline–fitted to the desired number of points and point distribution using NURBS formulations. A user needs to specify the distribution on four edges for the pressure side. Blade surfaces are spline–fitted with the desired number of points and the user–specified distribution. The same distribution will be copied to the suction side.

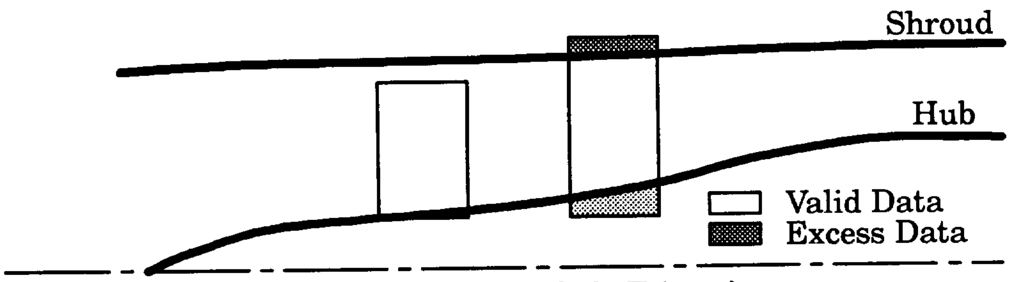


Figure 3.4 Blade Trimming

Non–Uniform Rational B–Spline (NURBS)

The spline–fitting routines used in TIGER are based on the NURBS definition. NURBS has drawn a lot of attention in the area of geometry modeling, computer graphics, and grid generation because of its powerful geometry properties such as the convex hull, local control, variation diminishing and affine invariance. It also has useful geometry tools such as knot insertion, degree elevation, and splitting[40]. Convex hull property allows the control of the geometry shape. The local control property allows editing of the geometry affects the geometry locally. Most of all, it allows accurate geometry representation, which is very important in geometry modeling.

A NURBS curve of m+1 control points can be defined by an order $\boldsymbol{k}$, a set of control points $\{\boldsymbol{b}_i,\ i=0,..,m\}$, a set of knots $\{\boldsymbol{u}_i,\ i=0,..,m+k\}$, and a set of weights $\{\boldsymbol{w}_i,\ i=0,..,m\}$. The following equations represent a NURBS curve:

$$c(u) = \frac{\sum_{i=0}^{m} w_i\, b_i\, N_i^k(u)}{\sum_{i=0}^{m} w_i\, N_i^k(u)} \tag{3.3}$$

where the basis functions N_i^k are defined as

$$N_i^k(u) = \frac{(u - u_i)N_i^{k-1}(u)}{u_{i+k-1} - u_i} + \frac{(u_{i+k} - u)N_{i+1}^{k-1}(u)}{u_{i+k} - u_{i+1}} \tag{3.4a}$$

$$\begin{aligned} N_i^1(u) &= 1 \quad \text{if} \quad u_i \leq u < u_{i+1} \\ &= 0 \qquad i = 0,\ 1,\ \ldots,\ m \end{aligned} \tag{3.4b}$$

A NURBS surface with (m+1)X(n+1) control points can be expressed by two orders $\boldsymbol{k}_u$ and $\boldsymbol{k}_v$, a set of control points $\{\boldsymbol{b}_{ij},\ i=0,..,m,\ j=0,..,n\}$, a set of knots $\{(\boldsymbol{u}_i),\ i=0,..,m+k_u,\ (\boldsymbol{v}_j),\ j=0,..,n+k_v\}$, and a set of weights $\{\boldsymbol{w}_{ij},\ i=0,..,m,\ j=0,..,n\}$. The following equation represents the NURBS surface:

$$s(u,v) = \frac{\sum_{i=0}^{k_u}\sum_{j=0}^{k_v} w_{i,j}\, b_{i,j}\, N_i^{k_u}(u)\, N_j^{k_v}(v)}{\sum_{i=0}^{k_u}\sum_{j=0}^{k_v} w_{i,j}\, N_i^{k_u}(u)\, N_j^{k_v}(v)} \qquad (3.5)$$

where the basis functions ($N_i^{k_u}$, $N_j^{k_v}$) are defined in a similar fashion as (3.4a) and (3.4b).

An expression for NURBS volume can also be obtained by using the similar criteria that was used to generate expressions for NURBS curve and surface. The NURBS volume formulation has been used to obtain analytic volume grids[43] [Ref. Yu and Soni]. The De Boor algorithm[40,44] [Ref.] can be used to evaluate the NURBS curves and the NURBS surfaces with known evaluation order(s), control points, weights, knot sequences, and hence the basis functions.

Inverse Approach

There is an on–going effort initiated by NASA IGES committee to define NINO[45] (NASA IGES NURBS ONLY) — a subset of the IGES[39] data format with only the NURBS formulations. The intention is to use NINO as a standard format for data transfer between the CAD/CAM community and the CFD community. However, despite this fact, the geometry data obtained for grid generation are not often in the form of NURBS representation but sets of discretized data. Therefore, the ability to convert such geometry data in discretized form into NURBS representation is needed. In order to do so, an inverse algorithm is used to evaluate control points and knot sequence from the given data set. In this work, the evaluation order is always set to be four, which gives a resulting cubic that maintains continuous second derivatives.

The inverse algorithm for a curve can be expressed as follows: Given a set of m data points $\boldsymbol{P} = \{\boldsymbol{p}_i=(x_i, y_i, z_i), i=1,..,m)\}$, one can obtain a set of knots $\boldsymbol{U} = \{\boldsymbol{u}_i, i=3,..,m+2\}$ evaluated by the normalized arc length of the given data set $\boldsymbol{P}$, and $u_0, u_1, u_2, u_{m+3}, u_{m+4}, u_{m+5}$ can be obtained by extrapolation. With the knots, a set of basis functions N_i^4 can be obtained. Assume all the weights associated with the control points are all unity, i.e. $\{\boldsymbol{w}_i=1.0, i=0,..,m+1\}$. The target solution is a set of control points $\boldsymbol{B} = \{\boldsymbol{b}_j, j=0,..,m+1\}$ that would be able to represent this set of data points. (3.3) can be written as

$$c(u_{i+2}) = \sum_{j=0}^{m+1} b_i \, N_i^4(u_{i+2}) = p_i \tag{3.6}$$

where $i=1,...,m$. (3.6) yields a system of m equations with $m+2$ unknowns. Therefore, two more equations are needed to solve for the $\boldsymbol{B}$ vector. These two extra equations can be obtained by specifying the slopes p_1' and p_m' at both ends of the curve. A simple way to obtain these slopes is to use a first order, one–sided finite difference method. The entire system to be solved becomes

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & & & \\ N_0^4(u_3) & N_1^4(u_3) & N_2^4(u_3) & & & \\ & N_1^4(u_4) & N_2^4(u_4) & N_3^4(u_4) & & \\ & & \ddots & \ddots & \ddots & \\ & & & N_{m-1}^4(u_{m+2}) & N_m^4(u_{m+2}) & N_{m+1}^4(u_{m+2}) \\ & & & a_{m+1,m-1} & a_{m+1,m} & a_{m+1,m+1} \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_m \\ b_{m+1} \end{bmatrix} = \begin{bmatrix} p_1' \\ p_1 \\ p_2 \\ \vdots \\ p_m \\ p_m' \end{bmatrix} \tag{3.7}$$

where

$$a_{00} = \frac{-\,3N_1^3(u_3)}{u_4 - u_1}$$

$$a_{01} = \frac{3N_1^3(u_3)}{u_4 - u_1} - \frac{3N_2^3(u_3)}{u_5 - u_2}$$

$$a_{02} = \frac{3N_2^3(u_3)}{u_5 - u_2}$$

$$a_{m+1,m} = \frac{3N_m^3(u_{m+2})}{u_{m+3} - u_m} - \frac{3N_{m+1}^3(u_{m+2})}{u_{m+4} - u_{m+1}}$$

$$a_{m+1,m+1} = \frac{3N_{m+1}^3(u_{m+2})}{u_{m+4} - u_{m+1}}$$

$$a_{m+1,m-1} = \frac{-\ 3N_m^3(u_{m+2})}{u_{m+3} - u_m} \tag{3.8}$$

One can use algebraic matrix manipulation for (3.7) to obtain a tri–diagonal matrix, which is computationally inexpensive to solve.

The analogous extension of this scheme using tensor products[40] leads to the calculation of control points for surfaces. Let $\boldsymbol{P} = \{\boldsymbol{p}_{ij}=(x_{ij}, y_{ij}, z_{ij}), i=1,..,m, j=1,..,n\}$ represent a surface patch. The analog algorithm is to sweep all the curves in the ξ–direction to obtain an intermediate control net $\bar{\boldsymbol{B}}=\{\bar{\boldsymbol{b}}_{ij}, i=0,..,m+1, j=0,..,n\}$ and substitute it as the right–hand–side in (3.7) to obtain the final control net $\boldsymbol{B}=\{\boldsymbol{b}_{ij}, i=0,..,m+1, j=0,..,n+1\}$. Mathematically, this algorithm can be expressed as following

$$\left[\mathcal{M}_i^{(m+2,m+2)}\right]_0 \left[\mathcal{B}_i^{(m+2,1)}\right]_0 = \left[\mathcal{P}_i^{(m+2,1)}\right]_0 \tag{3.9}$$

where $\left[\mathcal{M}_i^{(s,t)}\right]_j$ represents a s×t matrix $\mathcal{M}$ with its j–component looping. It is possible to sweep curve by curve in the η–direction to obtain an intermediate control net, for j=1,...,n,

$$\left[\bar{\mathcal{B}}_i^{(m+2,1)}\right]_j = \left[\mathcal{M}_i^{(m+2,m+2)}\right]_j^{-1} \left[\mathcal{P}_i^{(m+2,1)}\right]_j \tag{3.10}$$

Then sweeping in the ξ–direction, for i=0,...,m+1,

$$\left[\mathfrak{B}_j^{(n+2,1)}\right]_i = \left[\overline{\mathcal{M}}_j^{(n+2,n+2)}\right]_i^{-1} \left[\overline{\mathfrak{B}}_j^{(n+2,1)}\right]_i \qquad (3.11)$$

The resulting control net ***B*** has a size of $(m+2,n+2)$. The elements of the associate weights ***W*** are assumed to be all unity. Note that p_1' and p_m' in (3.7) can be obtained with different approaches to control the slope in the surface case.

<u>Curve and Surface Re–Distribution</u>

The inverse scheme is used to re–distribute points on a given curve. The procedure is extended utilizing tensor product formulation for surface calculation. NURBS formulation is used to spline–fit each radial curve on the blade after comparing with the interaction curves to make sure that the number of points along each curve is consistent so that a surface spline can be applied. The same algorithm is also used in the interactive steps allowing the user to move a breaking node that splits a segment of curve into two sections along a prescribed curve. This is accomplished by increasing or decreasing the parametric value of the breaking node, u, and using (3.3) with control points obtained from (3.7) and (3.8). NURBS formulation provides the flexibility for grid manipulation while maintaining geometry fidelity. Attention should be paid to the fact that the weights associated with each control point is unity. That is, for the curve definition in (3.3), the denominator becomes

$$\sum_{i=0}^{m} w_i \, N_i^k(u) = \sum_{i=0}^{m} N_i^k(u) = 1$$

due to the unity property. Therefore the scheme is equivalent to the non–uniform non–rational B–Spline

$$c(u) = \sum_{i=0}^{m} b_i \, N_i^k(u)$$

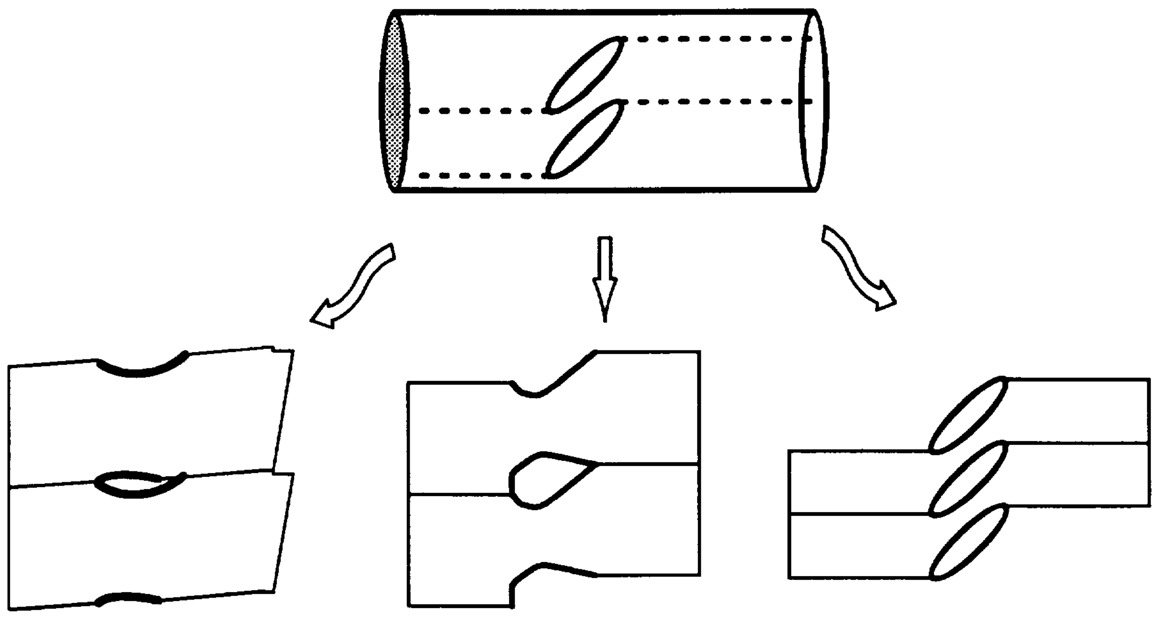


Figure 3.5 Cascade Surface Transformation
(a) Non–Periodic
(b) Direct Spread–Out
(c) Current Approach

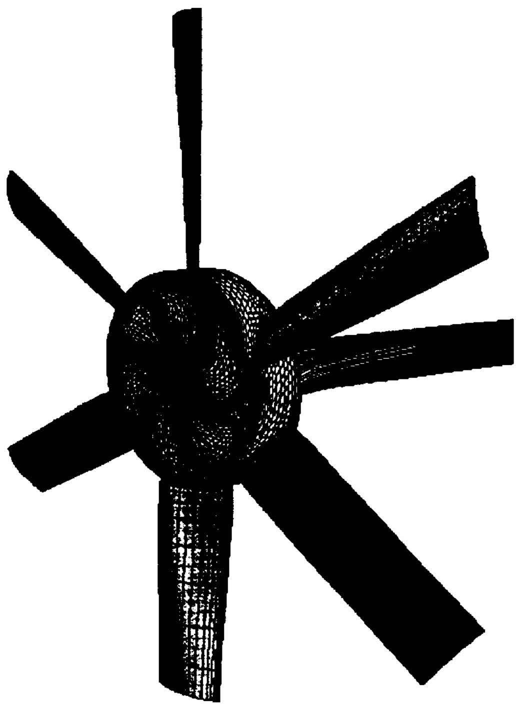


Figure 3.6 Numerical Model of an Industrial Ventilation Fan

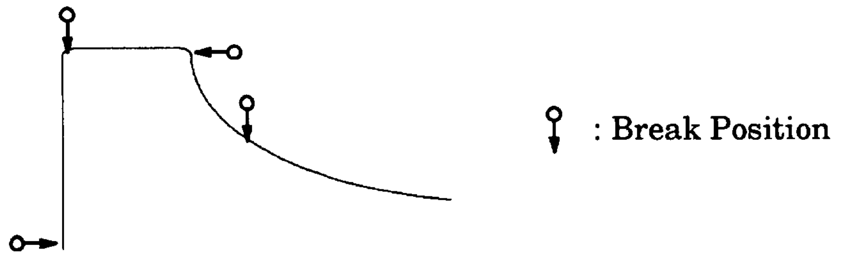


Figure 3.7 Hub Geometry Associated with the Industrial Ventilation Fan

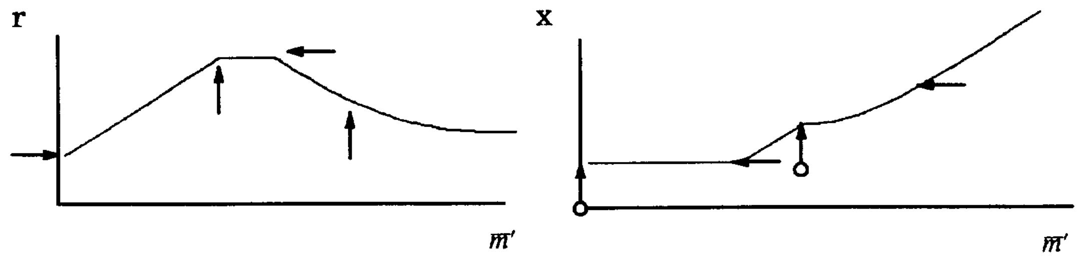


Figure 3.8 “Standard” Maps

Cascade Surface Mapping

Interactive manipulation of a grid on a three–dimensional surface is always a formidable task: maintaining the geometry definition alone is a big challenge, and at times, there can be difficulty in perceiving depth and geometry in a perspective view. Moreover, in order to maintain reasonable interactive response, the algorithm calculation must not be numerically expensive. Therefore, there have been efforts to generate a grid on a two–dimensional parametric space and transform it onto the target three–dimensional surface[46,47]. Fortunately, as mentioned in Chapter II, it is a reasonable assumption that the hub and shroud of a turbomachine are surfaces of revolution, i.e. the definition of such a surface can be simplified from $\mathcal{T}=\{(x,r,\theta)\}$ to $\mathcal{S}=\{(x,\theta)\}$ since for a generatrix g=(x,r), r is a function of x, or $r=f(x)$. Therefore, it would be possible to transform the cascade surface from three–dimensional space $\mathcal{T}$ into a two–dimensional parametric space $\mathcal{S}'=\{(m',\sigma)\}$, perform interactive manipulation in $\mathcal{S}'$, and transform it back to three–dimensional space.

There are many attempts to obtain a transformation between three–dimensional space $\mathcal{T}=\{(x,y,z)\}$ and two–dimensional parametric space $\mathcal{S}'=\{(m',\sigma)\}$. For a three–dimensional cascade surface of $(n_i+1)\times(n_j+1)$ on a surface of revolution, as shown in Figure 3.5, one can integrate the arc length of the ξ–boundaries and assign them as the m' coordinate in $\mathcal{S}'$, and use $r\theta$ as the σ coordinate. For a given set of surface grid $p(x_{ij},r_{ij},\theta_{ij})$, the transformation can be obtained by

$$\begin{aligned} m'_{0j} &= 0 \\ m'_{i,j} &= \sqrt{(x_{i,j}-x_{i-1,j})^2+(r_{i,j}-r_{i-1,j})^2} \\ \sigma_{i,j} &= \bar{r}\theta_{i,j} \end{aligned} \tag{3.12}$$

where

$$\bar{r} = \left\{\sum_{i=0}^{n_1} r_{i,0}\right\}/(n_i+1) \tag{3.13}$$

$$\begin{aligned} i &= 0,..,n_i \\ j &= 0,..,n_j \end{aligned}$$

This approach, although it has one–to–one correspondence for any associated generatrix, is not appropriate since the ξ–boundaries can contain chambered blade profiles. As a consequence, the arc length parametric coordinates m' are not equal for the lowest σ and the highest σ boundaries, as illustrated in Figure 3.5(a). In other words, periodicity no longer holds true with this transformation.

Another approach is to use the (x,θ) coordinates directly as (m',σ) to spread out the cascade surface into a two–dimensional plane. This approach, although it retains periodicity, loses the sense of the blade shapes in two–dimensional space. When the associated generatrix is not a single–valued function, this transformation also loses one–to–one correspondence. Hence, this is not a viable approach.

In the current work, the transformation relationship used in Reference 22 is adopted and enhanced. Similar to the first approach, this transformation is obtained by integrating the arc–length of the two–dimensional generatrix $g=(x_i, r_i)$ that defines the current surface of revolution, instead of the three–dimensional ξ–boundaries. An example of an industrial ventilation fan geometry is shown in Figure 3.6. The associated hub geometry is shown in Figure 3.7. These arc–length parametric coordinates maintain a one–to–one correspondence with every point (x_i, r_i). In other words, any point on this surface of revolution can find a one–to–one corresponding point by comparing the location with the associated "standard" maps, as shown in Figure 3.8. This approach eliminates the non–periodic problem experienced with the first approach. For a generatrix $g=g(\hat{x}(u), \hat{r}(u))$, its arc–length is integrated to be the "standard" m' coordinate by

$$\overline{m}'_i = \int_{u_0}^{u_i} g(\hat{x}(u), \hat{r}(u))\, du \qquad \overline{m}'_0 = 0.0 \qquad (3.14)$$

Each point $p(x_{ij}, r_{ij}, \theta_{ij})$ on the associated surface of revolution is compared with the maps shown in Figure 3.8 to obtain the proper m'–coordinate. However, if a generatrix is not a single–valued function, such as the example shown in Figure 3.7, there is a risk of losing the one–to–one correspondence. Therefore, a strategy is taken to safeguard this critical one–to–one correspondence by decomposing the generatrix into various sections based on the slope information. Arrows in Figures 3.7 and 3.8 represent the section break positions. For a generatrix section, if the slope $|\mu| < 1.0$, $\overline{m}'$–x map is used, otherwise $\overline{m}'$–r

map is utilized. Therefore, for each *point* $p(x_{ij}, r_{ij}, \theta_{ij})$, m'_{ij} is obtained by comparing with either $\overline{m}'$–x map or $\overline{m}'$–r map, and

$$\sigma_{i,j} = \bar{r}\theta_{i,j} \tag{3.15}$$

where

$$\bar{r} = \left\{\sum_{i=0}^{n_i} \hat{r}_i\right\} / (n_i + 1) \tag{3.16}$$

$$i = 0,..,n_i$$
$$j = 0,..,n_j$$

A transformed region associated with the industrial ventilation fan is demonstrated in Figure 3.9.

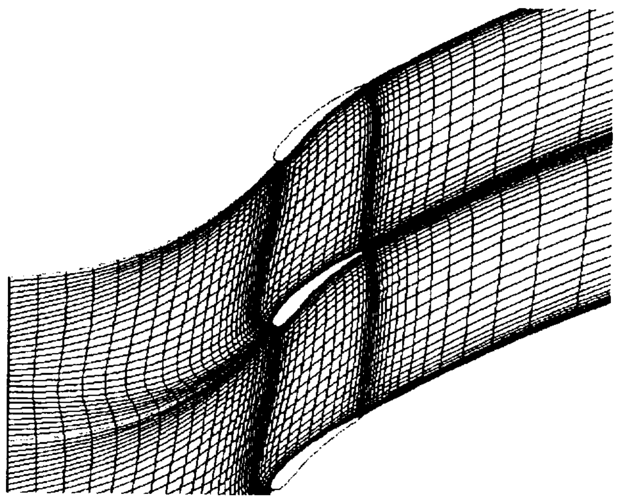


Figure 3.9 Transformed Two–Dimensional Region (Hub)

The inverse mapping is to obtain the cylindrical coordinates of each point $p(x_{ij}, r_{ij}, \theta_{ij})$ corresponding to each point in the parametric space $S'=\{(m',\sigma)\}$. For a point (m'_{ij}, σ_{ij}), the inverse mapping can be achieved by the following steps:

(*i*)Spline–fit m'_{ij} using $\overline{m}'$–x map to obtain x_{ij};

(*ii*)Spline–fit m'_{ij} using $\overline{m}'$–r map to obtain r_{ij};

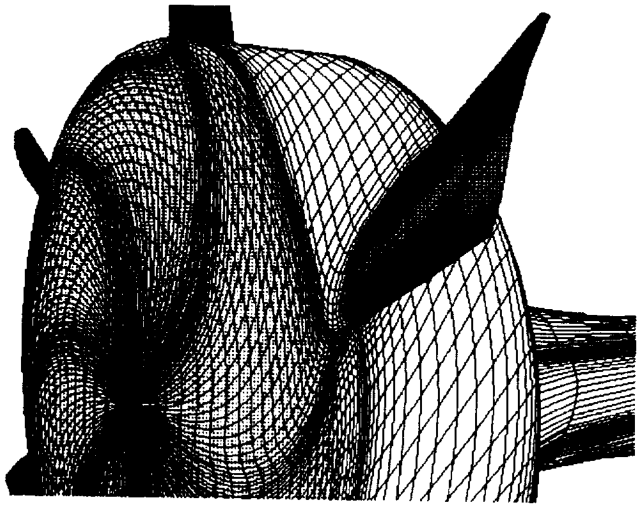


Figure 3.10 Resulting three–dimensional surface

$$(iii)\theta_{i,j} = \frac{\sigma_{i,j}}{\bar{r}}.$$

The resulting hub surface is shown in Figure 3.10. It is clearly demonstrated the success of this transformation scheme.

The cascade surface mapping scheme developed in this work is designed to spline–fit each point from (m',σ) space to desired surface of revolution point by point. It is therefore possible to allow surface grid generation with different topology. An example of (H+C)–type mixed–topology cascade surface is shown in Figure 3.11. C–type grid was used to generate the grid around the blade, while the upstream and downstream are constructed with H–type grid. An additional H–type grid block, however, may be inserted in between two neighboring C–type grid blocks. Demonstrated in Figure 3.12 is the capability of generating cascade surface grid in unstructured fashion. Presented in this example is simply a structured grid in an unstructured fashion, the grid quality is therefore not acceptable. However, it is clearly demonstrated that the scheme developed in this work is capable of handling a large variety of grid topologies accurately. It is possible for further development to re–construct the unstructured grid patch with either the Delaunay triangulation approach or the front propagation approach. Note that only two–dimensional unstructured grid generation scheme is needed to generate the three–dimensional surface grid on the surfaces of revolution, such as the hub.

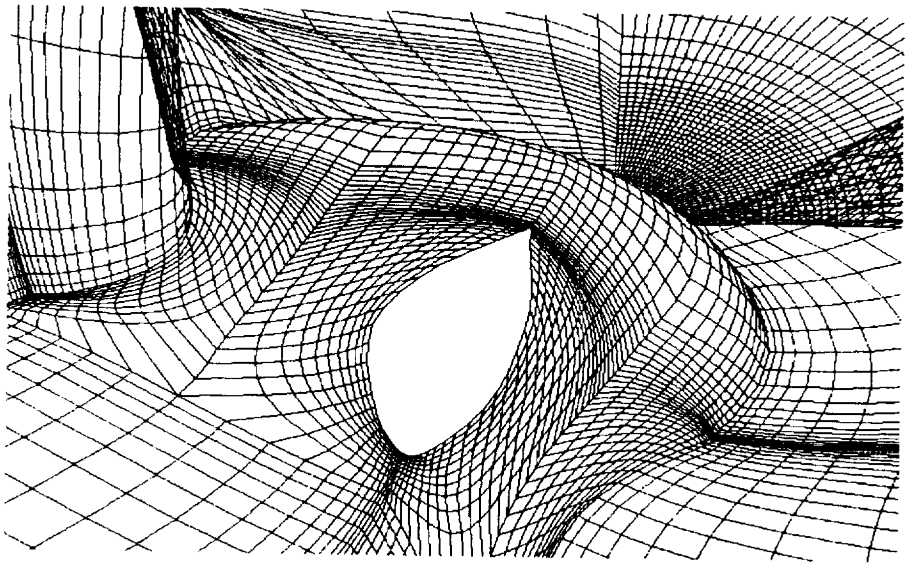


Figure 3.11 H–C Mixed–Topology Approach (Hamilton–Standard Single Rotation Propfan SR–7)

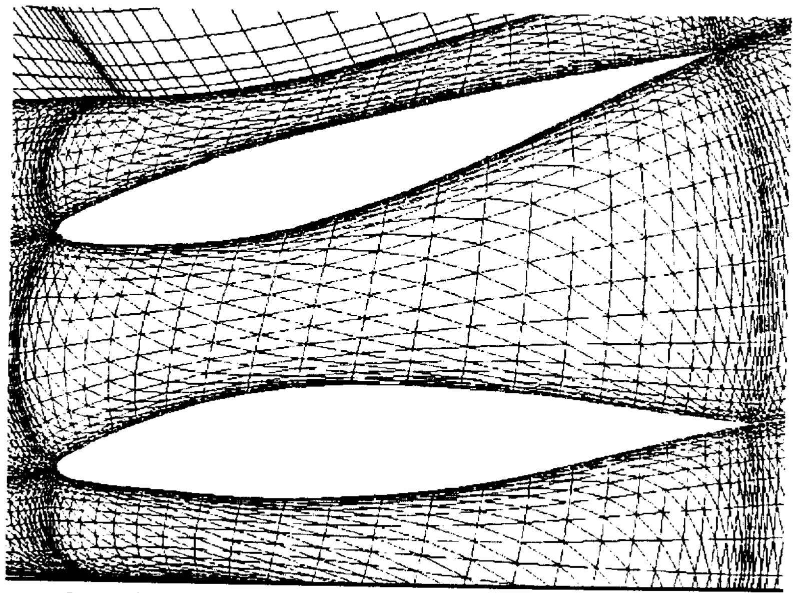
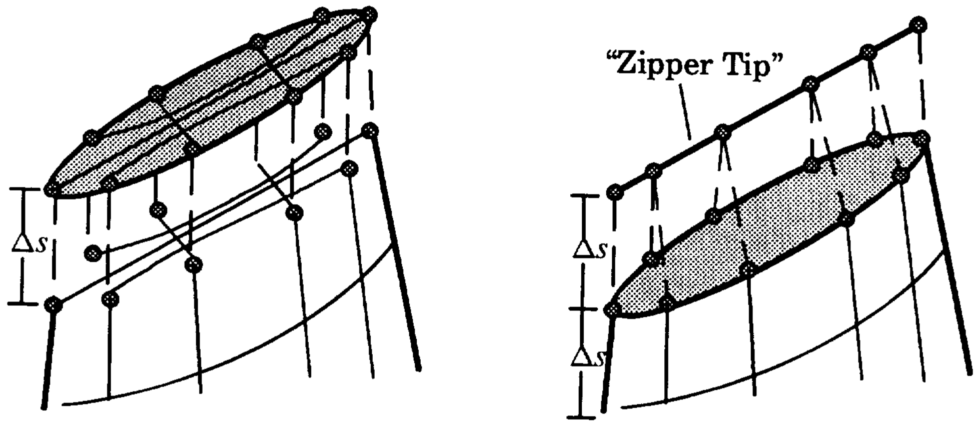


Figure 3.12 Unstructured Surface Grid for Hamilton–Standard Single Rotation Propfan SR–7

Blade Tip Modeling

In general, a blade can be defined without the description of the tip region. If an open–tip blade definition is obtained, there is a need to computationally model this region to close the blade tip. The modeling of the blade tip can be accomplished in many ways. Ideally, the surface on the blade tip



(a) Extra Block to Model Blade Tip (b) Current Approach

Figure 3.13 Blade Tip Modeling

should be modeled as it was designed, either flat or slightly rounded. However, such modeling would require an additional grid block that extends from the blade, as shown in Figure 3.13(a). Currently the algorithm is designed to take a blade with open tip and closes the blade with a single grid line ("zipper tip" as used later) that runs through the blade, which is approximately the mean camber line of the last blade profile on the tip, with an elevation Δs which is calculated from the arc length of the neighboring grid lines on the blade, as shown in Figure 3.13(b). This approach is apparently not a perfect one since the blade diameter will be extended $2\Delta s$ in length, creating a wedged tip surface. A result is presented in Figure 3.14. For a blade with large thickness, the blade tip region will become devoid of grid points. Fortunately, as experi-

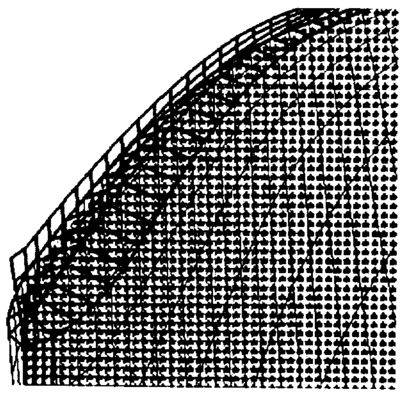


Figure 3.14 Blade Tip Modeling (Wedge Approach)

enced, it does not cause a significant discrepancy between the calculated flow solution and the experimental data[48].

Tip Clearance Modeling

A rotor blade in a turbomachinery design is the blade row that produces the pressure rise or work. The rotor blade rotates with a tip clearance between its tip and shroud. In this work, any blade is treated as a rotor blade if it is shrouded with a tip clearance, as discussed in §2.5.

Tip clearance is a very crucial issue for a turbomachinery design, as discussed in Chapter II. Since the tip clearance gap δ is usually less than 1% of the blade diameter (in some cases, δ can be even lower than 0.15% of the blade diameter), a minor discrepancy of grid distribution or grid behavior between the cascade surfaces of the rotor tip and the shroud will cause severe skewness in the grid. Therefore, extreme care needs to be taken to match the grid behavior between these two grid surfaces in order to eliminate such skewness. It becomes, however, tedious work to generate a grid with a general–purpose grid generation code. To resolve this problem, the algorithm is designed to generate the surface grid on the blade zipper tip level ($\eta-$ztip) by using the same distribution mesh as used on the blade tip level (η – tip). The distribution mesh is obtained by integrating the arc–length of each curve and normalized them between 0 and 1. This approach shows improvement on grid quality, viz., the skewness. However, it still produces significant skewness, as illustrated in Figure 3.15. This is because the generation of surface grid on η – ztip with WTFI using arc length distribution on η – tip is not enough, since

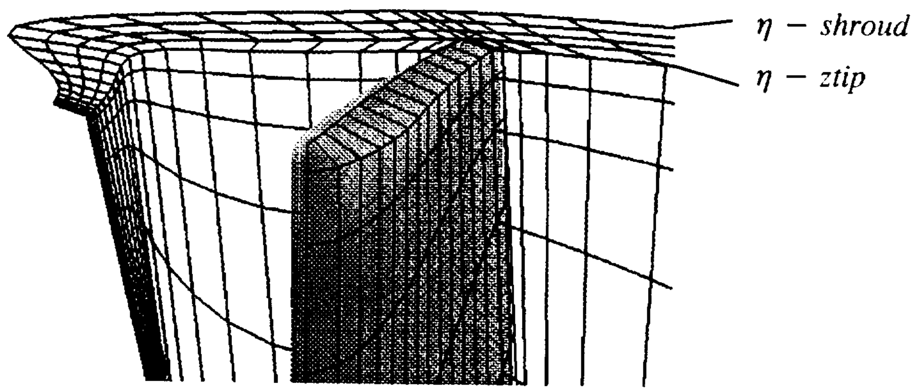


Figure 3.15 Kinks Induced by Non–Matching Grid Lines Between η – ztip and η – shroud Surfaces

the blade section is open on η – tip and it is sealed as a curve on η – ztip. There is no way for the η – ztip WTFI surface grid to realize the grid behavior on η – tip. Therefore, one more step is taken in TIGER to accomplish the quality improvement. Grid points from each circumferential curves on η – tip are taken and added to this curve with two end–points A and B in Figure 3.16. Points A and B are obtained by averaging the corresponding points on the suction side and the pressure side of the blade. The NURBS curve spline–fit algorithm is used to spline fit the curve with the arc length distribution obtained from η – tip. That is, the algorithm is designed to fit N_ζ+2 points with N_ζ points and proceeds the spline process curve by curve. The resulted η – ztip surface is then projected onto the shroud surface (η – shroud) by finding the

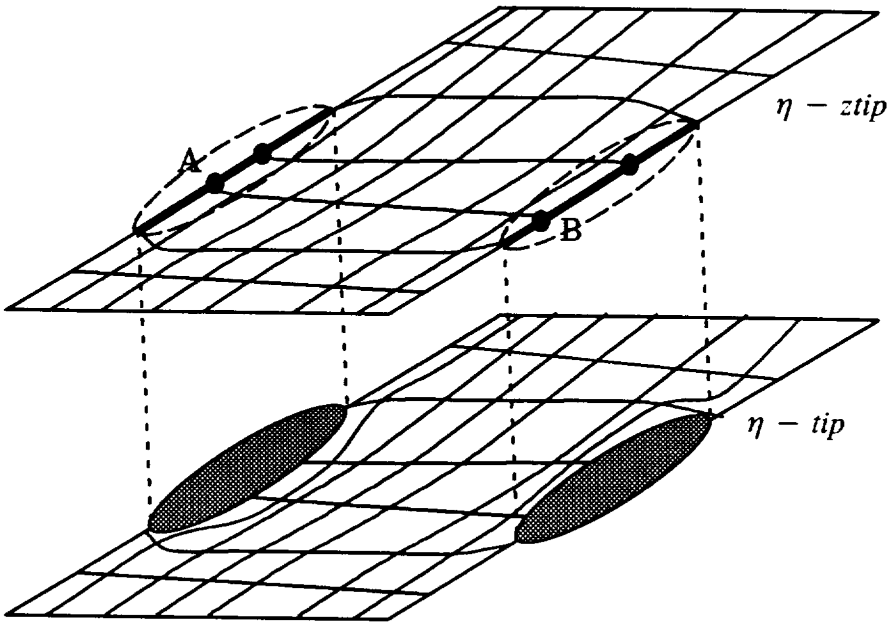


Figure 3.16 Automatic Surface Grid Matching

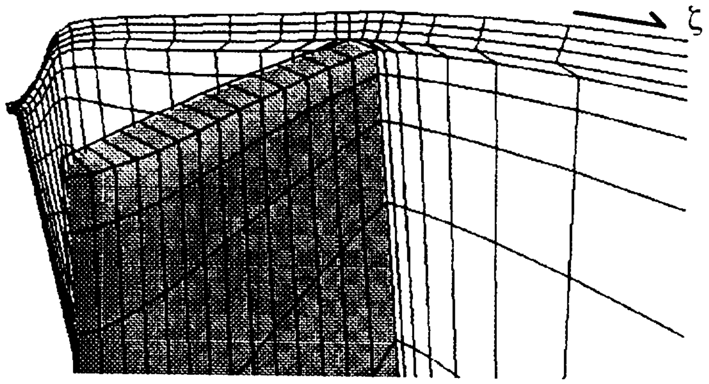


Figure 3.17 Automatic Projection

shortest distance of each point to the η−shroud surface of revolution to obtain the correct radius. The result is shown in Figure 3.17. Comparing Figures 3.15 and 3.17, it is evident that the grid quality of the η−ztip surface has been improved with this approach. The skewness in ζ–direction shown in Figure 3.17 is the result of the wedged–tip approach.

Unfortunately, when there is a concave region on the η−shroud surface such as the nacelle duct–lip region as shown in Fig. 3.18, this scheme will fail. It is because that for a point near the concave region, the shortest distance can not be the concave point, which is usually a critical geometry location. As shown in Figure 3.18, points A and B are supposed to project to points C and

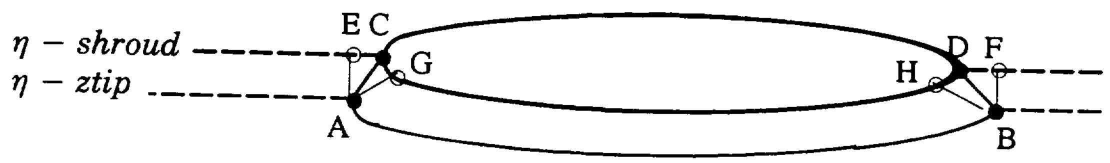


Figure 3.18 Failure Case: Concave Regions

D, respectively, to retain the geometry fidelity. However, by finding the shortest distance, they will be projected to points E and F (or G and H) instead. Failure to keep these corners means the failure of keeping geometry fidelity. In order to overcome this shortcoming, a checking routine is developed to safeguard this region. If a concave region like Figure 3.18 is detected, then after projecting section AB from η−ztip to EF or GH on η−shroud, a NURBS spline algorithm is used to spline fit this projected section back to section CD on η−shroud, using the arc length relative distribution obtained from section EF or GH to distribute the points. Figure 3.19 shows part of an engine nacelle. Figure 3.20 demonstrates the surface without safeguard process, and Figure 3.21 shows the surface with safeguard process.

LE/TE Circle Modeling

There are incidents that a blade is defined with only the surface data on the pressure side and suction side, but not with the LE/TE circles information[50]. In some cases, the data points are too sparse that they could not de-

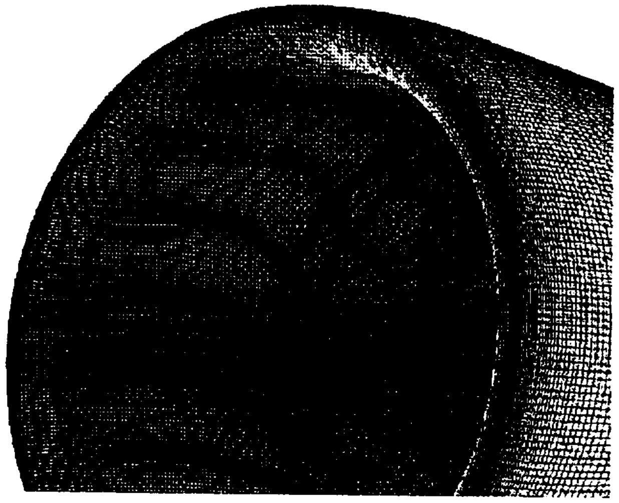


Figure 3.19 An Engine Nacelle

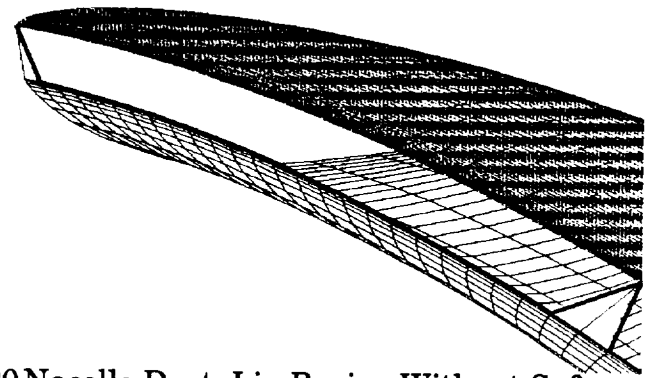


Figure 3.20 Nacelle Duct–Lip Region Without Safeguard Algorithm

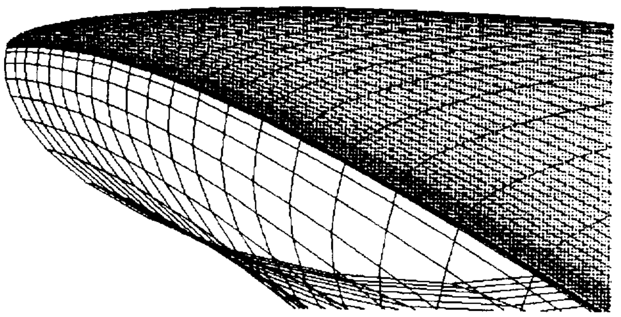


Figure 3.21 Nacelle Duct–Lip Region With Safeguard Algorithm

scribe the blade geometry properly. In these incidents, there is a need to fit the LE/TE circles based on the given data points. This tool takes a blade with open–ends or closed–ends and fits the circles inside.

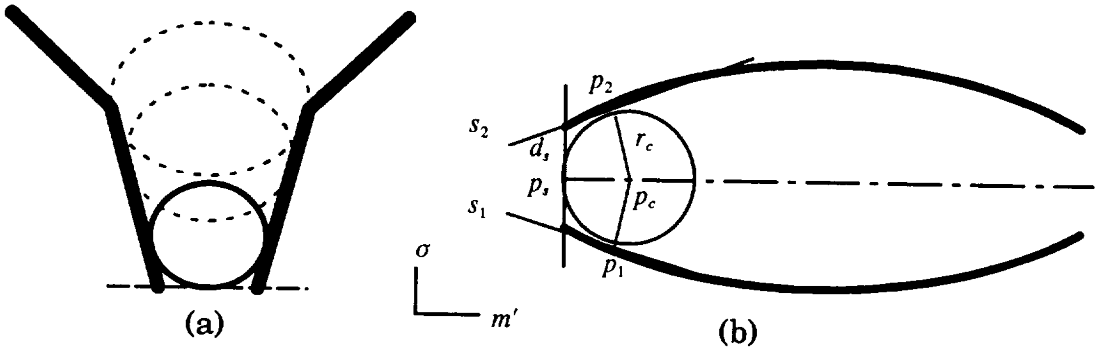


Figure 3.22 Leading Edge Circle Fitting
(a) Find the Proper BB Ball in the Funnel
(b) Two–Dimensional Airfoil

The algorithm is analog to dropping BB balls of various radii through a funnel, and trying to find the BB ball that will be tangent to the funnel's surface and the exit plane of the funnel, as demonstrated in Figure 3.22(a). The analog application for a two–dimensional airfoil Figure 3.22(b). The initial algorithm developed by Shumann[51], and re–coded by Chima[52], has various constraints on the input data, such as the number of points on both side must be the same, and their chord–wise coordinates must be the same, etc.. This algorithm has been generalized and these constraints have been removed in this work.

Referring to Figure 3.22(b), let the mid–point of the first point on both sides be $p_s=(m'_s,\sigma_s)$; the tangent points be $p_1=(m'_1,\sigma_1)$ and $p_2=(m'_2,\sigma_2)$; the distance between these two points be $2d_s$. The initial guess for the center of the leading edge circle $p_c=(m'_c,\sigma_c)$, and the radius of the leading edge circle be r_c. Then the algorithm to obtain the leading edge circle parameters can be expressed as following:

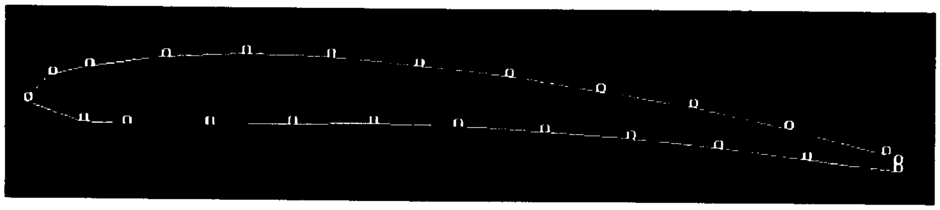
$(i)\ r_c = d_s$
$(ii)\ m_c = m_s + r_c$
$(iii)\ spline - fit$ for s_1 and s_2
$(iv)\ m_1 = m_c - r_c \times \sin(\tan^{-1}(s_1))$
$(v)\ m_2 = m_c + r_c \times \sin(\tan^{-1}(s_2))$
$(vi)\ spline - fit$ for σ_1 and σ_2
$(vii)\ \hat{r}_c = (\sigma_1 - \sigma_2)/(cos(\tan^{-1}(s_1)) + \cos(tan^{-1}(s_2))$
$(viii)\ update\ r_c$ if $|r_c - \hat{r}_c| > tolerance$

Step *(ii)*–*(viii)* are iterative. Figure 3.23 shows the result of a 2D airfoil LE and TE circles fitting.

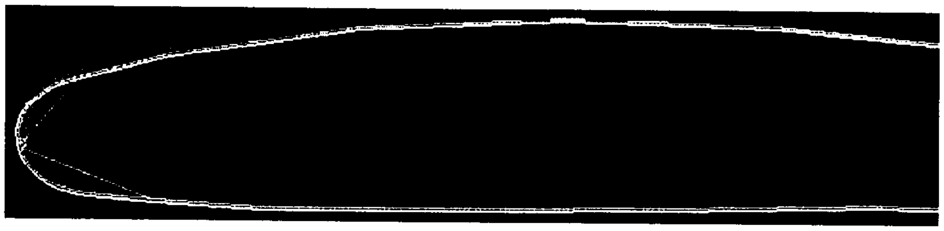
Even though this algorithm works in a two–dimensional space, it is also applicable to a three–dimensional blade. For application to the three–dimensional blades, each cross–section of the blade profiles is transformed into two–dimensional space (m',σ). Better results will be obtained if for each profile, the data points for both sides of the blade are lying on the same surface of revolution. Otherwise, the algorithm will take the average of the data points from both sides and use this averaged curve to form the surface of revolution. It is obvious then that there will be distortion for the resulting circles. An example for a three–dimensional blade is shown in Figures 3.24. Wireframe in Figure 3.24(a) represents the original surface data, and the shaded surface shows the rounded trailing edge. Note that the trailing edge is confined inside the original blade and is tangent to the plane where the original blade surface ends. Figure 3.24(b) shows the surface grid on the hub around the rounded blade trailing edge

User Interactions

User interactions in TIGER allows the user to manipulate the grid through the aide of GUI and graphics. The algorithm is designed to set up an

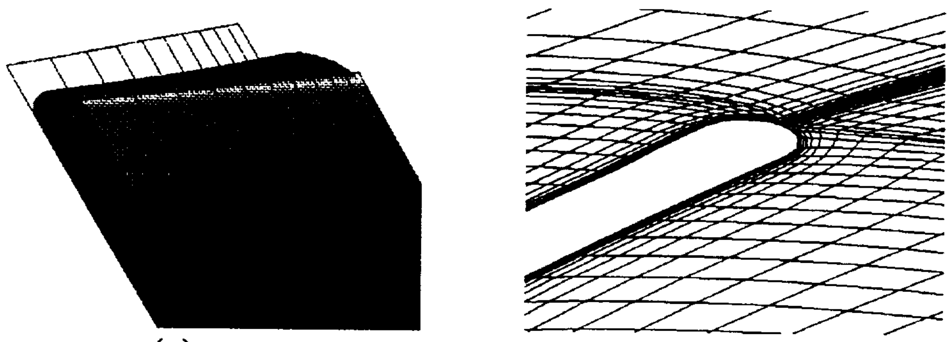


(a) Before Circle Fitting



(b) After Circle Fitting

Figure 3.23 Circle Fitting on a 2D Airfoil



(a) (b)

Figure 3.24 Results of Circle Fitting on a 3D Blade

initial framework of the grid and connecting each breaking nodes with straight lines for each field segments. However, for those geometry segments, such as the curve that defines the domain, the raw data points are spline–fitted to the desired number of points based on the index information. Even spacing is default to distribute points on all of the segments. Therefore, The algorithm expects a user to identify segments that needs modification by picking it with mouse, and provides necessary manipulations, such as modifying the curve in Bezier fashion, and information, such as distribution function. A distribution function can be provided by choosing the proper button on the GUI, or it can be provided interactively by manipulating a distribution function curve.

Bezier Curve Application

Bezier curve has been used in grid generation softwares[10,23,53,56] to allow local manipulation for better grid behavior. In this work, it has been applied substantially in various aspects to assist the user for better control on the grid, such as the design of generatrix in the field to connect each geometry entities, or the manipulation of the circumferential curve on the cascade surface. A Bezier curve of degree m can be defined as

$$\boldsymbol{c}(u) = \sum_{i=0}^{m} \boldsymbol{b}_i \, \boldsymbol{B}_i^m(u) \tag{3.17}$$

where

$$\boldsymbol{B}_i^m(u) = \binom{m}{i} (1-u)^{m-i} \, u^i \tag{3.18}$$

are Bernstein Polynomials of degree m, $\boldsymbol{b}_i$ are the Bezier points, and u is the parametric value with $0 \leq u \leq 1$. In practical grid generation applications, four Bezier points are sufficient, with two end points which connects to the geometry and two inner Bezier points that control the shape and slopes of the curve. An example is shown in Figure 3.25. A Bezier curve has several useful properties that are meaningful in grid generation. One interesting and important feature of a Bezier curve is that, from (3.17) with m=3,

$$\begin{aligned} c(0) &= b_0 \\ c(1) &= b_3 \\ c'(0) &= 3(b_1 - b_0) \\ c'(1) &= 3(b_2 - b_3) \end{aligned} \tag{3.19}$$

which means that the slopes at the end points of the curve are controlled by

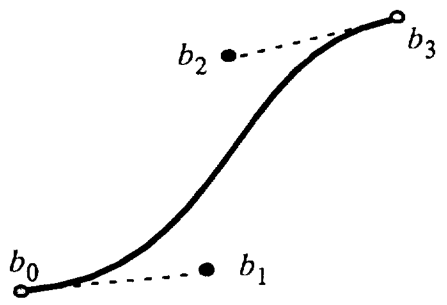


Figure 3.25 A Bezier Curve With Four Bezier Points

the inner Bezier points. This is a very useful feature for interactive grid generation since a user can adjust these two end points to obtain near–orthogonal grid.

A Bezier curve with four Bezier points can be obtained by setting m=3 in (3.17) and (3.18), and can be expressed in matrix form

$$c(u) = U\ M\ R \tag{3.20}$$

where

$$U = [\ 1\ u\ u^2\ u^3\ ]\ ,$$

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ 1 & 3 & -3 & 0 \end{bmatrix}\ , \textit{and}$$

$$R = [\ b_0\ b_1\ b_2\ b_3\ ]^T$$

or in polynomial form

$$c(u) = (1 - u)^3 b_0 + 3u(1 - u)^2 b_1 + 3u^2(1 - u) b_2 + u^3 b_3 \tag{3.21}$$

Stretching Functions

There are six basic stretching functions that are available to the user. They are (*i*) even spacing, (*ii*) exponential function, (*iii*) hyperbolic tangent function, (*iv*) geometry progression function, (*v*) hybrid function, and (*vi*) interactive, Bezier–curve–based function. Most of these functions distribute the points either on one end, or both ends. The exponential and hyperbolic tangent stretching function routines are adopted from GENIE^{++}. Their formulations and discussions can be found in many literatures[57–60]. Geometry progression function, hybrid function, and the interactive, Bezier–curve–based function are developed in this work.

Relative spacing *ds* is used to specify the stretching in this work. However, this relative spacing scheme uses the relative value with respect to the even spacing interval instead of the whole segment as usually used. For a curve with 11 points, for example, *ds*=0.1 in TIGER is equivalent to *ds*=(1./(11–1)) x 0.1 = 0.01 in the usual reference. This approach changes the perception of relative spacing from global sense to local sense, which frees the user from tedious mind calculation.

Geometry Progression Point Distribution

By specifying an initial interval ds_1 and a maximum progression ratio $G_{r\,max}$, one can express the relationship as

$$ds_i = ds_1 \times (G_r)^{i-1} \tag{3.22}$$

where $i = 1,2,\ldots,n$, and $(n+1)$ is the number of points of this segment and $G_r \leq G_{r\,\max}$ which can be easily obtained from

$$\sum_{i=0}^{n} ds_1 \times (G_r)^i = \Psi \tag{3.23}$$

by Newton iteration with $\Psi = 1.0$. If G_r obtained from (3.25) is less than or equal to $G_{r\,max}$, this option will consider that the inputs are valid and return appropriate distribution array $\mathcal{E} = \{ \hat{e}_i, i = 0, n \}$. However, if a user specifies G_r $_{max}$ that is too large and $(n+1)$ that is too small, or $G_{r\,max}$ that is too small and $(n+1)$ that is too large, this option will fail and even spacing will be returned.

Hybrid Point Distribution

In some cases, a user may want to use geometry progression for a portion of the inner region yet proceed the distribution in the outer region with hyperbolic tangent function. A user needs to specify the percentage of the region that is going to be distributed with geometry progression, G_p, and the progression ratio G_r. The algorithm is developed to calculate the number of points n needed to satisfy geometry progression in (3.23) with specified G_r and $\Psi = G_p$. The last interval in geometry progression region is used as the ds in the outer region with specified function, such as hyperbolic tangent. Distinct from the pure geometry progression, this option uses exactly the user–specified G_r as the progression ratio to progress the distribution until it reaches the specified G_p. When $G_r = 1.0$, the inner region will maintain even spacing and proceed with specified function in the outer region. This option can be applied to one–sided or two–sided point distribution. If two–sided point distribution is desired, a symmetric distribution for both sides will be attempted. A comparison between the exponential, hyperbolic tangent, geometry progression, and hybrid approach is shown in Figure 3.26.

Interactive Point Distribution

Other than using an analytic function which produces only one distribution array $\mathcal{E} = \{ \hat{e}_i, i = 0, n \}$ with given conditions, viz. number of points,

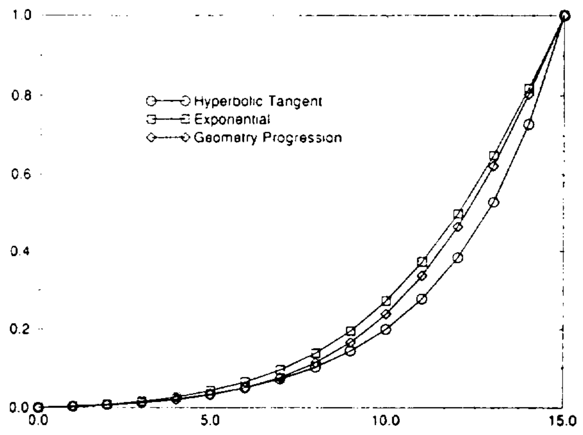


Figure 3.26 Comparison of Exponential, Hyperbolic Tangent, and Geometry Progression Approaches

ds_0 and/or ds_m, a user may sometimes want to "edit" a stretching function curve in order to obtain a special set of point distribution. The algorithm is designed to provide a graphically interactive stretching capability, allowing the user to use the mouse to manipulate a cubic Bezier curve with four Bezier points to obtain a desirable set of point distribution. There are two approaches to obtain $\mathcal{S}$, namely, the box constrained approach and the line constrained approach, as shown in Figure 3.27.

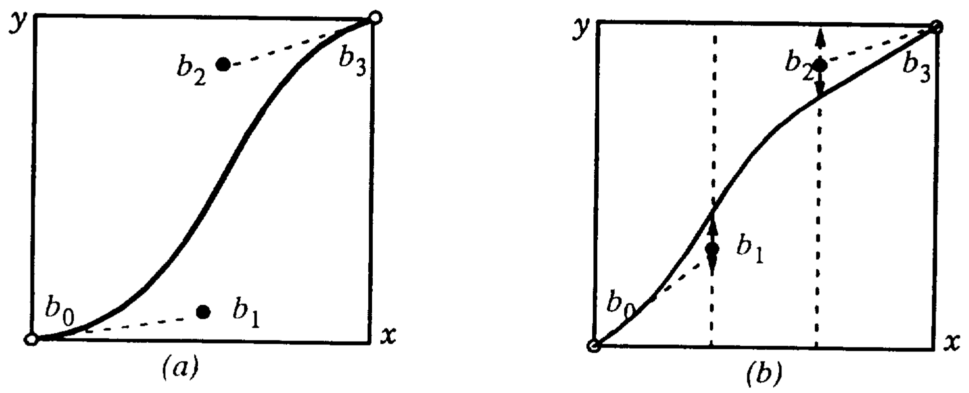


Figure 3.27 Interactive Distribution: (a) Box Constrained
(b) Line Constrained

The box constrained approach is achieved by setting the first Bezier point $\boldsymbol{b}_0$ at (0,0, 0.0) and the last Bezier point $\boldsymbol{b}_3$ at (1.0, 1.0), and constrains the second and the third Bezier point within this unit square box. In order to calculate the associated distribution value $\hat{e}_i$ of the Bezier curve, $\boldsymbol{c}(u)$ array is assigned in such a way that its first element is the index value of the point normalized between 0 and 1. That is, x_i={(float) i} / {(float) n }. Its second element is the parametric to be solved. With the first element known, (3.21) can be de–coupled and solved for the corresponding $\hat{u}$ value. That is, (3.21) can be re–written as

$$A\hat{u}^3 + B\hat{u}^2 + C\hat{u} + D = 0 \tag{3.24}$$

where

$$\begin{aligned} A &= -b_0 + 3b_1 - 3b_2 + b_3 \\ B &= 3b_0 - 6b_1 + 3b_2 \\ C &= -3b_0 + 3b_1 \\ D &= b_0 - c(u) \end{aligned}$$

$\hat{u}$ can be solved by Newton iteration for each point index. Applying it in (3.21) one can obtain the associated distribution value $\hat{e}_i$. Since all of the four Bezier control points are constrained inside this unit box, this approach can never produce illegal parametric distribution, i.e. this Bezier curve will always be a single–valued function. This can be proved by showing the first derivative of this Bezier curve is always positive. In other words, there is no point of inflection along the distribution curve. The derivative vector of (3.17) can be written as

$$\frac{dc(u)}{du} = m \sum_{i=0}^{m-1} \Delta \boldsymbol{b}_i \boldsymbol{B}_i^{m-1}(u) , \tag{3.25}$$

where

$$\Delta \boldsymbol{b}_i = \boldsymbol{b}_{i+1} - \boldsymbol{b}_i \ . \tag{3.26}$$

(3.25) can be expanded as

$$\frac{dc(u)}{du} = 3 \ [\ \Delta \boldsymbol{b}_0 u^2 + \Delta \boldsymbol{b}_1 u(1 - u) + \Delta \boldsymbol{b}_2 (1 - u)^2 \ ] \tag{3.27}$$

For the worst case, i.e. $\boldsymbol{b}_1$ = (1.0, 0.0) and $\boldsymbol{b}_2$ = (0.0, 1.0), then $\Delta \mathbf{b}_0$ = (1.0, 0.0), $\Delta \mathbf{b}_1$ = (–1.0, 1.0) and $\Delta \mathbf{b}_2$ = (1.0, 0.0). Obviously (3.27) becomes

$$\frac{dc(u)}{du} = \begin{bmatrix} 9u^2 - 9u + 3 \\ 3u - 3u^2 \end{bmatrix} = \begin{bmatrix} (3u - 1)^2 + 2 - 3u \\ 3u(1 - u) \end{bmatrix} \tag{3.28}$$

Since $0 \leq u \leq 1$, it is obvious that the slope of (3.17) is always positive. Therefore, using this approach allows the generation of valid distribution as long as the Bezier points $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ are constrained within the unit box. Some extreme cases with this approach are shown in Figure 3.28. Even spacing is represented by curve (1). Clustering points toward both ends is represented by curve (2), while generating curve (3) allows us to produce a distribution array that clusters points in the middle of the line segment. If a user would like to pack points toward the lower end of the line segment, both inner Bezier points should be placed on the right–lower corner to produce curve (4). Note that these extreme cases can produce a legal yet undesirable distribution. Detail discussion on the Bezier point setting will be provided in the later sections.

This interactive approach is fairly inexpensive in view of computing time. On a SGI Personal Iris workstation with 20MHz CPU and 16MB memory, a user can interactively distribute the point on a curve without much drag. This operation actually includes: (*i*)graphical display of the distribution curve, (*ii*)calculation of Bezier curve, (*iii*)Newton iterations to obtain proper parametric $\hat{u}$ for each point, (*iv*)spline–fitting the curve using NURBS algorithm, (*v*)surface grid interpolation, and(*iv*)image update for the entire dis-

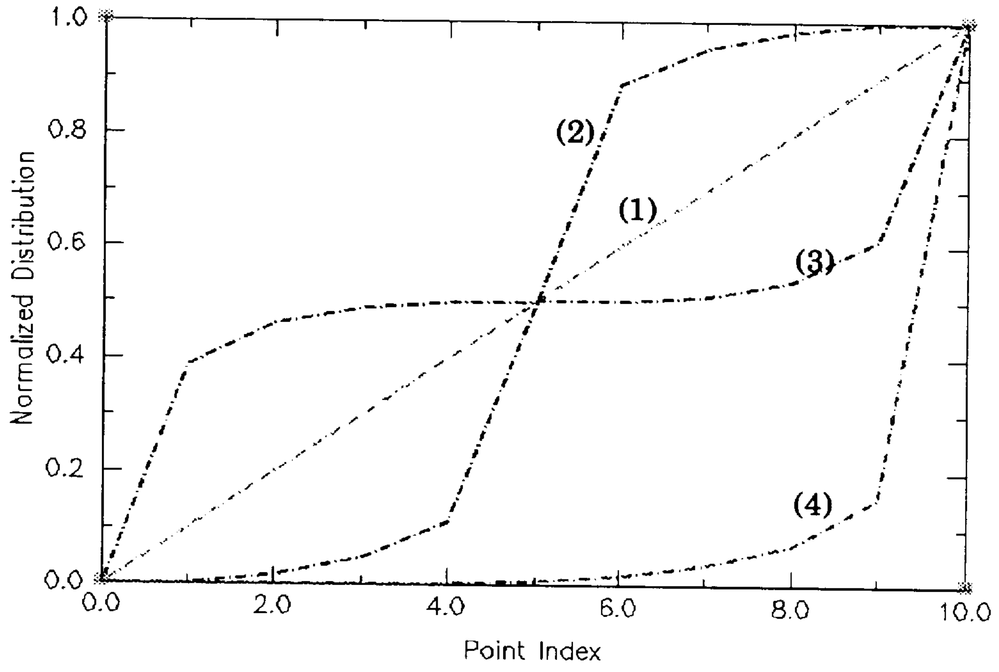


Figure 3.28 Extreme Cases With Box Constrained Approach

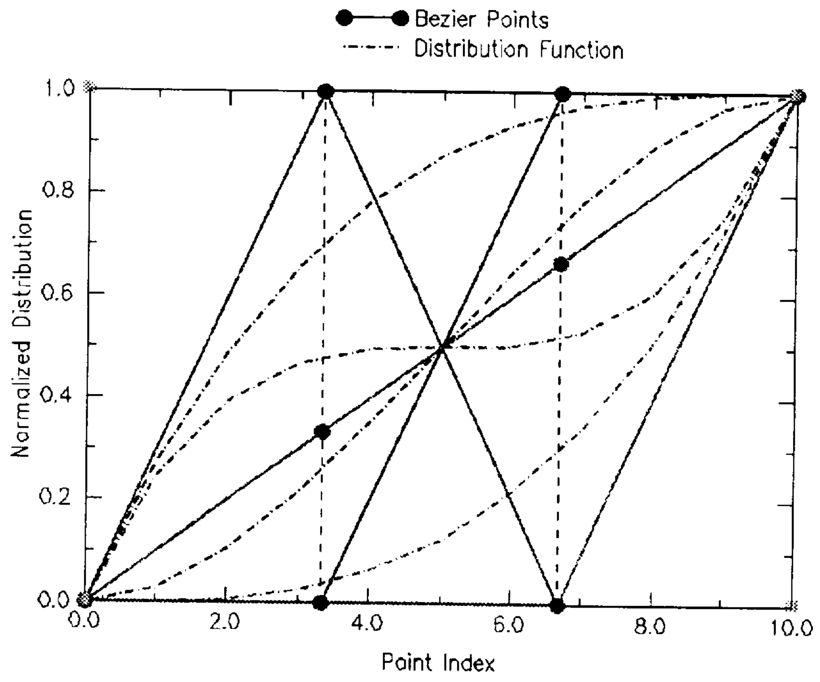


Figure 3.29 Distribution Curve With Line Constrained Approach

play. However, the calculation of distribution array $\mathcal{E}$ from a Bezier curve can be further modified to facilitate the algorithm by using the line constrained approach.

If the second and the third Bezier points $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ are constrained along u=1/3 and u=2/3 lines, respectively, as shown in Figure 3.27(b), each element in $\mathcal{E}$ can be obtained by

$$\hat{e}_i = \sum_{j=0}^{3} y_j \, B_j^3(u_i) \tag{3.29}$$

where y_j is the value of the second component of array $\boldsymbol{b}_j$. Obviously (3.29) incurs less float–point operations than the first approach. Figure 3.29 shows some extreme cases.

With the gain in speed, however, there is a price to pay. The drawback in this approach is the minimum $\hat{e}_i$ available. Since both inner Bezier points $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ are constrained to the lines at u=x=1/3 and u=x=2/3, respectively, the minimum interval that it is able to achieve would be larger than that available with box constrained approach. Detailed comparisons are discussed below.

Minimum Available Spacing For Box Constrained Approach

Table 3.1 Bezier Point Setting for Minimum Spacing (Box Constrained, One–End)

	b_0	b_1	b_2	b_3
x	0	1	1	1
y	0	0	0	1

The minimum one–end–packing (Lower End) relative spacing of box constrained approach for a curve with n points can be obtained by

$$ds_{\min} = \hat{u}^3 \tag{3.30}$$

where $\hat{u}$ is the solution of

$$\hat{u}^3 - 3\hat{u}^2 + 3\hat{u} - \frac{1}{(n-1)} = 0 \tag{3.31}$$

with the control points setting specified in Table 3.1. For example, consider n=11, the minimum relative interval available for box constrained approach is 0.00041 with respect to the even–spacing interval, or 0.000041 with respect to the overall segment.

Table 3.2 Bezier Point Setting for Minimum Spacing (Box Constrained, Two–End)

	b_0	b_1	b_2	b_3
x	0	1	0	1
y	0	0	1	1

The minimum two–end–packing relative spacing of box constrained approach for a curve with n points can be obtained by

$$ds_{\min} = -2\hat{u}^3 + 3\hat{u}^2 \tag{3.32}$$

where $\hat{u}$ is the solution of

$$4\hat{u}^3 - 6\hat{u}^2 + 3\hat{u} - \frac{1}{(n-1)} = 0 \tag{3.33}$$

with the control points setting specified in Table 3.2. Consider n=11, the minimum relative interval available for box constrained approach is 0.03762 with respect to the even–spacing interval, or 0.003762 with respect to the overall segment.

Minimum Available Spacing For Line Constrained Approach

Table 3.3 Bezier Point Setting for Minimum Spacing (Line Constrained, One–End)

	b_0	b_1	b_2	b_3
x	0	1/3	2/3	1
y	0	0	0	1

The minimum relative spacing with line constrained approach for a curve with n points can be obtained by

$$ds_{\min} = u^3 \qquad (3.34)$$

with the control points setting specified in Table 3.3. Note that $u=1/(n-1)$. Consider n=11, the minimum relative interval available for box constrained approach is 0.01 with respect to the even–spacing interval, or 0.001 with respect to the overall segment.

Table 3.4 Bezier Point Setting for Minimum Spacing (Line Constrained, Two–End)

	b_0	b_1	b_2	b_3
x	0	1/3	2/3	1
y	0	0	1	1

The minimum relative spacing with line constrained approach for a curve with n points can be obtained by

$$ds_{\min} = -2u^3 + 3u^2 \qquad (3.35)$$

with the Bezier points setting specified in Table 3.4. Consider n=11, the minimum relative interval available for box constrained approach is 0.28 with respect to the even–spacing interval, or 0.028 with respect to the overall segment.

A comparison of two extreme cases for both the box constrained approach and the line constrained approach is shown in Figure 3.30. Obviously,

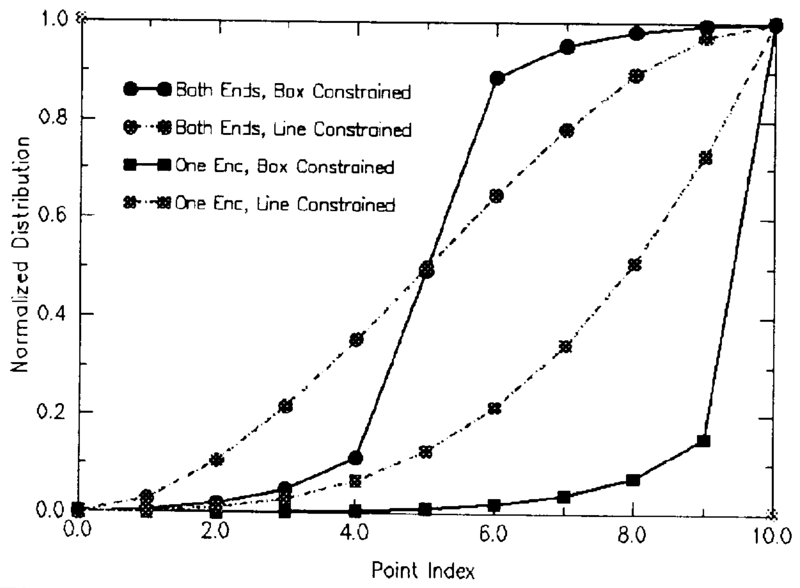


Figure 3.30 Comparison of Box Constrained Approach And Line Constrained Approach

using the line constrained approach allows a more moderate increase of grid spacing. Such comparison is also demonstrated on a straight line in Figure 3.31. Devoid of points due to the application of box constrained approach is not a favorable feature.

Grid Generation Systems

Methodologies for structured grid generation can be classified into direct methods, such as algebraic methods, or indirect methods, such as solving a set of PDEs. Elliptic systems are the most common PDE approach for generating structured grids. In this work, both algebraic and elliptic grid generation systems are available to the user for generating a grid. The algebraic system is used to generate the initial grid, and the elliptic system is used to refine

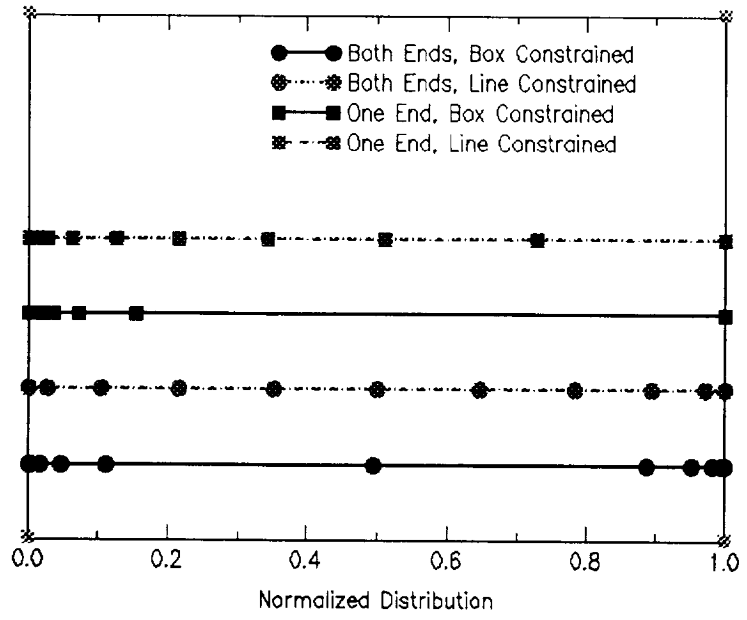


Figure 3.31 Comparison of Box Constrained Approach And Line Constrained Approach

the initial grid for better grid quality. This section discusses the applications of these systems to obtain complicated grid for general turbomachinery configurations.

Algebraic Grid Generation System

Interior grid points $\vec{p}(\xi,\eta,\zeta)$ of a computational domain can be constructed by interpolation from the boundaries. Such coordinate generation procedures are referred to as algebraic generation systems. These are many interpolation methods, some of them consider only the function values, such as the Lagrange interpolation, some of them consider the first derivatives as well, such as the Hermite method. The simplest algebraic interpolation method is the Lagrange interpolation. It can be expressed as, in one–dimension general form

$$\vec{p}(\xi) = \sum_{a=1}^{n} \phi_a \left(\frac{\xi}{I}\right) \vec{p}_a \tag{3.36}$$

where n is the degree of the polynomial, $(I+1)$ is the number of points in ξ–direction, $0 \leq \xi \leq I$, and ϕ_a are the interpolation functions or blending functions. If the lagrange interpolation polynomials are choosed, then

$$\phi_a\left(\frac{\xi}{I}\right) = \prod_{b=1}^{n} \frac{\xi - \xi_b}{\xi_a - \xi_b}, \quad b \neq a \tag{3.37}$$

The linear interpolation form of (3.38), i.e. n=2, can be expanded to be

$$\vec{p}(\xi) = \left(1 - \frac{\xi}{I}\right) \vec{p}_1 + \left(\frac{\xi}{I}\right) \vec{p}_2 \tag{3.38}$$

where

$$\begin{aligned} \phi_1 &= 1 - \frac{\xi}{I} = 1 - \mu \\ \phi_2 &= \frac{\xi}{I} = \mu \end{aligned} \tag{3.39}$$

<u>Transfinite Interpolation</u>

The most common algebraic interpolation method in grid generation is the transfinite interpolation (TFI). A general three–dimensional TFI can be expressed in projector form as

$$\begin{aligned} F_{\xi^1}(\vec{p}) \oplus F_{\xi^2}(\vec{p}) \oplus F_{\xi^3}(\vec{p}) &= F_{\xi^1}(\vec{p}) + F_{\xi^2}(\vec{p}) + F_{\xi^3}(\vec{p}) \\ &- F_{\xi^1}[F_{\xi^2}(\vec{p})] - F_{\xi^3}[F_{\xi^3}(\vec{p})] - F_{\xi^1}[F_{\xi^3}(\vec{p})] + F_{\xi^1}[F_{\xi^2}[F_{\xi^3}(\vec{p})]] \end{aligned} \tag{3.40}$$

where $F_{\xi^i}(\vec{p})$ is an one–dimensional interpolation function in the ξ^i–direction. Thompson[58] expressed (3.40) in a general form. If one considers (3.40) using the Lagrange interpolation only, then

$$F_{\xi^m}(\vec{p}) = \sum_{a=1}^{n} \phi_a^m (\mu^m) \vec{p}_a^{\,m} \tag{3.41}$$

$$F_{\xi^m}\left[F_{\xi^h}(\vec{p})\right] = \sum_{a=1}^{n} \sum_{b=1}^{n} \phi_a^m (\mu^m) \phi_b^h (\mu^h) \vec{p}_{ab}^{\,mh} \tag{3.42}$$

$$F_{\xi^1}\left[F_{\xi^2}\left[F_{\xi^3}(\vec{p})\right]\right] = \sum_{a=1}^{n}\sum_{b=1}^{n}\sum_{c=1}^{n} \phi_a^1(\mu^1)\,\phi_b^2(\mu^2)\,\phi_c^3(\mu^3)\,\vec{p}_{abc} \quad (3.43)$$

where

$$\mu^1 = \frac{i}{I}, \quad \mu^2 = \frac{j}{J}, \quad \mu^3 = \frac{k}{K}, \quad (3.44)$$

$(I+1)$, $(J+1)$, $(K+1)$ *are the numbers of points in* $\xi-$, $\eta-$, and $\zeta-$ *directions, respectively*

i,j,k *are the point indices,* $i = 0,I;\ \ j = 0,J;\ \ k = 0,K;$

$$\phi_1^m(\mu^m) = (1-\mu^m) \quad (3.45a)$$

$$\phi_2^m(\mu^m) = \mu^m \quad (3.45b)$$

$$\vec{p}_a^{\,1} = \vec{p}(\xi_a,\eta,\zeta), \quad \vec{p}_a^{\,2} = \vec{p}(\xi,\eta_a,\zeta), \quad \vec{p}_a^{\,3} = \vec{p}(\xi,\eta,\zeta_a) \quad (3.46a)$$

$$\vec{p}_{ab}^{\,12} = \vec{p}(\xi_a,\eta_b,\zeta), \quad \vec{p}_{ab}^{\,13} = \vec{p}(\xi_a,\eta,\zeta_b), \quad \vec{p}_{ab}^{\,23} = \vec{p}(\xi,\eta_a,\zeta_b) \quad (3.46b)$$

and

$$\vec{p}_{abc} = \vec{p}(\xi_a,\eta_b,\zeta_c) \quad (3.47)$$

Unfortunately, TFI method is not a robust scheme since it does not take into account any geometry information except the coordinates of the boundary points. This defect can be improved by including the arc length information of the boundaries and use them as interpolant array $\mathcal{T}=\{(\mathcal{T}^1,\mathcal{T}^2,\mathcal{T}^3)\}$ instead of the uniform interpolant array $\mu=\{(\mu^1,\mu^2,\mu^3)=(i/I, j/J, k/K)\}$. In order to simplify the derivation for the interpolant array $\mathcal{T}$, consider only a two–dimensional surface $\vec{p}(u,v) = (x^1(u,v)\ ,\ x^2(u,v)\ ,\ x^3(u,v)\ )$ with $(I+1)$x$(J+1)$ points. The ξ–component in the interpolant array $\mathcal{T}$ can be defined as

$$\mathcal{T}_i^1 = \frac{\displaystyle\int_{u_0}^{u_i} \vec{p}(x^1(u,v_m),\ x^2(u,v_m),\ x^3(u,v_m))\,du}{\displaystyle\int_{u_0}^{u_I} \vec{p}(x^1(u,v_m),\ x^2(u,v_m),\ x^3(u,v_m))\,du} \quad (3.48)$$

where

$$\mathcal{T}_0^1 = 0.0 \ ,$$
$$m = 0 \text{ and } J$$

The η – component of $\mathcal{T}$ can be obtained in a similar fashion. Therefore, (3.45a) and (3.45b) can be re–written as

$$\begin{aligned} \phi_1^1 &= 1 - \mathcal{T}^1 \\ \phi_2^1 &= \mathcal{T}^1 \end{aligned} \qquad (3.49)$$

Three–dimensional derivation can be extended in a similar fashion.

An algebraic interpolation method using the blending functions defined in the fashion of (3.49) is referred to as weighted transfinite interpolation (WTFI). It can be extended to three–dimension easily. This procedure, though is more expensive in terms of computer time and storage, is a better and more robust way for generating grid with interpolation method, since it takes into account the distribution on the boundaries. In this work, WTFI approach is

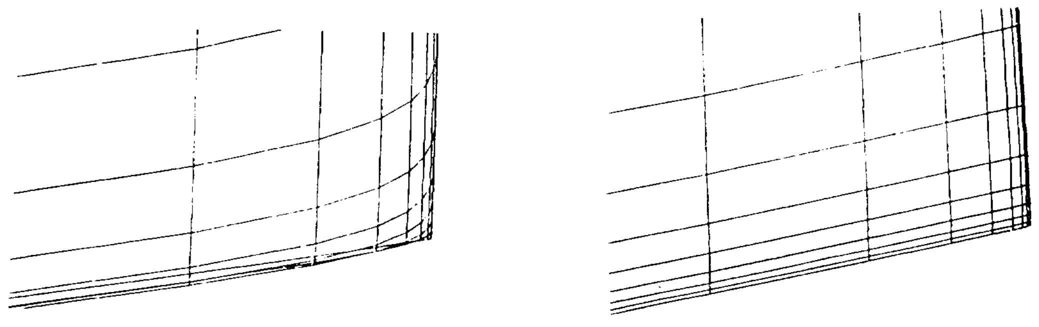
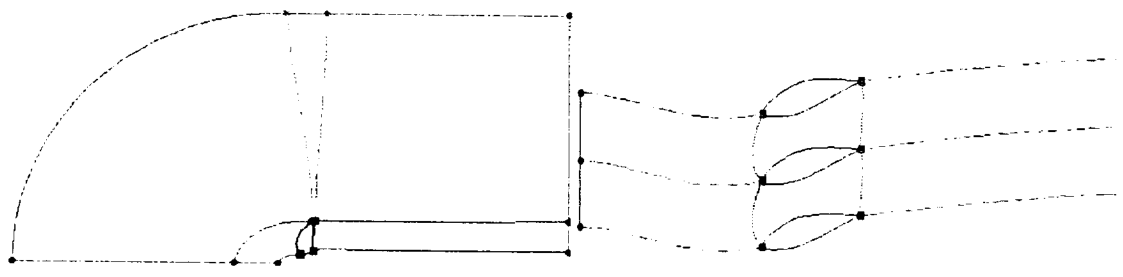

(a) TFI (b) WTFI

Figure 3.32 Comparison of Interpolation Results

used to generate both surface grid and volume grid. The advantage of WTFI over TFI is more significant when generating viscous grid for complicated geometries. A comparison of the TFI approach and WTFI approach is shown in Figures 3.32(a) and 3.32(b). The grid is part of the ξ–constant surface in the passage. The grid points is clustered toward the blade surface. The result using TFI interpolation shows overlapping grid, while the result using WTFI interpolation shows satisfactory grid behavior

Automatic Sub–Block Decomposition

A volume grid can be obtained by interpolation based on the entire six block faces. However, in some cases, this is not a desired approach. Within a grid block, there can exist difficult regions such as the tip clearance. These regions can create numerical difficulty for WTFI algorithm in obtaining valid grid. One way of enhancing the robustness of WTFI algorithm is to decompose the grid block into smaller units or sub–blocks.


(a) Meridional Sub–Blocking (b) Cascade Sub–Blocking
Figure 3.33 Sub–Block System in 2D

Sub–block decomposition, though provides more robustness to WTFI algorithm, requires the generation of more interior boundaries and surfaces, and hence more labor. Fortunately, the algorithm of TIGER is developed to use each index as a breaking node, and sub–divide the domain automatically. Consequently, a user does not have to manually decompose the domain and create the inner surfaces. If a user, however, prefer to decompose the sub–block differently, it can be done in the manipulation step after the initial grid has been created. Sub–block decomposition in both meridional and cascade frames for the Hamilton–Standard's SR–7 are shown in Figure 3.33. An example sub–block system of an engine configuration is demonstrated in Figure 3.34. Each sub–block has been manually translated a small offset distance to allow better view.

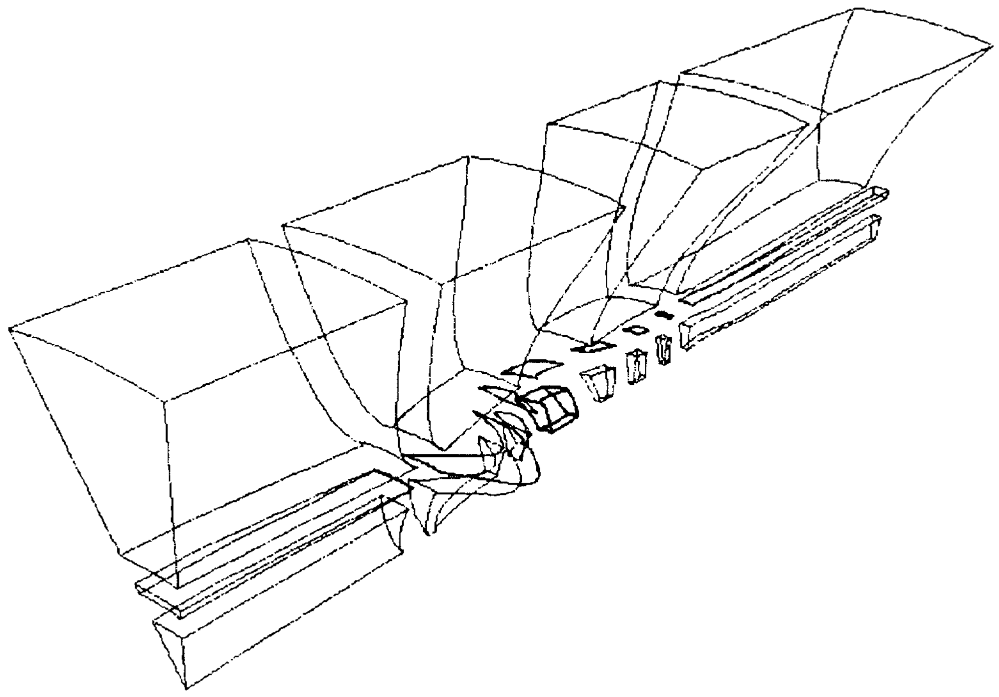


Figure 3.34 Sub–Block System for an Engine Configuration

Generating grids with an algebraic method is faster than the other methods, such as PDE methods or variational methods. However, its main disadvantage is the non–smooth result. That is, the discontinuity on the boundaries will propagate into the field. Therefore, TIGER also provides the elliptic grid generation system to smooth the grid.

Elliptic Grid Generation System

PDE methods for grid generation include elliptic, hyperbolic, and parabolic systems. Among these approaches, elliptic system is the most commonly used approach. It is used in many codes[8–29]. Unlike the result of an algebraic method, the discontinuities on the boundaries will not be propagated into the field if a grid is generated by an elliptic system. For a homogeneous elliptic system, or the Laplace equation, it generates the smoothest grid possible. Since the Laplace equation is a harmonic function, it has the extremum principle property: the extrema of solutions cannot occur within the field but on the boundary. This implies, in terms of grid generation, that the extrema of the curvilinear coordinates cannot occur in the interior of the physical region, and hence ensures the non–overlapping grid lines. However, the equidistribution of grid points in the field by the nature of the Laplace equation is generally not a desirable feature for point distribution. User–specified clustering of points near the region of interest is more desirable. Therefore, Poisson–type grid generation system is more appropriate for pratical applications.

The Poisson–type elliptic grid generation system in the computational domain is [Ref.]

$$\sum_{i=1}^{3}\sum_{j=1}^{3} g^{ij}\,\vec{p}_{\xi^i\xi^j} + \sum_{j=1}^{3} g^{jj}\,\mathcal{P}_j\,\vec{p}_{\xi^j} = 0 \qquad (3.50)$$

where $\mathcal{P}$ is the control function triplet, $\vec{p}$ is the postion vector of a grid point, ξ^i $(i = 1,2,3)$ are the curvilinear coordinates, and $g^{ij} = \underline{\nabla}\xi^i \bullet \underline{\nabla}\xi^j$ are the elements of the contravariant metric tensor. However, these contravariant components can be replaced by its covariant counter–parts $g_{ij} = \vec{p}_{\xi^i} \bullet \vec{p}_{\xi^j}$ which can be calculated directly:

$$g^{ij} = (g_{km}g_{ln} - g_{kn}g_{lm}) \;/\; g \qquad (3.51)$$
$$(i,k,l)\ \textit{cyclic},\quad (j,m,n)\ \textit{cyclic}$$

where g is the square of the Jacobian and is given by

$$\sqrt{g} = \sqrt{\det |g_{i,j}|} = \vec{p}_{\xi^1} \bullet (\vec{p}_{\xi^2} \times \vec{p}_{\xi^3}) \tag{3.52}$$

The control function triplet $\mathcal{P}$ in (3.50) can be fashioned to control the spacing and orientation of the grid lines. However, Poisson equation is not a harmonic function, and therefore it does not have the extremum principle property. In other words, inproper evaluation of the control functions can result in a grid with negative Jacobians. Therefore, the evaluation of $\mathcal{P}$ has been investigated by many researchers, such as Thompson et. al.[58–60], Soni[61], Sorenson[16–19], and Thomas and Middlecoff[62]. In this work, Thomas–Middlecoff–type and Thompson/Soni–type control functions are available. In two–dimensional form, they are:

- Thomas–Middlecoff[62]

$$\mathcal{P}_i = -\frac{\vec{p}_{\xi^i\xi^i} \bullet \vec{p}_{\xi^i}}{\vec{p}_{\xi^i} \bullet \vec{p}_{\xi^i}} \qquad i = 1,2 \tag{3.53}$$

- Thompson[58–60]/Soni[61]

$$\mathcal{P}_i = -\frac{\vec{p}_{\xi^i\xi^i} \bullet \vec{p}_{\xi^i}}{\vec{p}_{\xi^i} \bullet \vec{p}_{\xi^i}} - \frac{\vec{p}_{\xi^j\xi^j} \bullet \vec{p}_{\xi^i}}{\vec{p}_{\xi^j} \bullet \vec{p}_{\xi^j}} \qquad i = 1,2; \;\; j = 1,2; \;\; i \neq j \tag{3.54}$$

One can use finite difference approach to obtain the values for $\vec{p}_{\xi^i}$ and $\vec{p}_{\xi^i\xi^i}$ along a constant ξ^j boundary (i≠j) for both (3.53) and (3.54). In order to evaluate $\vec{p}_{\xi^j\xi^j}$ in (3.54), however, Thompson et. al. uses iterative approach to update $\vec{p}_{\xi^j\xi^j}$. Soni uses the grid spacing approach. In this work, Soni's approach is adopted.

Ripple Diffusion on Control Functions For Cascade Surface Applications

The aforestated control function approaches, however, are derived for general configurations. When applied to the cascade surface of a turbomachine, they all do fairly well on the surface. However, the concentration of grid points in the middle of passage near blade leading edge and trailing edge along the flow direction is not a desirable feature. Therefore, this work inserts a "ripple diffusion" weighting procedure to multiply a weight factor vector Ω with its components ranging between 0 and 1 to the control functions prior the actual execution of the elliptic solver.

Let $\mathcal{P}$ be the control function vector obtained from the previous section. The weight factor in η–direction is set to be 1 throughout the grid. However, the weight factor in ξ–direction will maintain the value of 1.0 along the η=0 and η=n curvilinear coordinate line, and decays its value linearly toward the middle of the passage, where at the middle of the passage, the value decays to 0.0. That is

$$\mathcal{P}_i = \mathcal{P}_i \times \Omega_j^i \qquad i = 1,2 \tag{3.55}$$

where

$$\begin{aligned} \Omega_j^1 &= [n/2]^{-1} \times j \\ \Omega_0^1 &= \Omega_n^1 = 1.0 \\ \Omega_{n/2}^1 &= 0.0 \\ \Omega_j^2 &= 1.0 \qquad j = 1,n \end{aligned} \tag{3.56}$$

Consequently, the altered control function $\mathcal{P}$ will distribute the grid points in η–direction as usual, but it will tend to equidistribute the grid points in ξ–direction. The initial algebraic grid is shown in Figures 3.35. Presented in Figure 3.36 is the elliptic grid with regular $\mathcal{P}$ obtained from (3.53). The elliptic grid with "ripple diffusion" on the same control function vector $\mathcal{P}$ is demonstrated in Figure 3.37. The contour maps of the control functions after weighting is shown in Figure 3.38.

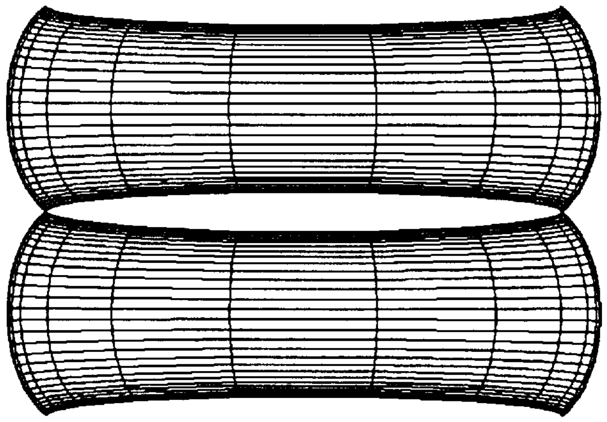


Figure 3.35 Initial Algebraic Grid

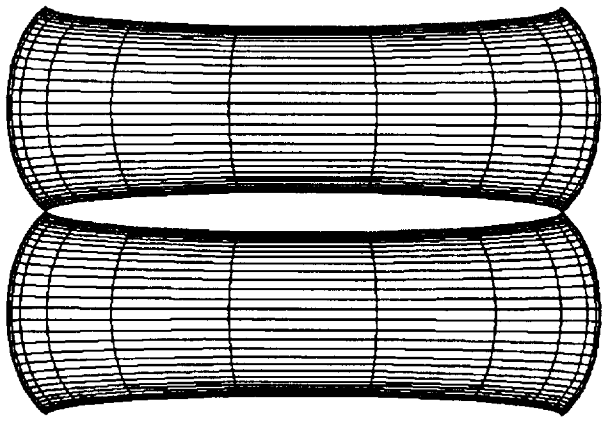


Figure 3.36 Elliptic Grid Before Ripple Diffusion

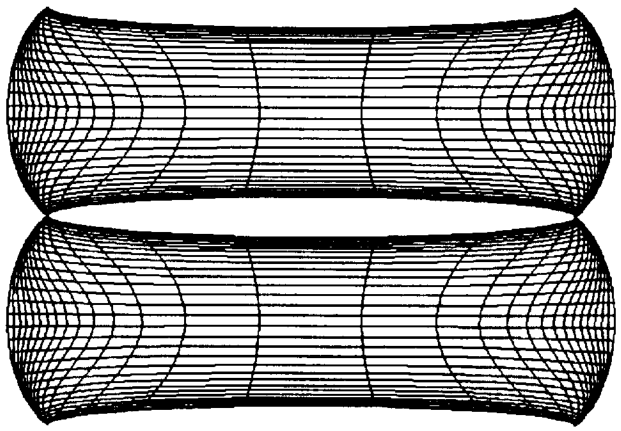


Figure 3.37 Elliptic Grid After Ripple Diffusion

(a) Without Diffusion (b) With Diffusion

Figure 3.38 Control Function Contours Comparison

<u>Cascade Surface Generation With Elliptic Smoothing and NURBS Spline–Fitting</u>

With the aforstated techniques described in §3.2.3, §3.2.4, and §3.4.2, the cascade surfaces are generated and optimized. A three–dimensional cascade surface lying on a body of revolution, e.g. the hub, is tranformed into two–dimensional parametric space (m',σ), as in the example shown in Figure 3.39 for the hub surface of an impeller configuration. However, for a blade with low stagger angle and large blade LE circle radius, it is a very difficult region to maintain non–crossing grid with WTFI approach. Examples of this problem is demonstrated in Figures 3.40(a). A common practice to resolve this problem is to apply elliptic system with proper control functions, such as the ones in (3.53) and (3.54). Results of elliptic smoothing with Thomas–Middlecoff–type control function are presented in Figures 3.40(b). In this figure, it is clearly demonstrated that the elliptic smoothing eliminates the problem region. However, as demonstrated in the close–up regions, the grid spacing is being pulled off the concave region and being pushed closer to the convex

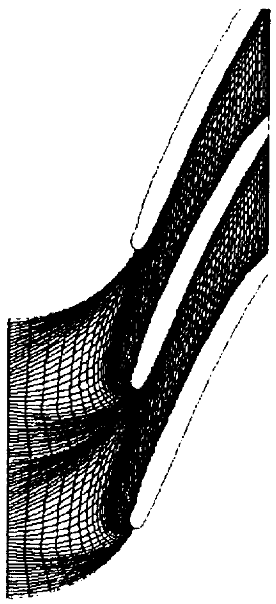


Figure 3.39 Transformed 2D Cascade Surface

boundary, creating a sudden change in grid spacing, as shown in Figure 3.40(b).

In order to improve the grid quality, one more step is taken in the algorithm: the surface with elliptic smoothing is spline–fitted with NURBS surface formulation as derived in §3.2.3. Elliptic smoothing is necessary because the NURBS formulation requires a non–overlapping initial grid to begin with. As demonstrated in Figure 3.40(c), the grid quality has been improved. It is shown in Figures 3.41 that for this particular case, the grid quality near the blade surface is improved as well. Another comparison for a different geometry is shown in Figures 3.42 to validate this approach.

Quality Checking

The basic quality requirement for a structured grid is that the grid Jacobian $\sqrt{g} = \vec{p}_{\xi^i} \bullet (\vec{p}_{\xi^j} \times \vec{p}_{\xi^k})$, i.e. the volume of the grid cell, must not be negative. The method for calculation of a general hexahedron developed by Davies

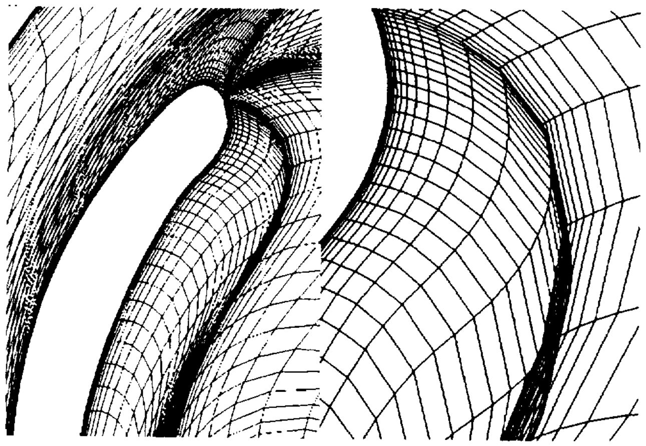


Figure 3.40a WTFI Approach

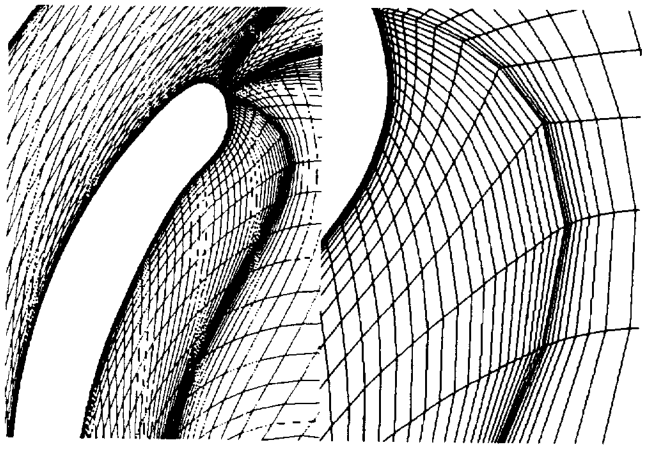


Figure 3.40b Elliptic Approach

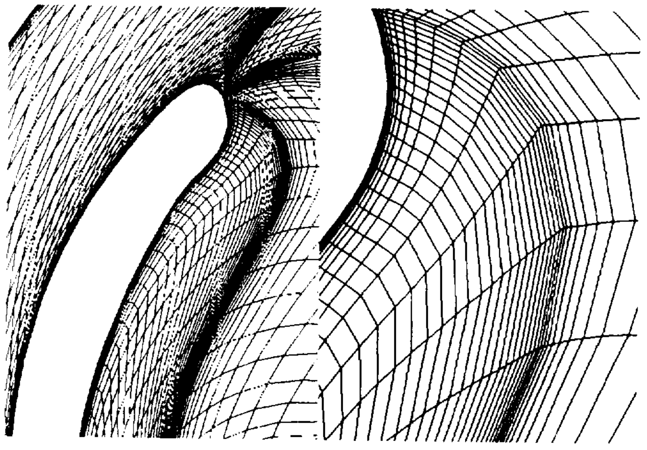


Figure 3.40c Elliptic + NURBS

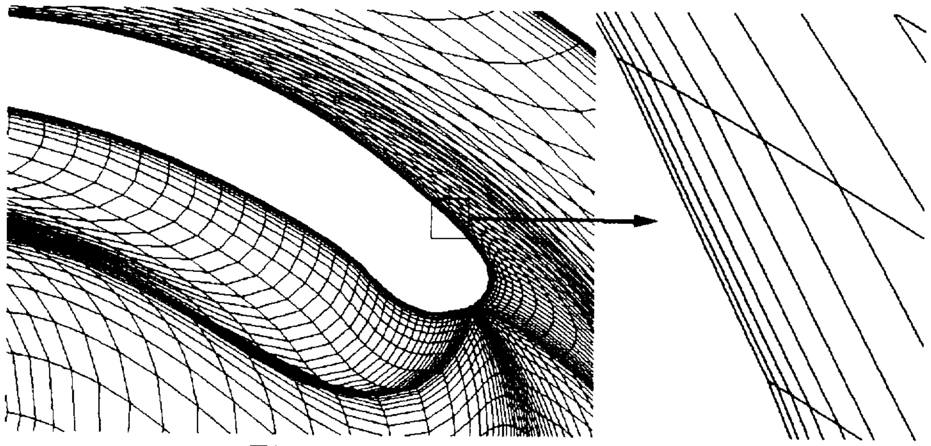


Figure 3.41a WTFI Approach

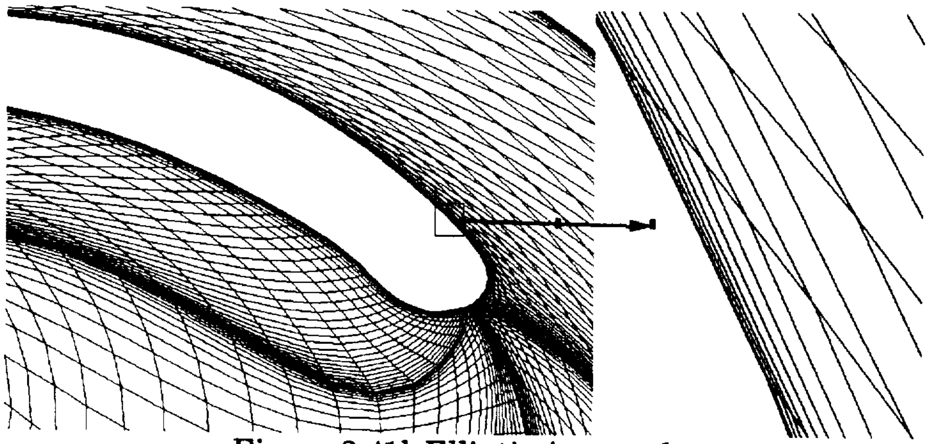


Figure 3.41b Elliptic Approach

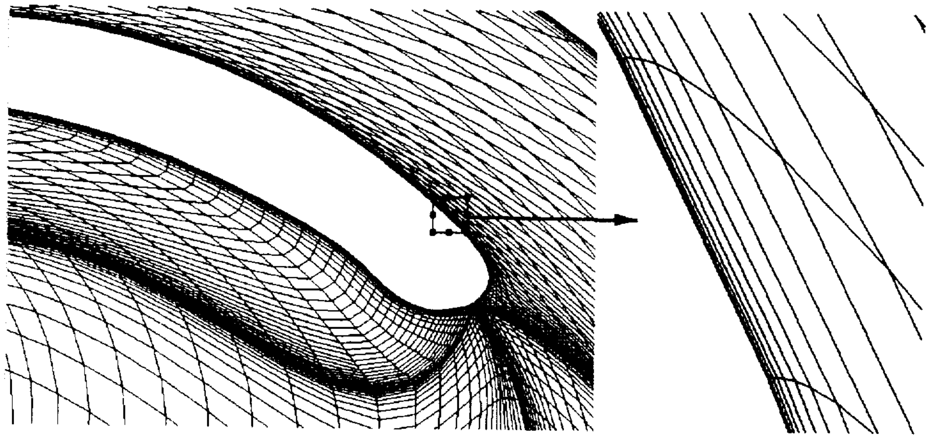


Figure 3.41c Elliptic + NURBS

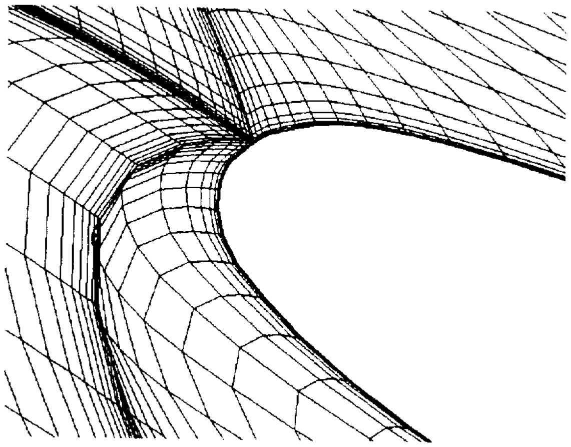


Figure 3.42a WTFI Approach

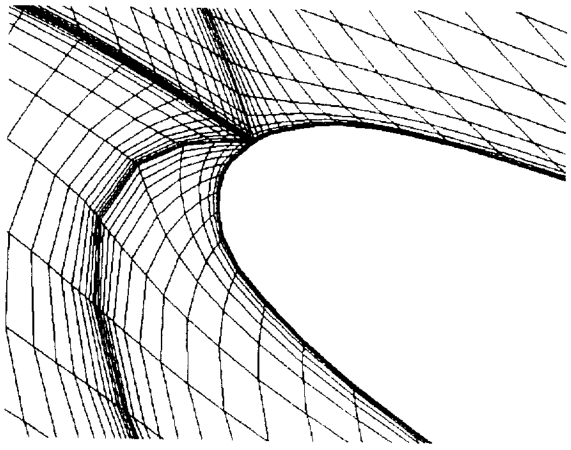


Figure 3.42b Elliptic Approach

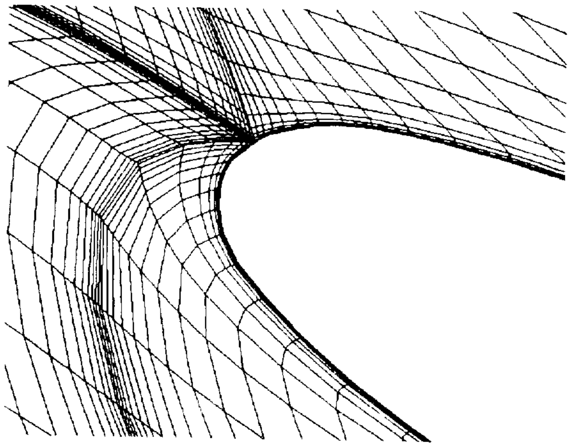


Figure 3.42c Elliptic + NURBS

and Salmond[63] is adopted in this system to perform the quality check. If one considers a pyramid RABCD as shown in Figure 3.43(a), the vector $\vec{RQ}$ can be obtained by using the tensor product interpolation. Using the projector expression, one can write $\vec{RQ} = P_u P_v$, where u and v are the surface parametric coordinates, $0 \leq u \leq 1$, and $0 \leq v \leq 1$. Therefore,

$$P_u = \vec{RA} + v(\vec{AD}) = \vec{RA} + v[\vec{RD} - \vec{RA}] \tag{3.58}$$

$$P_v = \vec{RA} + u(\vec{AB}) = \vec{RA} + u[\vec{RB} - \vec{RA}] \tag{3.59}$$

therefore

$$\begin{aligned} P_u P_v &= \vec{RA} + u[-\vec{RA} + \vec{RB}] + v[-\vec{RA} + \vec{RD}] + \\ &\quad uv[\vec{RA} - \vec{RD} - \vec{RB} + \vec{RC}] \\ &= \vec{RA} + u\vec{AB} + v\vec{AD} + uv[\vec{AC} - \vec{AD} - \vec{AB}] \end{aligned} \tag{3.60}$$

The surface area ABCD can be obtained by

$$\begin{aligned} \vec{n}_p dS_p &= \left\{ [\frac{\partial(\vec{RQ})}{\partial u} \times \frac{\partial(\vec{RQ})}{\partial v}] / |\vec{RQ}| \right\} du\ dv \\ &= [\vec{AB} + v(\vec{AC} - \vec{AB} - \vec{AD})] \times [\vec{AD} + u(\vec{AC} - \vec{AB} - \vec{AD})] du\ dv \end{aligned} \tag{3.61}$$

The volume of the pyramid V_{RABCD} is therefore

$$V_{RABCD} = \frac{1}{3} \int_0^1 \int_0^1 \vec{RQ} \bullet \vec{n}_p dS_p$$

$$\begin{aligned} &= \frac{1}{3} \int_0^1 \int_0^1 \Big\{ [\vec{RA} \bullet (\vec{AB} \times \vec{AD})] + \\ &\quad u[\vec{RA} \bullet (\vec{AB} \times \vec{DC}) + \vec{AB} \bullet (\vec{AB} \times \vec{AD})] + \\ &\quad v[\vec{AD} \bullet (\vec{BC} \times \vec{AD}) + \vec{AD} \bullet (\vec{AB} \times \vec{AD})] + \\ &\quad u^2[\vec{AB} \bullet (\vec{AB} \times \vec{DC})] + v^2[\vec{AD} \bullet (\vec{BC} \times \vec{AD})] + \\ &\quad u^2 v[(\vec{AC} - \vec{AB} - \vec{AD}) \bullet (\vec{AB} \times \vec{DC})] + \\ &\quad uv^2[(\vec{AC} - \vec{AB} - \vec{AD}) \bullet (\vec{BC} \times \vec{AD})] + \\ &\quad uv[\vec{AB} \bullet (\vec{BC} \times \vec{AD}) + \vec{AD} \bullet (\vec{AB} \times \vec{DC}) + \end{aligned}$$

$$(\vec{AC} - \vec{AB} - \vec{AD})\bullet(\vec{AB} \times \vec{AD})] \} \, du \, dv$$

$$= \frac{1}{6} \vec{RA}\bullet[\vec{DB} \times \vec{AC}] + \frac{1}{12}\vec{AC}\bullet[\vec{AD} \times \vec{AB}] \qquad (3.62)$$

The total volume of a general hexahedron ABCDEFGH as shown in Figure 3.43(b) is therefore the sum of six pyramids

$$V = V_{RABCD} + V_{RAEFB} + V_{RADHE} + V_{RGFEH} + V_{RGHDC} + V_{RGCBF} \qquad (3.63)$$

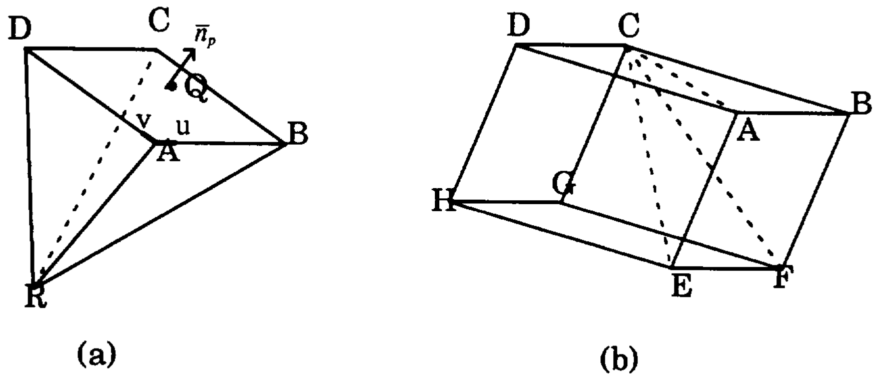


Figure 3.43 Volume Presentation

CHAPTER IV
ALGORITHM DEVELOPMENT

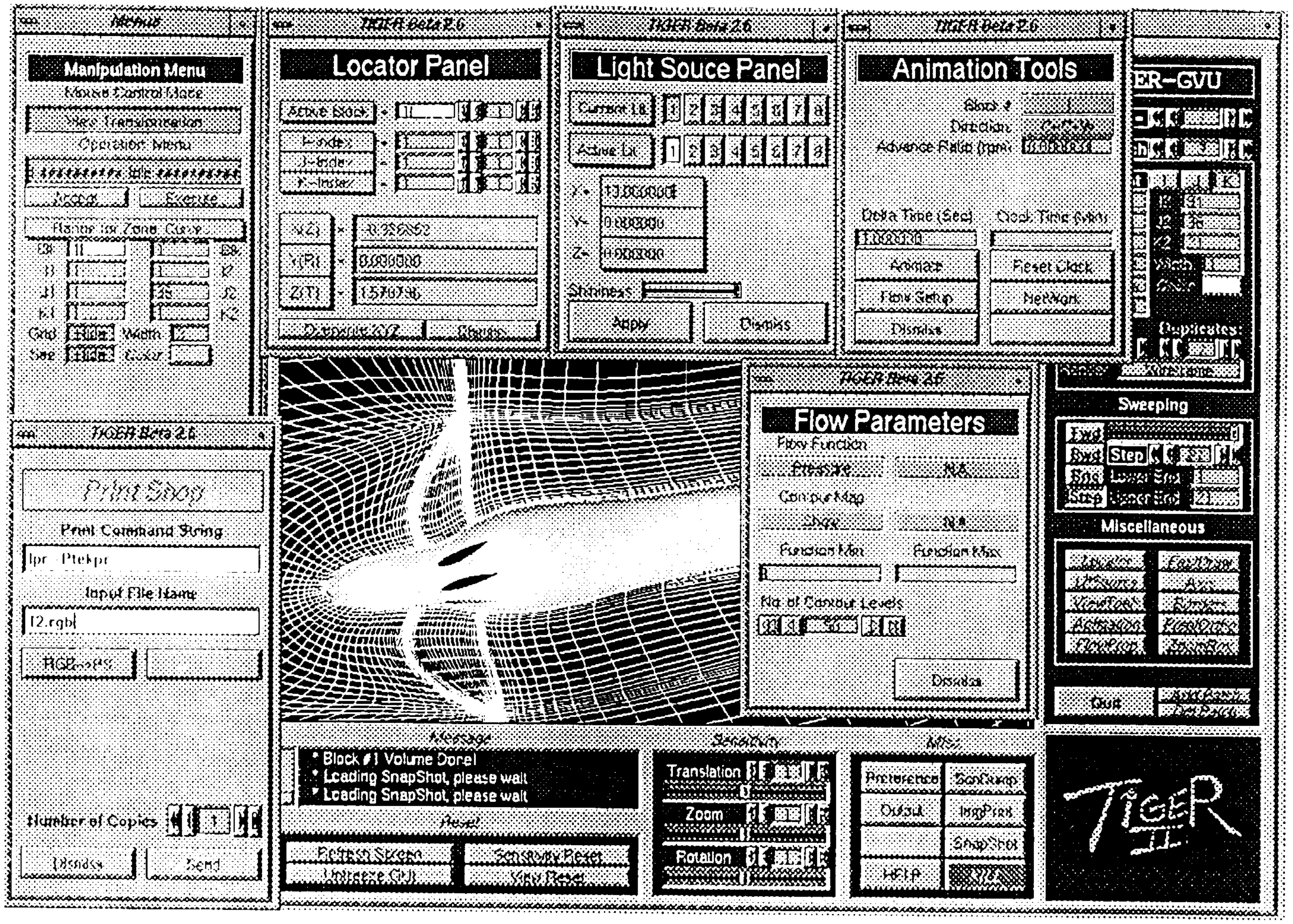
The development of TIGER has evolved from the original grid generation code into a comprehensive simulation system that contains six modules: (*i*) grid generation module, (*ii*) visualization module, (*iii*) network module, (*iv*) simulation module, (*v*) toolbox module, and (*vi*) flow module. Each module may be accessed independently to perform different tasks. The grid generation module allows construction of turbomachinery grids. The visualization module provides a sophisticated scientific visualization for both grid and flow solution. The network module allows data communication in a heterogeneous environment. The simulation module offers the user a near real–time flow simulation capability. Various utility routines have been developed to support the overall task. These utility routines are collected in the toolbox module. The flow module is an open–end module that performs flow field calculation, and can be replaced by other flow solvers with minor modifications. These modules are linked together with a common graphical user interface as shown in Figure 4.1, which is discussed in the following section.

Design of the Graphical User Interface

Graphical user interface (GUI) design has become one of the major issues in the development of contemporary CFD software. User–friendliness is the key to the success of a GUI design while maintaining the functionality of the program. A GUI must be able to provide a user with a friendly environment allows easy access to the functions supported by the code. For general purpose software, it must allow the user to access those functions randomly, while for a customized software, it should provide a sequence of panels that guide the user through the process.

TIGERS' GUI, as shown in Figure 4.1, is developed using FORMS Library[64]. FORMS Library is a user–friendly GUI toolkit software in the public domain with an attractive widget outlook and condensed size. It is a library developed using the Silicon Graphics Inc. (SGI) Graphics Library (GL)[65], and hence is executable only on SGI workstations. FORMS provides a graphical design module which allows the user to interactively design the interface. It has a large variety of commonly–used widgets such as buttons, sliders and input fields.

TIGER's GUI is developed by constructing a global panel window, an image display window, various sub–panels and auxiliary panels. The global panel, where the logo, message browser, image controls widgets, and other global function buttons reside, is the main panel of the whole GUI. The image display window sits in the left–hand upper corner of the global panel and is the window that allows the user to view the grid and the solution. Sub–panels such as the "GVU" panel, pop up on the right hand portion of the global panel as the algorithm requests user inputs. They provide the necessary widgets for a particular group of user inputs. Auxiliary panels, such as the "Animation Tools" and "Flow Property" panels, pop up only upon the users' request for some customization, such as light source, locator, and viewing parameter modification. There is also a help window which pops up upon request. It consists of a large browser window to display the contents of the help files, and several buttons for the user to select topics.



* Block #1 Volume Done!
* Loading SnapShot, please wait
* Loading SnapShot, please wait

lpr -Ptekpr

I2.rgb

Figure 4.1 Graphical User Interface

Index Systems

Two index systems have been developed in TIGER. They are the global index system (GIS) and the local index system (LIS). GIS is an indexing system which considers the whole computational domain as a single block, even though this domain may contain many grid blocks. The indices of critical points, such as the hub nose, blade LE/TE, and block interfaces, are independent of the actual block they are in, but refer to a common computational origin. GIS is used while specifying the index information for blades and other entities such as hub and duct. The advantage of this index system is that it unifies the overall geometry into one global index reference system, which allows the user to have better control of the size of the overall grid, as well as the control of overall point distribution.

Contrary to GIS, LIS is an indexing system in which the indices are block–dependent. In other words, the indices refer to the local curvilinear coordinate origins of each block. LIS exists only in the visualization routines. Since TIGER's visualization module is generalized for multi–blocked grids of any configuration, GIS may not be able to constitute, for certain cases, a single full rectangular computational block by collecting all the blocks. LIS is then a clear choice for TIGER to identify any point in any block.

For example, if the LE ξ–index for the stator in Block–II, as shown in Figure 3.1, is 65 in GIS, and if Block–I has a grid size of 55x22x21, then the stator's LE ξ–index will be 11 in LIS.

Block System

Multi–block capability is necessary so that each grid block will be of suitable size for computer systems to run the flow code with limited access of memory. It is also required for counter–rotating machines or rotating–stationary configurations with an uneven blade count. A user may specify block index ranges in GIS, and execute the algorithm to obtain the meridional surface grid. This surface is then taken apart into patches based on the block index ranges, and rotated individually for a certain angle α to form the other side of the passage. The angle α is determined by Equations (3.1) and (3.2), as discussed in §3.2. A pictorial presentation of a mockup missile block system is shown in Figure 4.2. The physical domain of the missile is decomposed into

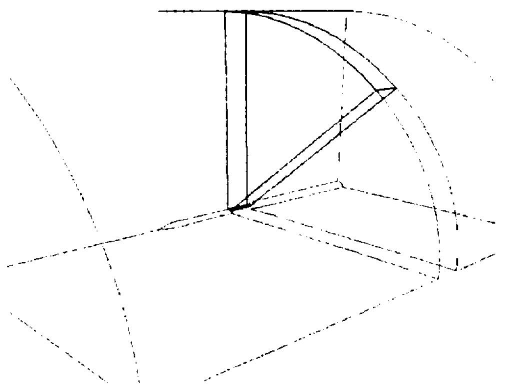


Figure 4.2 Block System for a Missile Mockup with Three Rows of Fin

three axial blocks, which are presented by their block border lines. Other than the block index ranges, the construction of the block system is automatically done by the developed algorithm. This algorithm is developed in such a way that it breaks the domain into several sub–blocks based on the index information, constructing six surface boundaries for each sub–block, and performing interpolation for volume grid. This automatic algorithm eliminates part of the labor required to construct the entire grid.

Grid Generation Module

This is the main module of TIGER system. Its contribution is to allow grid construction about turbomachinery configurations in a time–efficient manner. In order to achieve this goal, the algorithm of this grid generation module is set up in such a way that the user simply has to follow the steps and manipulate the default grid setting when necessary. These default setups are general enough to handle most turbomachinery configurations. There are three main steps for constructing a grid: (*i*)computational modeling, which requires the user to input proper parameters to identify the topology; (*ii*) geometry modeling, which includes the design of generatrices, blade surface construction, and surface mapping techniques; and (*iii*) grid generation, which include the manipulation of two–dimensional framework, and three–dimensional grid manipulation. The details of these steps are discussed in the following sections.

Parameter Input

The execution of the grid generation module requires the user to provide certain necessary grid generation parameters. These parameters include (*i*)the grid size in ξ, η, and ζ directions; (*ii*)indices for critical locations, such as the blade LE/TE and tip; (*iii*)computational domain transformation topologies, such as C–type or H–type, as discussed in §3.1.1; (*iv*)number of blade rows, and number of blades for each row which decides the circumferential domain expansion; and (*v*)number of blocks and their index ranges, which are used in the automatic block decomposition algorithm. These parameters are crucial for the algorithm to establish appropriate default settings for grid construction. There are two modes available for the user to input this information, they are "interactive mode" and "history mode". When the system is in interactive mode, a user has to input each of the parameters on a sequence of sub–panels using mouse and keyboard. If history mode is activated, the data contained in a history file will be duplicated to these sub–panels on the screen, and a user may change any of the parameters before acceptance. These sub–panels, however, may be skipped when in history mode.

The algorithm developed to construct the default grid contributes significantly to reducing the labor required to construct a turbomachinery grid. The outer surface for external flow configurations, for example, is usually a surface of revolution by a straight line or a composite curve of an elliptic arc and a straight line, as illustrated in Figures 3.1 and 3.2. This particular construction is performed by the automatic algorithm developed in this module. The upstream, downstream, and radial extension factors are used to determine the size of the domain.

These parameters, either specified interactively by a user or read in from a history file, will be output to a backup history file "T2bkup.HSTRY". This file may be used as a history file for the next execution. A sample of this journal file for an impeller is listed in Appendix B.

Generatrix Setup

A generatrix is the curve that rotates about the axis of revolution and forms the surface of revolution, as shown in Figure 4.3. It is analogous to an

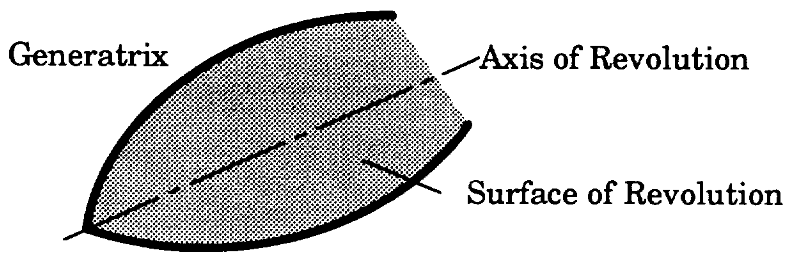


Figure 4.3 Generatrix and Axis of Revolution

η=constant grid line, which rotates about the axis of rotation to form a surface of revolution, such as the hub and the nacelle, as shown in Figure 4.4. All the geometry data, including the hub, the shroud and all the blades, should be presented in one file. The algorithm is designed to read all the geometry data from an external file. Based on their index information, these geometry entities are arranged to establish their relationship for this interactive step. During this step, only those generatrices that run through critical locations such as blade root/tip and shroud/nacelle are required for manipulation. These lines are defined on the meridional plane, i.e. (x,r) plane. The initial generatrices are set up in the straight line fashion. A user may use the mouse and the buttons on GUI to manipulate these generatrices in the Bezier curve fashion to improve grid line behavior. These generatrices, once defined and accepted, will not change in the following steps. They serves as borders of the sub–patches for TFI.

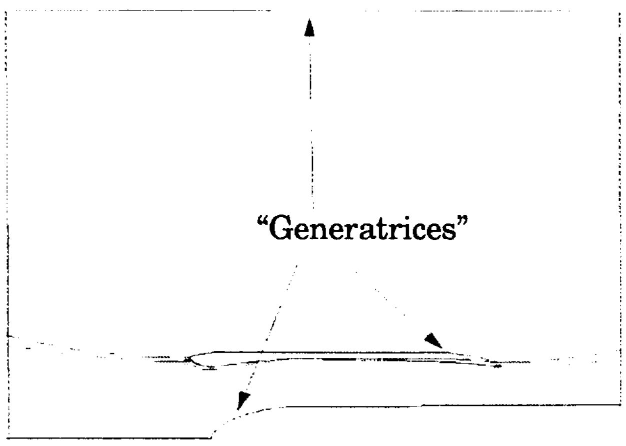


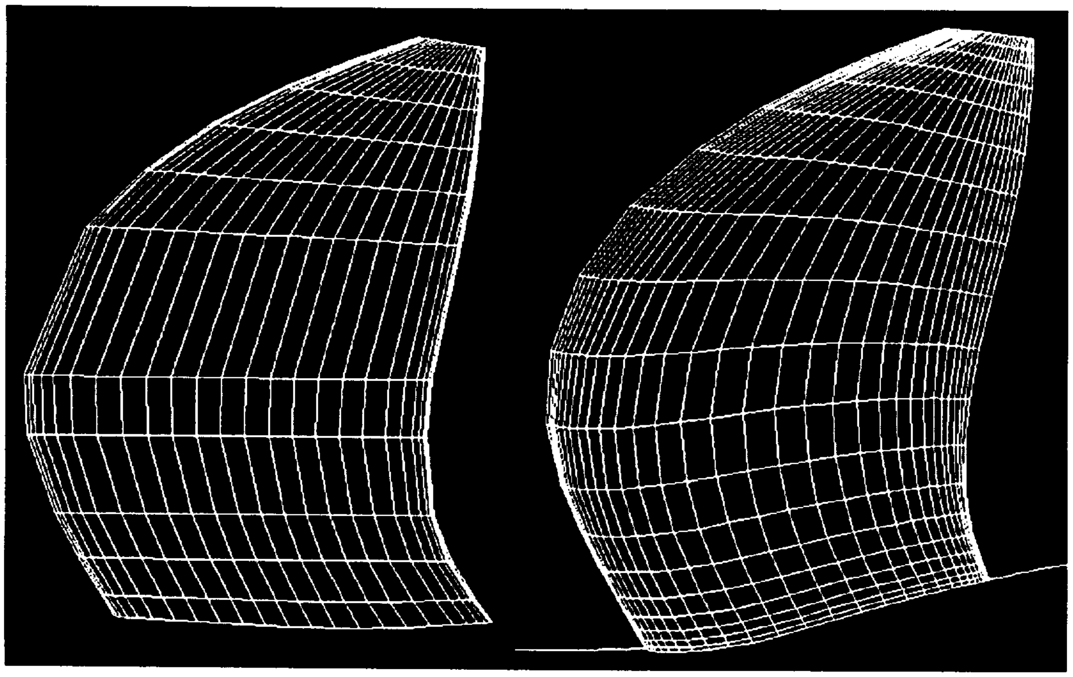
Figure 4.4 Generatrix Design

A journaling scheme is developed for this interactive step. Each event queued by pushing a button or moving the mouse is recorded in a sequential manner. The movement of the cursor in both vertical and horizontal directions are traced in terms of the screen coordinates (pixel). This sequential information will be re–played to construct the generatrices exactly. A sample file for generating a missile mockup is listed in Appendix B.

Blade Surface Construction

Once the user–defined generatrices are accepted, the algorithm is designed to construct the blade surfaces. This includes the trimming of the blade surfaces with these generatrices and the blade surface spline–fitting using NURBS formulations. Based on η–index information, the algorithm will decide if intersection is necessary for the blade and a certain generatrix. The intersection is performed between each blade surface with the surfaces revolved by the generatrices. Newton–Raphson algorithm[12,13] is used for intersection. Once the intersection is performed, each raw data point on the blade will be compared to these intersection curves for trimming, proceeds curve by curve. NURBS surface algorithm is applied to spline these surfaces to the desired point distribution. A user needs to specify the distribution parameters for the pressure side of each blade section, and the algorithm will copy the distribution to the suction side. The constructed surface data is stored on ζ=1 and $\zeta=N_\zeta$ plane in the computational domain as shown in Figures 3.1 to 3.2. They are also stored in a scratch file for later use. Note that this step is an automatic process excepts the control on point distribution. Figure 4.5(a) shows a propeller blade defined in raw data, and Figure 4.5(b) shows the same blade after construction with this automatic algorithm.

Geometry modeling is a tedious step for construction of a grid. The automatic algorithm developed in TIGER substantially reduced the amount of labor involved in the geometry modeling for turbomachinery applications. This automatic algorithm includes the procedures of generating surfaces of revolutions based on the generatrices, intersection of these surfaces with the



(a) Raw Data (b) After Construction
Figure 4.5 Blade Surface Construction

blade surfaces, extraction of intersection curves, and the blade surface trimming and re–construction.

Frame Setup

After the completion of the blade surface construction, all critical points, such as the blade LE/TE, and hub nose, etc., are linked together to form two–dimensional frames. A user will see a full–domain frame on the meridional plane, with each of the geometry entities and critical nodes linked within a frame. A user may toggle between a meridional frame, as shown in Figure 4.6 and a cascade frame, as shown in Figure 4.7, by a single selection on the GUI . Each node is a breaking point that decomposes the entire domain into segments for better control on the grid behavior. This setup is designed to allow the user to provide point distribution information for any segment in a concise environment. When the frame is set to be the meridional plane, the

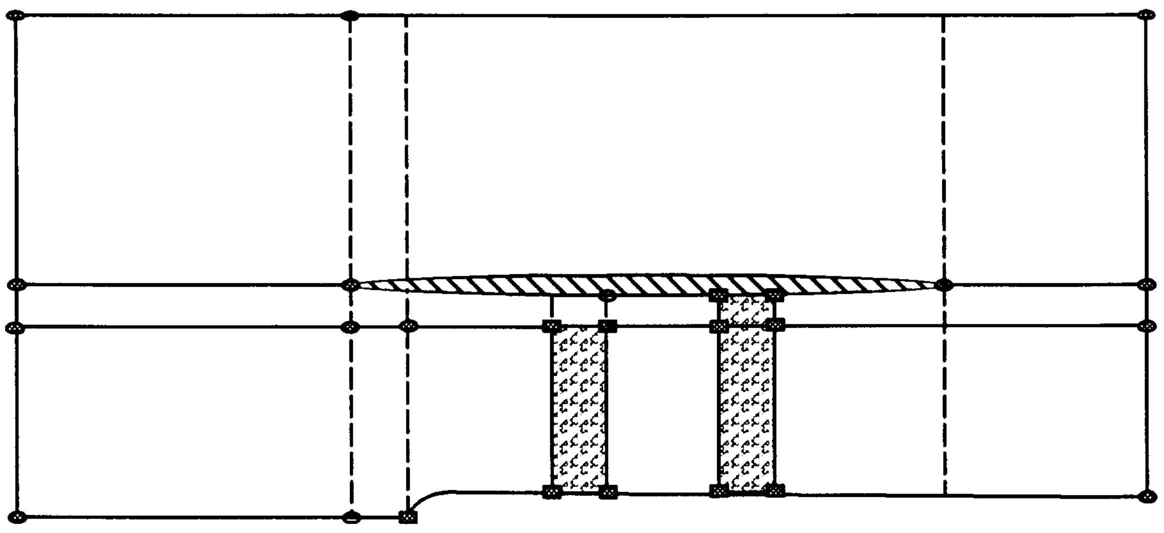


Figure 4.6 Meridional Frame with Break Nodes

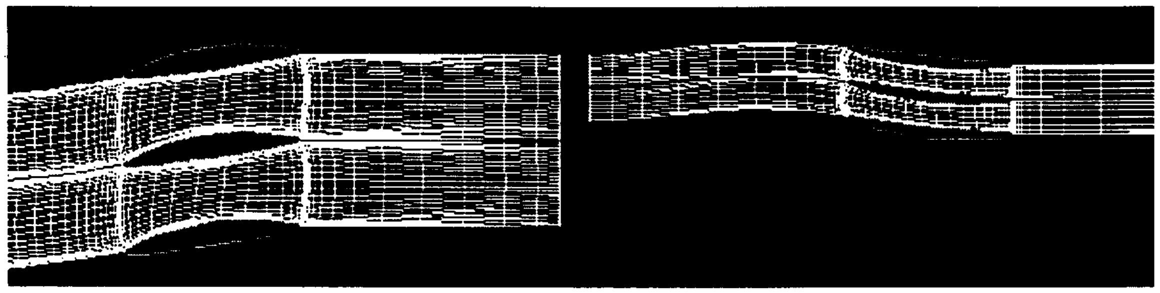


Figure 4.7 Cascade Frame with Mesh Presentation

user may pick a node and slide it along the generatrix to a position that is better for the overall grid. This is done by utilizing NURBS curve formulation described in §3.2. The algorithm spline fits this particular node onto the generatrix by changing its parametric value, and re–generate both sides of the generatrix segment accordingly with NURBS formulations. A geometry node such as the duct lip or the hub nose, however, is not allowed for such manipulation in order to maintain the geometry definition. A radial segment between a lower generatrix and a upper generatrix is allowed to be manipulated in Bezier curve fashion, except for the segments that contain the blade sections or the domain borders. This radial segment is re–linked with a straight line when one of its corresponding nodes is moved. A user needs to re–shape this segment with Bezier curve if necessary. The point distribution information remains intact. Manipulation on the meridional plane allows a user to set up $\boldsymbol{x}$ and $\boldsymbol{r}$ coordinates for meridional surfaces of the passage, i.e. $\zeta=1$ and $\zeta=N_\zeta$ surfaces.

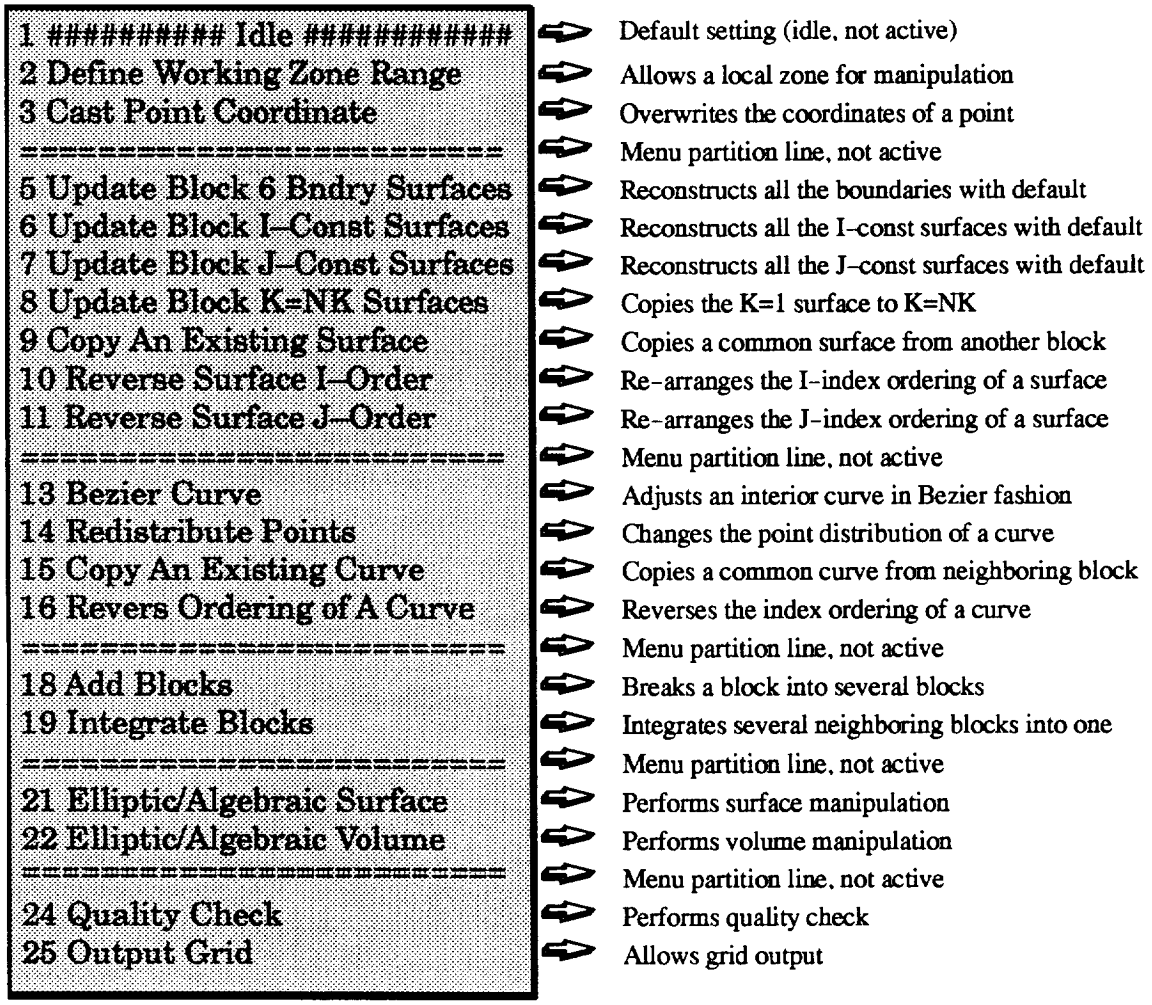
The third coordinate, θ, is obtained from the cascade frame manipulation. When a user toggles the plane from meridional view to cascade view, The program shows the cascade surfaces for all the blocks along the flow path. This cascade view is the result of transformation from a three–dimensional space $(\boldsymbol{x},\boldsymbol{r},\theta)$ to a two dimensional space $(\boldsymbol{m}',\sigma)$ as discussed in §3.2. Each block is separated with a small offset in $\boldsymbol{m}'$ direction, d$\boldsymbol{m}'$, in order to visually distinguish each block, as the gap illustrated in Figure 4.7. However, when mapping from $(\boldsymbol{m}',\sigma)$ space to $(\boldsymbol{x},\boldsymbol{r},\theta)$ space, each d$\boldsymbol{m}'$ will be discarded so that the geometry won't be altered. In this cascade frame, a user may manipulate any of the nodes and segments other than the geometry entities such as the blade LE and TE. Once the manipulation for these two frames is performed, a user may toggle a *"Try It"* button on the GUI to see the result of current setting. With this entry, only six boundary surfaces are generated for each sub–block and are displayed in three–dimensional view. This allows a user to manipulate the image for close examination. A user may go back to the frame setup procedure and modifies the setting further more for better grid. This iteration process allows a user to get the surface grid right without restarting from the beginning.

Journal capability of this step is achieved by using a serious of link lists, which contains the actual coordinates and necessary information such as the indices. These link list structures record those information for each breaking node in the frame, including their Bezier points, distribution informations, etc.. These information are written to a backup file "T2bkup.FRAME". When reading the history, this information will be duplicated to the link lists, and will be adjusted according to the current geometry. Therefore, if there is any minor variation in the geometry, such as a minor variation in blade stagger angle, the algorithm is designed to adjust the coordinate value for each associated breaking nodes. A sample file is listed in Appendix B.

Three–Dimensional Manipulation

Once the user accepts the frame setting, the boundary surfaces for each sub–block are generated, the procedure is also designed to activate volume grid interpolation with WTFI without any user–interaction. The resulting grid is usually satisfactory because of the automatic sub–block decomposition strategy. However, it may not be near–perfect all the time. In some difficult regions such as the cascade surface near the blade leading edge and trailing edge, there may have concentrated or kinky grid lines if the blade setting angle is low and the radii for LE/TE circles are large. This module provides the user a mean to do the final touch–up on the grid. A menu of these functions is shown and explained in Figure 4.8.

The grid is displayed in a three–dimensional perspective view. A user may pick a local zone for manipulation (Figure 4.8, #2). Various functions are provided for such manipulation, such as Bezier curve, point redistribution, surface as well as volume smoothing, etc.. If the surface grid within the zone presents kinky grid lines, a user may use the Bezier curve option (Figure 4.8, #13), as described in §3.3, to manipulate any curve within the zone for smooth or orthogonal grid behavior. This zone will be divided into two individual patch, and WTFI surface generation algorithm will be applied to generate grid



1 ########## Idle ############	Default setting (idle, not active)
2 Define Working Zone Range	Allows a local zone for manipulation
3 Cast Point Coordinate	Overwrites the coordinates of a point
=========================	Menu partition line, not active
5 Update Block 6 Bndry Surfaces	Reconstructs all the boundaries with default
6 Update Block I–Const Surfaces	Reconstructs all the I–const surfaces with default
7 Update Block J–Const Surfaces	Reconstructs all the J–const surfaces with default
8 Update Block K=NK Surfaces	Copies the K=1 surface to K=NK
9 Copy An Existing Surface	Copies a common surface from another block
10 Reverse Surface I–Order	Re-arranges the I-index ordering of a surface
11 Reverse Surface J–Order	Re-arranges the J-index ordering of a surface
=========================	Menu partition line, not active
13 Bezier Curve	Adjusts an interior curve in Bezier fashion
14 Redistribute Points	Changes the point distribution of a curve
15 Copy An Existing Curve	Copies a common curve from neighboring block
16 Revers Ordering of A Curve	Reverses the index ordering of a curve
=========================	Menu partition line, not active
18 Add Blocks	Breaks a block into several blocks
19 Integrate Blocks	Integrates several neighboring blocks into one
=========================	Menu partition line, not active
21 Elliptic/Algebraic Surface	Performs surface manipulation
22 Elliptic/Algebraic Volume	Performs volume manipulation
=========================	Menu partition line, not active
24 Quality Check	Performs quality check
25 Output Grid	Allows grid output

Figure 4.8 Menu Description for Three–Dimensional Manipulation

for these two patches independently. This update procedure is achieved on the real–time base. A user may want to re–distribute points on this curve (Figure 4.8, #14), instead of manipulate it in Bezier curve fashion. This can be achieved in a similar fashion. NURBS curve spline–fit algorithm as discussed in §3.3 is used to re–distribute the points on this particular curve, and WTFI surfaces are generated accordingly. A user may access the interactive point distribution algorithm at this step.

If a volume zone was specified, a user may not interactively adjust the grid inside the zone. However, a user may access the volume manipulation option (Figure 4.8, #22) to re–construct the volume grid. Quality of the grid (Figure 4.8, #24) may be checked in this step with the Davies–Salmond approach described in §3.4. Grid blocks may be output (Figure 4.8, #25) to the designated files with previously defined formats. Unfortunately, the journaling capability for this module has not been established.

GVU Module

GVU module is the visualization module of the TIGER system. Originally designed for visualization of turbomachinery grids, this module has been extended and generalized. It is currently a comprehensive and sophisticated post–processing module which allows visualization for both grid and flow solution of any general configuration with any number of blocks. It allows the visualization of unsteady grid and solution as well. GVU module allows the user to read in grid/solution data from multiple files, each can be in different format and coordinates system, and each can contains any number of blocks. Both Cartesian and cylindrical coordinates are supported. Wireframe and Gouraud shading rendering are both available for visualizing the grid. Maximum of 7 light sources are available. Color–coded wireframe, line contour, solid contour, and vector field rendering are available for solution visualization. Different solution rendering methods demostrated with a solution obtained by Taylor[66] are shown in Figure 4.9. This figure also demonstrates the generality of the GVU module to any configurations. Time–dependent grid and/or solution may be visualized dynamically. An animation entry is available for this module to graphically animate the rotation of each block for a user to visualize the relative motion of each block. This is especially useful for the counter–rotating configurations. A user can also access the network module and simulation module of the TIGER system through this module.

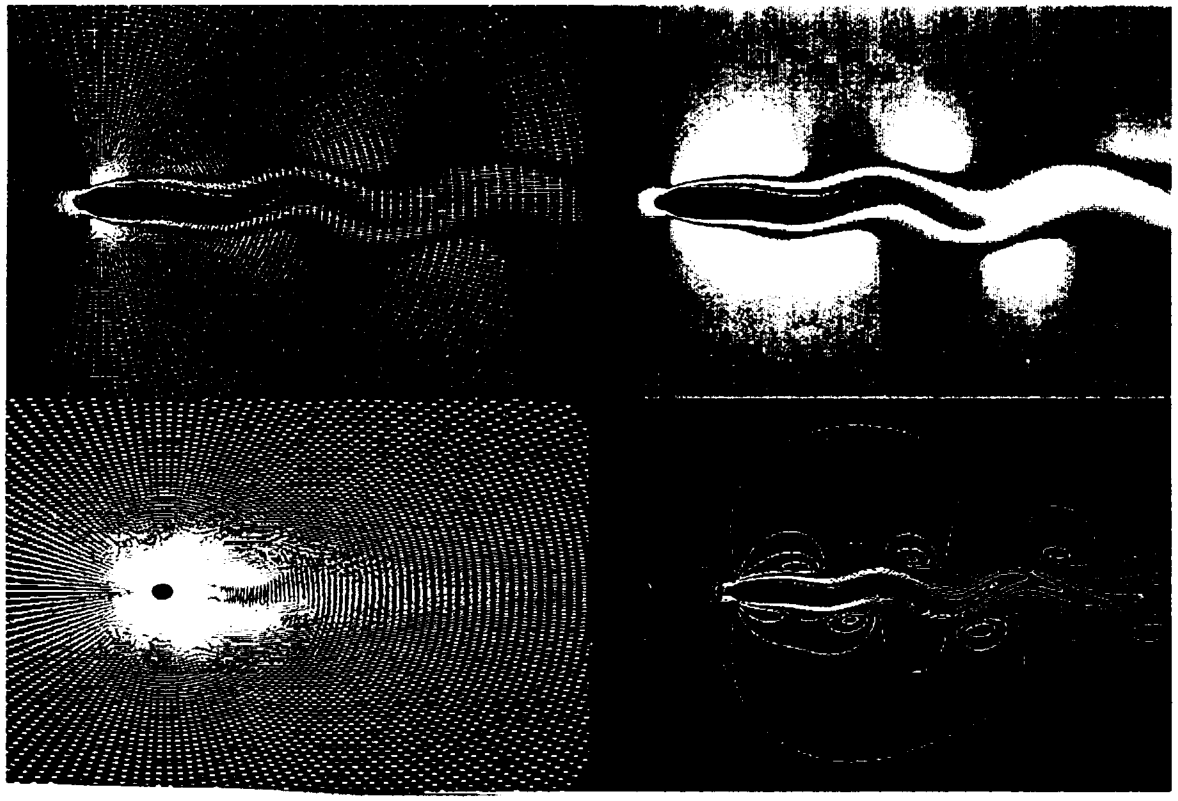


Figure 4.9 Rendering Methods for Flow Solution Visualization

Data Structure

Data structure design is a very important element in software system development. The data structure used in this GVU module contributes significantly to its success. Schematic presentations of the data structure are shown in Figures 4.10 and 4.11. The data structure for this module is arranged in such a way that no grid points, solution data, or function data is stored in each node of the link list. Only indices and related pointers are stored. The entire actual data are stored in a one dimensional array. This arrangement avoids the duplication of data storage, and hence has higher efficiency in terms of memory management. This allows a user to draw any number of surface patches in any number of blocks.

If a user exit the system by choosing the "Quit" button on GVU panel, a history file will be dumped to the file "T2gvu.lst.bkup". This file contains the detailed information about each surface patch of each block. A user may want to move this file to "T2gvu.lst" and execute the GVU module to restore the entire surface setting for the image. An example file is listed in Appendix B.

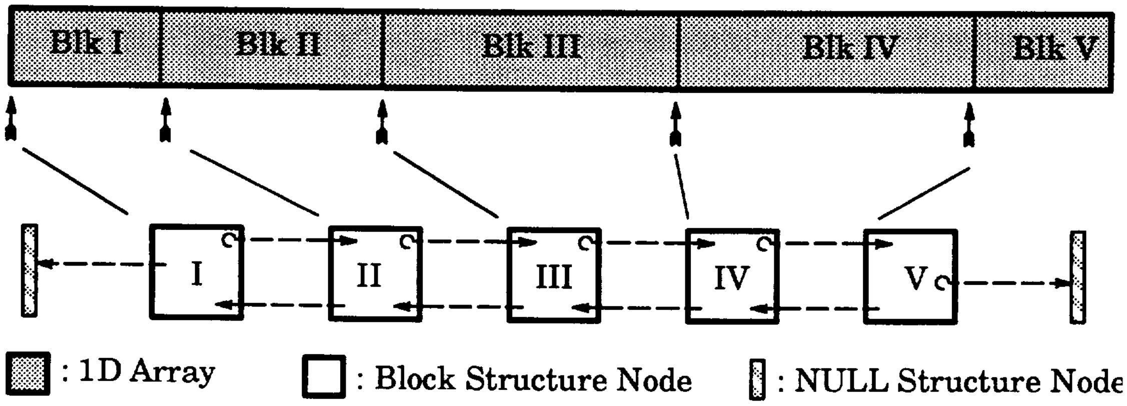


Figure 4.10 Array Distribution for Grid Only

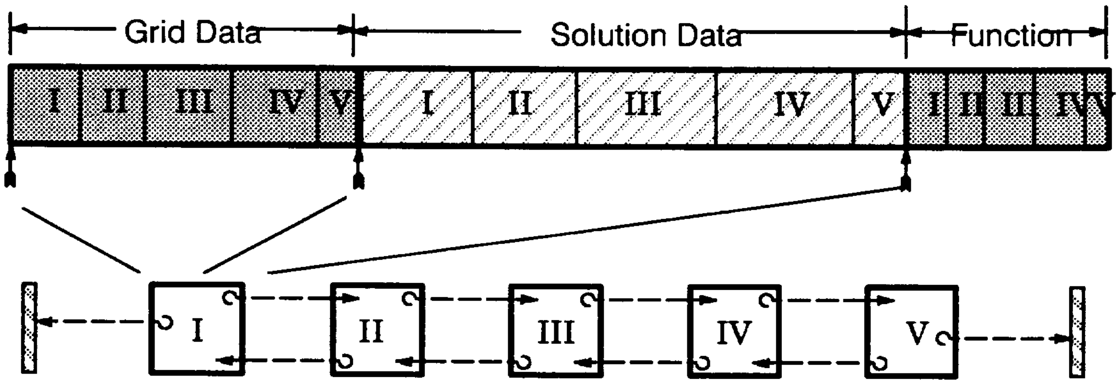


Figure 4.11 Array Distribution for Grid, Solution, and Current Function

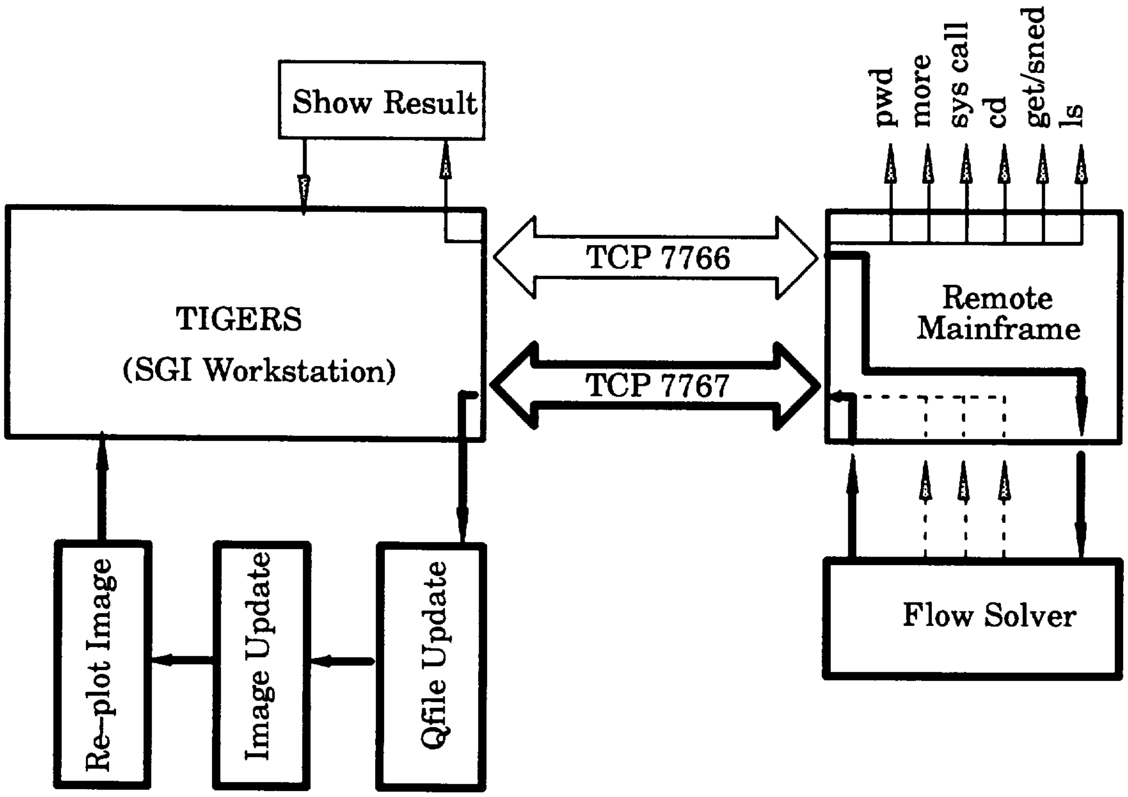


Figure 4.12 Scheme for the Network Module

Network Module

In order to accomplish the data communication in a heterogeneous environment, a network module is needed to create a bridge between different machines. A network module was developed for TIGER system to achieve this goal. This module uses the TCP/IP communications protocols[67] and allows file transfer service. This module consists of two sub–modules: the client sub–module and the server sub–module. The client sub–module resides in TIGER as a function and the server sub–module is a set of routines which reside independently on the remote mainframe. The scheme is illustrated in Figure 4.12.

The client/server model is the most commonly–used paradigm in constructing distributed applications, where the client applications request services from a server process. A server process normally listens for service requests. That is, a server process remain dormant until a connection is requested by a client's connection to the server's address, which activates the server process and perform the service.

Client/server model works well for applications that only require the server to service the request from the client. However, in order to maintain two–way communication between the client and the server so that the flow solution can be sent back to local workstation from the remote mainframe whenever the solution dump is completed, TIGER's network module established a two–way client–server/server–client relationship. That is, the client sub–module also served as server to receive the incoming solution data flow, while the server sub–module served as a client as well to send the updated solution data, and activate the client sub–module to service its request.

Non–blocking sockets, the UNIX communications abstraction similar to files, are used to avoid algorithm waiting for data flow. In other words, a socket marked as non–blocking will skip the waiting for the completion of the socket operation. Instead, it returns an error signal. This allows the program to return to the main algorithm and maintain the communication with the user. Therefore, on the local workstation, a user is able to manipulate the image objects on the screen, and the client sub–module will not “block” the socket until the data flow is transferring back and being updated. Such non–blocking approach allows user to maintain the control for most of the time.

However, this approach also creates another problem, namely, the loss of dormancy. For a typical client/server model, the server process remains dormant until the wait–up call from the client's connection. In this case, the server uses minimal CPU time since it wakes up only when it is necessary. Once non–blocking approach is applied, the server becomes a non–dormant process, which continuously accumulates the CPU time. This is not a favorable situation, since most mainframes impose a CPU limit for all interactive processes. In order to get around this problem, non–blocking sockets are still used to avoid the waiting, but for each error signal returned from the non–blocking sockets, i.e. no data flow is currently transferring, the server process is forced to “sleep” for 2 seconds. Even though this approach creates a minor lag for the server to realize the client's request, it uses much less CPU time due to this forced–dormancy. During the actual applications, such time lag is almost unnoticeable.

Simulation Module

The simulation module is accomplished by combining the GVU module and the network module. The GVU module provides the image visualization update of the geometry and the flow solution, while the network module handles the communication of the flow solver on the remote mainframe and TIGER on the local graphics workstation. A sketch of the setup is illustrated in Figure 4.13. The communication between the flow solver and the server module is accomplished by two *instruction* files, “srv2slvr.ins” and “slvr2srv.ins”. The first instruction file allows the server module to send the instruction sym-

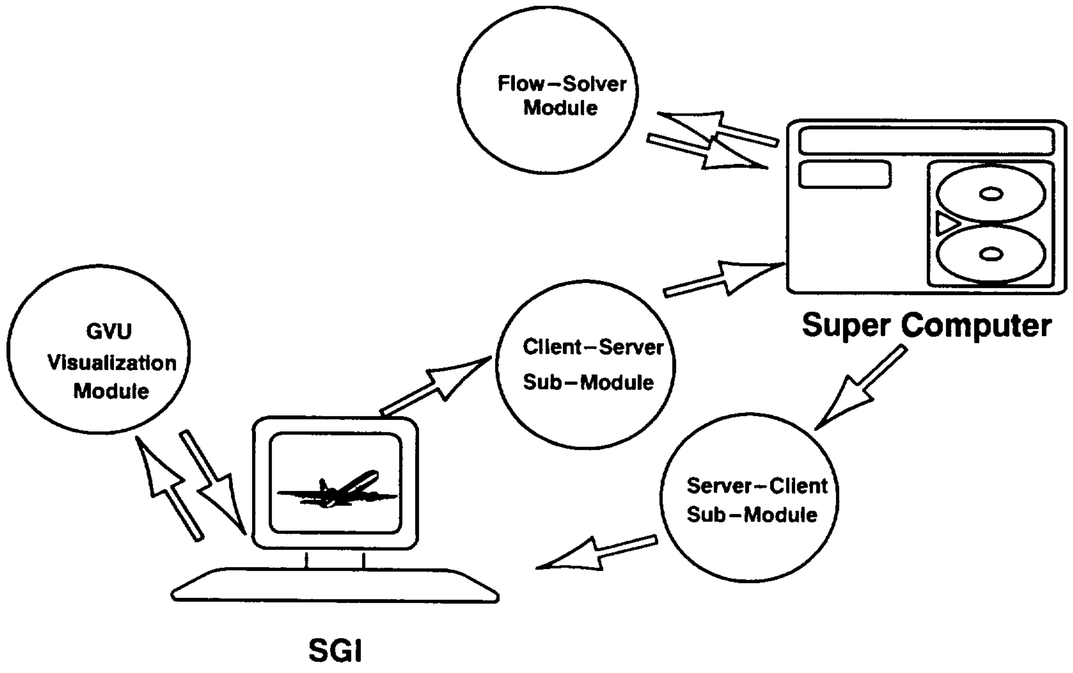


Figure 4.13 Simulation Module Setup

bols to the flow solver and control its action, such as "sleeping" the flow solver to prevent it overwriting the solution files while the server module is sending them back. And "slvr2srv.ins" file contains the information to instruct the server module when to transfer the solution files back to local graphics workstation. These two files synchronize the server and the flow solver.

Once the solution files have been transferred back to local graphics machine, the client module will send the signal through another instruction file, "clt2tiger.ins", on local workstation to TIGER, which will actuate the update process in TIGER. This update process includes reading the new solution to proper array location, updates the image objects based on the new solution, and redraw the image on the screen. The update process will "black–out" the communication between the user and TIGER, it will also freeze all the activities in TIGER.

ToolBox Module

This module provides the user with certain supporting utility tools related to the task, which may or may not be needed for each application. These utility tools are collected into this ToolBox module on the side. This allows the future extension or addition of the tools without modifying the main algorithm. In this module, the following tools are provided:

Format Conversion

This is a tool for the user to convert various geometry formats defined by different standards. Geometry data with an "alien" format is read in, converted to cylindrical coordinates, and displayed graphically in the graphic window, which allows the user to verify the geometry. This data may be output to another file in TIGER format. This tool may also read in a data file in TIGER format for the user to verify the raw data. This tool currently accepts 8 different commonly used formats, either in Cartesian coordinates or cylindrical coordinates.

LE/TE Circle Fitting

As discussed in §3.2.6, a given blade may not have well–defined LE and TE. Therefore, the algorithm developed in §3.2.6 is collected in the ToolBox module to allow the modeling of the LE/TE circle. For a two–dimensional airfoil, the data points should be given in the form of 2 curves, namely, the pressure side and the suction side. Each curve is spline–fitted independently. These two curves, however, are required to be single valued functions assume the form $\sigma = f(m')$.

MICE

Over the course of the development for TIGER system, a utility routine, *MICE* (Manipulation of grId CElls), was developed to facilitate the grid applications. *MICE* is a collection of various routines that allows a user to convert various data formats into desired format. This tool allows any number of block from any number of files to be manipulated and output to designated files. Manipulations such as extraction of a grid into various grids, combination of several grids into one grid, integration of grids into multi–block format, scaling/translation/rotation of an existing grid, duplication of turbomachinery passages, Cartesian and cylindrical coordinates conversion, and format conversions are salient features of this tool. This tool, however, is an external module to the entire TIGER system. In other words, a user can only access it externally.

Flow Module

This module is an open–ended module in the TIGER system. The flow solver that resides on the mainframe may be any appropriate software. Minor modifications will be needed to the input and output routines of the flow solver so that it will recognize the instruction file logic. Currently, a multi–stage unsteady turbomachinery code (MSUTC)[48,49] is adopted as the flow module. This code applies the thin–layer approximation with a Baldwin–Lomax turbulence model to solve the Reynolds–averaged Navier–Stokes equations. It is an implicit finite volume scheme with flux Jacobians evaluated by flux–vector–splitting and residual flux by Roe's flux–difference–splitting. A modified two–pass matrix solver based on the Gauss–Siedel iteration was used to replace the standard LU factorization to enhance the code's stability. This code, uses localized grid distortion technique[30,48,49] to the buffer zone between the blade rows to achieve time–accurate solution for compressible flow.

This code, still in its research format, needs to hard–wire several parameters case by case. It therefore poses a problem for TIGER to integrate it fully. Further development would be expected to generalize MSUTC so that the integration can be improved.

CHAPTER V

BATCH AND DYNAMIC GRID GENERATION

With the user–friendly GUI and the customized grid generation procedures, grid generation using current work for complicated turbomachinery configurations requires less labor involvement. However, it is not always necessary for a user to execute TIGER with the full–capability of its graphics. Instead, sometimes it may be more user–friendly if a user can execute the code without the graphics by running the history files in a batch mode. This promotes the development in the first section of this chapter.

The ultimate goal of CFD is to optimize design in view of desired performance. This goal, however, cannot be achieved with one single design because numerous corrections to the geometry are needed in order to obtain desired performance. These geometry perturbations, may require the re–construction of numerical grids. Therefore, this chapter is also going to discuss the development of a dynamic grid generation version that allows time–dependent pitch change. This version also paves the way for future development to simulate dynamic unsteady flow fields.

Batch Version

The batch version of TIGER inherits the basic algorithm of the grid generation module in the aforestated interactive version. This batch version does not contain any of the graphics capabilities or the overall interactive nature; this simplifies its algorithm and condenses its size. Consequently, this batch version has the portability that the interactive version does not have. It reads in existing history files produced by its interactive version and repeats what was done to re–generate the grid.

This is especially advantageous when a large size grid needs to be transferred to a remote mainframe to perform a flow calculation. A user may transfer fairly small history files and re–generate the grid on the remote mainframe instead of transferring a large grid file. Or, when a grid is not needed for the current time but will be used later, only history files are needed to be stored on a machine with limited disk space. Moreover, it would be advantageous for generation of viscous grids where very tight spacings are desired and double–precision is essential. Executing the batch version of TIGER on a 64–bit machine allows a user to obtain grids with double precision which are necessary for high–Reynolds–number flow fields.

The batch version of current work has been ported and successfully tested on machines like the SGI Personal IRIS™, SGI Challenge™, Cray XMP–2/216™, and Cray YMP–8/8126™.

Dynamic Grid Generation

A variation of TIGER's batch version was tested for dynamic grid generation about turbomachinery configurations. Blades of a turbomachine may change their power setting by varying the blade stagger angle during the operation; they may flutter because of the force interaction between the blade structure and the flow. This section demonstrates the capability of generating dynamic turbomachinery grids using current work.

Minor modifications were made to the batch version of current work. These modifications include making TIGER as a subroutine to a main driver routine which calls TIGER iteratively, passing different blade angles in each cycle. Consequently, TIGER generates the grids with variation of blade stagger angles.

Changing the blade stagger angle, however, changes the blade position from where the grid was initially generated using the interactive version. Therefore, the algorithm must be able to modify its history setting accordingly. This is done by adding an adjustment routine that will check the current position of the blade and its designed position and adjust the neighboring segments and breaking nodes accordingly. The adjustments occur mainly in the frame setting process. Once the blade position is changed, the arc–lengths of the generatrices associated with the blade will be influenced, changing the m'–coordinates of each associated nodes. The associated averaged radii $\bar{r}$, will be changed as well. Therefore, this adjustment algorithm is designed to take the current parameters and offset the parametric coordinates (m',σ) associated with the nodes to their new location.

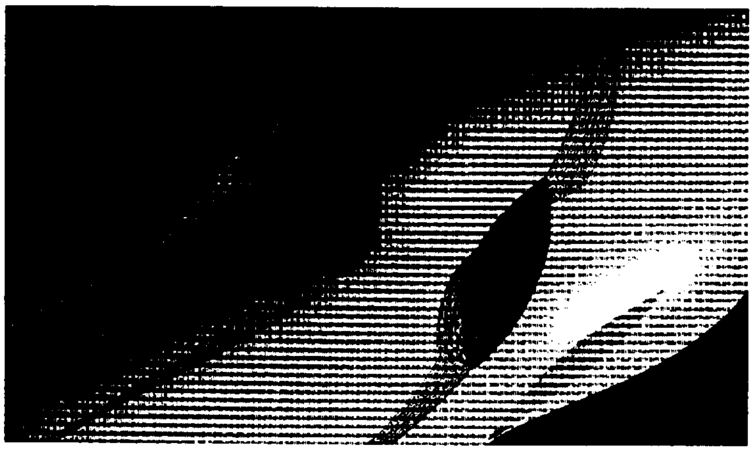


Figure 5.1 Trace of the Moving Periodic Boundaries

The case that is to be presented in this section is a Hamilton–Standard single–rotation propfan, SR–7 with dynamic pitch changing according to time.

This geometry was modelled at $\beta_{3/4} = 58.0$ degrees, and the initial grid was generated interactively on an SGI workstation to create the history files. These history files are then ported to a SGI Challenge machine. The automatic adjustment algorithm proved to be so successful that the grid is able to maintain positive volume everywhere with a stagger angle range of 58 ± 15 degrees. In other words, the power setting changing may have 30 degrees of variation range. Further improvements may be achieved by adjusting more on the neighboring segments near the blade. The trace of the moving passage boundary is shown in Figure 5.1. The surface grid on the hub with $\beta_{3/4} = 43.0$ degrees is shown in Figure 5.2, while the same surface with $\beta_{3/4} = 73.0$ is shown in Figure 5.3. Note that this blade is twisted, therefore, the blade angle presents higher angle than $\beta_{3/4}$. Superimposed image of both cases is shown in Figure 5.4.

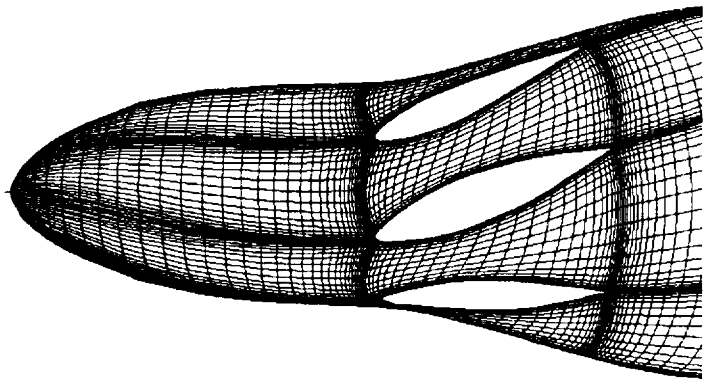


Figure 5.2 Surface Grid of the Hub at 43 Degrees

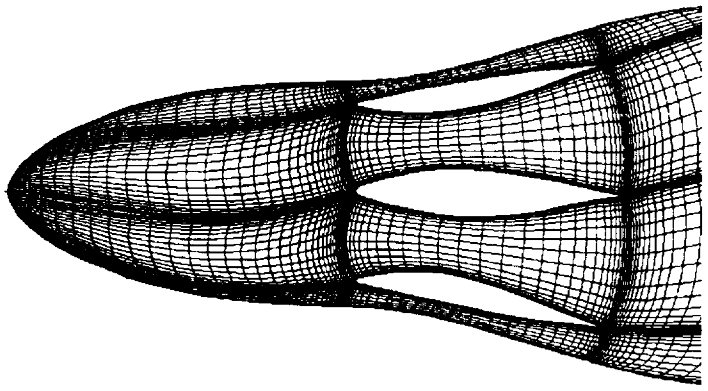


Figure 5.3 Surface Grid of the Hub at 73 Degrees

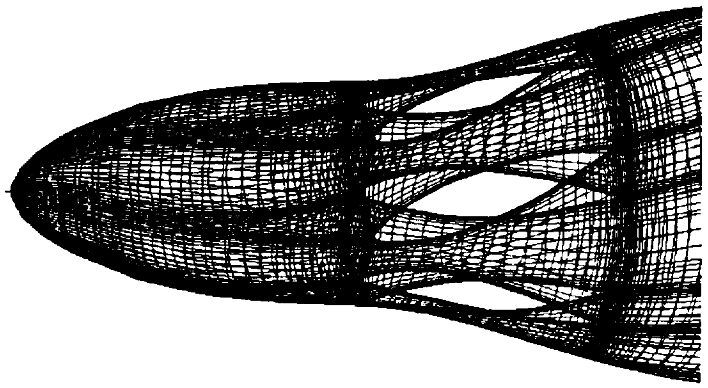


Figure 5.4 Superimposition of Hub Surfaces for Both Angles

CHAPTER VI
APPLICATIONS AND RESULTS

TIGER has been used to generate grids for various real–world complicated turbomachinery configurations. Its algorithm has been validated to be very time–efficient and robust. These applications include internal flow, external flow, and internal–external flow configurations. Axial, radial, and mixed—flow designs were used to validate its grid generation capability. Two geometries were used to validate its integrated simulation system. This chapter summarizes these applications in two main categories: (*i*) grid applications and (*ii*) simulation applications.

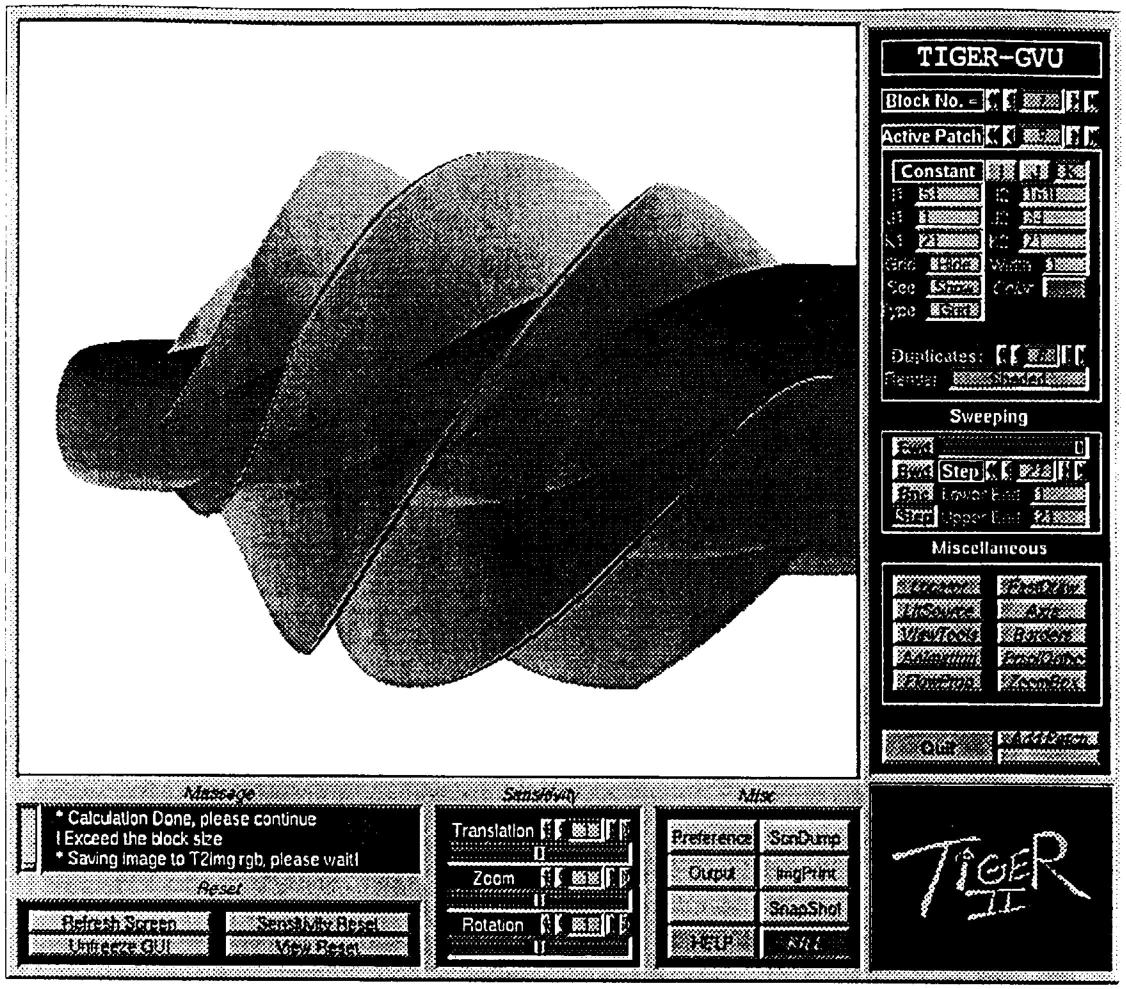
Grid Applications

Internal Flow Field

A geometry is characterized as an internal flow field configuration if the entire domain of computation is confined by solid geometry except the inlet and outlet. Several cases presented in this category include a water pump inducer, a centrifugal compressor, a turbopump impeller, a high–Reynolds–number stage, and a mockup geometry designed for validation.

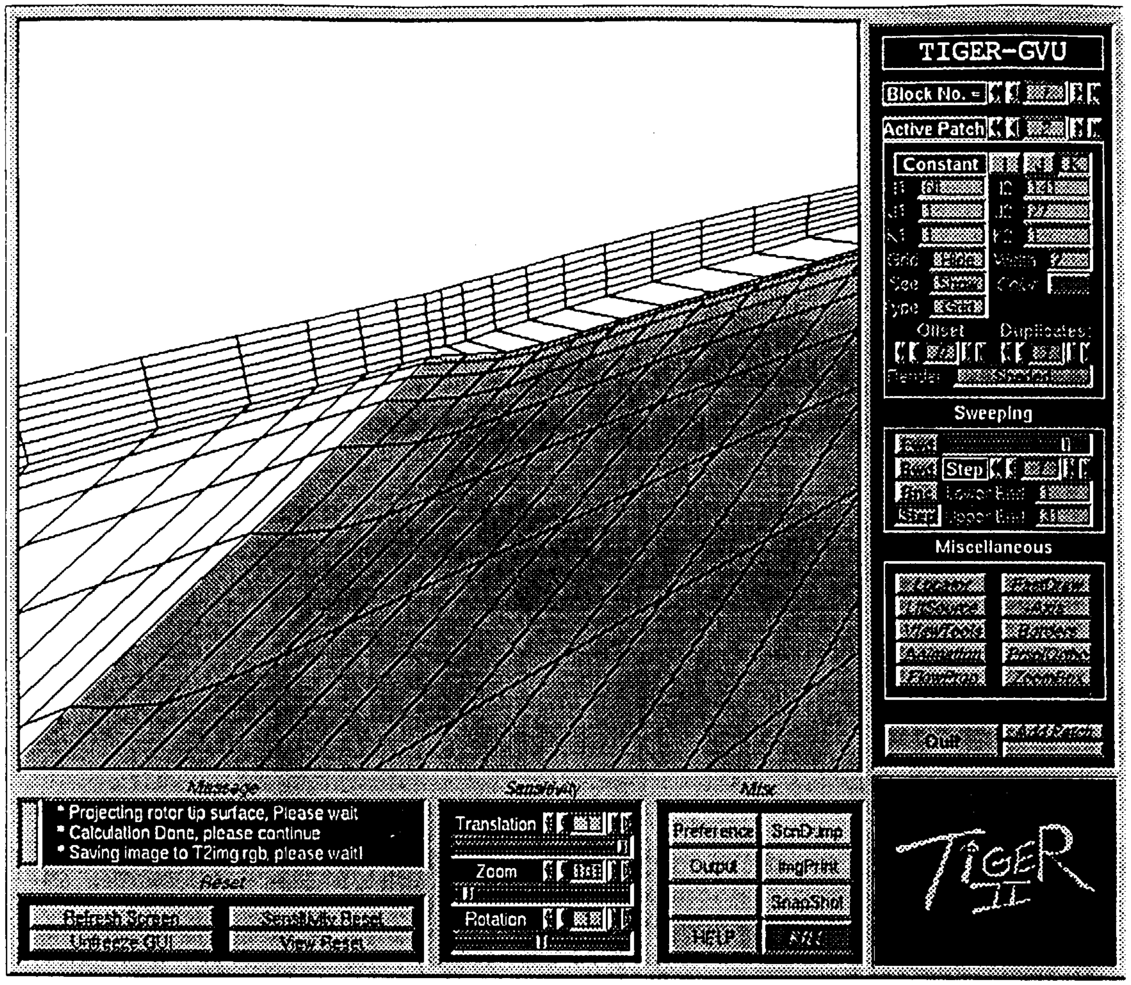
Water Pump Inducer

The first case for internal flow field is an inducer for a water pump. This inducer has 6 blades which wrap around the hub, as shown in Fig. 6.1(a). A tip clearance of 0.008 inches (0.1334% of the blade diameter) is modelled in



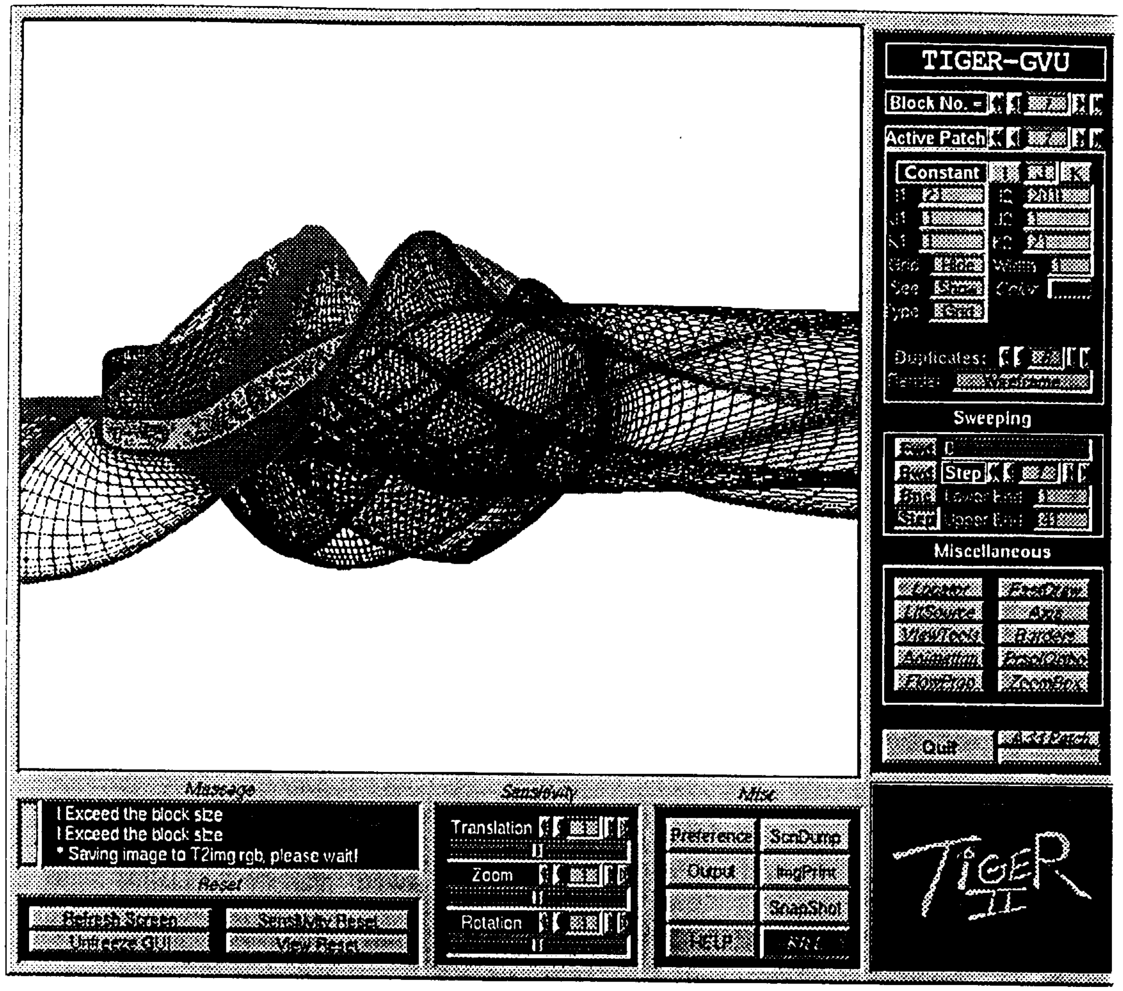
* Calculation Done, please continue
! Exceed the block size
* Saving image to T2img.rgb, please wait!

Figure 6.1 (a) Water Pump Inducer



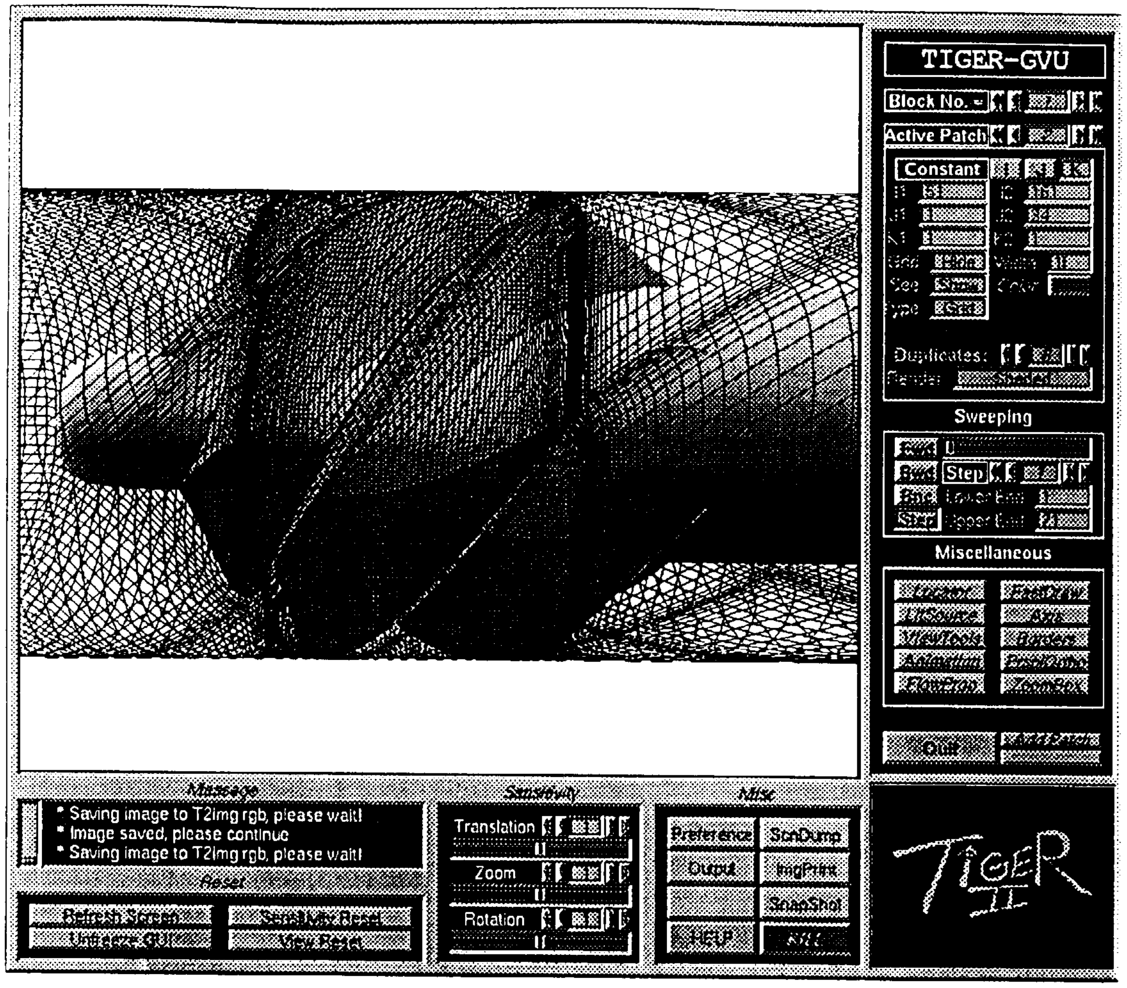
```
* Projecting rotor tip surface, Please wait
* Calculation Done, please continue
* Saving image to T2img.rgb, please wait!
```

Figure 6.1 (b) Grid Behavior Inside Tip Clearance



! Exceed the block size
! Exceed the block size
* Saving image to T2img.rgb, please wait!

Figure 6.1 (c) Overall Grid Behavior (Hub)

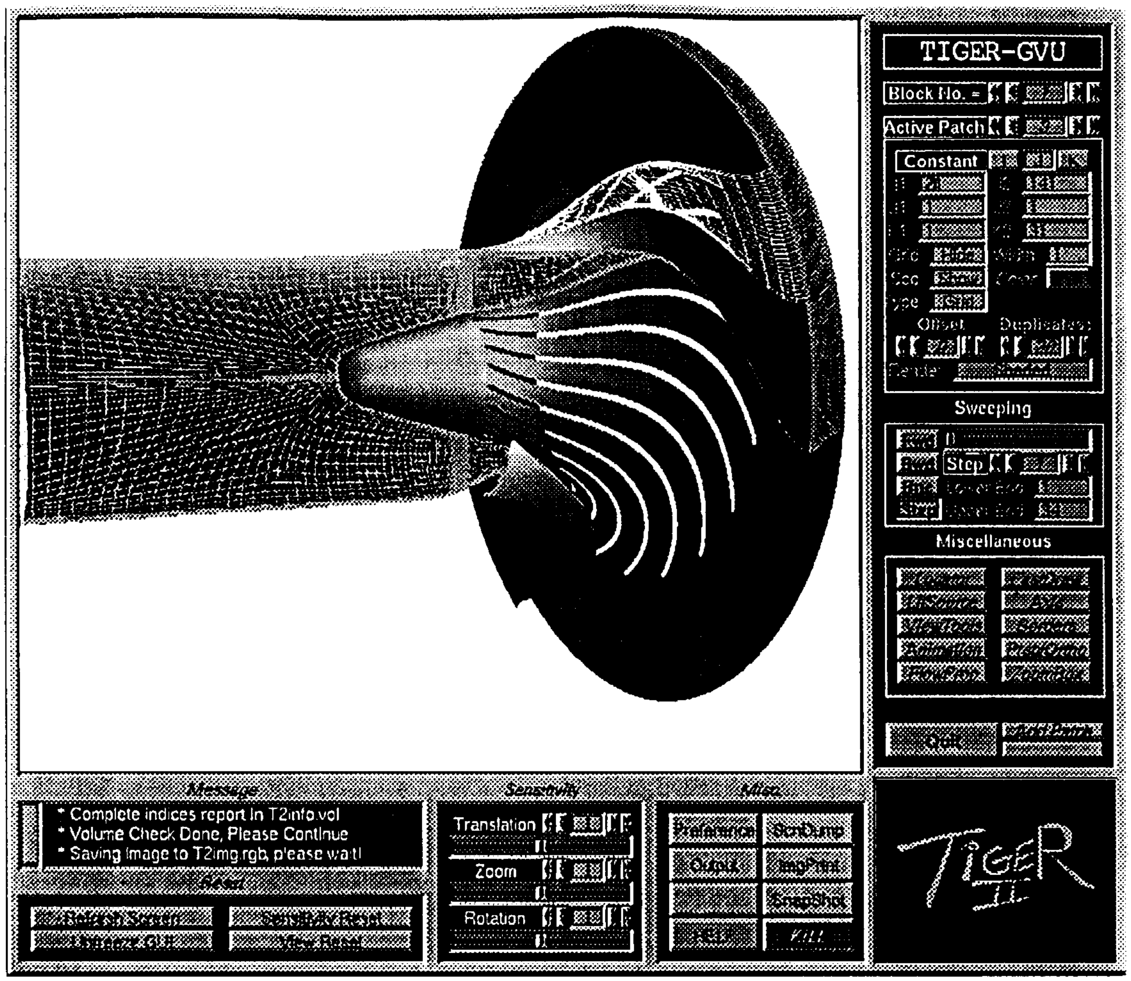


* Saving image to T2img.rgb, please wait!
* Image saved, please continue
* Saving image to T2img.rgb, please wait!

Figure 6.1 (d) Overall Grid Behavior (Shroud) the way discussed in §3.3. The grid behavior inside the gap near the leading edge is illustrated in Figure 6.1(b). The grid size is 156x34x31 per passage with 6 grid cells in the tip clearance in η–direction (radial direction). The point distribution is packed toward the solid surfaces to resolve the turbulence flow. The overall grid behavior on the hub and the shroud surfaces are shown in Figures 6.1(c) and 6.1(d), respectively. The existence of the blunt bull–nose on the hub activates TIGER to break the hub generatrix into 2 sections so that the cascade mapping can be accurately executed. This grid was accomplished in about 3 man–hours starting from scratch.

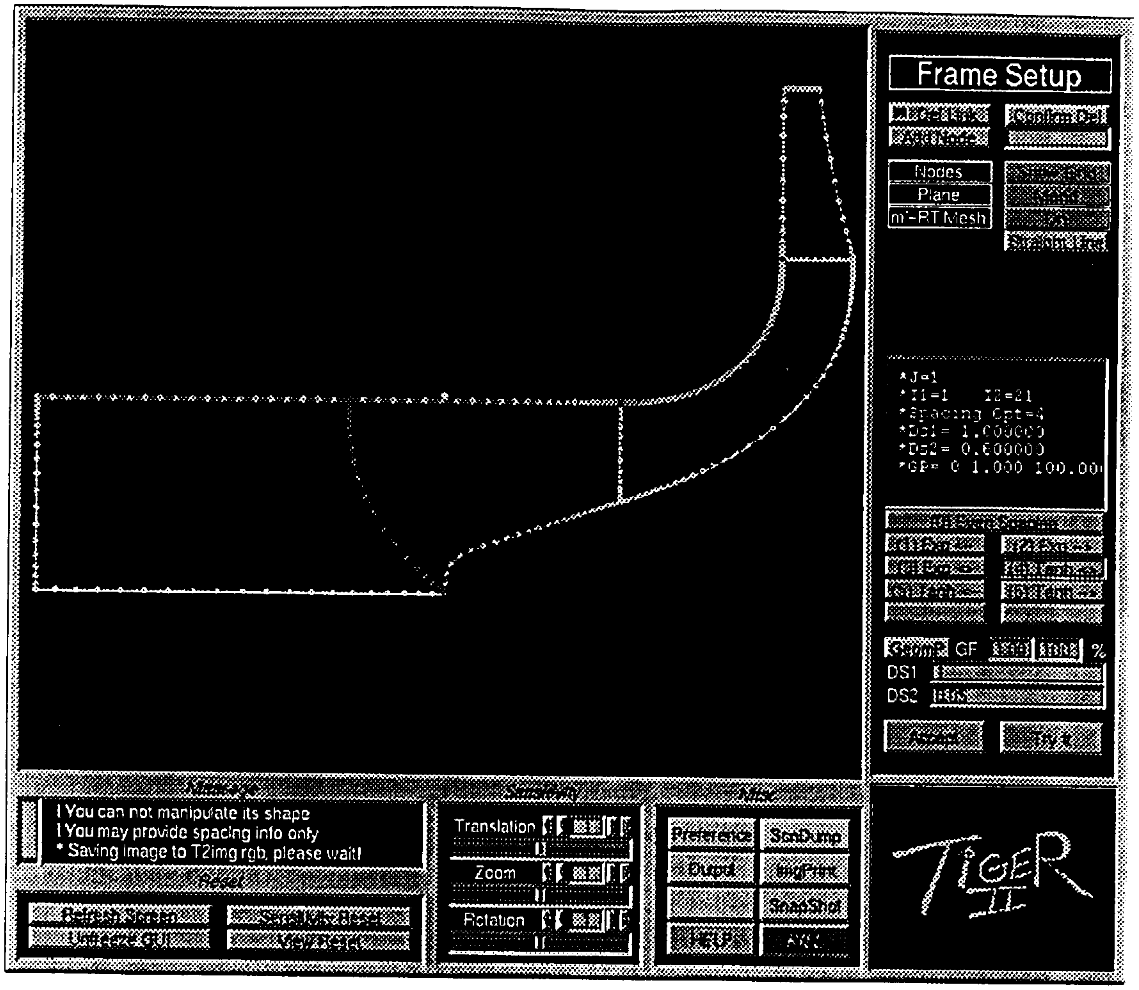
NASA Low Speed Centrifugal Compressor

This facility is built to obtain detailed flow field measurements for computational fluid dynamic code assessment and flow physics modeling[68,69]. It has 20 full blades with a contraction region at the exit as shown in Figure 6.2(a). Due to the discrepancy between the given raw data and an existing grid of this geometry at the contraction region, a decision was made to use the contraction design as in the existing grid. Therefore, the blade data for this case was extracted from the raw data, while the generatrix geometry for hub and shroud were extracted from an existing grid. A tip clearance of 0.1 inches (0.1667% of the blade diameter) is modelled. The meridional frame setting is shown in Figure 6.2(b). The cascade frame setting using the transformation procedure described in §3.5 is shown in Figure 6.2(c). Demonstrated in Figure 6.2(d) is the overall grid. The grid size is 141x34x31 per passage with 6 grid cells in the tip clearance in η–direction. Grid points were clustered toward the solid surfaces to resolve the turbulence flow. The hub generatrix were decomposed into three sections while the shroud generatrix were decomposed into two sections to perform the cascade mapping logic. This grids was



* Complete indices report in T2info.vol
* Volume Check Done, Please Continue
* Saving image to T2img.rgb, please wait

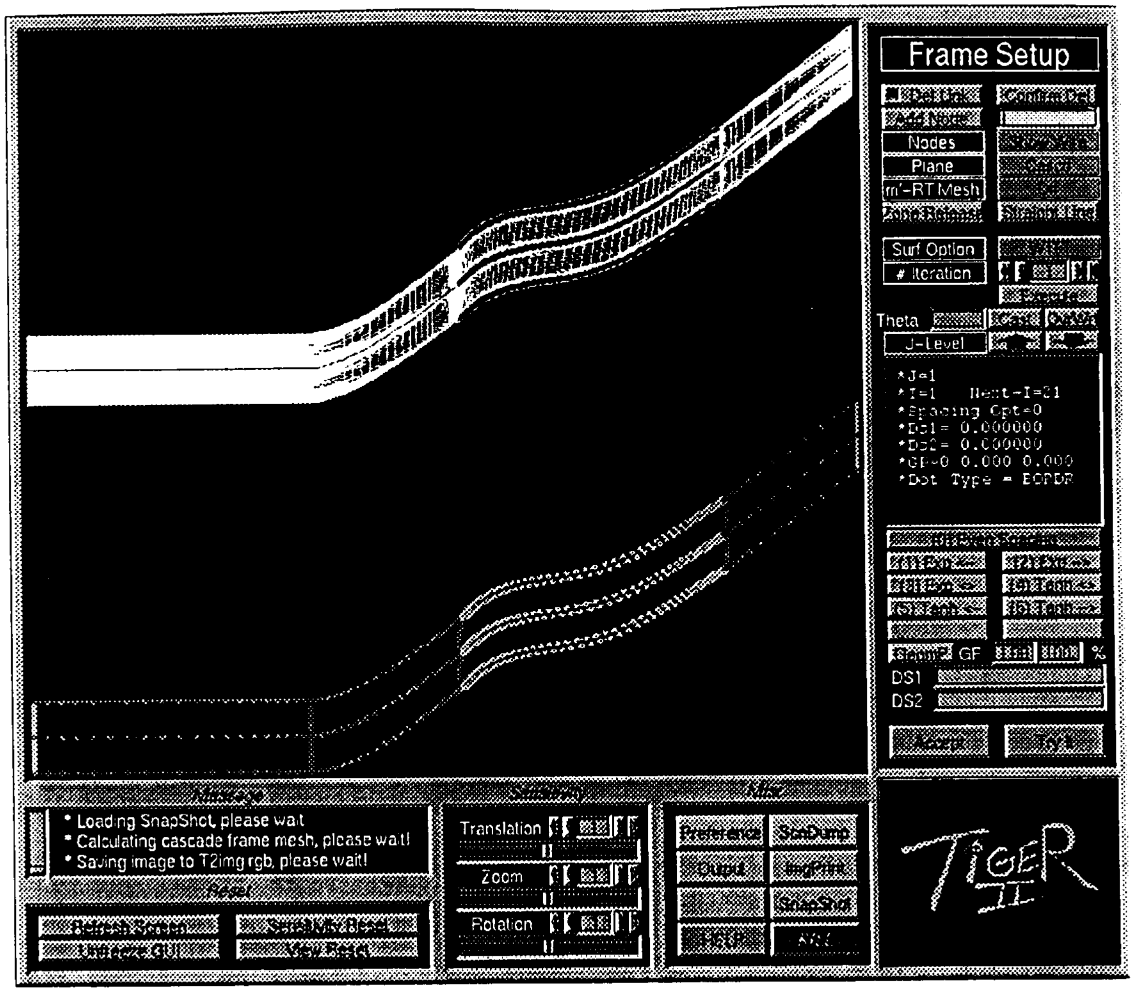
Figure 6.2(a) NASA Low Speed Centrifugal Compressor



```
*J=1
*I1=1     I2=21
*Spacing Opt=4
*DS1= 1.000000
*DS2= 0.600000
*GP= 0 1.000 100.00
```

```
! You can not manipulate its shape
! You may provide spacing info only
* Saving image to T2img rgb, please wait!
```

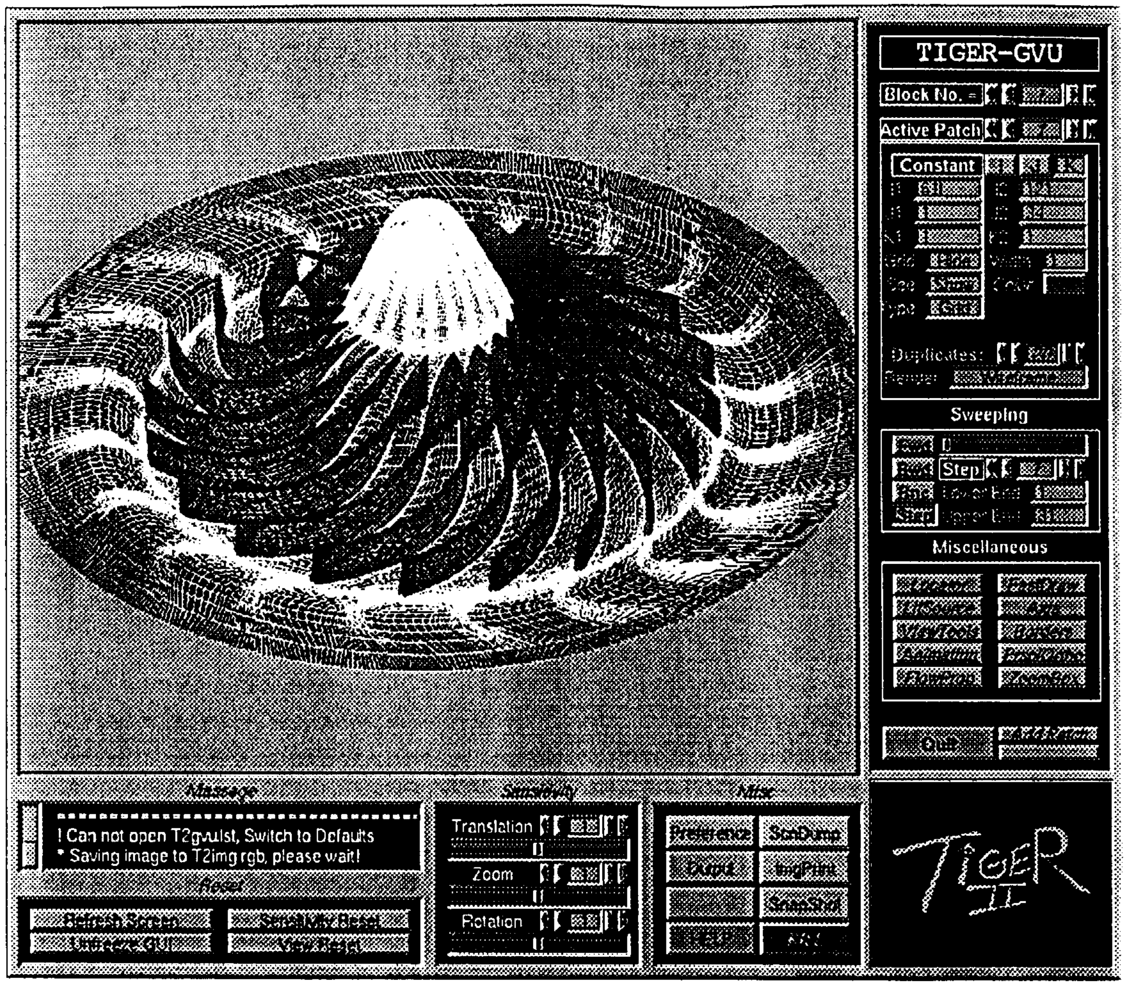
Figure 6.2(b) Meridional Frame Setting



```
*J=1
*I=1     Next-I=21
*Spacing Opt=0
*DS1= 0.000000
*DS2= 0.000000
*GF=0 0.000 0.000
*Dot Type = BORDR
```

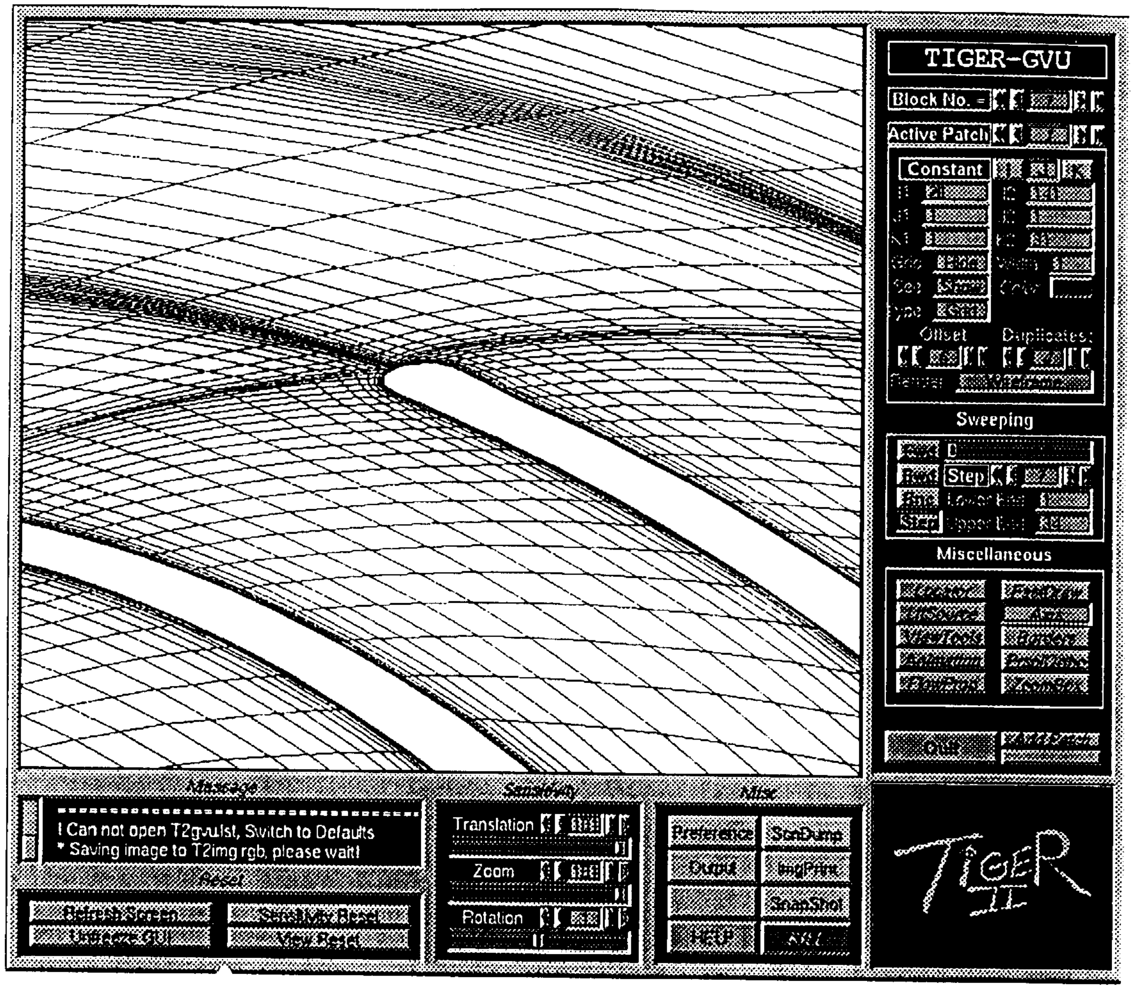
```
* Loading SnapShot, please wait
* Calculating cascade frame mesh, please wait!
* Saving image to T2img rgb, please wait!
```

Figure 6.2 (c) Cascade Frame Setting



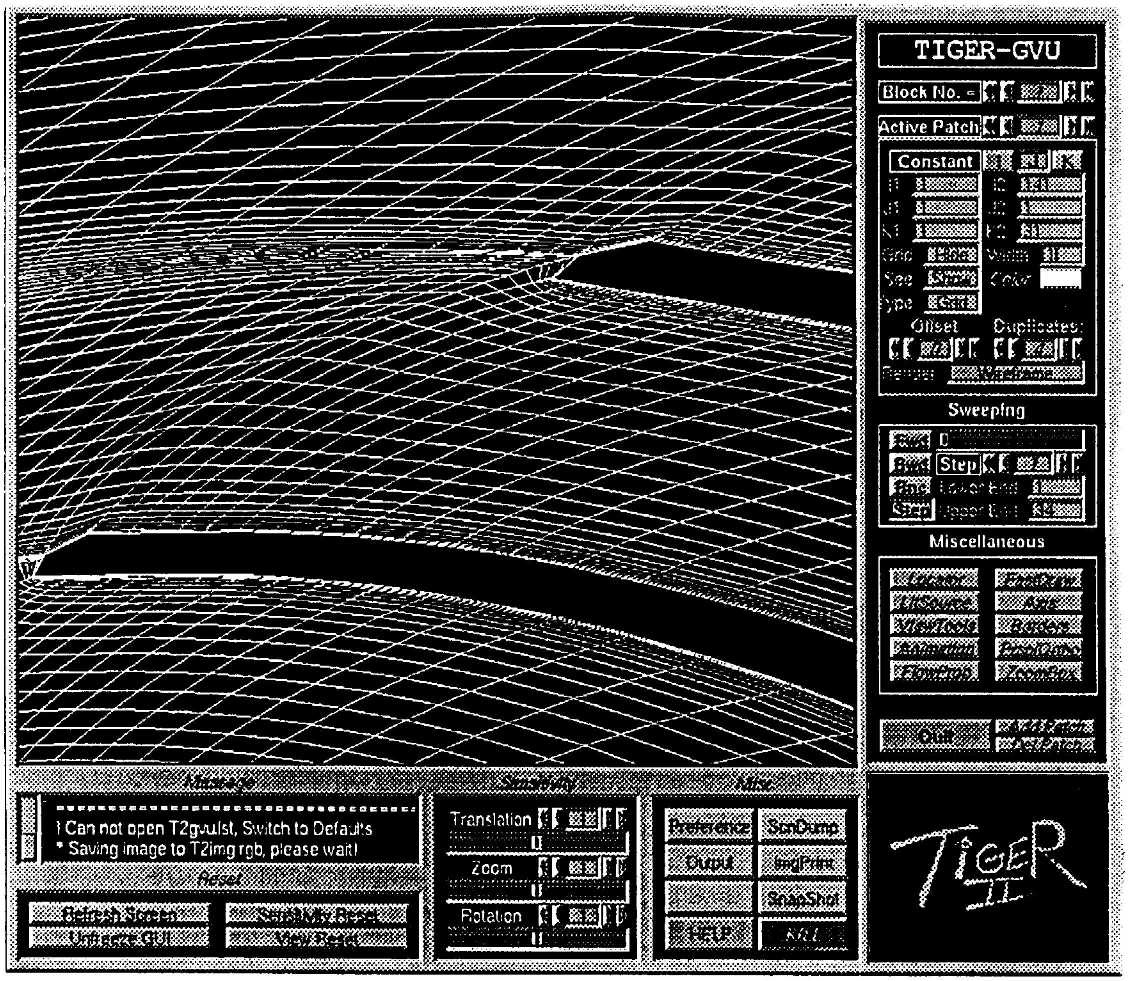
! Can not open T2gvulst, Switch to Defaults
* Saving image to T2img.rgb, please wait!

Figure 6.2 (d) Overall Grid Behavior



! Can not open T2gvulst, Switch to Defaults
* Saving image to T2img rgb, please wait!

Figure 6.2 (e) Rounded Trailing Edge Approach



! Can not open T2gvulst, Switch to Defaults
* Saving image to T2img rgb, please wait!

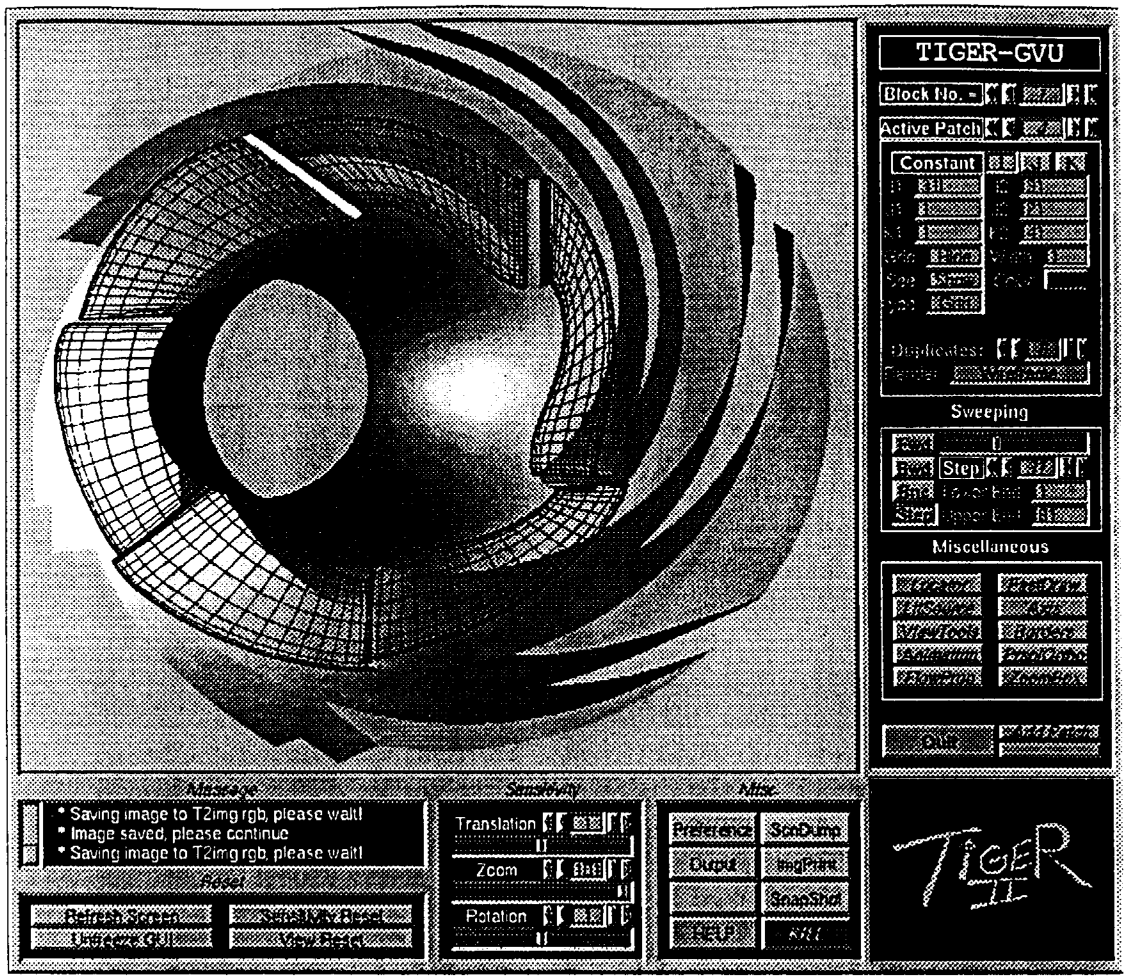
Figure 6.2 (f) Wedged Trailing Edge Approach also accomplished in about 3 man–hours. This presented grid has a rounded trailing edge using the fitting technique discussed in §4.4.2. A close–up image near this region is shown in Figure 6.2(e). However, because of the difficulties experienced during the flow calculation, the trailing edge has been modified to be wedge–shaped, as shown in Figure 6.2(e), to eliminate the turbulence flow induced by the rounded trailing edge.

SIMPLEX Turbopump Impeller

This geometry, provided by NASA MSFC, is an on–going design. Shown in Figure 6.3(a) is the overall configuration of this impeller. This grid was generated using 81x21x41 grid points per passage. Six blades were modeled without tip clearance. Due to the low stagger angle and fairly blunt leading edge, it is a very difficult region to generate a cascade surface grid with H–type topology. Severe grid skewness is unavoidable, and it becomes worse on the shroud. Bezier curves were constructed near this region to eliminate the line crossing. Figure 6.3(b) shows the hub surface grid without Bezier curve manipulation. Figure 6.3(c) shows the grid behavior on the hub (wheel). Figure 6.3(d) shows the close-up of the leading edge region. The grid quality has been improved using the scheme combined the elliptic smoothing and NURBS spline–fitting described in §3.4.3. The trailing edge, however, is treated as an open edge because of the straight cut of the trailing edge. Extra blocks may be attached downstream. They can be obtained with a general–purpose grid generation code, such as EAGLEView and GENIE^{++}.

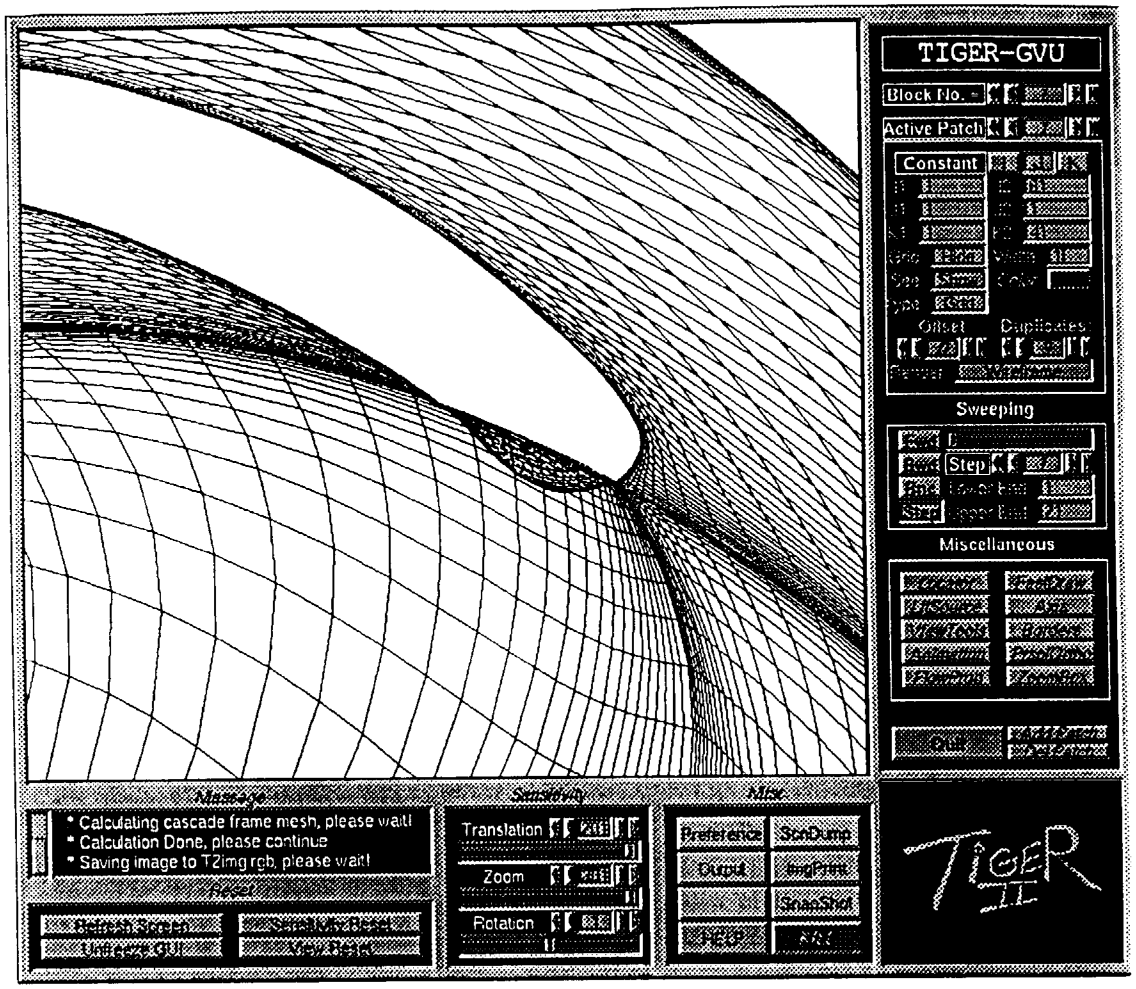
High–Reynolds–Number Pump

This configuration is designed to study the wake and blade interaction in incompressible flow[70]. The geometry is illustrated in Figure 6.4(a). This



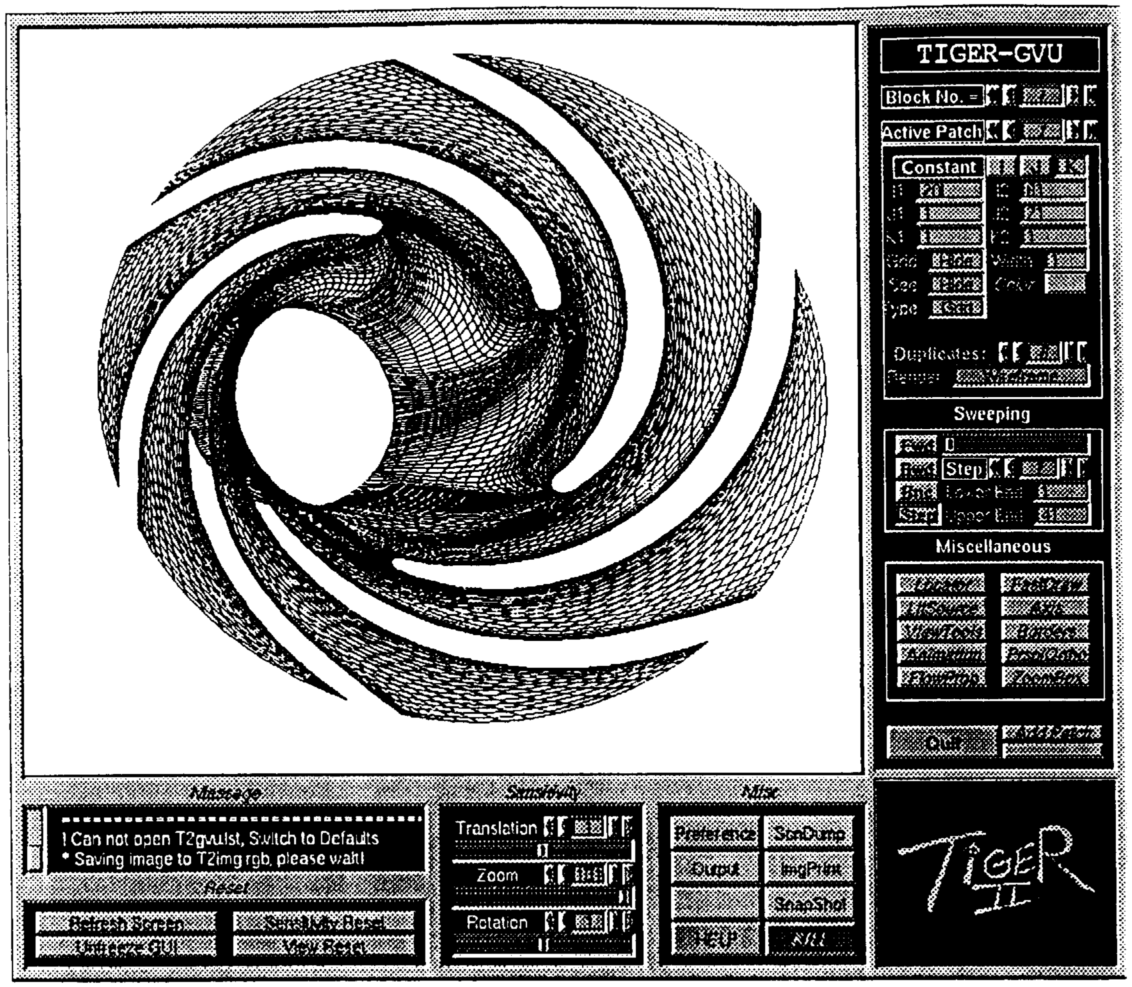
```
* Saving image to T2img.rgb, please wait!
* Image saved, please continue
* Saving image to T2img.rgb, please wait!
```

Figure 6.3 (a) SIMPLEX Turbopump Impeller



* Calculating cascade frame mesh, please wait!
* Calculation Done, please continue
* Saving image to T2img.rgb, please wait!

Figure 6.3 (b) Hub Surface Grid without Bezier Curve Manipulation



```
! Can not open T2gvulst, Switch to Defaults
* Saving image to T2img rgb, please wait!
```

Figure 6.3 (c) Hub Surface Grid with Bezier Curve Manipulation

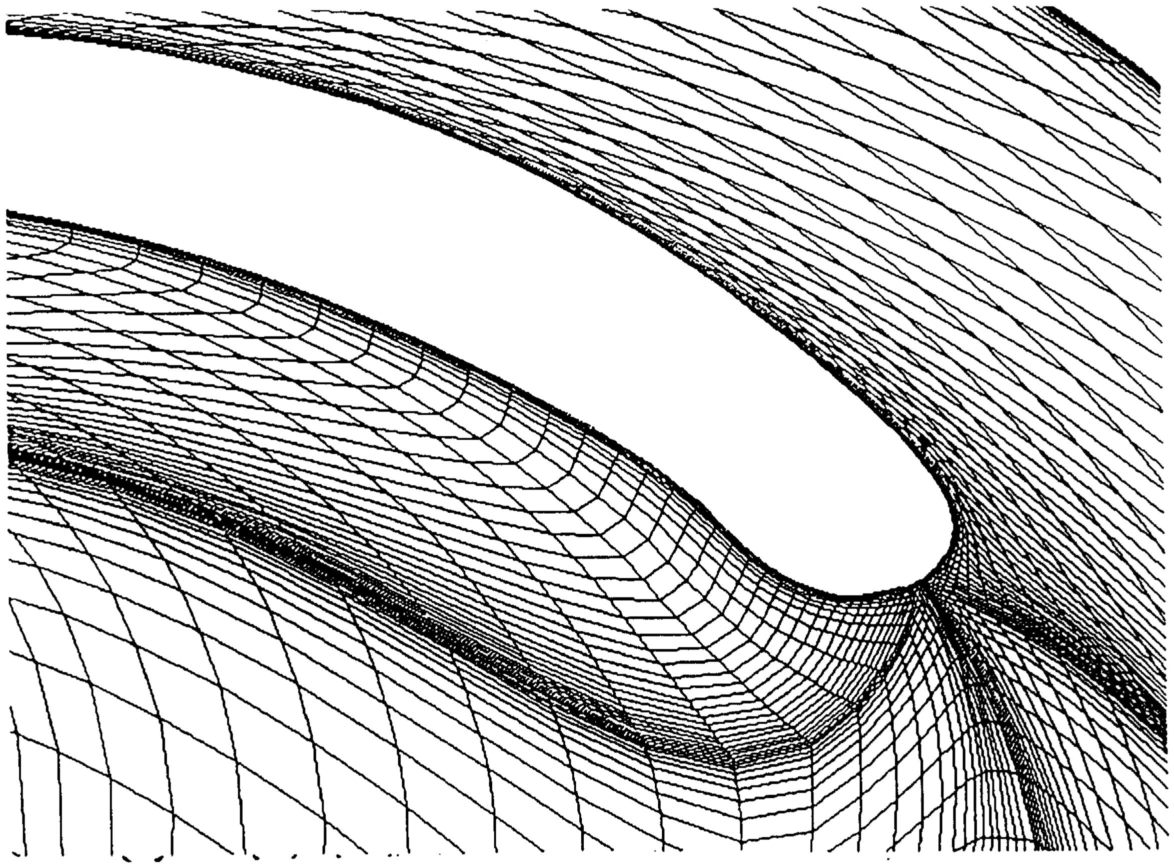
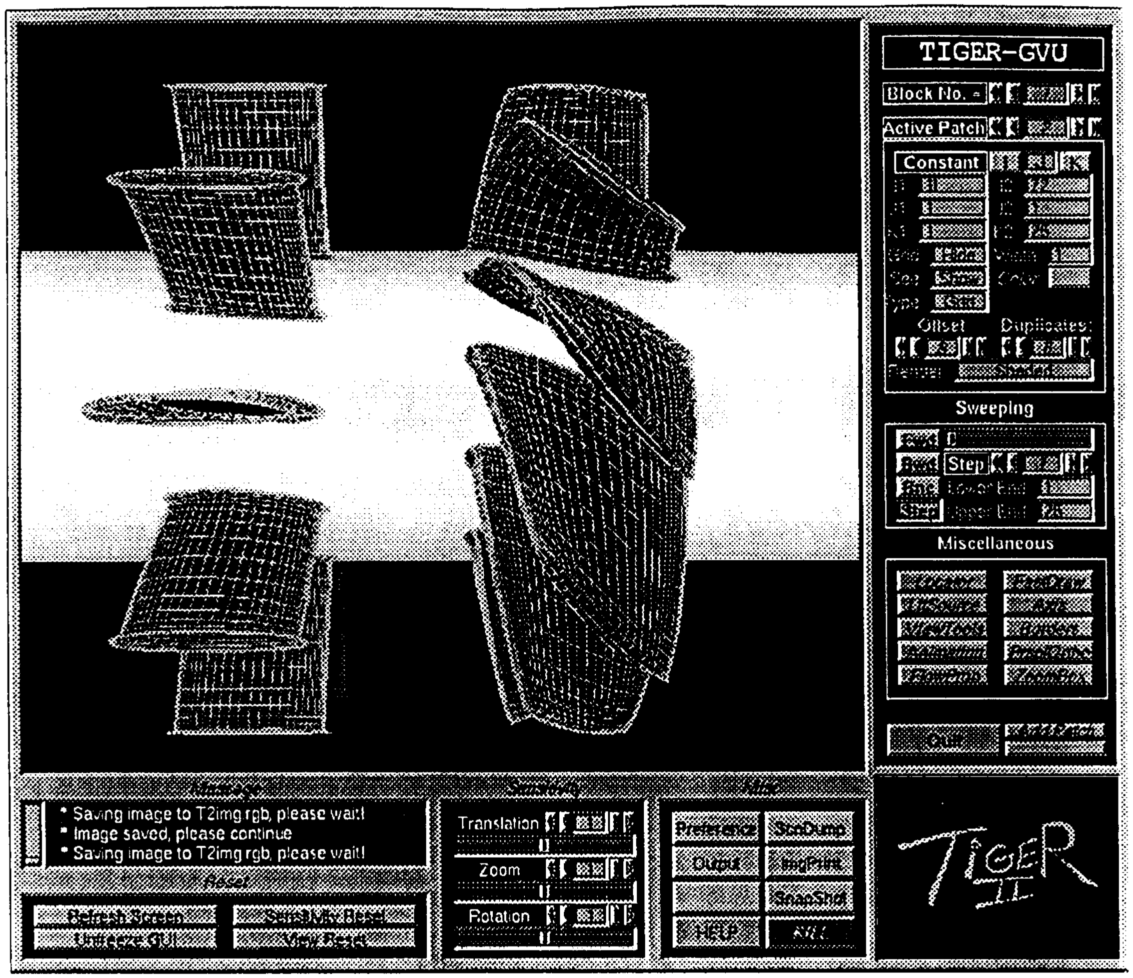


Figure 6.3 (d) Close–up View near Blade Leading Edge

single–stage geometry has actually 13 stator blades upstream followed by 7 rotor blades. However, this configuration is numerically expensive since the flow solver would need to model the full geometry which is a formidable task in terms of computer capacity. Therefore, an initial geometry having a blade count of 11–11 was used to verify that the flow solver was working correctly. There are two different designs for this geometry. The first one has no fillets on both stator and rotor blade surfaces, while the second one has fillets for both the stator and rotor. The design presented is the later. The overall grid behavior is shown in Figure 6.4(b). Illustrated in Figure 6.4(c) is the grid near the stator and the rotor. A viscous grid was constructed to allow turbulence flow modeling. However, the grid generation near the junction of the fillet and the hub is challenging. As shown in Figure 6.4(d), grid lines in three directions are almost lying on the same (r,θ) plane. This junction area might cause numerical difficulty if attempts are made to generate the grid in Cartesian coordinates. Fortunately, TIGER did not experience any difficulty with its WTFI and cylindrical coordinate approach, even though it is a single precision code. However, in order to achieve 0.1% relative radial grid spacing with respect to the blade radius while maintaining reasonable significant digits, batch version of TIGER is used to produce 64–bit grids.

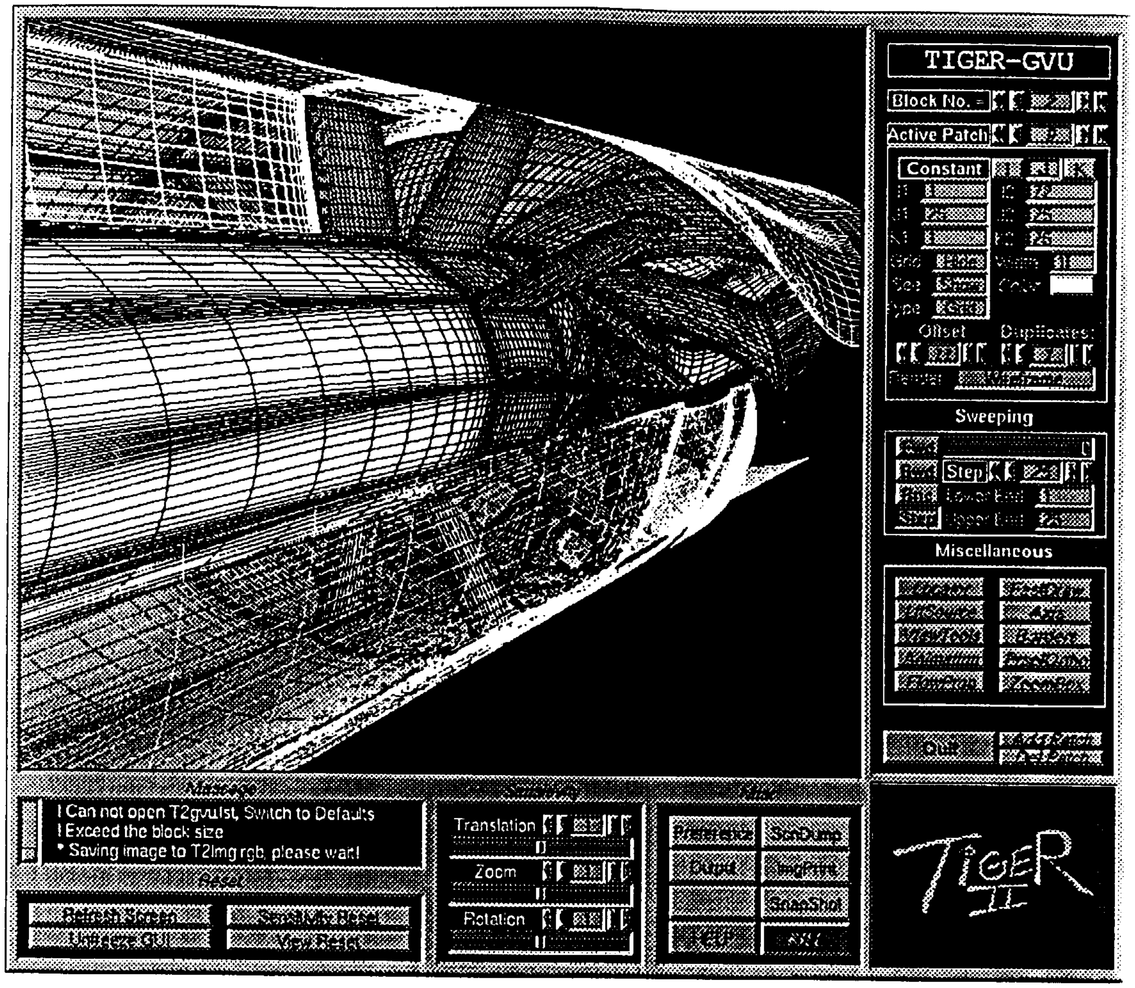
Axial–Radial Configuration (Mockup)

A research mockup geometry is presented in Figure 6.5(a). The intention of this mockup is to test the robustness of the algorithms of domain framework setup and cascade mapping. This geometry was created by combining available components. The inducer portion is the simplified geometry of the inducer used in SIMPLEX turbopump. It has three blades. A circular bull–nose was created to close the front end of the hub, while a straight line



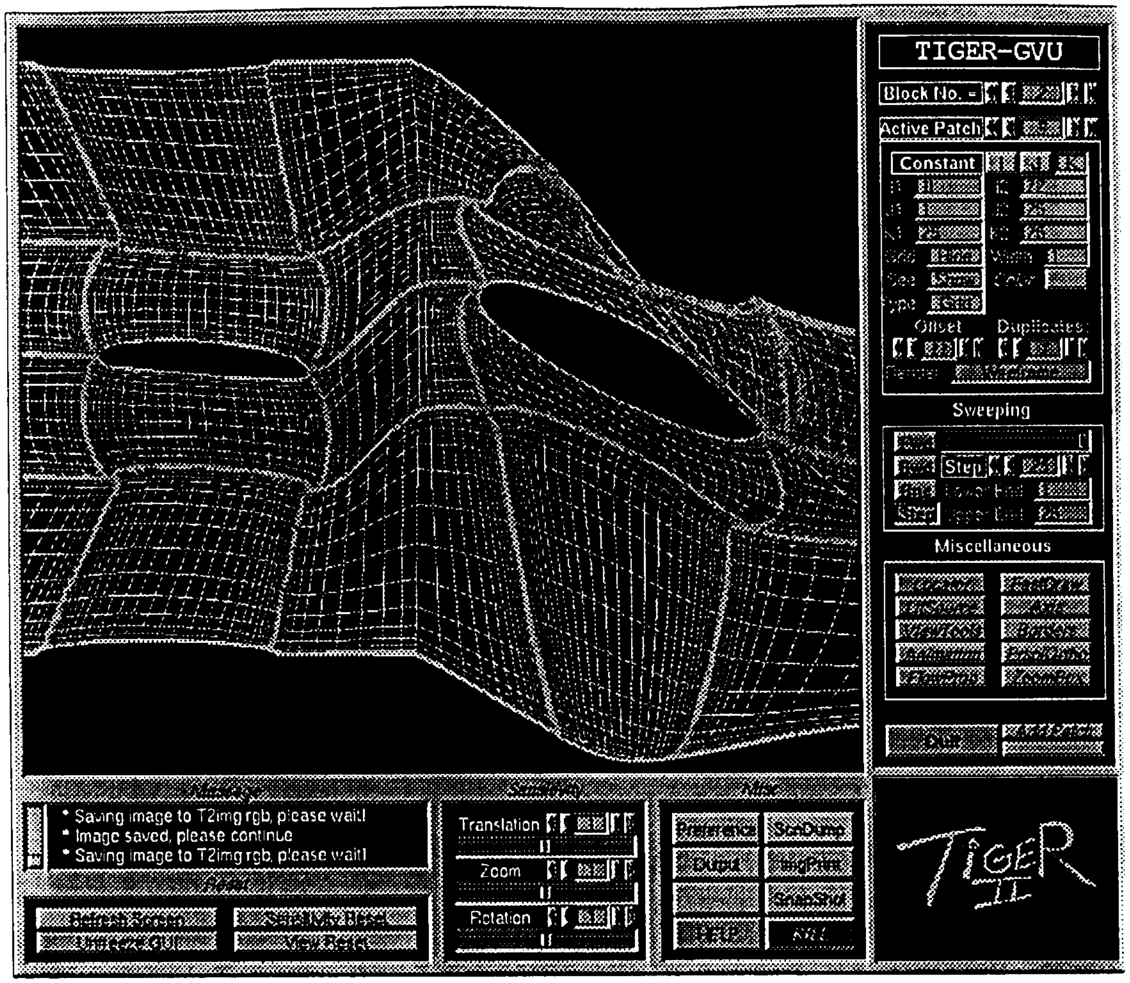
- Saving image to T2img.rgb, please wait!
- Image saved, please continue
- Saving image to T2img.rgb, please wait!

Figure 6.4 (a) High–Reynolds–Number Pump



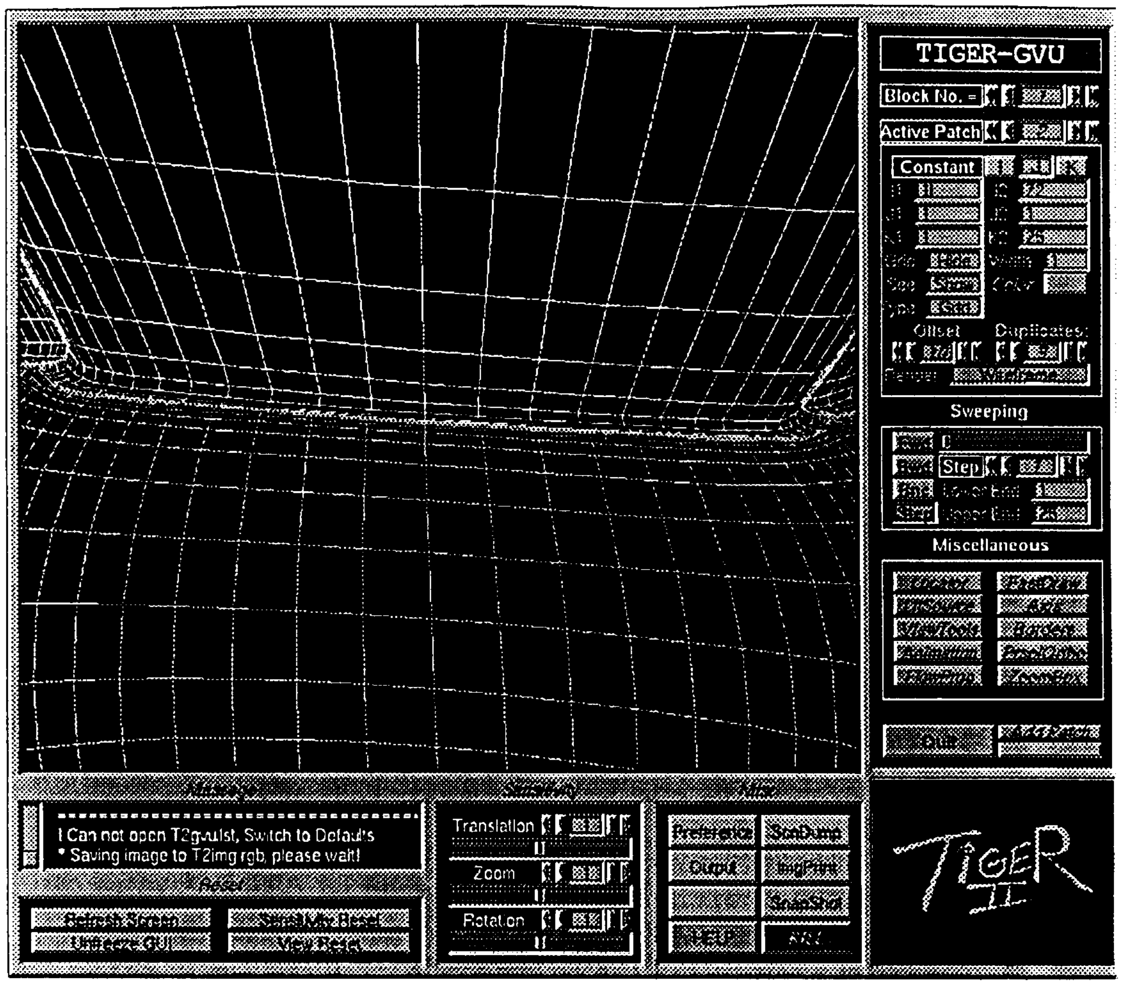
! Can not open T2gvulst, Switch to Defaults
! Exceed the block size
* Saving image to T2Img.rgb, please wait!

Figure 6.4 (b) Overall Grid Behavior



```
* Saving image to T2img.rgb, please wait!
* Image saved, please continue
* Saving image to T2img.rgb, please wait!
```

Figure 6.4 (c) Surface Grid near the Blades



I Can not open T2gvulst, Switch to Defaults
* Saving image to T2img rgb, please wait!

Figure 6.4 (d) Surface Grid near the Blade Fillet Region was extended upstream for the shroud. The impeller was also taken from the SIMPLEX turbopump design as presented above. The distance between the inducer and the impeller, however, was not accurately modeled since the data was not available. The generatrix for hub and the shroud were extended downstream from the exit of the impeller to form a turn–around duct that faces upstream. The meridional frame setting of this geometry is shown in Figure 6.5(b).

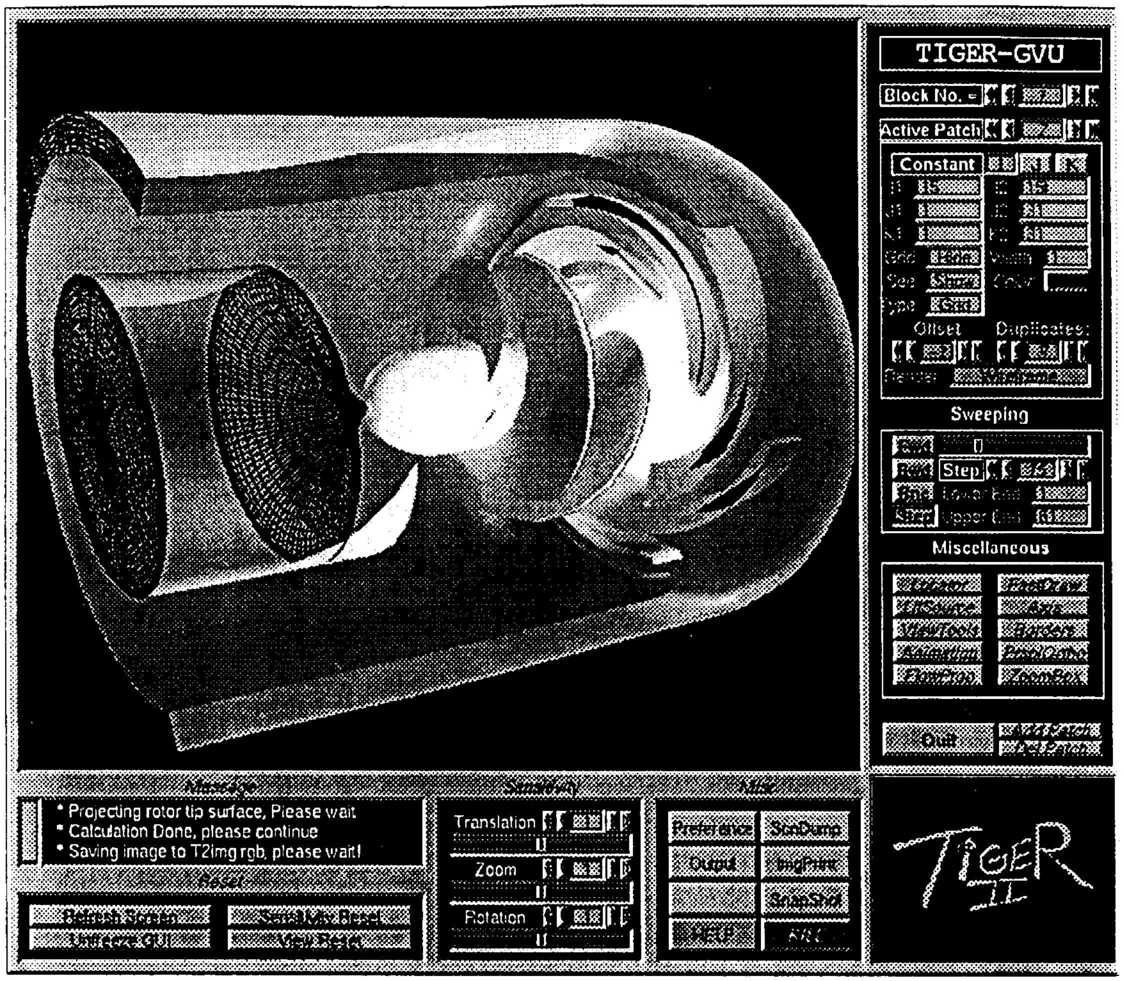
This grid was generated in two blocks. The first block has a grid size of 61x31x31 per passage, while the second block has a grid size of 61x31x16. A tip clearance of 1% of the blade diameter was modeled with three grid cells inside. The grid of the entire geometry is demonstrated in Figure 6.5(c). The impeller was modeled without tip clearance. The close-up of the inducer and the impeller geometry is demonstrated in Figure 6.5(d). This grid, however, does not have detailed manipulation and quality control, since the purpose is to test the robustness of the overall algorithm.

External Flow Field

A geometry is characterized as an external flow field configuration if its outer domain of computation is not confined by solid geometry but free–stream. Two cases are presented in this category. They are a missile mockup, and an underwater weapon system.

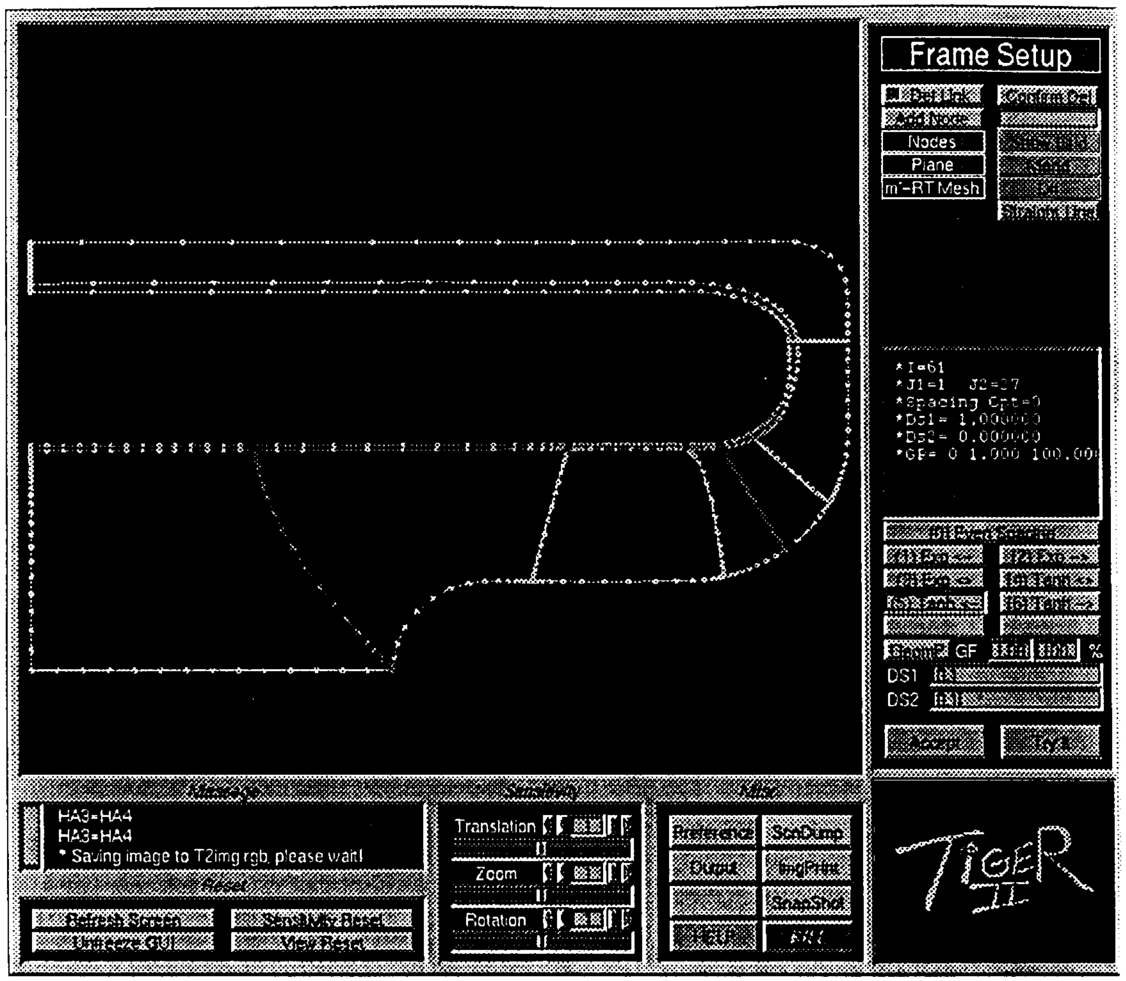
Missile Configuration

A missile with fins may be treated as a "turbomachinery" configurations as well, as long as the arrangement of the fin retains periodicity. This example is an missile mockup which has three rows of fins (4+6+4) as shown in Figure 6.6(a). This geometry is generated in three axial blocks, with their



* Projecting rotor tip surface, Please wait
* Calculation Done, please continue
* Saving image to T2img.rgb, please wait!

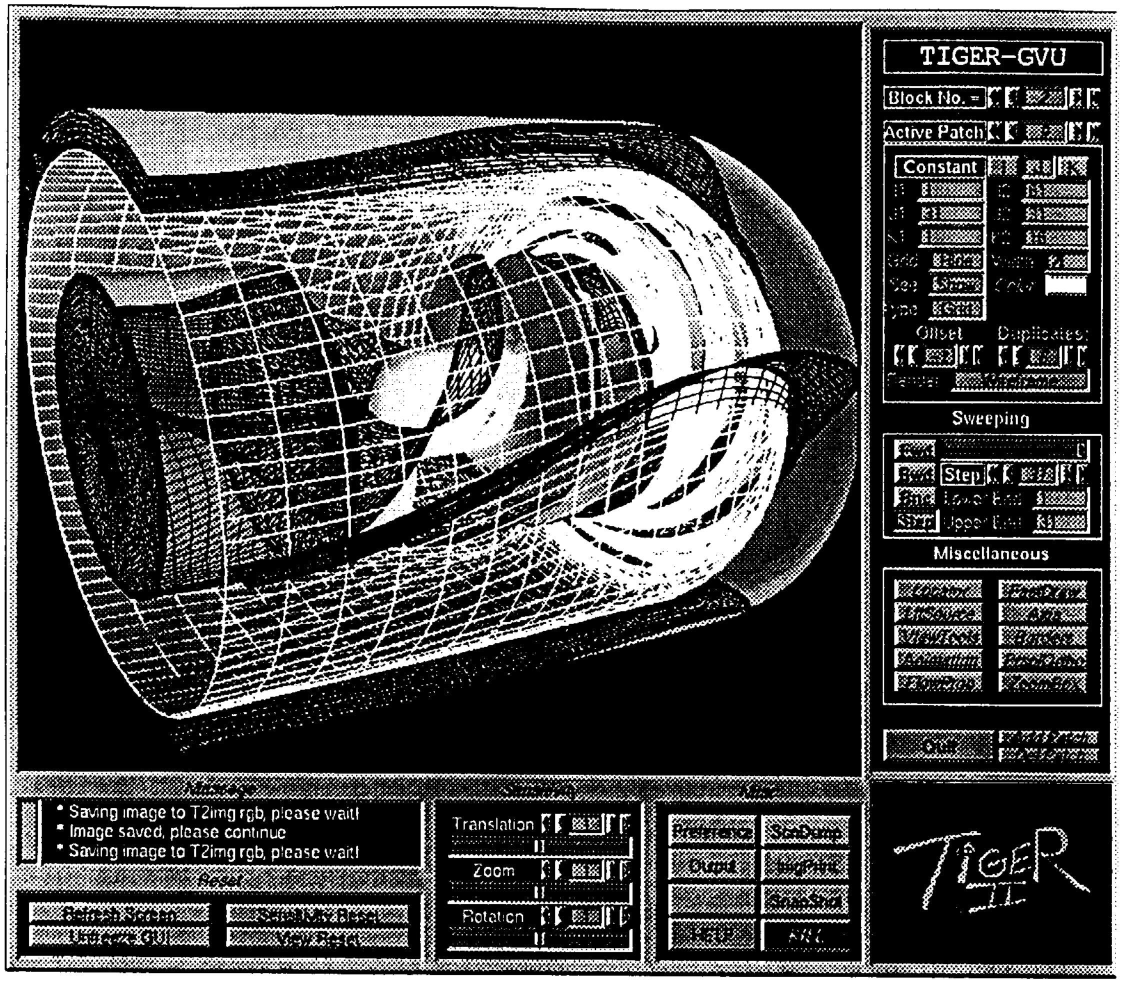
Figure 6.5 (a) Research Configuration: Axial–Radial Mixed–Flow Configuration



```
*I=61
*J1=1   J2=27
*Spacing Cpt=0
*DS1= 1.000000
*DS2= 0.000000
*GF= 0 1.000 100.00
```

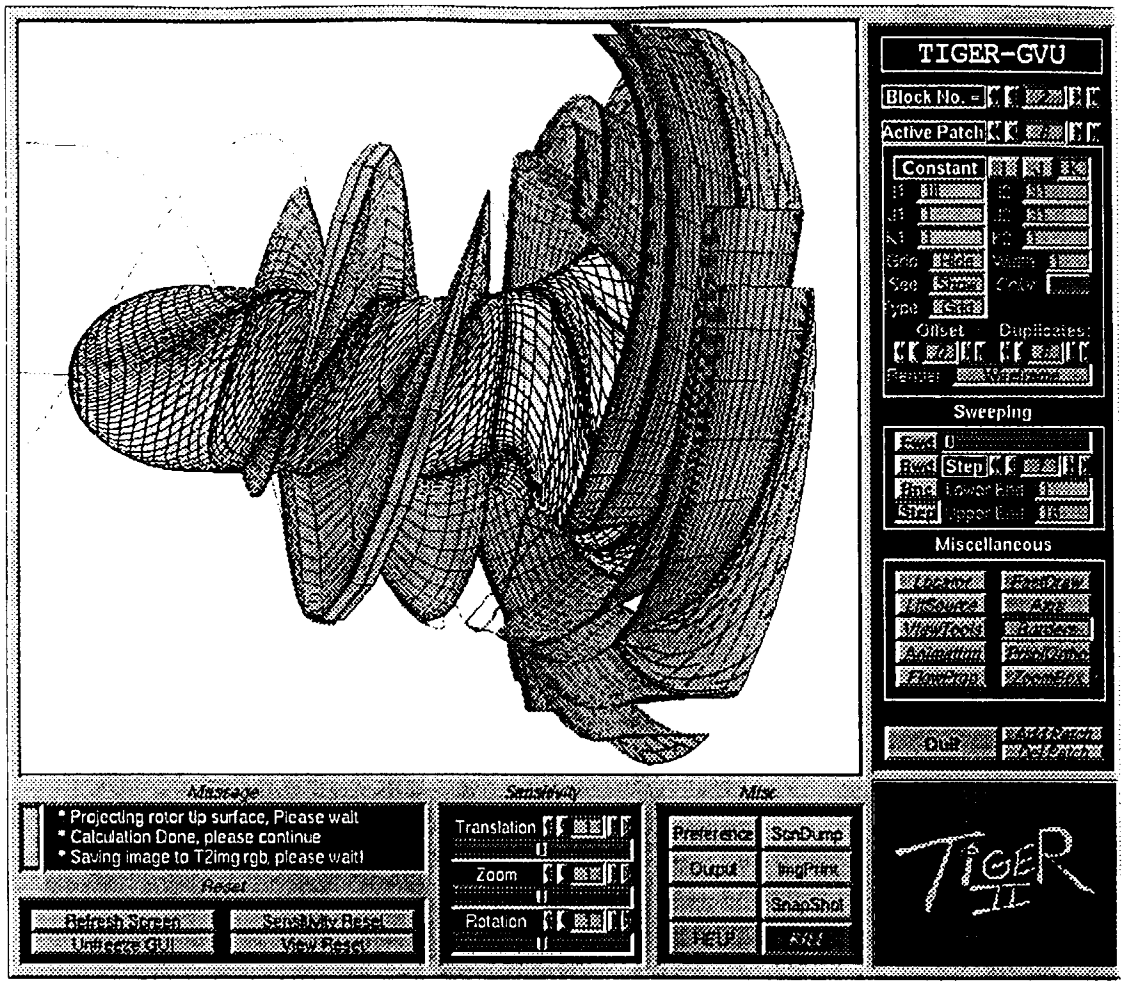
```
HA3=HA4
HA3=HA4
* Saving image to T2img.rgb, please wait!
```

Figure 6.5 (b) Meridional Frame Setting



• Saving image to T2img rgb, please wait!
• Image saved, please continue
• Saving image to T2img rgb, please wait!

Figure 6.5 (c) Overall Grid Behavior



* Projecting rotor tip surface, Please wait
* Calculation Done, please continue
* Saving image to T2img.rgb, please wait!

Figure 6.5 (d) Blade Region Close–up

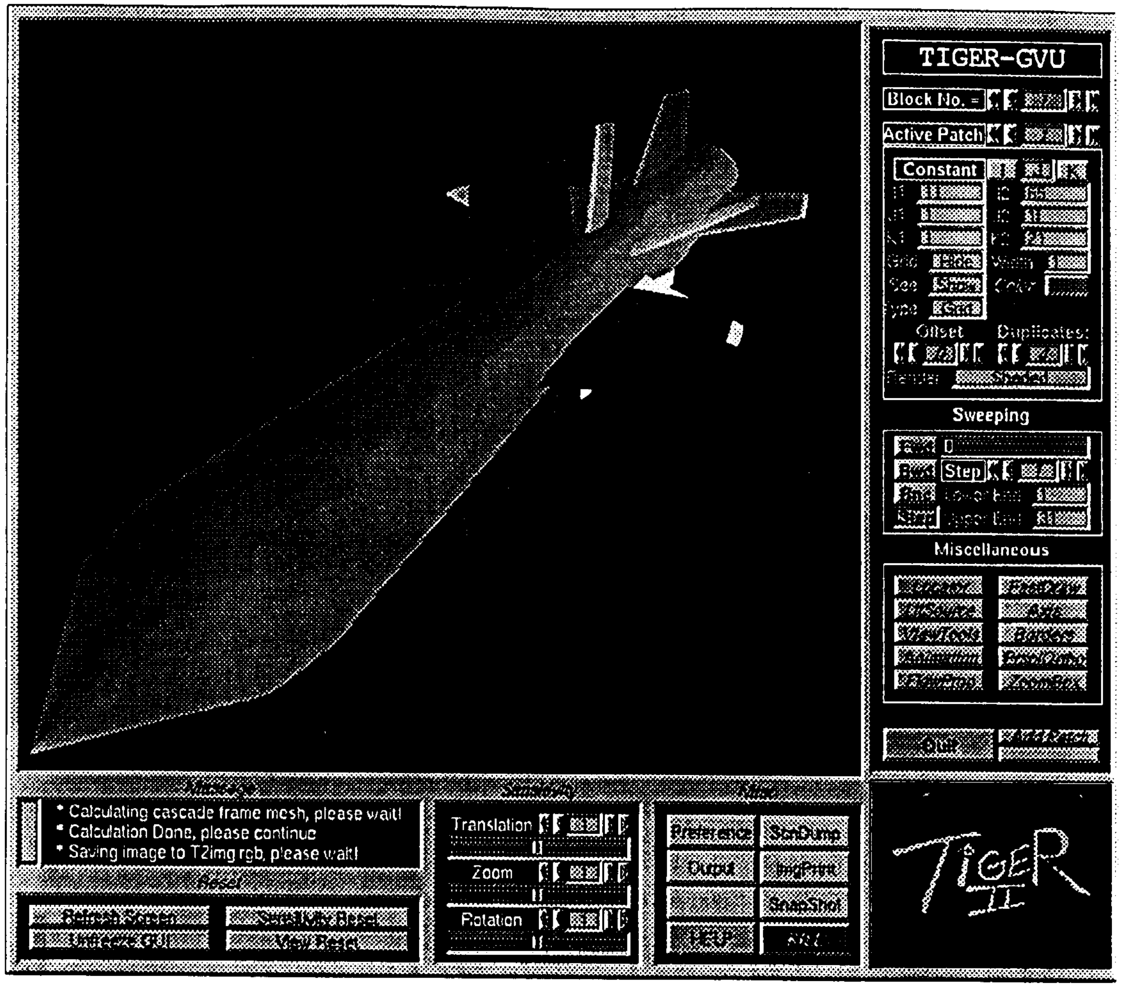
sizes of 65x31x22, 37x31x15, and 81x31x22. Note that the number of total circumferential grid cells for each block is exactly 84. The surface grid is shown in Figure 6.6(a). This geometry is generated in three axial blocks, with their sizes of 65x31x22, 37x31x15, and 81x31x22. Note that the number of total circumferential grid cells for each block is exactly 84. The surface grid is shown in Figure 6.6(b), while a ξ=constant plane is illustrated in Figure 6.6(c). This grid was accomplished in 40 minutes.

Underwater Weapon System

This geometry has four fins, followed by eight stators, and six rotors (propulsors). As illustrated in Figure 6.7(a), the stators have 22.5 degree offset from the position of the fins. TIGER requires the first circumferential grid line to be continuous due to the constraint of the flow code, TURBO[30]. Therefore, the grid was generated without the offset angle and explicitly turned the stator block by 22.5 degrees with *MICE* after the completion of grid construction. Blunt nose and tail of the body creates another difficulty for surface mapping. The body was decomposed into three sections by the designed algorithm, and therefore was able to retain the sharp corner. However, the highly swept rotor blades induces kinks while trying to match the grid lines with the stator block. Only an Euler grid was obtained for this geometry. The total grid was constructed with three grid blocks.

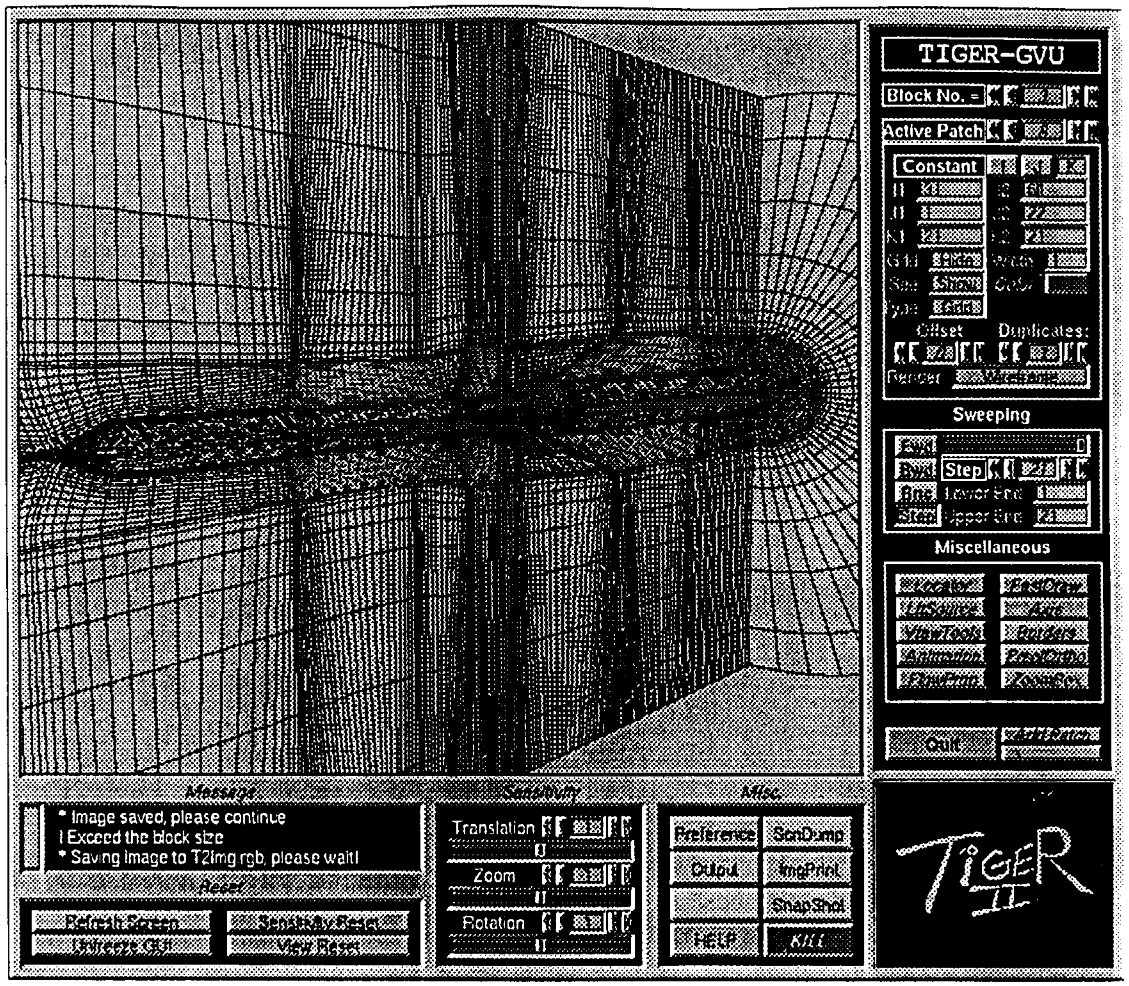
Internal–External Flow Field

A geometry is characterized as an external flow field configuration if its outer domain of computation does not have the presence of solid geometry, but part of its domain is confined by a solid geometry. The cases presented in this



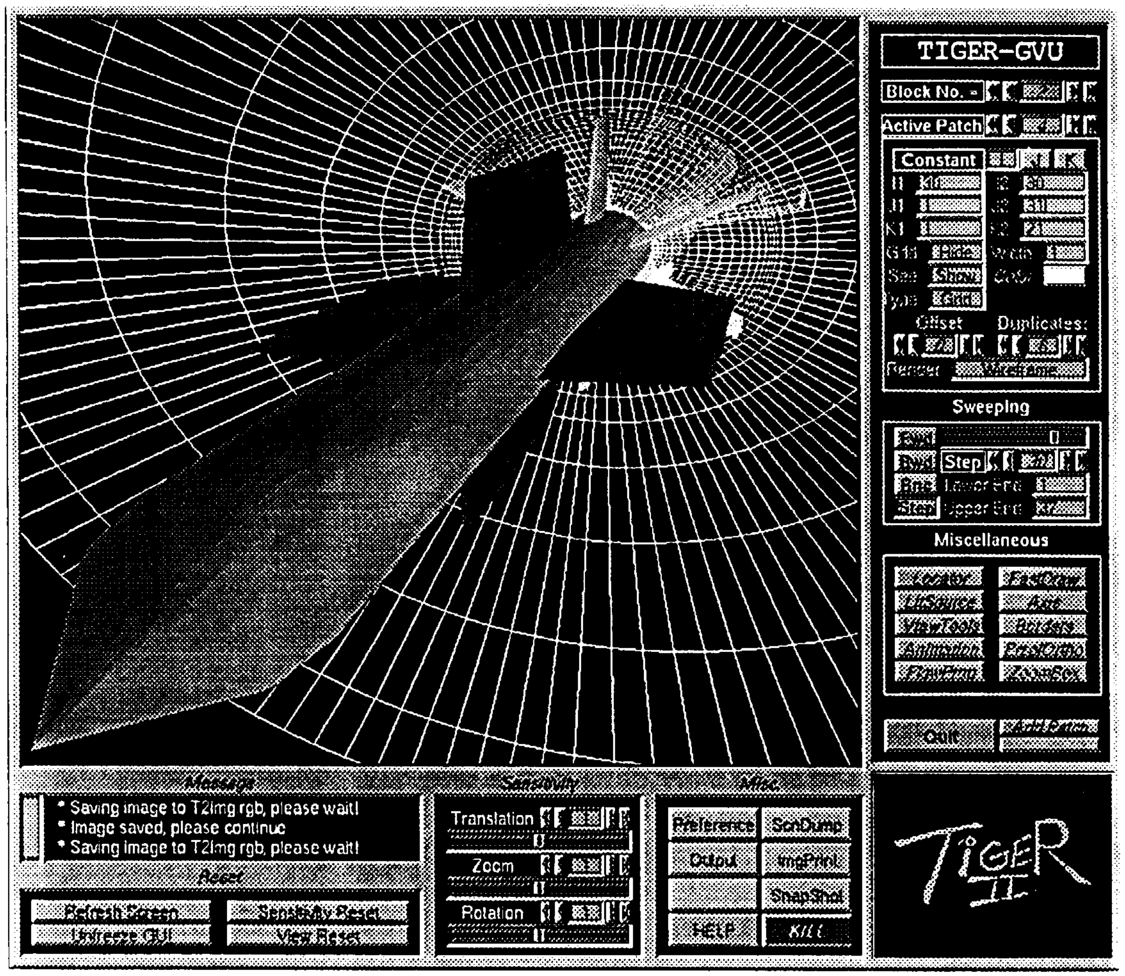
* Calculating cascade frame mesh, please wait!
* Calculation Done, please continue
* Saving image to T2img rgb, please wait!

Figure 6.6 (a) Research Geometry: Missile Mockup



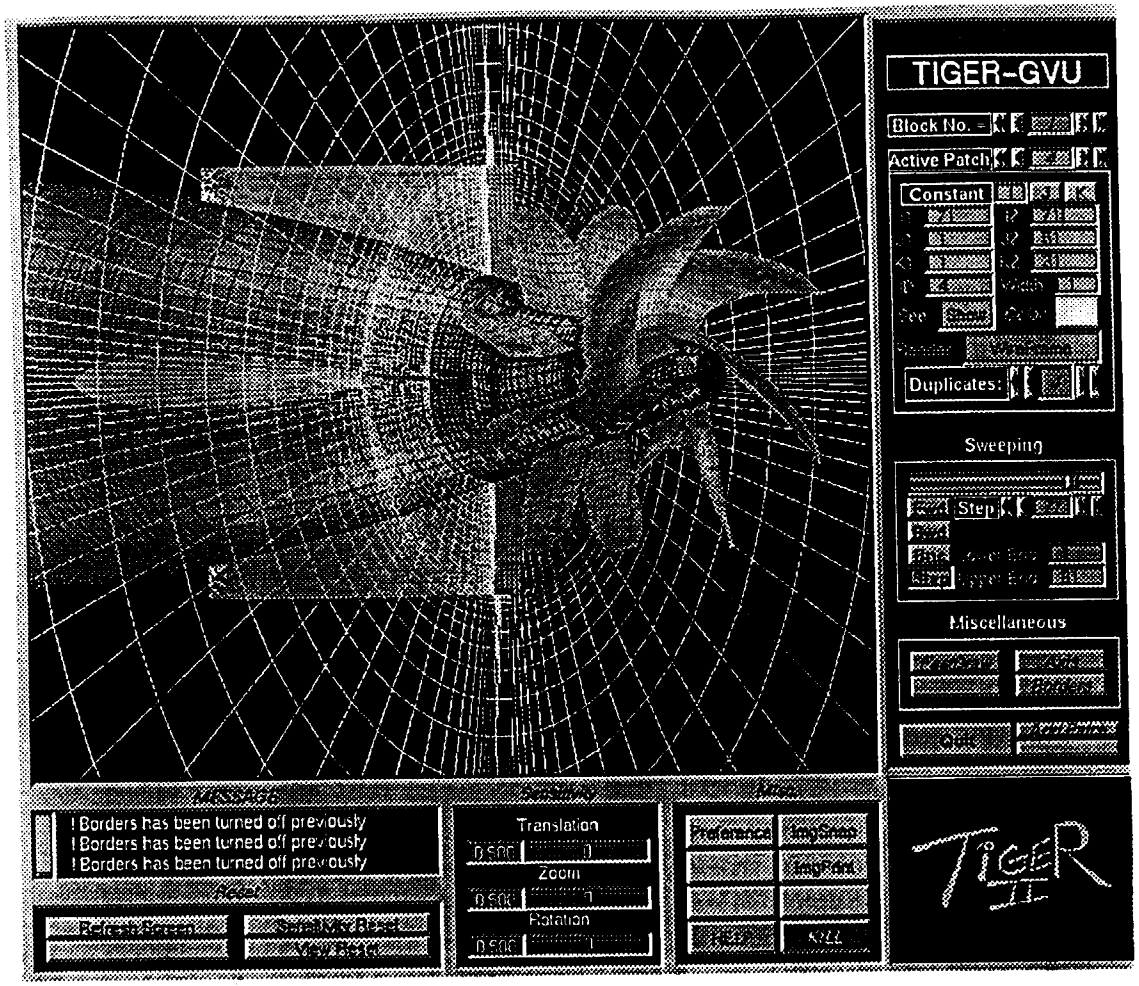
```
* Image saved, please continue
! Exceed the block size
* Saving Image to T2img.rgb, please wait!
```

Figure 6.6 (b) Surface Grid Behavior



```
* Saving image to T2img rgb, please wait!
* Image saved, please continue
* Saving image to T2img rgb, please wait!
```

Figure 6.6 (c) Cross–Section Grid Behavior along the Flow Direction



! Borders has been turned off previously
! Borders has been turned off previously
! Borders has been turned off previously

Figure 6.7 (a) Underwater Weapon System

category are two ultra–high bypass ratio turbofans, and an engine mockup with multiple ducts.

NASA 1.15 Pressure Ratio Fan Stage

The first configuration in this category is a 1.15 pressure ratio fan stage tested extensively at NASA[71–74]. A technical description of this 25:1 ultra–high bypass ratio turbofan is illustrated in Figure 6.8(a). The actual geometry contains 12 rotor blades and 32 stator blades. However, it was numerically modeled as 16 rotor blades with 40 stator blades due to the consideration of numerical expense. An H–grid of four blocks is generated for the full configuration. The block structure is the same as shown in Figure 3.1(a). The grid sizes of each block per passage are: 81x25x26, 81x25x11, 81x16x26, and 81x16x11. The meridional frame after manipulation is shown in Figure 6.8(b). The cascade frame after manipulation has been illustrated in Figure 3.12. The corresponding meridional surface grid is demonstrated in Figure 6.8(c).

An automatic surface projecting technique as described in §3.2.4 is used to create the grid in a timely fashion. Results are also shown in Chapter III. Total man–hours to obtain a decent grid is no more than four to six hours for this configuration.

Modern Ultra–High Bypass Ratio Turbofan

Designed by a major engine manufacturer, this configuration has 16 rotor blades followed by 22 stator blades. A wind tunnel model configured with short nacelle is presented in Figure 6.9(a). Illustrated in Figure 6.9(b) is the shaded numerical model configured with long nacelle. Due to the numerical constraint of one of the flow codes, ADPAC[75], that is going to be used for the calculation, the hub nose has been modified such that there is a finite radius

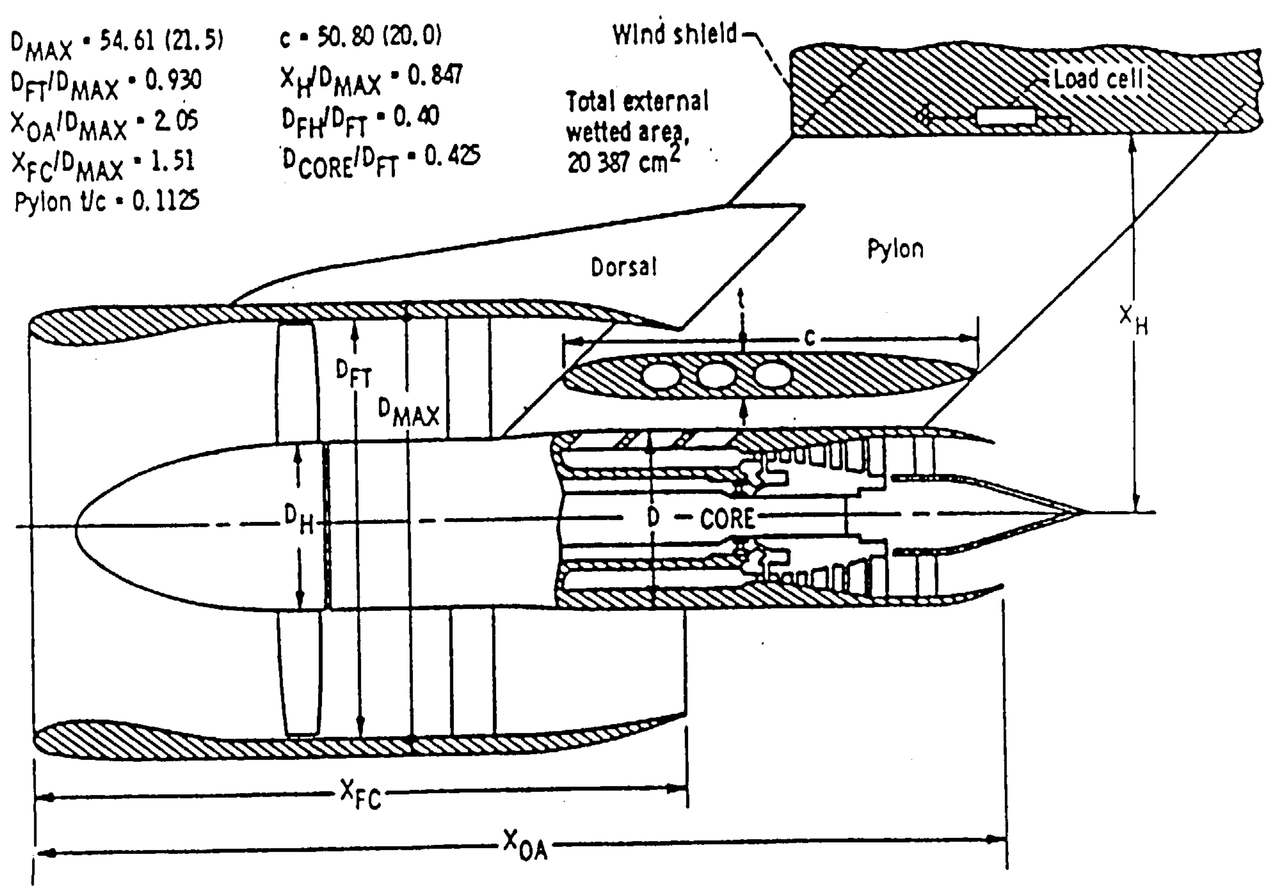
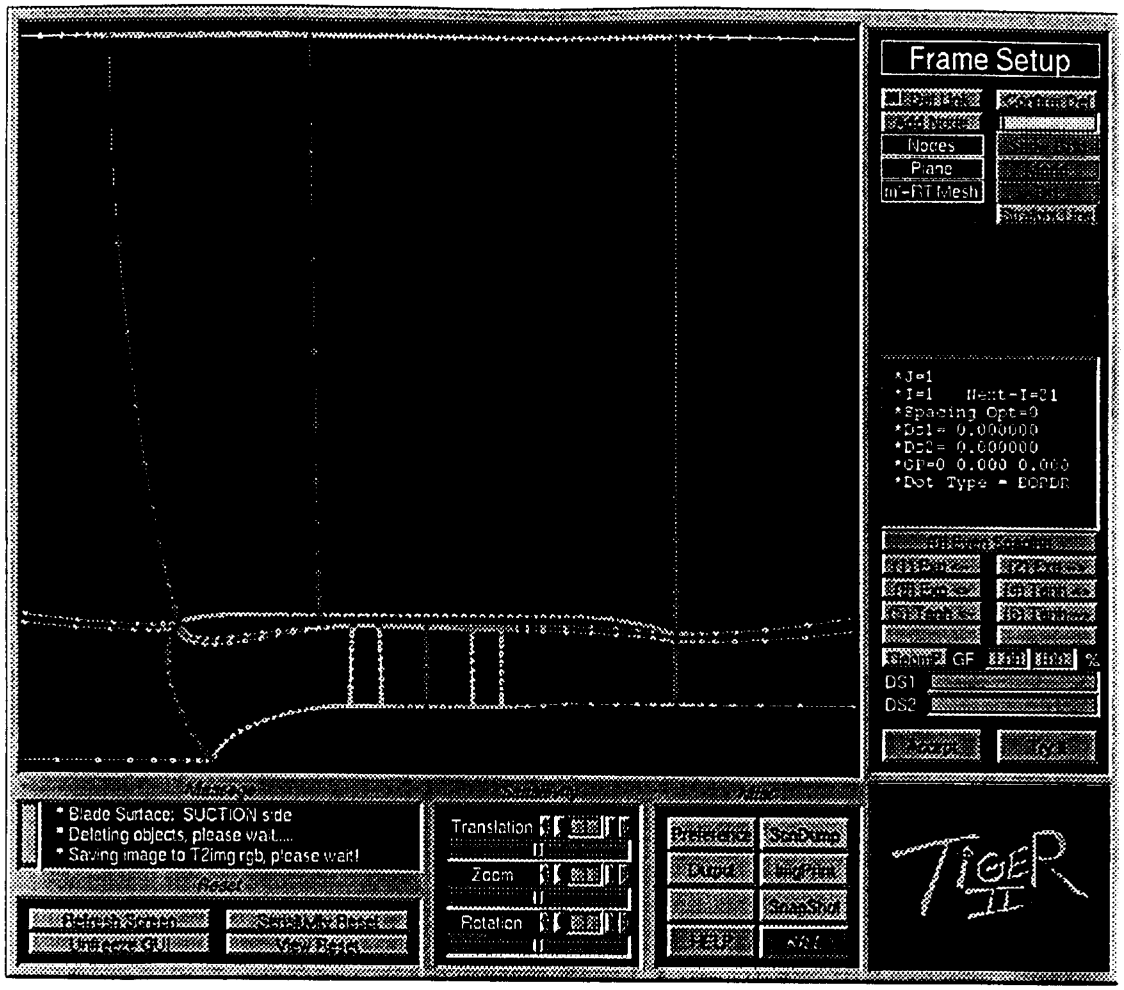


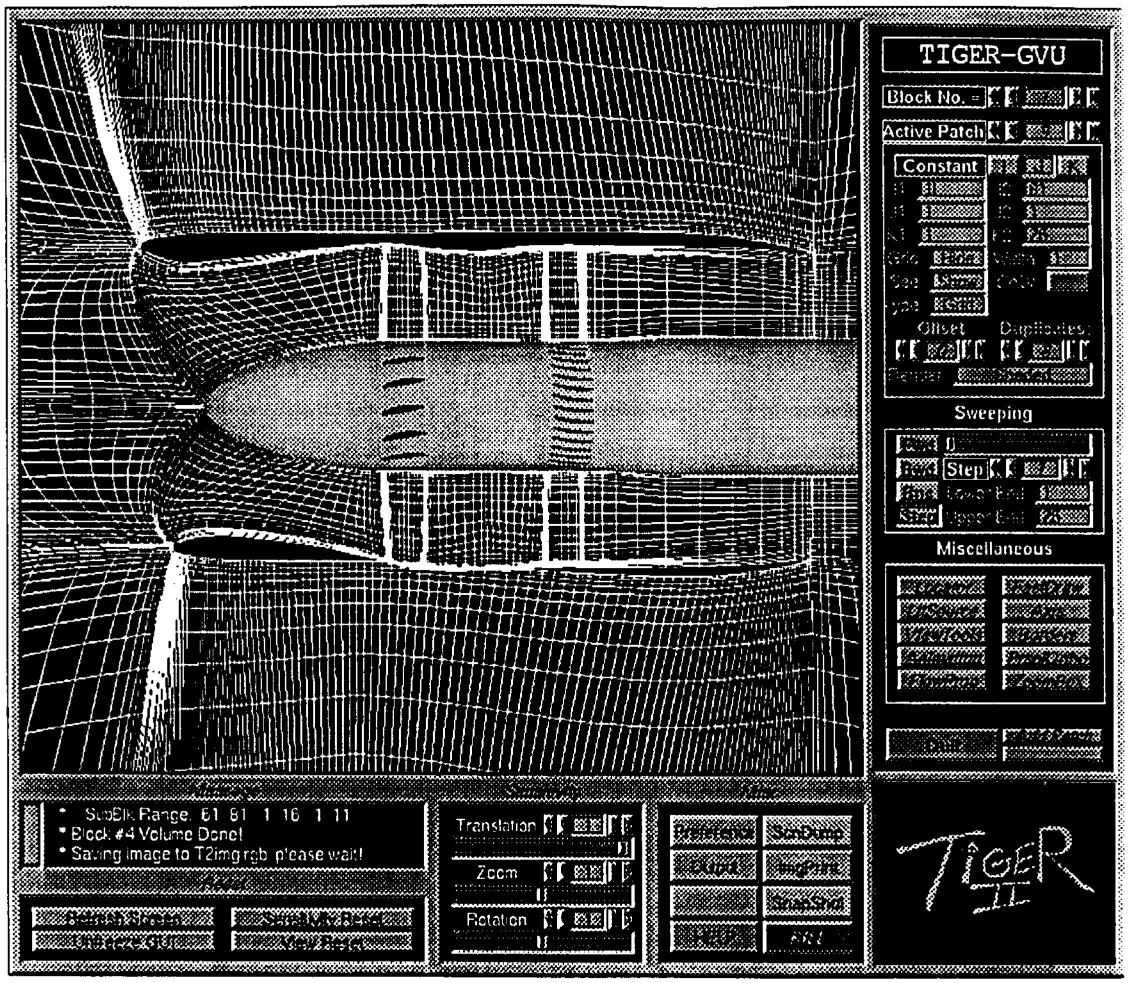
Figure 6.8 (a) NASA 1.15 Pressure Ratio Fan Stage



```
*J=1
*I=1    Next-I=21
*Spacing Opt=0
*DS1= 0.000000
*DS2= 0.000000
*CP=0 0.000 0.000
*Dot Type = DOPDR
```

* Blade Surface: SUCTION side
* Deleting objects, please wait....
* Saving image to T2img.rgb, please wait!

Figure 6.8 (b) Meridional Frame Setting



```
* SubBlk Range: 61 61 1 16 1 11
* Block #4 Volume Done!
* Saving image to T2img.rgb please wait!
```

Figure 6.8 (c) Meridional Surface Grid

sting extended upstream of the nose, with a smooth slope transition section. An H–grid that consists of four blocks was generated. The block structure is the same as shown in Figure 3.1(a). Two grids with different grid sizes were generated. The fine grid, used for ADPAC to obtain time–averaged solution, has the grid sizes of 129x37x23, 93x37x17, 129x37x23, and 93x37x17 per passage. ADPAC's time–averaging approach needs to model only one passage for each block. This grid, however, is too big for the flow code TURBO since it is a time–accurate flow code that needs to model half of the geometry due to the 16x22 blade count configuration. A total number of about three million grid points would be needed for the calculation using TURBO if fine grid was used. Therefore, a coarse grid was built with the size of 75x31x23, 65x31x17, 75x17x23, and 65x17x17 per passage. Using this coarse grid reduces the required memory by 2/3. Both grids were built in the same fashion, despite the difference in grid size, in order to be able to compare the results. The grid presented in this section is the fine grid. The meridional surface grid is shown in Figure 6.9(c). The cut–out of the geometry is presented in Figure 6.9(d). Several ξ=constant surfaces are shown in Figure 6.9(e).

Engine Mockup

In an attempt to validate the applicability of this work to more complicated engine configurations, namely, multiple ducts, an engine mockup was constructed for testing. The technical description of the true engine that this mockup emulates is shown in Figure 6.10(a). This mockup, though contains non–smooth curves as generatrix due to the digitization error, retains the general shape of an engine. The guide vane was created by stacking a series of blade profiles. Only six guide vanes were modeled in order to form a wider passage allowing better view on the inner structure. This geometry is as-

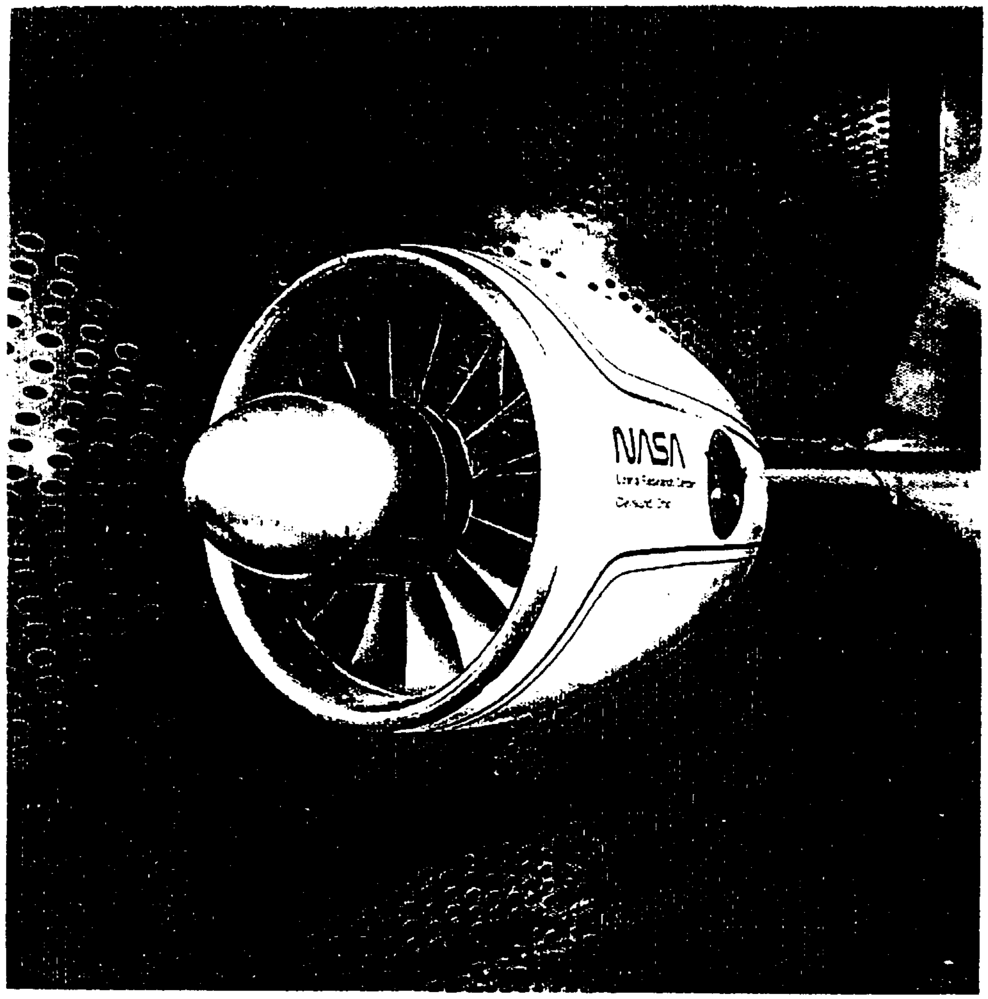
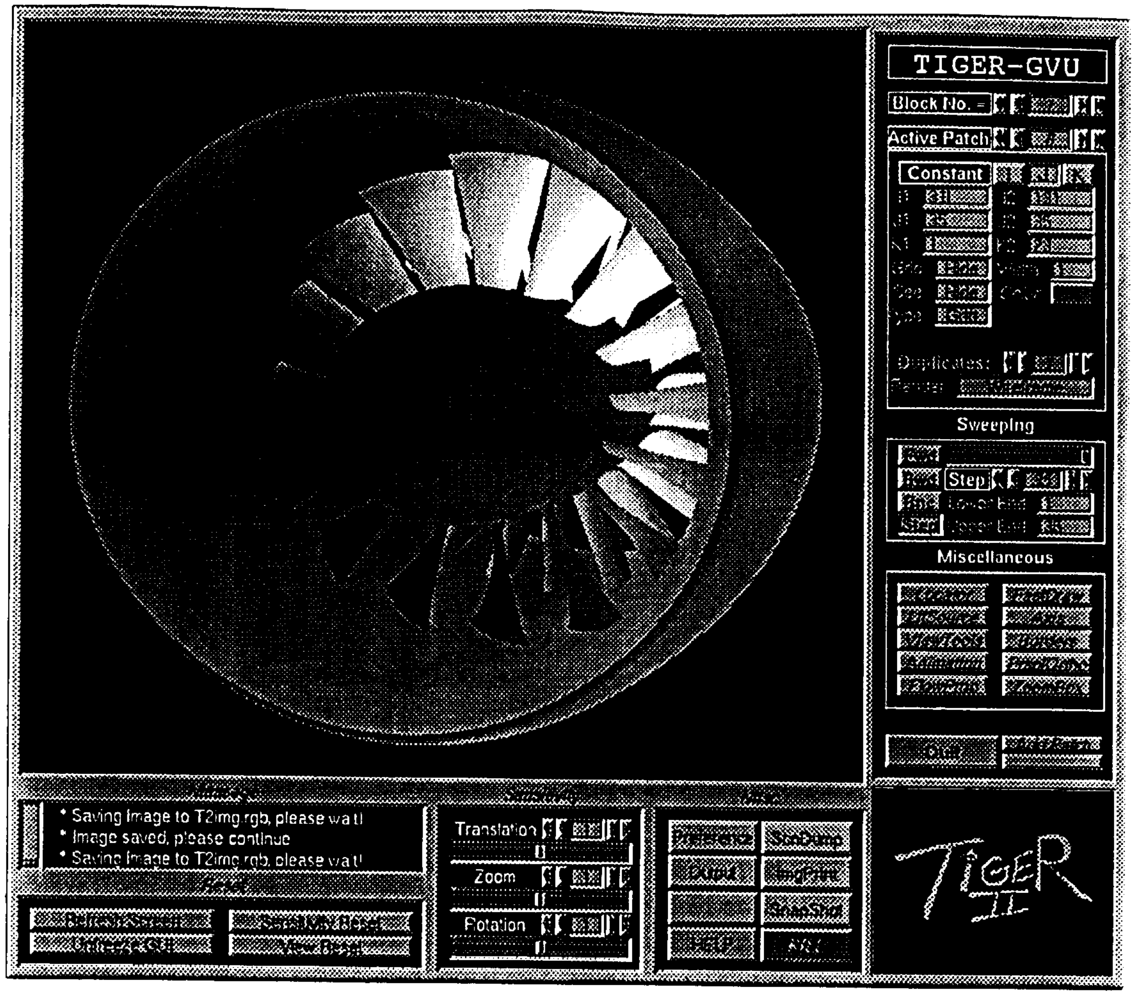
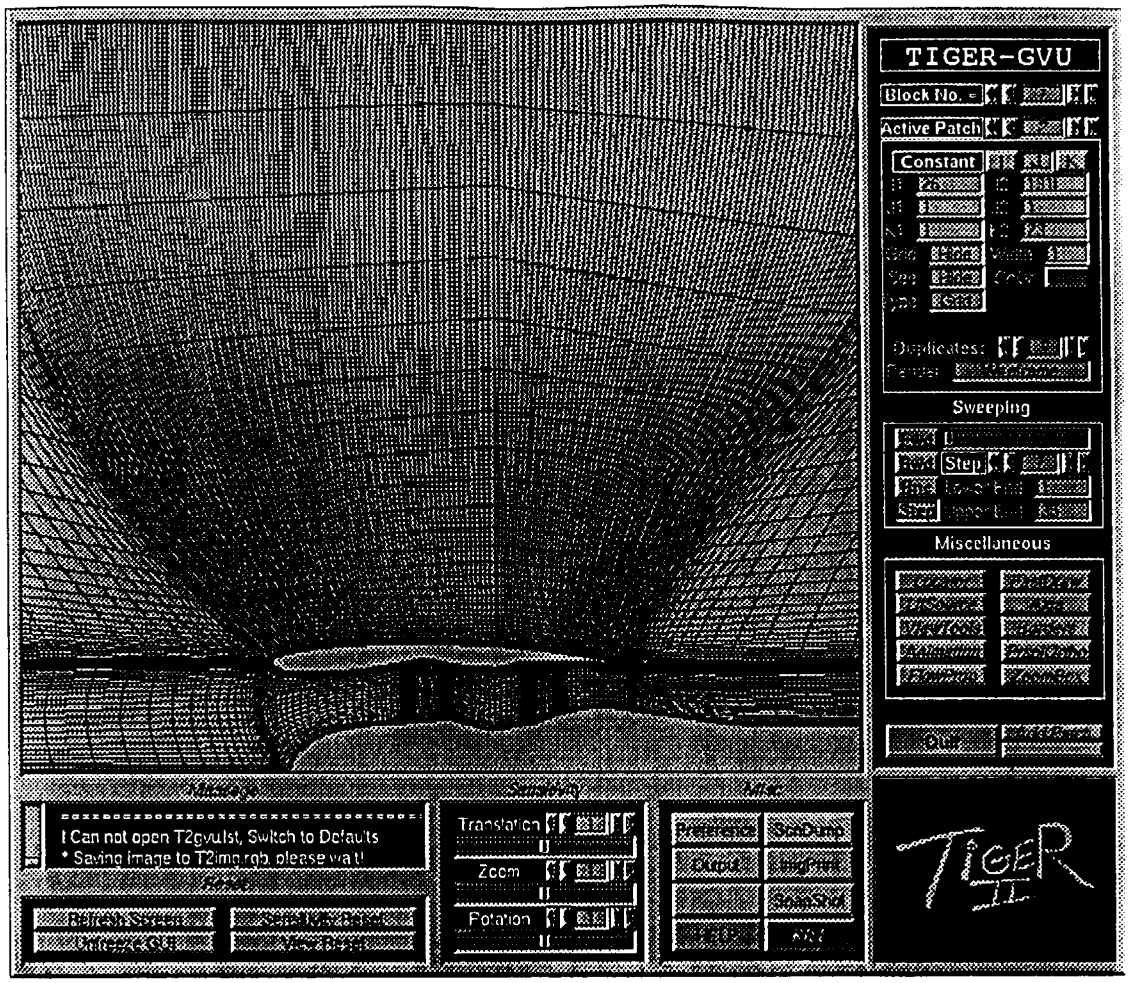


Figure 6.9 (a) Wind Tunnel Model of a Modern Ultra–High Bypass Ratio Turbofan



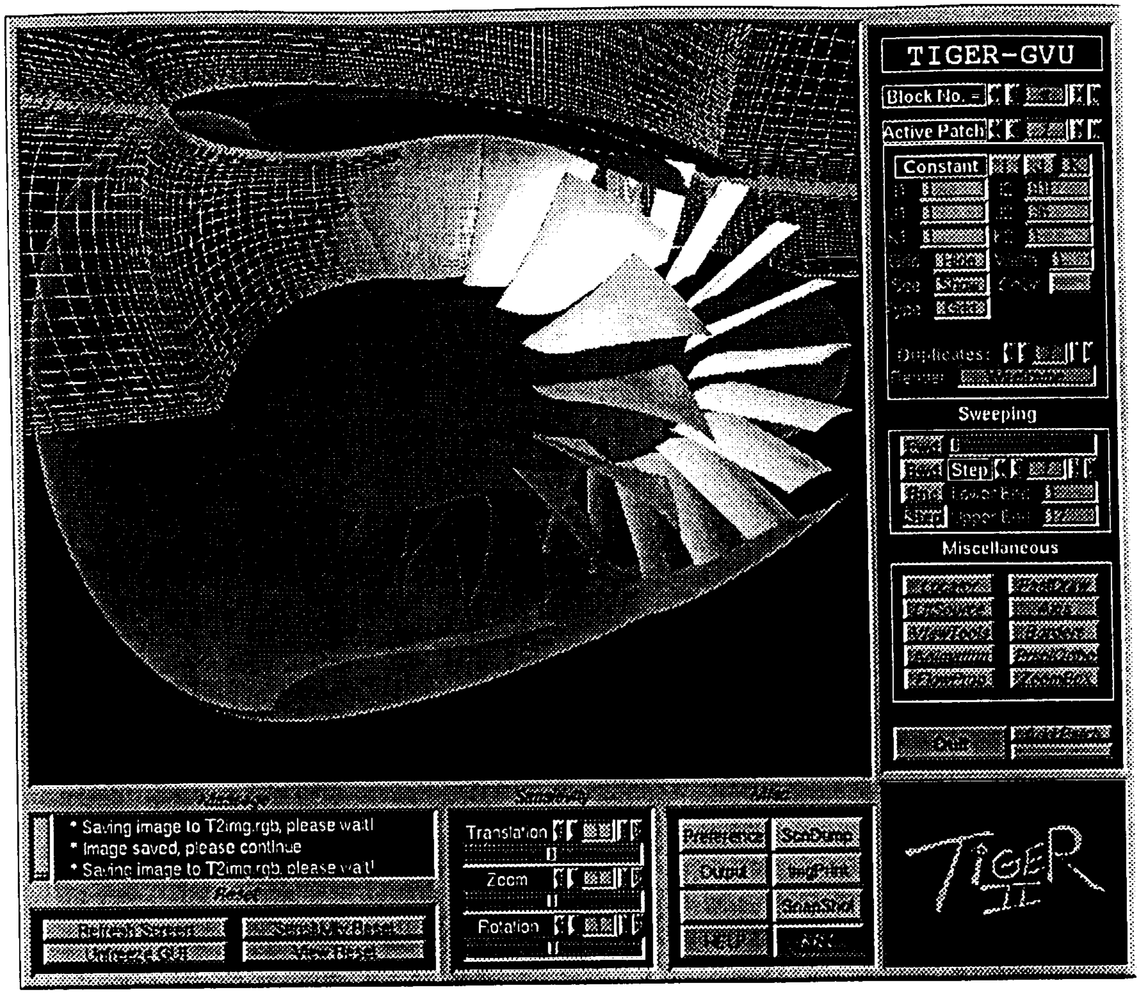
- Saving Image to T2img.rgb, please wait!
- Image saved, please continue
- Saving Image to T2img.rgb, please wait!

Figure 6.9 (b) Numerical Model



! Can not open T2gvu.lst, Switch to Defaults
* Saving image to T2img.rgb, please wait!

Figure 6.9 (c) Meridional Surface Grid



```
* Saving image to T2img.rgb, please wait!
* Image saved, please continue
* Saving image to T2img.rgb, please wait!
```

Figure 6.9 (d) Geometry Cut–out View

sumed to have 2 core flow splitters and an outer nacelle. A total of five radial grid blocks were used to construct the entire grid. The overall structure of the engine flow path is demonstrated in Figure 6.10(b). Demonstrated in Figure 6.10(c) is the overall grid.

Simulation Applications

The simulation module of TIGER attempts to reduce the man–hours wasted during the iteration in submitting a flow calculation job on a mainframe, transferring the solution data back to local graphics workstation, and then executing a post–processing software to visualize the solution during the initial phase of flow calculation. This idea has been exercised on two external geometries. However, it is only a matter of the setup of the flow code rather than the modification of TIGER to go beyond these simple configurations for more complicated applications. In the following two cases, the turbomachinery flow code, MSUTC (Multi–Stage Unsteady Turbomachinery Code) is used as the flow module. This flow solver may be replaced by other appropriate flow codes as well. Only minor modifications are needed.

Fin–Body Geometry

The first case presented is a fin–body geometry shown in Figure 6.11(a). This mockup has four fins attached to the body. A 41x36x21 C–type grid is generated and transferred to a remote machine through TIGER's network module. Although MSUTC is a compressible turbomachinery flow code capable of modeling turbulence flow, this geometry is modeled by setting the advance ratio J=0. The Euler solution was obtained for an axial flow Mach number of 1.9. Each iteration of flow calculation will take about 30 seconds on a SGI Challenge machine with 4x100MHz processors. The density contours of

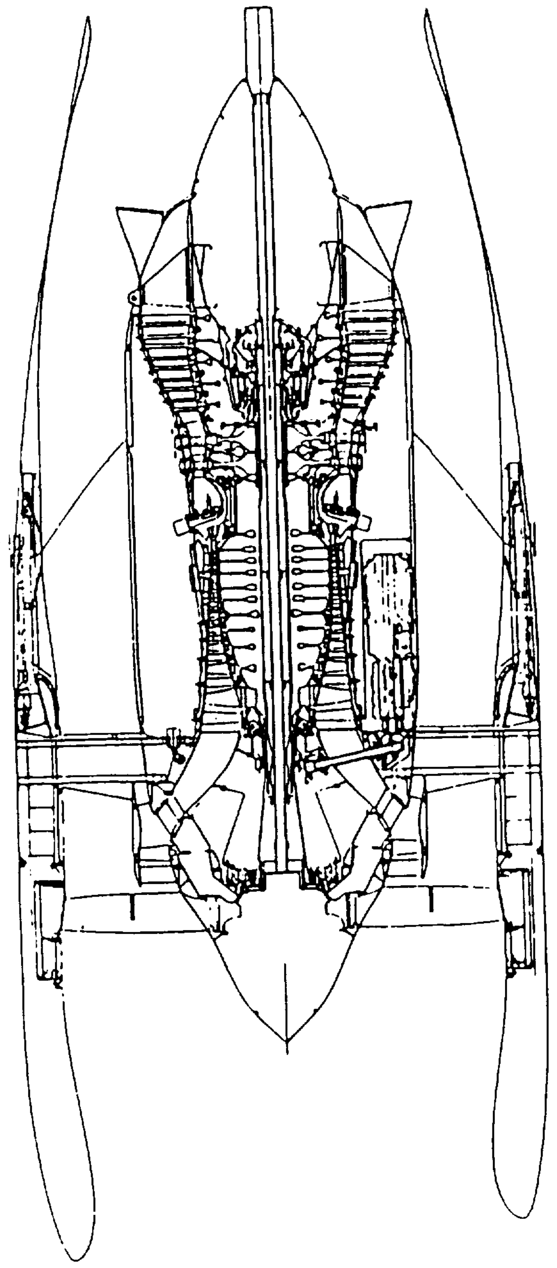
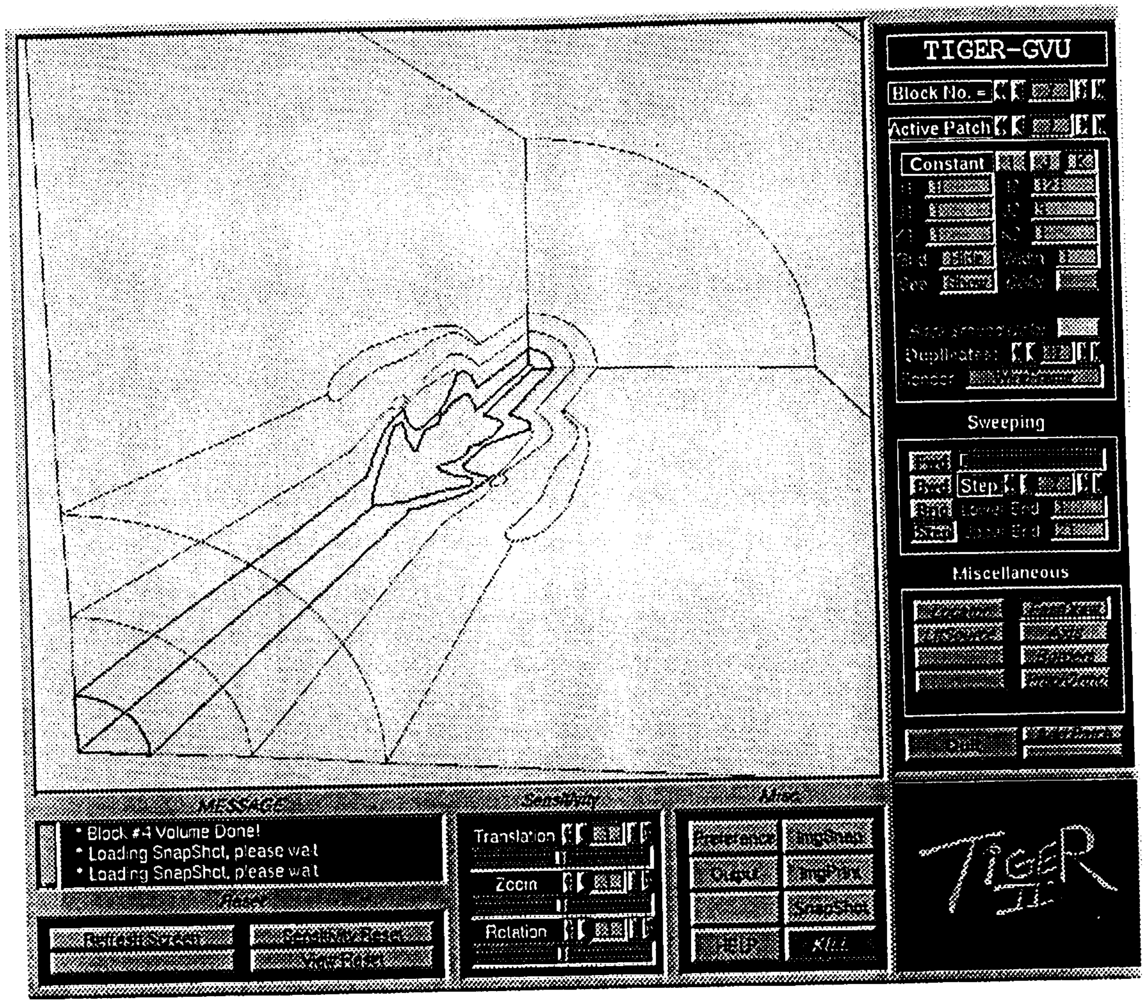
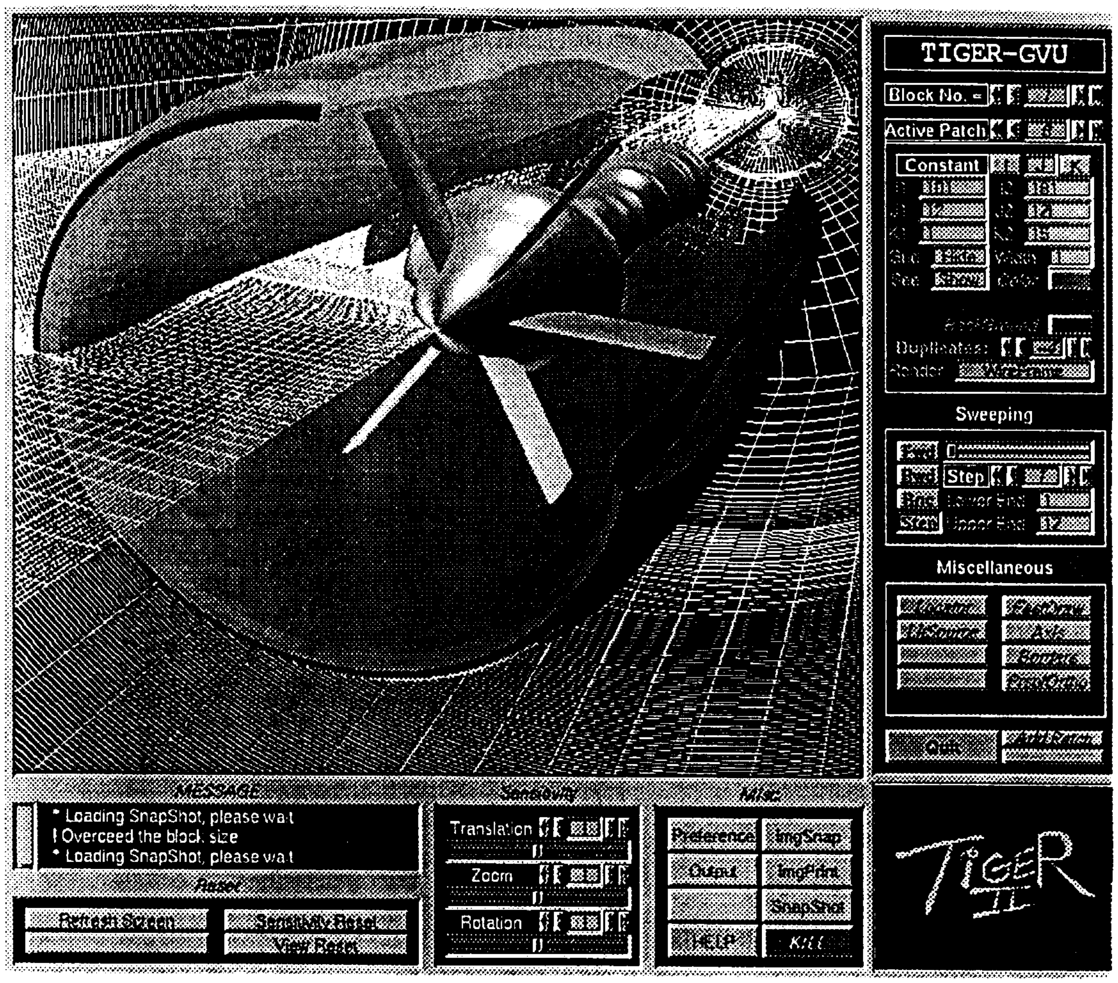


Figure 6.10 (a) Technical Layout of a Modern Engine



• Block #4 Volume Done!
• Loading SnapShot, please wait
• Loading SnapShot, please wait

Figure 6.10 (b) Flow Path Layout



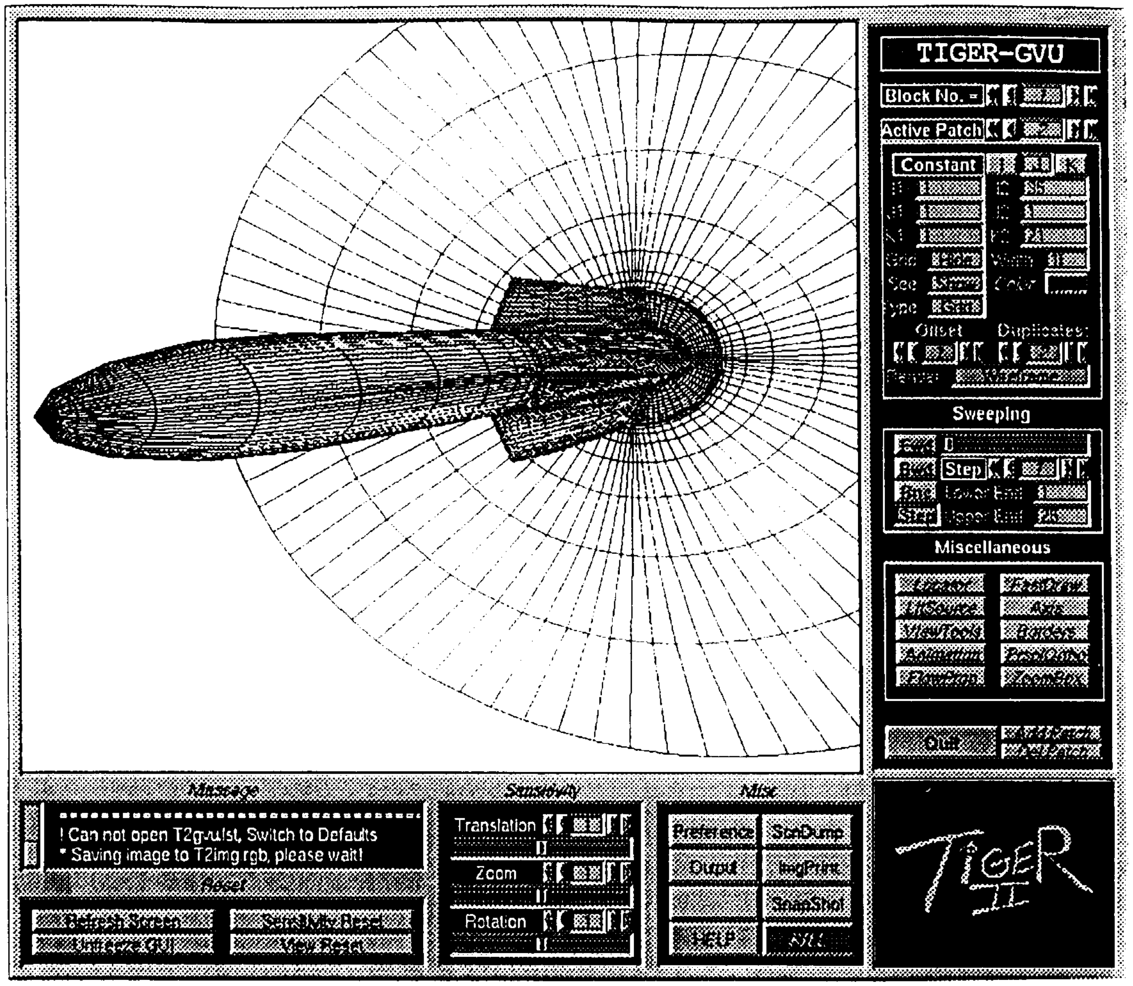
* Loading SnapShot, please wait
! Overceed the block size
* Loading SnapShot, please wait

Figure 6.10 (c) Overall Grid Behavior

the 1st, 20th, and 2000th iteration are shown in Figures 6.11(b) to 6.11(d), respectively. The first two snapshots were taken consecutively after the grid was generated and the flow field was simulated. After the 20th iteration, TIGER was terminated while the flow module was still active. To take a snapshot for the 2000th iteration, TIGER was executed again after the flow calculation was completed. This was done by accessing TIGER's GVU visualization module, and re–establishing the network connection using the network module. Once the network connection has been restored, a user may push one button labelled "*Update*" on the screen to activate the automatic update algorithm. The final solution file is automatically transferred back to the local graphics workstation to update the solution array. This algorithm eliminates the tedious *ftp* procedure. It also frees the local machine from the connection with the mainframe while it is executing a long run.

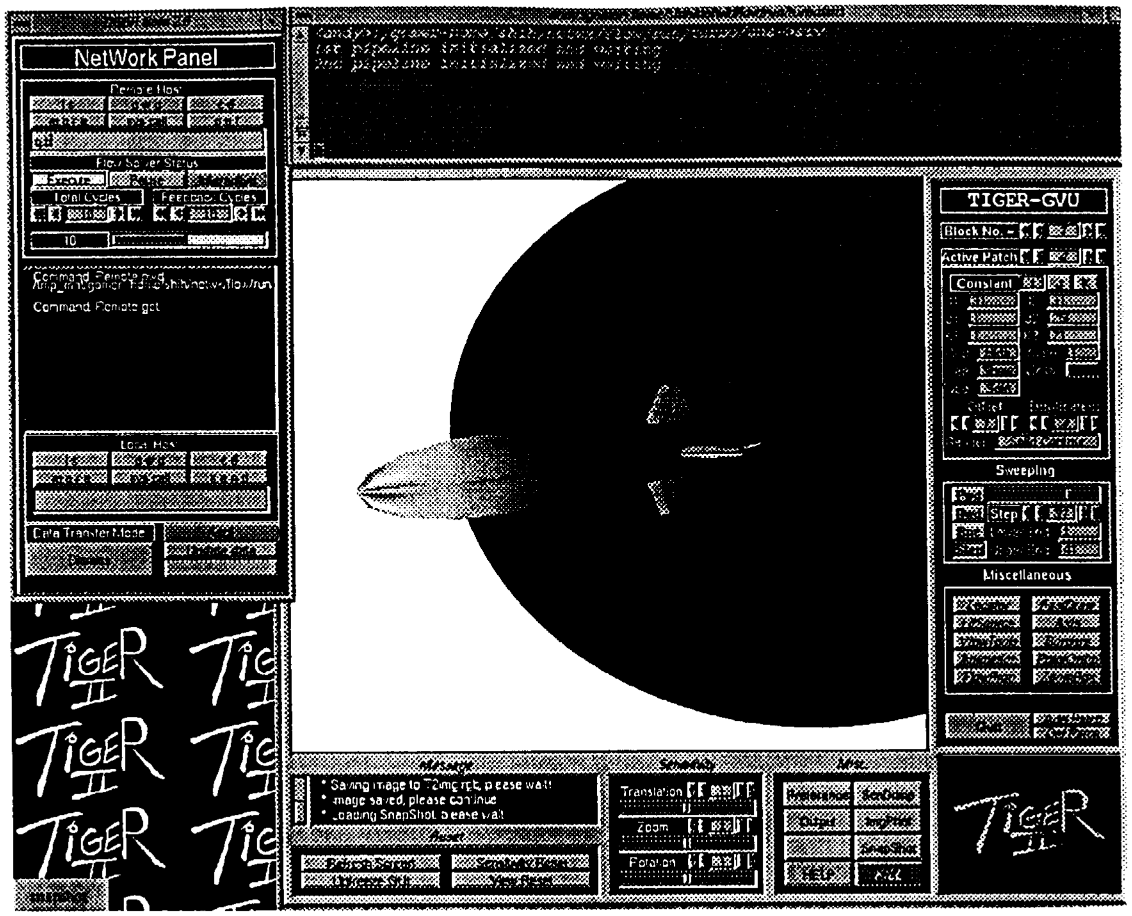
Hamilton–Standard Single Rotation Propfan SR–7

The other case for the simulation application is a single rotational propfan, SR–7[42]. This geometry is modeled with C–type domain, as illustrated in Figure 6.12(a), with a grid size of 31x16x9. This geometry has 8 blades and the blade stagger angle is set to be 54.97 degrees, with the advance ratio J=3.07 (rpm=1840) and a free–stream Mach number 0.78. Each iteration of flow calculation for Euler solution will take about 5 seconds on the same SGI Challenge machine. The density contour of the 1st, 20th, and 2000th iteration are shown in Figures 6.12(b) to 6.12(d), respectively. Similar procedure was taken to capture the snapshots of the flow solution evolution.



! Can not open T2gvu.lst, Switch to Defaults
* Saving image to T2img.rgb, please wait!

Figure 6.11 (a) Numerical Model of a Fin–Body Geometry

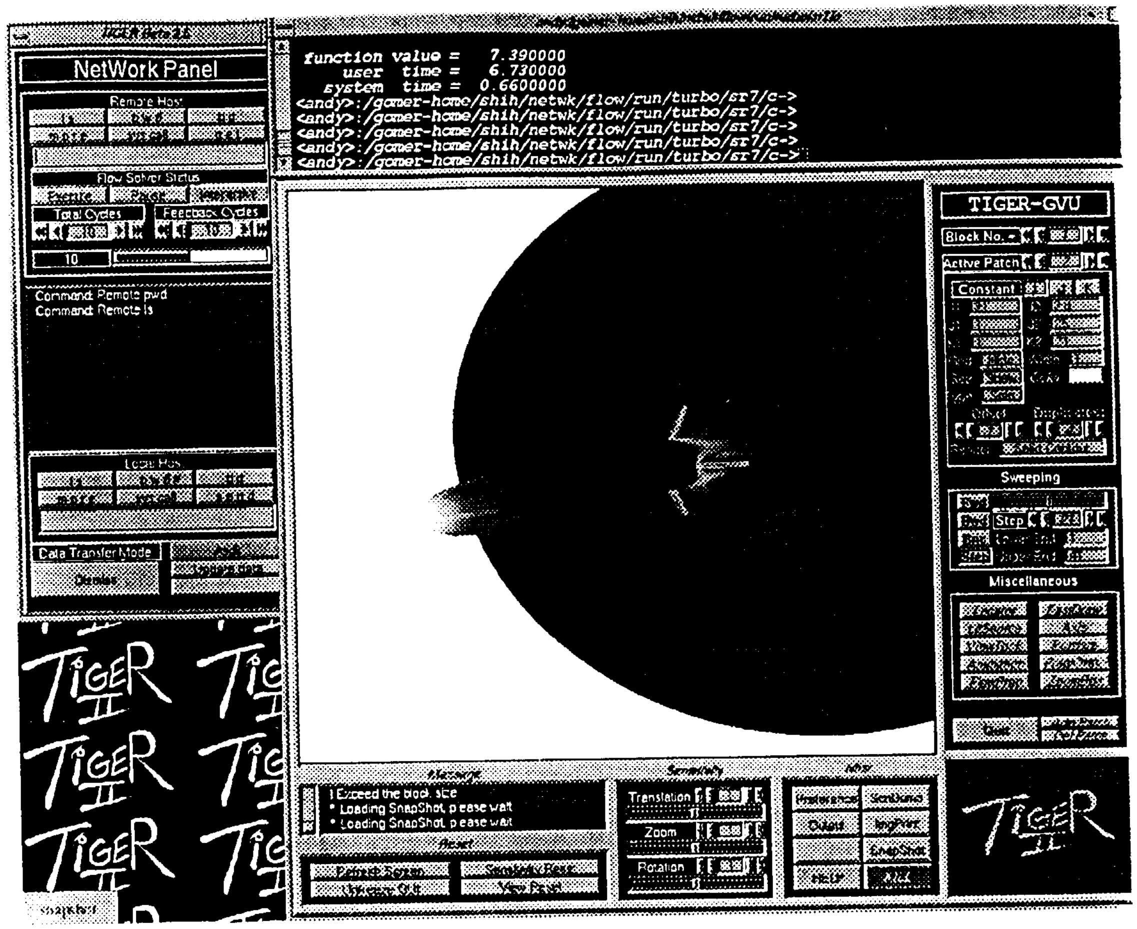


```
[illegible]
1st pipeline initialized and waiting
2nd pipeline initialized and waiting
```

```
Command Remote mvd
/tmp_mnt/gomer/home/shih/active/flow/run
Command Remote get
```

```
• Saving image to T2img.rgb, please wait!
• Image saved, please continue
• Loading SnapShot, please wait
```

Figure 6.11 (b) Density Contour of the 1st Iteration Simulation

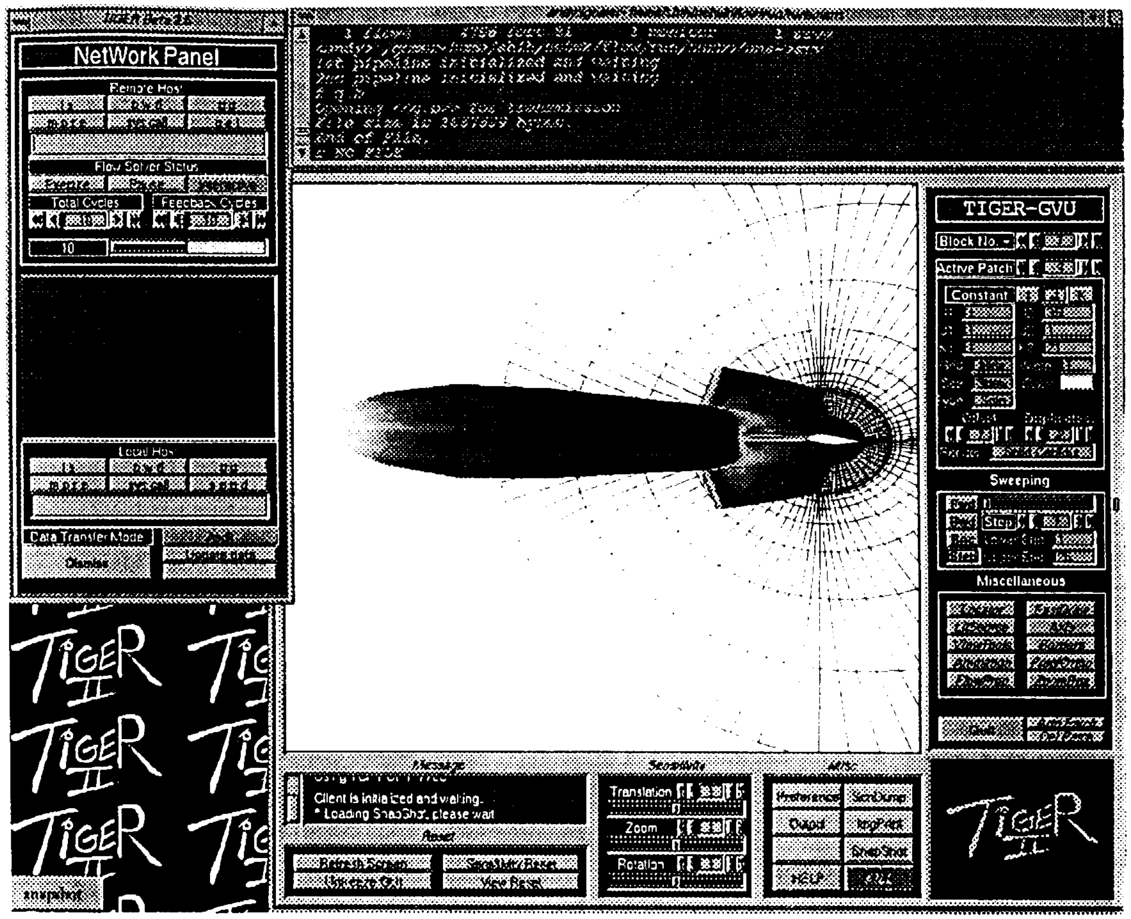


```
function value =    7.390000
    user  time =    6.730000
  system  time =   0.6600000
<andy>:/gomer-home/shih/netwk/flow/run/turbo/sr7/c->
<andy>:/gomer-home/shih/netwk/flow/run/turbo/sr7/c->
<andy>:/gomer-home/shih/netwk/flow/run/turbo/sr7/c->
<andy>:/gomer-home/shih/netwk/flow/run/turbo/sr7/c->
<andy>:/gomer-home/shih/netwk/flow/run/turbo/sr7/c->
```

```
Command: Remote pwd
Command: Remote ls
```

```
! Exceed the block size
* Loading SnapShot, please wait
* Loading SnapShot, please wait
```

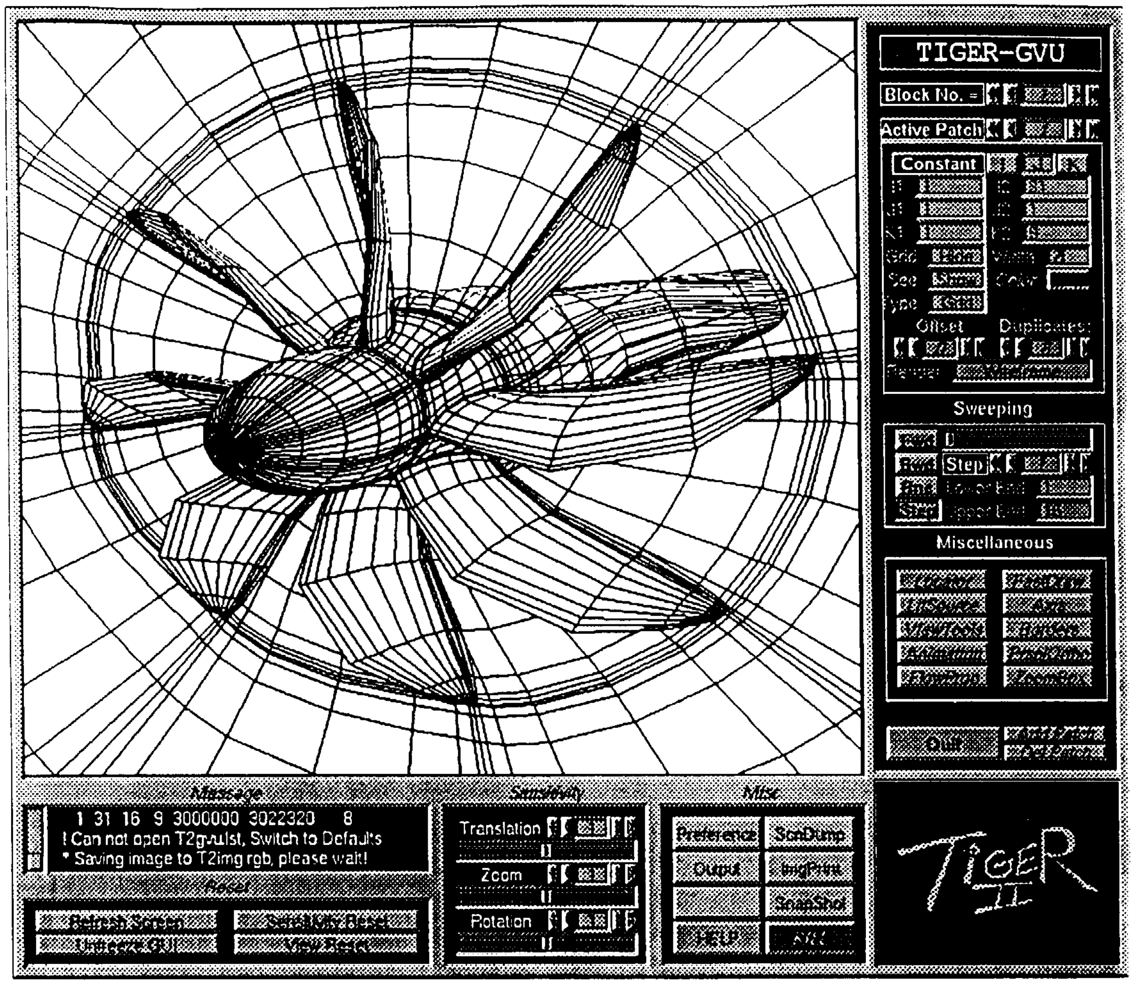
Figure 6.11 (c) Density Contour of the 20th Iteration Simulation



```
[illegible]
1st pipeline initialized and waiting
2nd pipeline initialized and waiting
[illegible]
Opening [illegible] for transmission
File size is 2007609 bytes.
End of file.
[illegible]
```

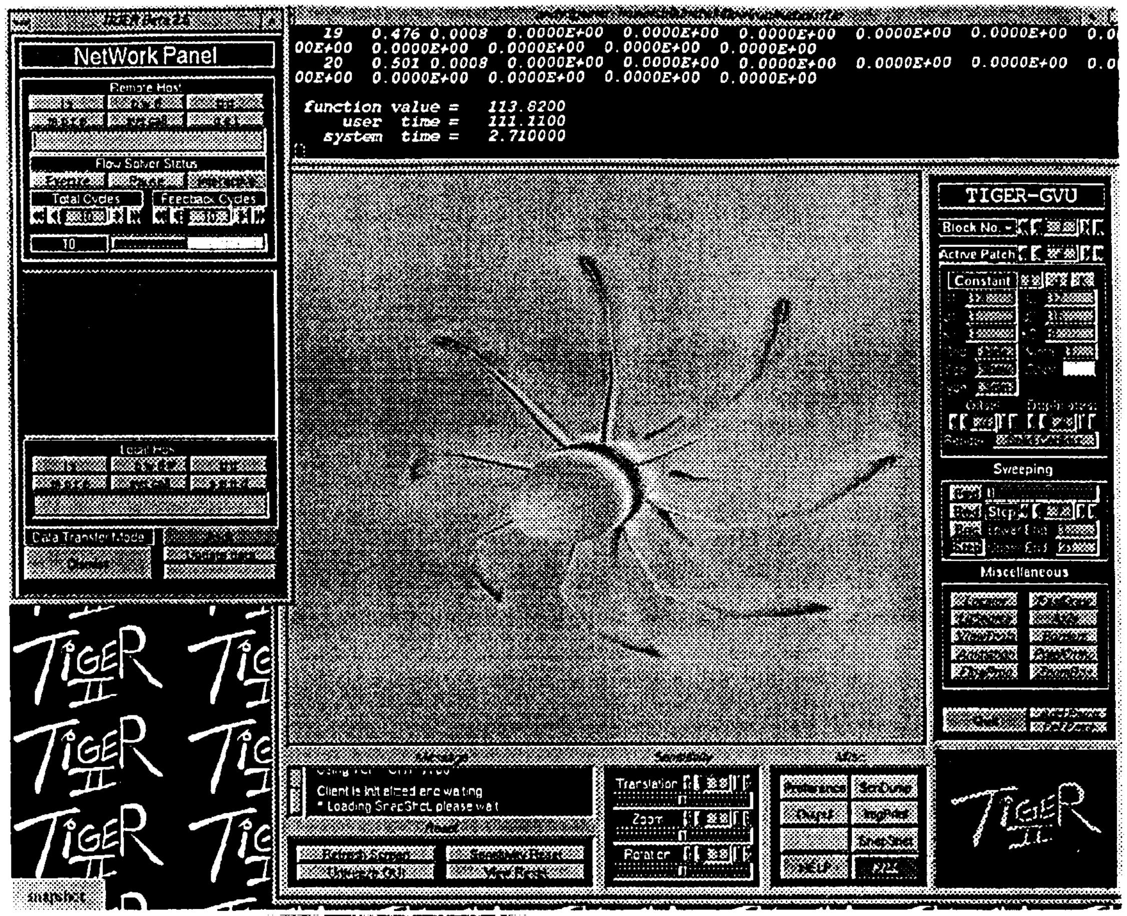
```
Client is initialized and waiting.
* Loading SnapShot, please wait
```

Figure 6.11 (d) Density Contour of the 2000th Iteration of Simulation



```
1 31 16 9 3000000 3022320    8
! Can not open T2gvulst, Switch to Defaults
* Saving image to T2img rgb, please wait!
```

Figure 6.12 (a) Numerical Model of the Hamilton–Standard SR–7 Prop–Fan

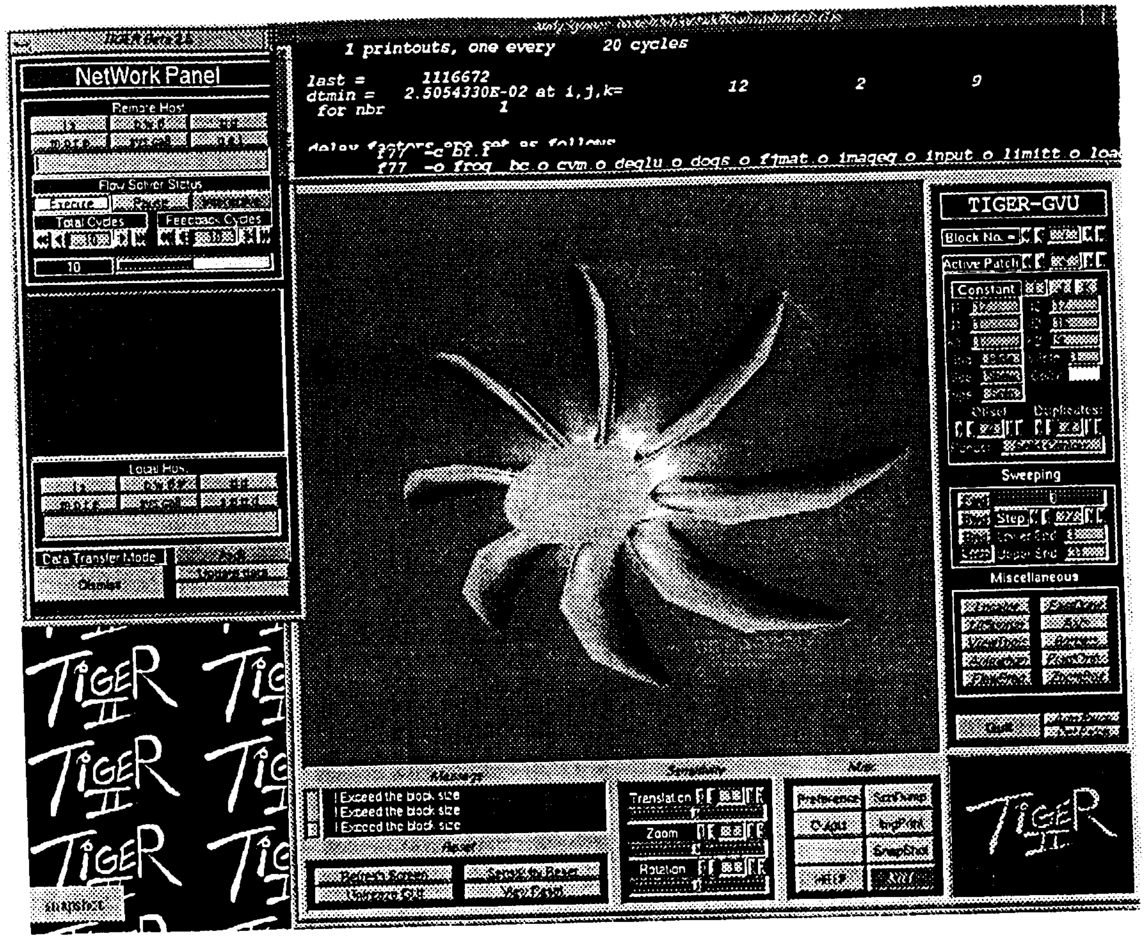


```
    19    0.476 0.0008  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0
00E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
    20    0.501 0.0008  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0
00E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00

 function value =    113.8200
     user  time =    111.1100
   system  time =    2.710000
```

```
Client is initialized and waiting
* Loading SnapShot, please wait
```

Figure 6.12 (b) Denisty Contour of the 1st Iteration Simulation

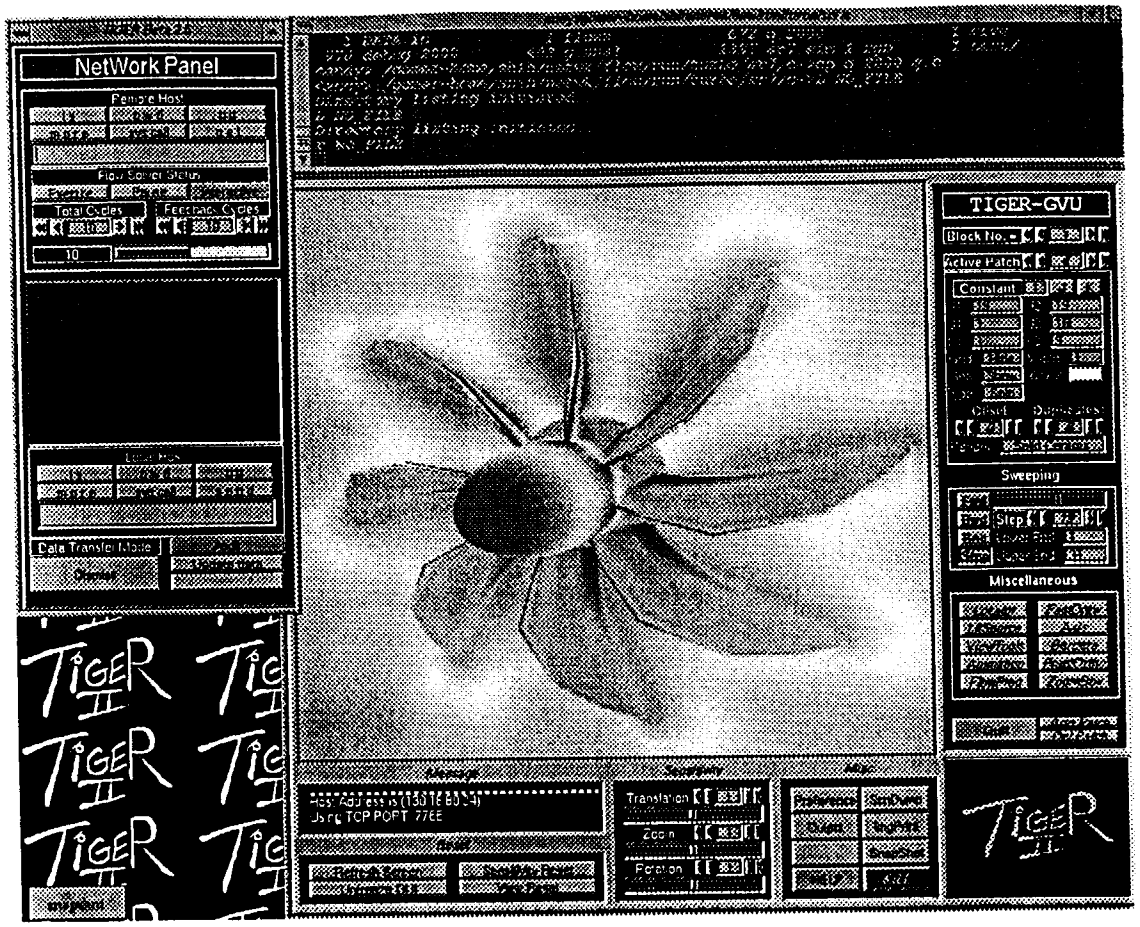


```
 1 printouts, one every     20 cycles

last =        1116672
dtmin =   2.5054330E-02 at i,j,k=           12          2          9
 for nbr            1

delay factors are set as follows
       f77  -c bf.f
       f77  -o frog  bc.o cvm.o deglu.o dogs.o fjmat.o imageq.o input.o limitt.o loa
```

Figure 6.12 (c) Denisty Contour of the 20th Iteration Simulation



RESTADD ESS IS (130.18.60.34)
Using TCP PORT 7766

Figure 6.12 (d) Density Contour of the 2000th Iteration Simulation

CHAPTER VII
CONCLUSION

A comprehensive CFD system for general turbomachinery configurations has been developed. This system consists of a customized grid generation module that dramatically reduces the time and labor required to construct a turbomachinery grid; a general–purpose visualization module for both grid and solution data; a simple yet versatile network module that bridges the gap between different computer systems across the network; a simulation module that allows near real–time flow simulation and visualization for turbomachinery configurations; a toolbox module that allows this system to accept various geometry formats and massage the blade definition if necessary; and finally, an open–ended flow module that can be easily replaced by any other state–of–the–art turbomachinery code.

System Summary

The grid generation module has been extensively tested with the real world applications as well as academic research geometries. Customized algorithms include: (*i*)automatic computational modeling, (*ii*)automatic blade surface construction, (*iii*)automatic sub–block decomposition, (*iv*)cascade surface transformation, and (*v*)combination of elliptic smoothing and NURBS spline–fit algorithm to achieve better grid quality. These customized algorithms allow the construction of a turbomachinery grid in a fraction of the time that is necessary when using a general–purpose code. Yet it is general enough to handle most turbomachinery configurations. This capability is demonstrated in the first portion of Chapter VII. The GVU visualization module, initially developed for turbomachinery grid visualization only, has evolved into a general–purpose CFD visualization software that provides both static and dynamic visualization capability. The network module is developed to allow a heterogeneous environment for this system. This module achieves two–way communication between a local graphics workstation and a remote mainframe. This system, therefore, allows TIGER to be able to access the enormous computing power of a remote supercomputer, yet retains the horsepower of local graphics engine. The simulation module of TIGER is the one that turns such advantages into real–world flow simulations for complicated turbomachinery configurations on the near real–time basis. The ToolBox module provides the user with the necessary utilities to manipulate/validate the raw data, while the open–ended flow module retains the flexibility of integrating with state–of–the–art turbomachinery flow solver softwares.

Future enhancement

Though TIGER is a comprehensive system that brings all the technologies together centering around the turbomachinery applications, there are several issues that needs to be addressed. Aspects for future enhancement in each module are suggested below:

Grid Generation Module

The grid generation module in TIGER is fairly robust for most of the geometries. However, its cascade frame manipulation capability still needs to be enhanced because getting a good surface at this stage will promote better grid quality for the overall grid. Moreover, the restriction of aligning the first K–surfaces along the axial blocks should be eliminated so that a user may have a choice either to align the first K–surfaces or to create a totally mis–matched grid. Blade tip surface using an extra block would allow more accurate modeling of this critical region. This should also be addressed in the future development. Finally, hybrid/mixed–topology grid algorithms, as demonstrated in Chapter VI, should be implemented and generalized as they will broaden the TIGER's applicability to the real–world applications.

Visualization Module

This module has been recently extended to allow flow solution visualization. Even though it has four most commonly used rendering methods, i.e. color–coded wireframe, solid contour, line contour, and vector field, it is desirable to have other more sophisticated methods like particle trace, stream lines, etc.. Furthermore, instead of storing the grid or flow solution data into an one–dimensional array, it should be modified in such a way that it will store them in disk (temporary file) so that there won't be any restriction on the memory, and hence can achieve broader range of time steps for unsteady flow visualization. Unstructured grid visualization capability should also be implemented.

Network Module

TIGER's network module allows handshaking between different machines. It brings back the entire solution file back from a remote host, which allows the user to freely examine every corner within the flow field. However, the main drawback of this scheme is that the data transferring speed is constrained by the hardware data throughput capacity, and therefore prolongs the "black–out" period. One way to get around this problem is to send the indices of user–specified surface patches, and request that the flow module dump out the solution for these surfaces. This scheme will make the data transferring a much lighter burden for the network module, and hence shorten the "black–out" time. Its drawback, however, would be the lag for the user to visualize different regions of the field, since the solution in this new region will not be updated until the next solution batch is sent back.

Simulation Module

The simulation module is currently serving as a dispatcher and a receiver. It does not have the capability to modify input files or change boundary conditions. Therefore, a boundary condition/ flow condition panel needs to be implemented in this module so that the user may specify the flow conditions interactively without opening another window. However, this approach may require the modification of the flow module in order to coordinate the effort. Qualitative performance representation is also needed to provide the user a more meaningful analysis during the simulation.

ToolBox Module

The main roadblock of TIGER in terms of geometry data acceptance is the absence of the capability to accept IGES definitions. Fortunately, there is an on–going research on developing an IGES converter, CAGI[76], at Mississippi State University sponsored by NASA Marshall Space Flight Center. It would enhance TIGER's applicability if this product can be installed into TIGER's ToolBox.

REFERENCES

[1] Warsi, Z.U.A., "Fluid Dynamics: Theoretical and Computational Approaches," CRC, Ann Arbor, 1993.

[2] White, F.M., "Viscous Fluid Flow," McGraw–Hill, New York, 1974.

[3] Pao, R.H.F., "Fluid Mechanics," John Wiley & Sons, New York, 1961.

[4] Anderson, D.A., Tannehill,, J.C., Pletcher, R.H., "Computational Fluid Machanics and Heat Transfer," McGraw Hill, New York, 1994.

[5] Sod, G.A., "Numerical Methods in Fluid Dynamics, Initial and Initial Boundary–Value Problems," Cambridge, New York, 1986.

[6] Whitfield, D.L., Janus, J.M., and Simpson, L.B., "Implicit Finite Volume High Resolution Wave–Split Scheme for Solving the Unsteady Three–Dimensional Euler and Navier–Stokes Equations on the Stationary or Dynamic Grids," Engineering & Industrial Research Station Report, MSSU–EIRS–ASE–88–2, Mississippi State University, Feb. 1988.

[7] Thompson, J.F. and Weatherill, N.P., "Structured and Unstructured Grid Generation," NSF Engineering Research Center for Computational Field Simulation, March 1993.

[8] Soni, B.K., "GENIE: Generation of Computational Geometry–Grids for Interal/External Flow Configurations," p.915, Numerical Grid Generation in Computational Fluid Dynamics, Sengupta, S., Hauser, J., Eiseman, P.R., Thompson, J.F. (Eds.), Proceedings of the second International Conference, Pineridge Press, 1988.

[9] Soni, B.K., "Two– and Three–Dimensional Grid Generation for Internal Flow Applications of Computational Fluid Dynamics," AIAA–85–1526, AIAA 17th Fluid Dynamics, Plasma Dynamics, and Laser Conference, Cincinnati, 1985.

[10] Soni, B.K., "Grid Generation for interanal Flow Configurations," Journal of Computers Math Apllications, Vol. 24, No. 5/6, pp. 191–201, 1992.

[11] Soni, B.K., Thompson , J. F., Stokes, M.L., and Shih, M. H., "GENIE^{++}, EAGLEVIWE and TIGER: General Purpose and Special Purpose Graphically Interactive Grid Systems," AIAA–92–0071, AIAA 30th Aerospace Sciences Meeting, Reno, NV, January 1992.

[12] Thompson, J.F.,"Program EAGLE Numerical Grid Generation System User's Manual: Surface Generation System," AFTAL–TR–87–15, Vol. II, March 1987.

[13] Thompson, J.F.,"Program EAGLE Numerical Grid Generation System User's Manual: Grid Generation System," AFTAL–TR–87–15, Vol. III, March 1987.

[14] Thompson, J.F., "The National Grid Project," Computer Systems in Engineering, Vol. 3, Nos. 1–4, pp. 393–399, 1992.

[15] Akdag, V. and Wulf, A., "Integrated Geometry and Grid Generation System for Complex Configurations," Proceedings of the NASA Workshop on Software Systems for Surface Modeling and Grid Generation, Hampton, Virginia, April 1992.

[16] Sorenson, R.L. and McCann, K., "GRAPEVINE: Grids about Anything by Poisson's Equations in a Visually Interactive Netwroking Environment," Proceedings of the NASA Workshop on Software Systems for Surface Modeling and Grid Generation, Hampton, Virginia, April 1992.

[17] Sorenson, R.L., "The 3DGRAPE Book: Theory, Users' Manual, Examples," NASA–TM–10224, 1989.

[18] Sorenson, R.L., "Three–Dimensional Zonal Grids About Arbitrary Shapes by Poisson Equation," NASA–TN–101018, 1988.

[19] Sorenson, R.L., "Three–Dimensional Elliptic Grid Generation About FighterAircraft for Zonal Finite–Difference Computations," AIAA–86–0429, AIAA 24th Aerospace Sciences Meeting, Reno, NV, January 1986.

[20] Steinbrenner, J.P. and Chawner, J.R., "Recent Enhancements to the GRIDGEN Structured Grid Generation System," Proceedings of the NASA Workshop on Software Systems for Surface Modeling and Grid Generation, Hampton, Virginia, April 1992.

[21] Steinbrenner, J.P., "Enhancements to the GRIDGEN System for Increased User Efficiency and Grid Quality," AIAA–92–0662, AIAA 30th Aerospace Sciences Meeting, Reno, NV, January 1992.

[22] Beach, T.A., "An Interactive Grid Generation Procedure for Axial and Radial Flow Turbomachinery," AIAA–90–0344, AIAA 28th Aerospace Sciences Meeting, Reno, NV, January 1990.

[23] Beach, T.A. and Hoffman, G., "IGB Grid: User's Manual (A Turbomachinery Grid Generation Code)," NASA Contractor Report 189104, January, 1992.

[24] Crook, A.J. and Delaney, R.A., "Investigation of Advanced Counterrotation Blade Configurations Concepts for High Speed Turboprop Systems. Task III – Advanced Fan Section Grid Generator Final Report and Computer Program User's Manual," NASA Contractor Report CR–187129, September, 1991.

[25] Shih, M.H., and Soni, B.K., "Geometry Modeling and Multi–Block Grid Generation For Turbomachinery Configurations," Proceedings of the NASA Workshop on Software Systems for Surface Modeling and Grid Generation, Hampton, Virginia, April 1992.

[26] Shih, M. H., "TIGER User's Manual," NSF/MSU ERCCFS, Mississippi State University, September 1991. (unpublished)

[27] Soni, B.K. and Shih, M.H., "TIGER: Turbomachinery Interactive Grid GenERation," Proceedings of the Third International Conference of Numerical Grid Generation in CFD, Barcelona, Spain, June 1991.

[28] Soni, B.K. and Shih, M.H., "TURBOGRID: Turbomachinery Applications of Grid Generation," AIAA–90–2242, 26th AIAA/SAE/ASME Joint Propulsion Conference, Orlando, Florida, July 1990.

[29] Shih, M.H., "TIGER: Turbomachinery Interactive Grid genERation," Master's Thesis, Mississippi State University, December 1989.

[30] Janus, J.M., "Advanced 3–D CFD Algorithm for Turbomachinery," Ph.D. Dissertation, Mississippi State University, Mississippi, May 1989.

[31] Whitfield, D.L., Swafford, T.W., Janus, J.M., Mulac, R.A. and Belk, D.M., "Three–Dimensional Unsteady Euler Solutions for Propfans and Counter–Rotating Propfans in Transonic Flow," AIAA–87–1197, June, 1987.

[32] Stokes, M.L., Huddleston, D.H., and Remotique, M.G., "A Proactical Model for Multidisciplinary Analysis Data and Algorithm," 6th SIAM Conference on Parallel Processing for Scientific Computing, Norfolk, VA, March 1993.

[33] Cooper, G.K. and Sirbough, J.R., "PARC Code: Theory and Usage," AEDC–TR–89–15, Arnold Engineering Development Center, Arnold AFB, 1989.

[34] Arabshahi, A. and Whitfield, D.L, "A Multiblock Approach to Solving the Three–Dimensional Unsteady Euler Equations about a Wing–Pylon–Store Configuration," AIAA–89–3401, AIAA Atmospheric Flight Mechanics Conference, Boston, Mass., August 1989.

[35] Laurien, E., Holthoff H., and Wiesbaum, J., "Interactive Numerical Simulation And Grid Adaption For Hypersonic Flow," Computer Systems in Engineering, Vol. 3, Nos. 1–4, 1992, pp. 401–411.

[36] Hawthorne, W.R., "Aerodynamics of Turbines and Compressors," Princeton, , New York, 1988.

[37] Hesse, W.J. and Mumford, N.V.S. Jr., "Jet Propulsion for Aerospace Applications," Pittman, New York, 1964.

[38] Kuiper, G., "Cavitation Inception on Ship Propeller Models," Publication No. 655, Netherlands Ship Model Basin, Wageningen, The Netherland, 1981.

[39] Kent, R., (Ed.) "The Initial Graphics Exchange Specifications (IGES) Version 5.1," National Computer Graphics Association, September 1991.

[40] Farin, G. "Curves and Surfaces for Computer Aided Geometric Design: A Pratical Guide," Third Edition, Academic Press, 1990.

[41] Yu, T.Z., "IGES Transformer and NURBS in Grid Generation," Master's Thesis, Mississippi State University, Aug. 1992.

[42] Cambell, W.A., Arseneaux, P.J., and Wainauski, H.S., "NASA Large Scale Advanced Prop–Fan (LAP) High Speed Wind Tunnel Test Report," Hamilton Standard Division, United Technologies Co., HSER–11894, 1987.

[43] Yu, T.Z. and Soni, B.K., "Geometry Transformer and NURBS in Grid Generation," to be presented, 4th International Conference in Numerical Grid Generation in Computational Fluid Dynamics and Related Fields.

[44] DeBoor, C., "A Practical Guide to Splines," Springer–Verlag, New York, 1978.

[45] Blake, M.W., Chou, J.J., Kerr, P.A., and Thorp, S.A., "The NASA–IGES Geometry Data Exchange Standard," Proceedings of the NASA Workshop on Software Systems for Surface Modeling and Grid Generation, Hampton, Virginia, April 1992.

[46] Jones, G.A., "Surface Grid Generation for Composite Block Grids," Ph.D. Dissertation, Mississippi State University, Mississippi, May 1986.

[47] Thomas, P.D., "Construction of Composite Three Dimensional Grids From Subregion Grids Generated by Elliptic Systems," AIAA–81–0996.

[48] Chen, J.P. and Whitfield, D.L., "Navier–Stokes Calculations for The Unsteady Flowfiled of Turbomachinery," AIAA–93–0676, AIAA 31st Aerospace Sciences Meeting, Reno, NV, January 1993.

[49] Chen, J.P., "Unsteady Three–Dimensional Thin–Layer Navier–Stokes Solutions for Turbomachinery in Transonic Flow," Ph.D. Dissertation, Mississippi State University, Mississippi, December 1991.

[50] Abbott, I.H. and Von Doenhoff, A.E., "Theory of Wing Sections," Dover, New York, 1959.

[51] Schumann, L.F., "Fortran Program for Calculating Leading– and Trailing–Edge Geoemtry of Turbomachine Blades," NASA TM X–73679, Cleveland, Ohio, June 1977.

[52] Chima, R.V., "TCGRID," November 1990.

[53] Shih' M.H., Yu, T–Y, and Soni, B.K., "Interactive Grid Generation and NURBS Applications," Journal of Applied Mathematics and Computations.

[54] Shih, M.H. and Soni, B.K., "TIGER: A User–Friendly Interactive Grid Generation System For Complicated Turbomachinery And Axis–Symmetric Configurations," Proceedings of the NASA MSFC 11th Workshop for Computational Fluid Dynamic Applications in Rocket Propulsion, Huntsville, AL, April 1993.

[55] Shih, M.H., and Soni, B.K., "TIGER: A Graphically Interactive Grid System for Turbomachinery Applications," Proceedings of the First European Computational Fluid Dynamics Conference, Brussels, Belgium, September 1992.

[56] Shih, M.H., and Soni, B.K., "Grid Generation for 3D turbomachinery Configurations," AIAA–92–3671, 28th AIAA/SAE/ASME Joint Propulsion Conference, Nashville, Tennessee, July 1992.

[57] Thompson, J.F., Mastin, C.W., "Order of Difference Expressions in Curvilinear Coordinate Systems," Journal of Fluids Engineering, Vol. 107, p.241, 1985.

[58] Thompson, J.F., "A Composite Grid Generation Code For General 3–D Regions," AIAA–87–0275, AIAA 25th Aerospace Sciences Meeting, Reno, NV, January 1987.

[59] Thompson, J.F., Warsi, Z.U.A. and Mastin, C.W., "Numerical Grid Generation: Foundations and Applications," North Holland, 1985.

[60] Thompson, J.F., Warsi, Z.U.A. and Mastin, C.W., "Boundary–Fitted Coordinate Systems for Numerical Solution of Partial Differential Equations – A Review," Journal of Computational Physics, Vol. 47, P. 1, 1982.

[61] Soni, B.K., "Elliptic Grid Generation System: Control Functions Revisited I," To Appear, Journal of Applied Mathematics and Computers.

[62] Middlecoff, J.F. and Thomas, P.D., "Direct Control of the Grid Point Distribution in Meshes Generated by Elliptic Equations," AIAA–79–1462, AIAA 4th Computational Dynamics Conference, Williamsburg, VA, 1979.

[63] Davis, D.E. and Salmond, D.J., "Calculation of the Volume of a General Hexahedron for Flow Predictions," AIAA Journal, Vol. 23, No. 6, June 1985.

[64] Overmars, M.H., "Forms Library: A Graphical User Interface Toolkit for Silicon Graphics Workstations," Version 2.1, Utrecht, Netherland, November 1992.

[65] Maitz, K, Oleson, L., and Walsh, K., "Graphics Library Reference Manual, C Edition," Version 3.0, Silicon Graphics Inc., Mountain View, CA, 1989.

[66] Taylor, L.K., "Unsteady Three–Dimensional Incompressible Algorithm Based on Artificial Compressibility," Ph.D. Dissertation, Mississippi State University, MS, May 1991.

[67] Kesner, G. and Smith, A.B.W., "TCP/IP User's Guide," Version 2.0, Silicon Graphics Inc., Mountain View, CA, 1988.

[68] Hathaway, M.D., Wood, J.R., and Wasserbauer, C.A., "NASA Low–Speed Centrifugal Compressor for 3–D Viscous Code Assessment and Fundamental Flow Physics Research," ASME 91–GT–140, International Gas Turbine and Aeroengine Congress and Exposition, Orlando, FL, June 1991.

[69] Hathaway, M.D., Chriss, R.M., Wood, J.R., and Strazisar, A.J., "Experimental and Computational Investigation of the NASA Low–Speed Centrifugal Compressor Flow Field," ASME 92–GT–213, International Gas Turbine and Aeroengine Congress and Exposition, Cologne, Germany, June 1992.

[70] Zierke, W.C., Straka, W.A., Taylor, T.D., "The High Reynolds Number Flow Through an Axial–Flow Pump," Pennsylvania State University Applied Research Laboratory, Technical Report TR 93–12, Nov. 1993.

[71] Osborn, W.M., and Steinke, R.J., "Performance of a 1.15 Pressure Ratio Axial–Flow Fan Stage with a Blade Tip Solidity of 0.5," NASA TM X–3052, 1974.

[72] Steffen, F.W., "Cruise Performance of an Isolated 1.15 Pressure Ratio Turbofan Propulsion Simulator at Mach Numbers from 0.6 to 0.85," NASA TM X–3064, 1974.

[73] Wesoky, H.L., Abbott, J.M., Albers, J.A., and Dietrich, D.A., "Low–Speed Wind Tunnel Tests of a 50.8 Centimeter (20–in.) 1.15 Pressure Ratio Fan Engine Model," NASA TM X–3062, 1974.

[74] Wesoky, H.L., and Steffen, F.W., "Wind Tunnel Tests of a 20 in. Diameter 1.15 Pressure Tatio Fan Engine Model," NASA TM X–71445, 1973.

[75] Hall, E.J. and Delaney, R.A., "Investigation of Advanced Counterrotation Blade Configurations Concepts for High Speed Turboprop Systems. Task 5 – Unsteady Counterrotation Ducted Propfan Analysis Computer Program User's Manual," NASA Contractor Report CR–187125, January, 1993.

[76] Soni, B., Yu, T.Y., and Vaughn, D., "CAGI:Computer Aided Grid Interface — A Work in Progress," Proceedings of 10th Workship for Computational Fluid Dynamics Applications in Rocket Propulsion, pp. 577–614, NASA MSFC, Hunstsville, AL, April 1992.

APPENDIX A

GEOMETRY FILE

A. TIGER Geometry Format

```
############################################################################
################    TIGER Beta 2.6 Geometry Format    ######################
############################################################################
# Desc: Geometry file sample of TIGER format
# Misc: Pound Sign = Comment
#
# number of curves
#
          1
#
# number of points on curve# 1
#
         34
#
#         Z            R
 -0.3268520     0.0000000
 -0.3240740     0.0101850
  .........     .........
  2.6612200     0.1546500
  3.0000000     0.1546500
############################################################################
#
#   1st  Blade Information: PCA=Pitch Change Axis
#
############################################################################
#   NI(Axial)  NJ(Radial)  Blade_Diam   Beta34       Z-PCA        T-PCA
############################################################################
    25          12          1.0000000    58.9071541    0.0078526    0.0000000
############################################################################
#           Pressure Side                          Suction Side
#
#   Z(Axial)     Radius       Theta         Z(Axial)     Radius       Theta
############################################################################
# Radial Cross-Section #1
   -0.020169     0.133628     0.922292     -0.020169     0.133628     0.922292
   -0.016996     0.133093     0.927614     -0.022391     0.132032     0.938412
    ........     ........     ........      ........     ........      .......
    0.036589     0.140201     2.278855      0.036589     0.140201     2.278855
# Radial Cross-Section #2
    ........     ........     ........      ........     ........      .......
# Radial Cross-Section #3
    ........     ........     ........      ........     ........      .......
    ........     ........     ........      ........     ........      .......
    ........     ........     ........      ........     ........      .......
# Radial Cross-Section #12
############################################################################
#
#   2nd  Blade Information: PCA=Pitch Change Axis
#
############################################################################
#   NI(Axial)  NJ(Radial)  Blade_Diam   Beta34       Z-PCA        T-PCA
############################################################################
    25          12          1.0000000    58.9071541    0.0078526    0.0000000
############################################################################
#           Pressure Side                          Suction Side
#
```

```
#   Z(Axial)     Radius       Theta          Z(Axial)     Radius       Theta
##########################################################################################
# Radial Cross-Section #1
   -0.020169     0.133628     0.922292      -0.020169     0.133628     0.922292
   -0.016996     0.133093     0.927614      -0.022391     0.132032     0.938412
    ........     ........     ........       ........     ........      .......
    0.036589     0.140201     2.278855       0.036589     0.140201     2.278855
# Radial Cross-Section #2
    ........     ........     ........       ........     ........      .......
# Radial Cross-Section #3
    ........     ........     ........       ........     ........      .......
    ........     ........     ........       ........     ........      .......
    ........     ........     ........       ........     ........      .......
# Radial Cross-Section #12
```

APPENDIX B

HISTORY FILES

A. Parameter History File

```
#==============================================================
#==> TOTAL AXIAL  POINTS FOR THE FULL GEOMETRY
#==============================================================
     IMAX =    111
#==============================================================
#==> TOTAL RADIAL POINTS FOR THE FULL GEOMETRY
#==============================================================
     JMAX =     21
#==============================================================
#==> NUMBER OF TOTAL GRID BLOCKS
#==============================================================
     NUMBER OF BLOCKS  =      1
#==============================================================
#==> GLOBAL INDICES: I1, I2, J1, J2, K1, K2
#==============================================================
     INDICES =      1  111     1    21     1    21     FOR BLOCK  1
#==============================================================
#==> CONVENTION OPTIONS=> 1:RIGHT-HANDED 2:LEFT-HANDED
#==============================================================
     CONVENTION DIRECTION =      1
#==============================================================
#==> COORDINATES OPTIONS=> 1:XYZ          2:ZRT
#==> OUTPUT FORM OPTIONS=> 1:UNFORMATTED  2:FORMATTED
#==> FORMAT      OPTIONS=> 1:PLOT3D
#==============================================================
     GRID COORDINATE     =     1 FOR BLK #  1
     GRID OUTPUT FORM    =     1 FOR BLK #  1
     GRID OUTPUT FORMAT =      1 FOR BLK #  1
     grd.file
#==============================================================
#==> THE FOLLOWING 3 PARAMETERS DEFINES THE OUTER DOMAIN
#==============================================================
     PARAMETER FOR     UPSTREAM BNDRY    = 46.0862694
     PARAMETER FOR   DOWNSTREAM BNDRY    = 1.0000000
     PARAMETER FOR RADIAL OUTER BNDRY    = 1.0000000
#==============================================================
#==> NORMALIZATION LENGTH, USUALLY THE BLADE DIAMETER
#==============================================================
     NORMALIZATION CHARACTERISTIC LENGTH= 1.0000000
#==============================================================
#==> OVERALL FLOW FIELD:1=External,2=Internal,3=External-Internal
#==============================================================
     FLOW FIELD =      2
#==============================================================
#==> OVERALL GRID TYPE:  1=H-type, 2=C-type
#==============================================================
     GRID TYPE  =      1
#==============================================================
#==> SAME BLADE ROW EXISTS PERIODICITY
#==============================================================
     NUMBER OF BLADE ROWS =      1
#==============================================================
#==> 1=PROPFAN(1ST), 2=PROPFAN(2ND), 3=ROTOR, 4=STATOR
#==============================================================
     BLADE TYPE =      3 FOR ROW #  1
#==============================================================
     I INDEX ON BLADE LEADING  EDGE        =    41 FOR ROW #  1
     I INDEX ON BLADE TRAILING EDGE        =    91 FOR ROW #  1
     J INDEX ON BLADE ROOT (LOWER  END)    =     1 FOR ROW #  1
     J INDEX ON BLADE TIP  (HIGHER END)    =    17 FOR ROW #  1
     J INDEX ON BLADE SEALING TOP          =    18 FOR ROW #  1
     NUMBER OF BLADES FOR THIS ROW         =    20 FOR ROW #  1
     BLADE INTERSECTED?     (1:NO, 2:YES)  =     1 FOR ROW #  1
     BETA34 USER-DEFINED?   (1:NO, 2:YES)  =     1 FOR ROW #  1
     USER-SPECIFIED BETA34 (IGNOR IF AS IS)=    0.0000000
#=================================================================
#==> INFORMATION FOR MERIDIONAL CURVES
#=================================================================
     NUMBER OF MERIDIONAL CURVES    =     2
#=================================================================
#==> INDICES FOR MERIDIONAL CURVES:   I-Lip     I-Tail     J-Level
#=================================================================
     INDICES FOR CURVE LEVEL   1  =      15         111           1
     INDICES FOR CURVE LEVEL   2  =       1         111          21
EOF
```

B. Generatrix Setup History File

```
#ID          ACTIVITY            MVPIX         MVPIY
#==================================================
  0 START-THE-ACTION                 0             0
108         LeftArrow                0             0
106           MoveInR                0             0
199         LeftMouse            -5478        -75054
199         LeftMouse             -301         -1470
108         LeftArrow                0             0
106           MoveInR                0             0
199         LeftMouse            -2371        -76904
199         LeftMouse              -53          -320
199         LeftMouse              160           865
108         LeftArrow                0             0
108         LeftArrow                0             0
199         LeftMouse             -116          -553
104 SwapControlPoint                 0             0
199         LeftMouse             -138           378
108         LeftArrow                0             0
199         LeftMouse               31          1396
199         LeftMouse            -3527          -558
199         LeftMouse                0             0
103 SwapControlPoint                 0             0
199         LeftMouse              414         -1292
108         LeftArrow                0             0
108         LeftArrow                0             0
106           MoveInR                0             0
199         LeftMouse              554        -14001
199         LeftMouse              -35         -1056
199         LeftMouse              -37            82
199         LeftMouse               11            87
199         LeftMouse                0         -1209
108         LeftArrow                0             0
106           MoveInR                0             0
199         LeftMouse             1069        -15452
108         LeftArrow                0             0
108         LeftArrow                0             0
108         LeftArrow                0             0
199         LeftMouse                0             0
111              Quit                0             0
```

C. Frame Setup History File

```
#============================================================
#============================================================
#============= FRAME MANIPULATION SETUP DATA ===============
#=============                               ===============
#============= Tiger Beta V2.6  Feb. 14,1993 ===============
#============================================================
#============================================================
#
       1 = Number of Frames
#
############################################################
############## AXIAL-SEGMENT DISTRIBUTION #################
############################################################
       3 = Number of HLINKs
#
#=====
       1    15.2443161  = J-INDEX & Rbar
#=====
       4 = Number of Segments
#====================================
       1      15 = I-INDEX RANGE
#====================================
      0.0000000    9.6257610    19.2515221    32.0858688 = BEZIER Z
      11.0716782    11.0716782    11.0716782    11.0716782 = BEZIER T
       4           1.0000000    0.5000000     = (OPTION,DS1,DS2)
       0           1.0000000    100.0000000    = (GEOMP ,GF ,GP)
#====================================
      15      41 = I-INDEX RANGE
#====================================
      32.0858688    37.3536911    44.4878426    49.6442833 = BEZIER Z
      11.0716782    11.0716782    17.9912701    24.0107365 = BEZIER T
       4           1.0000000    0.0500000     = (OPTION,DS1,DS2)
       0           1.0000000    100.0000000    = (GEOMP ,GF ,GP)
#====================================
      41      91 = I-INDEX RANGE
#====================================
      49.6452789    58.8389587    68.0326385    80.2908783 = BEZIER Z
      23.9995766    23.9995766    40.1137772    40.1137772 = BEZIER T
       0           0.0000000    0.0000000     = (OPTION,DS1,DS2)
       0           0.0000000    0.0000000     = (GEOMP ,GF ,GP)
#====================================
      91     111 = I-INDEX RANGE
#====================================
      80.3315201    84.1765137    92.7248840    95.8333969 = BEZIER Z
      40.1361465    43.3459892    50.4568596    53.0426292 = BEZIER T
       4           0.3000000    1.0000000     = (OPTION,DS1,DS2)
       0           1.0000000    100.0000000    = (GEOMP ,GF ,GP)
..................................................................
..................................................................
..................................................................
#
############################################################
############# RADIAL-SEGMENT DISTRIBUTION #################
############################################################
      10 = Number of VLINKs
#
```

```
#==================================================
      1      1     17 = (I-INDEX,J-INDEX RANGE)
#==================================================
   -46.0862694   -46.0862694   -46.0862732   -46.0862732 = BEZIER Z
     0.0000000     4.2480931    13.8063030    16.9923725 = BEZIER R
     0.7262824     0.7262824     0.7295887     0.7295887 = BEZIER T
      6            0.0500000     0.0300000     = (OPTION,DS1,DS2)
      0            1.0000000   100.0000000     = (GEOMP ,GF ,GP)
#
#==================================================
     15      1     17 = (I-INDEX,J-INDEX RANGE)
#==================================================
   -14.0003996   -16.6953945   -20.3267670   -20.4457550 = BEZIER Z
     0.0000000     2.8795271    11.7781773    17.0137043 = BEZIER R
     0.7262824     0.7262824     0.7262823     0.7262823 = BEZIER T
      4            0.0500000     0.0200000     = (OPTION,DS1,DS2)
      0            1.0000000   100.0000000     = (GEOMP ,GF ,GP)
#
#==================================================
     41      1     17 = (I-INDEX,J-INDEX RANGE)
#==================================================
     0.0070102     0.0072592     0.0051870     0.0051870 = BEZIER Z
     8.4552860    10.5991631    17.0307312    17.0307217 = BEZIER R
     1.5750616     1.5735682     1.5712843     1.5712843 = BEZIER T
      0            1.0000000     0.0000000     = (OPTION,DS1,DS2)
      0            1.0000000   100.0000000     = (GEOMP ,GF ,GP)
#
#==================================================
     91      1     17 = (I-INDEX,J-INDEX RANGE)
#==================================================
    18.3131180    16.9303627    12.7790813    12.7785130 = BEZIER Z
    29.9996643    29.9625435    29.9999619    29.9731083 = BEZIER R
     2.6328597     2.6286278     2.6215239     2.6203341 = BEZIER T
      0            1.0000000     0.0000000     = (OPTION,DS1,DS2)
      0            1.0000000   100.0000000     = (GEOMP ,GF ,GP)
#
#==================================================
    111      1     17 = (I-INDEX,J-INDEX RANGE)
#==================================================
    15.5525990    14.8938560    12.9176273    12.9176273 = BEZIER Z
    45.2400055    45.2400055    45.2400017    45.2400017 = BEZIER R
     3.4795012     3.4795012     3.4044256     3.4044256 = BEZIER T
      4            0.2000000     0.1000000     = (OPTION,DS1,DS2)
      0            1.0000000   100.0000000     = (GEOMP ,GF ,GP)
..........................................................
..........................................................
#
##########################################################
####### CIRCUMFERENTIAL-SEGMENT DISTRIBUTION ##############
##########################################################
#
#=====
#===== J-INDEX =    1
#=====
      5 = Number of Nodes
#=================================
      1      1     21 = (I-INDEX,K-INDEX RANGE)
#=================================
     0.0000000     0.0000000     0.0000000     0.0000000 = BEZIER Z
    11.0716782     9.6349354     8.1981907     6.2825336 = BEZIER T
    -46.0862694    0.0000000    0.7262824    0.0000000 = (Z,R,T,OFFSET)
     0            0.0000000    0.0000000    = (OPTION,DS1,DS2)
     0            0.0000000    0.0000000    = (GEOMP ,GF ,GP)
#===================================
      15      1     21 = (I-INDEX,K-INDEX RANGE)
#===================================
      32.0858688    32.0858688    32.0858688    32.0858688 = BEZIER Z
      11.0716782    9.6349354     8.1981907     6.2825336 = BEZIER T
     -14.0003996    0.0000000     0.7262824     0.0000000 = (Z,R,T,OFFSET)
       0            0.0000000     0.0000000     = (OPTION,DS1,DS2)
       0            0.0000000     0.0000000     = (GEOMP ,GF ,GP)
#===================================
      41      1     21 = (I-INDEX,K-INDEX RANGE)
#===================================
      49.6442833    49.3147469    48.7076111    49.6442833 = BEZIER Z
      24.0107365    22.5461102    19.2472210    19.2215939 = BEZIER T
      0.0070102     8.4552860     1.5750616     0.0000000 = (Z,R,T,OFFSET)
       4            0.1000000     0.1000000     = (OPTION,DS1,DS2)
       0            1.0000000     100.0000000    = (GEOMP ,GF ,GP)
#===================================
      91      1     21 = (I-INDEX,K-INDEX RANGE)
#===================================
      80.3315201    81.0230255    80.3315201    80.3315201 = BEZIER Z
      40.1361465    39.0723419    37.2626610    35.3470039 = BEZIER T
      18.3131180    29.9996643    2.6328597     0.0000000 = (Z,R,T,OFFSET)
       4            0.1000000     0.1000000     = (OPTION,DS1,DS2)
       0            1.0000000     100.0000000    = (GEOMP ,GF ,GP)
#===================================
     111      1     21 = (I-INDEX,K-INDEX RANGE)
#===================================
      95.8333969    95.8333969    95.8333969    95.8333969 = BEZIER Z
      53.0426292    51.6058846    50.1691399    48.2534828 = BEZIER T
      15.5525990    45.2400055    3.4795012     0.0000000 = (Z,R,T,OFFSET)
       0            0.0000000     0.0000000     = (OPTION,DS1,DS2)
       0            0.0000000     0.0000000     = (GEOMP ,GF ,GP)

.............................................................
.............................................................
EOF
```

D. GVU Visualization History File

```
#*************************************************************************
#*****************        TIGER-GVU Batch File      ***********************
#*************************************************************************
#
# *****
# NOTE:
# *****
#       (1) Width >= 1
#       (2) Ndup  >= 1
#       (3) Offset: Sets the initial postion of the wall
#       (4) Pid      1: Iconstant 2: Jconstant 3:Kconstant
#       (5) Type     1: Grid      2: Solution
#       (6) Render
#             -Grid 1: WireFrame 2: Shading
#             -Soln 1: WireFrame 2: Solid Contour  3: Line Contour  4: Vector
#       (7) See     0: Hide      1: Show
#       (8) Grid    0: Hide      1: Show
#
#*************************************************************************
#Blk  I1  I2  J1  J2  K1  K2 Pid Clr Wid Ndup Offs Type Render See Grid  Patch
# =======================================================================
   1   1 121   1  36   1   1   3   7   1    1    0    1      1   0    0      1
   1   1 121   1  36  21  21   3   7   1    1    3    1      1   0    0      2
   1  21 121   1   1   1  21   2  90   1    4    0    1      2   0    0      3
   1  55  85   1  21   1   1   3  91   1    1    0    1      1   0    0      4
   1  55  85   1  21  21  21   3  91   1    1    3    1      1   0    0      5
   2   1 121   1  36   1   1   3   7   1    1    0    1      1   0    0      1
   2   1 121   1  36  21  21   3   7   1    1    3    1      1   0    0      2
   2  21 121   1   1   1  21   2  90   1    4    0    1      2   0    0      3
   2  55  85   1  21   1   1   3  91   1    1    0    1      1   0    0      4
   2  55  85   1  21  21  21   3  91   1    1    3    1      1   0    0      5
```