

THREE-DIMENSIONAL HYBRID GRID GENERATOR AND UNSTRUCTURED  
FLOW SOLVER FOR COMPRESSORS AND TURBINES

A Dissertation

by

KYUSUP KIM

Submitted to the Office of Graduate Studies of  
Texas A&M University  
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

December 2003

Major Subject: Aerospace Engineering

THREE-DIMENSIONAL HYBRID GRID GENERATOR AND UNSTRUCTURED  
FLOW SOLVER FOR COMPRESSORS AND TURBINES

A Dissertation

by

KYUSUP KIM

Submitted to Texas A&M University  
in partial fulfillment of the requirements  
for the degree of

DOCTOR OF PHILOSOPHY

Approved as to style and content by:

---

Paul G. A. Cizmas  
(Chair of Committee)

---

Leland A. Carlson  
(Member)

---

Othon K. Rediniotis  
(Member)

---

David L. Rhode  
(Member)

---

Walter E. Haisler  
(Head of Department)

December 2003

Major Subject: Aerospace Engineering

## ABSTRACT

Three-dimensional Hybrid Grid Generator and Unstructured Flow Solver for  
Compressors and Turbines. (December 2003)

Kyusup Kim, M.S., Texas A&M University

Chair of Advisory Committee: Dr. Paul Cizmas

A numerical method for the simulation of compressible turbulent flows is presented. This method includes a novel hybrid grid generation for airfoil cascades and an unstructured mesh flow solver. The mesh tool incorporates a mapping technique and a grid smoothing method. The mapping technique is used to build an initial volume mesh and the grid smoothing method is used to improve the quality of the initial mesh. The grid smoothing is based on the optimization of mesh-quality parameters. The further improvement of the smoothed mesh is achieved by an edge-swapping and node-insertion technique.

The unstructured flow solver is developed for a hybrid grid. This flow solver uses a rotational frame of reference. The convective and viscous fluxes are numerically solved by an upwind scheme and an averaged nodal gradient. A higher-order spatial accuracy is achieved by a piece-wise linear reconstruction. An explicit multi-stage method is employed for integration in time. The Menter's  $k-\omega$  model is implemented to simulate the turbulence effects.

The flow solver is validated against the analytical and experimental results. A parametric study is performed for a high speed centrifugal compressor.

To my family

## ACKNOWLEDGMENTS

I thank my adviser Dr. Paul Cizmas for his guidance. He offered me invaluable insight and timely advice even while busy with other students. His advice was not limited to academic matters only, but included wisdom. Thank you.

I also would like to thank Dr. David Darmofal. He was the first mentor in my graduate study and introduced me the exciting world of CFD. Many lessons he has given to me on the unstructured meshes me were invaluable during the present study.

My many thanks must go to Dr. Zhenxue Han, Dr. Dragos Isvoranu, and Dr. Horia Flitan. They provided me with countless suggestions and constructive criticism, which in many ways helped my work. Dr. Han provided me with a base code for flow simulation.

I would like to thank my committee members, Dr. Leland Carlson, Dr. Othon Rediniotis, and Dr. David Rhode for their support during the course of my work. Friends in our CFD group have been of great help to me. Jason Guarnieri implemented sub-menu windows in GUI programming and also provided me with the graphics files used in GUI windows. Steven Chambers found many errors in the draft and helped me to correct them.

Finally, I am indebted so much to my family who have supported me for a long time and waited patiently through the course of my research.

## TABLE OF CONTENTS

CHAPTER		Page
I	INTRODUCTION . . . . .	1
	A. Statement of work . . . . .	1
	B. Mesh generation . . . . .	1
	C. Flow solver . . . . .	5
	D. Original contributions of the present work . . . . .	8
	E. Outline . . . . .	9
II	PHYSICAL MODEL . . . . .	10
	A. The Navier-Stokes equations . . . . .	10
	B. Reynolds-averaged Navier-Stokes equations . . . . .	12
	1. Nondimensional form . . . . .	15
	C. Turbulence model . . . . .	17
	1. The $k$ - $\omega$ model . . . . .	20
III	NUMERICAL METHOD . . . . .	23
	A. Hybrid mesh generation . . . . .	23
	1. Introduction . . . . .	23
	2. Mapping . . . . .	25
	3. Mesh smoothing . . . . .	26
	a. Sub-mesh . . . . .	27
	b. Centroid smoothing . . . . .	29
	c. Optimization-based smoothing . . . . .	32
	4. Edge swapping and node insertion . . . . .	37
	B. Flow solver . . . . .	39
	1. Spatial discretization . . . . .	41
	a. Introduction . . . . .	41
	b. Integral formulation . . . . .	47
	c. The Navier-Stokes equations in the rotational frame of reference . . . . .	47
	d. Semi-discrete form of the Navier-Stokes equations	50
	e. Inviscid flux . . . . .	53
	f. Gradient computation . . . . .	56
	g. Piecewise linear reconstruction . . . . .	62

CHAPTER	Page
h. Viscous flux . . . . .	64
2. Temporal discretization . . . . .	66
a. Implicit residual smoothing . . . . .	68
3. Boundary conditions . . . . .	69
C. Implementation of turbulence model . . . . .	76
D. Parallel implementation . . . . .	80
E. Graphical user interface . . . . .	84
IV RESULTS . . . . .	89
A. Hybrid mesh generation for an axial turbine rotor . . . . .	90
B. Supersonic inviscid vortex flow in a circular channel . . . . .	102
C. Transonic channel flow over a bump . . . . .	108
D. Laminar boundary layer flow over a semi-infinite flat plate . . . . .	112
E. Turbulent boundary layer flow over a semi-infinite flat plate . . . . .	120
F. Honeywell high speed centrifugal compressor . . . . .	124
V CONCLUSION AND FUTURE WORK . . . . .	132
REFERENCES . . . . .	135
VITA . . . . .	148

## LIST OF TABLES

TABLE		Page
I	Nondimensional variables. . . . .	17
II	Minimum and maximum angle (MS, ES and NI denote mesh smoothing, edge swapping, and node insertion, respectively). . . . .	94
III	Minimum of the quality metric $\tau$ (MS, ES and NI denote mesh smoothing, edge swapping, and node insertion, respectively). . . . .	94
IV	Comparison of transonic channel flow. . . . .	111
V	Maximum Mach number variation as a function of wheel speed. . . . .	126
VI	Size of computational grids. . . . .	129
VII	Parameter variation. . . . .	130



## LIST OF FIGURES

FIGURE	Page
1	Boundaries of background meshes with nodes in the lower inlet corner. . . . . 26
2	Sub-mesh for node $N_0$ . . . . . 28
3	Types of sub-mesh boundary. . . . . 30
4	Shadow at concave corner: intersection of half-spaces a) and b). . . . . 31
5	Types of kernel. . . . . 31
6	Centroid dual cell. . . . . 33
7	Two-dimensional example of median and centroid dual cell for a node $N_0$ . Solid lines denote the boundary of dual cells and thick dotted lines denote the boundary of the first-order stencil $\mathcal{S}_{N_0}^1$ of node $N_0$ . . . . . 43
8	Median-dual cells. The volume contributions to a median-dual cell for a vertex $i$ are shown. Sub-cells are defined by the surface crossing the cell centroid $cc$ , face centroid $fc$ and mid-edge points $i-j$ . . . . . 44
9	Face area contributions (shaded area) from two hexahedral cells to an edge $e_{ij}$ . . . . . 45
10	First-order stencil for 2-D Green-Gauss gradient computation. . . . . 58
11	Example of one-to-two mesh partitions. (a) The shaded area is the buffer layer ( $B_1 \cap B_2$ ) between the block mesh $B_1$ and $B_2$ . (b) The solid lines denote the slave data points ( $B_1^S$ and $B_2^S$ ) and the dotted lines denote the master data points ( $B_1^M$ and $B_2^M$ ). . . . . 83
12	GUI: Opening splash. . . . . 85
13	GUI: Main panel. . . . . 85

FIGURE	Page
14	GUI: Flow model panel. . . . . 86
15	GUI: Flow model panel. Inviscid option turns off laminar/turbulent options. . . . . 86
16	GUI: Boundary condition panel. . . . . 86
17	GUI: Solver control panel. . . . . 87
18	GUI: Input/Output control panel. . . . . 87
19	GUI: Geometry panel. . . . . 87
20	GUI: Execution control panel. . . . . 88
21	Test case airfoil. . . . . 91
22	Variation of airfoil cross sections. . . . . 92
23	Number of cells <i>vs.</i> quality measure $\tau$ . . . . . 95
24	Source mesh at the hub. . . . . 96
25	Hub layer. . . . . 96
26	Layer at 20% span from the hub. . . . . 97
27	Layer at 40% span from the hub. . . . . 98
28	Layer at 60% span from the hub. . . . . 99
29	Layer at 80% span from the hub. . . . . 100
30	Tip layer. . . . . 101
31	Structured mesh for supersonic vortex case. . . . . 103
32	Pressure contour of constant reconstruction case on a fixed frame of reference. $U_i = 2.25$ , $\rho_i = 1$ , $p_i = 1/\gamma$ , $r_i = 1$ , and $r_o = 1.384$ . . . . 106
33	Pressure contour of linear reconstruction case on a rotating frame of reference. $U_i = 2.25$ , $\rho_i = 1$ , $p_i = 1/\gamma$ , $r_i = 1$ , and $r_o = 1.384$ . . . . 107

FIGURE	Page
34	Pressure error of constant reconstruction case. . . . . 107
35	Schematic of transonic flow over a bump. . . . . 109
36	(71 × 31 × 2) mesh for transonic flow over a bump. . . . . 110
37	Iso-Mach contour plot of transonic flow over a bump. . . . . 111
38	Schematic of setup of Blasius boundary layer. . . . . 113
39	Closeup view of mesh for Blasius boundary layer. The plate leading edge is placed at $x = 0$ . . . . . 116
40	Constant reconstruction case. U velocity profile of Blasius boundary layer, $Re_x = 120000$ . Line denotes the Blasius solution and symbols denote the computed value. . . . . 117
41	Linear reconstruction case. U velocity profile of Blasius boundary layer, $Re_x = 120000$ . Line denotes the Blasius solution and symbols denote the computed value. . . . . 118
42	Comparison of skin friction coefficients. CR=Constant Reconstruction, LR=Linear Reconstruction, and BS=Blasius Solution. . . . . 119
43	Mesh for turbulent flow over a semi-infinite plate. . . . . 122
44	Tangential velocity profile for the $k - \omega$ model. The computed velocities are sampled at $Re_x = 1.0498 \times 10^6$ . The experiment data by Wieghardt & Tillman result is sampled at $Re_x = 1.0643 \times 10^6$ . . . . . 123
45	Comparison of the computed skin friction coefficients, $C_f$ , and the experiment data by Wieghardt and Tillmann. The computed result is based on the $k - \omega$ model. . . . . 123
46	Detail of the Honeywell centrifugal compressor impeller geometry. . . . . 125
47	Computational grid. . . . . 126
48	Meridional section through the grid on the back side of the impeller. . . . . 126
49	Leakage pressure location. . . . . 127

FIGURE		Page
50	Grid convergence test. . . . .	128
51	Compressor map. Lines with symbols denote experimental data. Open triangles denote the computed results. . . . .	130
52	Variation of axial thrust load with operating point. . . . .	131
53	Variation of leakage mass flow rate with operating point. . . . .	131

## CHAPTER I

### INTRODUCTION

#### A. Statement of work

The goals of the present work are: 1) to develop a versatile numerical tool for the simulation of compressible turbulent flows, and 2) to investigate the physics of turbomachinery flows. To achieve the goals, the following numerical algorithms have been developed: 1) a novel efficient hybrid mesh generator for airfoil cascades, and 2) an unstructured flow solver. The flow solver incorporates many elements of modern developments of Computational Fluid Dynamics (CFD). The usability of the flow solver is enhanced by addition of a Graphical User Interface (GUI). The flow solver is parallelized such that it can be applied to large problems for fast turnaround time.

#### B. Mesh generation

A mesh can be structured or unstructured, depending on the arrangement of the neighboring points. For a structured mesh, the pattern of the points arrangement is regular and the solution algorithms can take advantage of these regular patterns. A structured mesh is relatively simple to generate compared to an unstructured mesh and the inherent pattern in the mesh makes the solution algorithms simple to implement. A limitation of the structured mesh is that the localized enrichment of the mesh points at certain regions, such as a high curvature boundary region and a region of steep solution gradient, cannot be achieved without disrupting the natural structured pattern of the mesh. The selective local enrichments are necessary for an

---

The journal model is *AIAA Journal*.

efficient use of computational resources. This limits the applications of structured meshes to relatively simple geometries.

An unstructured mesh can be characterized by an irregular stencil pattern. Unlike a structured mesh, the repeated regular stencil patterns are not required for an unstructured mesh and this makes the unstructured mesh suitable for complex geometries and local adaptation. Unstructured mesh generation methods are much more expensive to implement than the methods for structured meshes. In addition, an unstructured mesh requires the explicit description of the irregular patterns. The required information is not only limited to the locations of nodes and the connectivity among nodes via edges, but also the cell faces and the volume cells. The most pronounced added cost is the increased storage requirement. Additional hidden run time cost is due to the increased access time to look up the connectivity table. The cost of the indirect references can be high because of the cache miss and the relatively slow memory access time [1]. Cache miss occurs when the data accesses are not localized and a memory access in a typical computer is approximately 100 times slower than a single CPU operation [2].

In summary, structured meshes and unstructured meshes have their own merits and limitations. The present work attempts to combine the simplicity of a structured mesh and the generality of a unstructured mesh into an efficient mesh generation method. Next paragraphs present a brief review of mesh generation methods related to this study.

Mixed element meshes are widely used in viscous CFD problems because they allow better control of the cell sizes in the viscous region [3]. Prism cells for the viscous region and tetrahedral cells for the inviscid region are a common combination for a three-dimensional (3-D) mesh. The advancing-layers method is a popular way to cluster the cells in the viscous region [4]. However, these advanced methods in

unstructured mesh generation techniques require high costs in their implementation.

Mapping, which is often referred as  $2\frac{1}{2}$ -D meshing, is a popular volume mesh generation method in finite element method analysis. Mapping is typically used for hexahedral meshing [5, 6, 7]. As the name suggests, the method maps a two-dimensional (2-D) source mesh in sweeping motion to generate a volume mesh. The boundary ribs, *i.e.*, series of ring-like closed boundaries, are initially generated along the sweeping direction to guide the mapping process. The spacing between the layers is specified prior to the mapping as is often done in a structured mesh generation process.

Staten *et al.* [7] used unstructured background mesh and linear interpolation to map the nodes from the source layer to the target layer. Mapping alone, however, cannot guarantee that the mesh will be of high quality. Even though the nodes are always projected within the interior, mapping cannot prevent the edges from becoming tangled, especially near a concave boundary. In such a case, a mesh should be corrected because the tangled edges lead to negative volume cells, which render the mesh unusable.

A smoothing method with guaranteed mesh quality improvement is extremely important to correct an invalid mesh and also to further enhance the quality of a valid mesh. Laplacian smoothing is the most widely used smoothing method because of its simplicity and effectiveness in moderate cases[8, 9]. One of the variations of Laplacian smoothing is based on tension spring analogy. Laplacian smoothing has been applied to the unsteady flow computation of a pitching blade [9]. The limitation of this method is that the quality of the smoothed mesh is not always improved [10]. Specifically, the Laplacian method and its variant methods can degenerate the mesh near a concave boundary by creating inverted cells or placing a node outside of the valid region. A “smart”Laplacian smoothing technique works around the problem by

rejecting the displacement of a candidate node computed by a conventional Laplacian method if the new node location decreases the quality measure.

Angle-based smoothing uses a torsion spring analogy and is designed to achieve a smooth variation of angles between neighboring cells [11]. The edge length, however, is not directly controlled and thus it appears that this method requires an initial grid of moderate quality. Another class of the mesh smoothing method, which has recently become popular, is based on the direct optimization of the quality measure [12, 13]. The optimization can be either a global or a local process, with the latter approach generally being cost-effective. The list of the quality measures can be found in Amenta *et al.* [14], who suggest that a mixture of quality measures be used. They show that a smoothing method coupled with a quality measure based only on edge-length can create collapsed cells.

This concludes the review of the mesh generation methods. The proposed hybrid mesh method in the present work combines the strengths of unstructured mesh techniques with the simplicity of structured mesh generation. The mapping method and the mesh optimization technique are also combined in order to develop a simple yet effective volume mesh generation tool for turbomachinery application. An initial mesh is created by mapping a two-dimensional (2-D) unstructured mesh in a predetermined direction, which in the case of turbomachinery is the spanwise direction of an airfoil. Mesh optimization is then applied to remove the invalid elements in the mesh and to increase the quality of the mesh. The proposed method extends the traditional two-dimensional node insertion and edge-swapping methods. The extended methods are designed to improve the quality of the proposed hybrid type of mesh.



### C. Flow solver

The hybrid mesh, which was discussed in the previous section, is the combination of unstructured grid and structured grid. The combined mesh is an unstructured mesh overall. This section reviews the computational methods for the flow models on unstructured meshes.

The Navier-Stokes equations together with the conservation equations of mass and energy describe the motion of a compressible viscous fluid. The equations are a nonlinear system of equations. Even though these equations have been known for centuries, the existence of a general solution is yet to be proved [15]. Aside from some simple cases, obtaining an analytical solution is not trivial because of the nonlinearity. Therefore, experiments are routinely performed to investigate the details of the flow field. With ever increasing computational power, CFD is increasingly used to solve the Navier-Stokes equations for many practical purposes.

Finite Volume Method (FVM) is a discretization method which uses an integral form of the conservation equations. Once a domain of interest is subdivided into a set of non-overlapping volumes, which are often referred to as control volumes, the governing equations must hold not only for the entire domain, but also for each control volume. The final solution obtained in this manner is a set of discrete values represented at the centers and/or the nodes of the control volumes.

FVM can be implemented in a cell-centered approach or a cell-vertex approach, depending on where solutions are stored. Solutions are stored at the centroids of cells in a cell-centered method whereas they are stored at the vertex in a cell-vertex method. The present work uses a cell-vertex method.

Given a three-dimensional (3-D) tetrahedral mesh, the ratio of the number of cells to the number of vertices ranges from approximately 5 to 6. This ratio translates

to the greater memory requirement for a cell-centered approach than a cell-vertex approach.

The computational requirements per unknown are proportional to the number of faces in a cell-centered method and to the number of edges in a cell-vertex method. The ratio of the number of faces to the number of edges is about 2 for a tetrahedral grid. Therefore, the computational resources required for a cell-centered method are considerably higher than for a cell-vertex method [16]. However, for a given mesh, higher quality solutions may be produced using a cell-centered technique [17]. The size of control volumes tends to be smaller in a cell-centered method than in a cell-vertex method and the smaller volumes contribute to the improved solution accuracy of a cell-centered method [18].

The flux in the Navier-Stokes equations is due to convection and diffusion. The present work uses the Roe's approximate Riemann solver to compute the convective flux and averaged gradients with directional derivative to compute the viscous flux [19, 20, 21, 22].

The flux exchange occurs on the surfaces of the control volumes. Consider a common surface shared by two adjacent control volumes. The discrete state vectors are stored at the center of the two volumes. When the state vectors are not equal, the evaluation of the flux function on the shared surface raises a question: which state vector should be used to compute the flux? Godunov's method treats the two distinct states across an interface as the initial conditions for a time dependent problem in order to compute the evolution of the states. The initial value problem is known as the Riemann problem. An exact solution of the nonlinear problem can be computed iteratively, but at high cost. For this reason, an approximated solution is typically used. One of the most used solutions was developed by Roe and its wide acceptance is due to its accuracy, especially its superior solution behavior in shear layers [23], p. 105.

The solution methodology in the Godunov method is as follows: 1) reconstruction, in which the interface states are specified, 2) evolution stage, in which the interaction of the two interface states progresses in time and the evolved solution is obtained as the Riemann solution, and 3) projection, in which the updated solution is distributed to correct the two states at the centers of control volumes.

The discrete states at the centers of control volumes are the volume-average values. The reconstruction is an inverse process of the volume-averaging. Based on the local subset of the averaged states, the variation of the states in a limited region can be expressed as a polynomial function. A constant function represents no variation of a state within a control volume. With a constant reconstruction, the reconstructed states on the cell boundary is equal to the averaged states at the center of the cell. This results in a first-order accurate solution in space. With a linear reconstruction, the solution achieves a second-order accuracy. The  $k$ -exact reconstruction by Barth is a generalization for arbitrary order of reconstruction where  $k$  is the order of reconstruction accuracy [24]. The computational cost for the reconstruction beyond the linear accuracy grows more steeply than the accuracy grows. For this reason, the reconstruction in the present work is limited to the linear reconstruction for second-order spatial accuracy.

The numerical solution based on reconstruction higher than a linear accuracy creates a nonphysical oscillatory response at local extrema. Godunov has shown that the monotonic solution is only possible for a linear method at first-order accuracy. Limiter functions are nonlinear by design to overcome the first order accuracy limitation imposed on a linear method. In one-dimensional sense, the limiter functions control the slope of the reconstructed states to prevent a new extrema from being created. The extension of the concept to a multi-dimension is originally proposed by Barth [25]. The present work uses a limiter function proposed by Venkatakrish-

nan [26].

Many flows in engineering problems are turbulent and consequently are characterized by random fluctuations. Even though an instant realization of a turbulent flow can be directly simulated with the Navier-Stokes equations, because of the small scales, a direct numerical simulation requires enormous number of data points. This makes the direct simulation extremely expensive in terms of the computational resources requirements and unrealistic in their applications to most compressible flows.

The mean motions of turbulent flows can be alternatively determined by the Reynolds-averaged Navier-Stokes equations at relatively low cost. The cost reduction is mainly due to the reduced spatial resolution. Averaging the flow variables, coupled with the nonlinearity of the Navier-Stokes equations, introduces the Reynolds stresses as new unknowns. The closure of the problem is completed with turbulence models. The present work employs the  $k - \omega$  model. This concludes the introduction to the solution methods.

#### D. Original contributions of the present work

- Development of a novel hybrid mesh generation technique, incorporating the mesh optimization methods.
- Development of a compressible flow solver designed for a mixed element unstructured mesh.
- Development of a parallelization strategy for flow solution.
- Prediction of axial thrust load of a centrifugal compressor.

## E. Outline

The next chapter presents the Navier-Stokes equations, the Reynolds-averaged Navier-Stokes equations, and the turbulence model used in the present work. The first part of Chapter III is devoted to a discussion of the hybrid mesh generation method. The mapping and mesh smoothing methods are presented. This is followed by the discussion of the edge-swapping and the node insertion techniques. The discussion of the solution algorithms for the flow solver are divided into the topics related to the spatial discretization and the temporal discretization.

Chapter IV presents the results of the hybrid mesh generation technique, the flow solver validation cases, and a numerical study of a high speed centrifugal compressor. The conclusions and the future work are presented in the last chapter.

## CHAPTER II

## PHYSICAL MODEL

## A. The Navier-Stokes equations

The Navier-Stokes equations in conjunction with the continuity equation and energy balance equation describe the conservation of mass, momentum and energy of a compressible heat conducting viscous fluid. This system of the equations will be collectively referred to as the Navier-Stokes equations for the remainder of this document. The Navier-Stokes equations can be written as

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u_i}{\partial x_i} = 0 \quad (2.1)$$

$$\frac{\partial \rho u_i}{\partial t} + \frac{\partial \rho u_i u_j}{\partial x_j} = \frac{\partial}{\partial x_j} (-p + t_{ij}) \quad (2.2)$$

$$\frac{\partial \rho E}{\partial t} + \frac{\partial \rho H u_j}{\partial x_j} = \frac{\partial}{\partial x_j} (t_{ij} u_i - q_j) \quad (2.3)$$

where the body force due to the gravity and the heat source are ignored.  $\rho$  is the density and  $u_i$  are the velocity components.  $p$  is the static pressure and  $t_{ij}$  is the viscous stress tensor.  $E$  is the total energy per mass,  $H$  is total enthalpy, and  $q_j$  is heat-flux. Einstein summation rule is assumed.

The total energy per mass,  $E$ , and total enthalpy,  $H$ , are defined as

$$E = e + \frac{1}{2}(u_i u_i), \quad H = h + \frac{1}{2}(u_i u_i) \quad (2.4)$$

where  $e$  is the internal energy and  $h$  is the enthalpy.

The momentum balance equations are based on Cauchy's equation of motion with a Newtonian linear stress-strain constitutive equation. With the latter relationship,

the viscous stress tensor  $t_{ij}$  is

$$t_{ij} = \mu \left( \frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) + \lambda \frac{u_k}{x_k} \delta_{ij} \quad (2.5)$$

where  $\mu$  is the dynamic viscosity and  $\lambda$  is the bulk viscosity. The dissipation due to viscosity,  $t_{ij}\partial u_i/\partial x_j$ , must be non-negative and this requires  $\lambda + 2\mu/3 \geq 0$ . Stokes hypothesis assumes that

$$\lambda + \frac{2}{3}\mu = 0 \quad (2.6)$$

such that  $t_{ij}$  vanishes when the fluid motion is in rest. The heat flux  $q_j$  follows the Fourier Law,

$$q_j = -k \frac{\partial T}{\partial x_j} \quad (2.7)$$

where  $k$  is the thermal conductivity and  $T$  is the temperature. Temperature, pressure and density are related by the equation of states. The equation of states for the ideal gas is

$$p = \rho RT \quad (2.8)$$

where  $R$  is the gas constant for air. The internal energy and enthalpy of a calorically perfect gas are functions of temperature only

$$e = c_v T, \quad h = c_p T \quad (2.9)$$

where  $c_v$  and  $c_p$  are the specific heat constants at constant volume and constant pressure, respectively. Using equation (2.9), the pressure in equation (2.8) can be expressed as

$$p = \rho RT = (\gamma - 1) \left( E - \frac{1}{2} u_i u_i \right) \quad (2.10)$$

where the following definitions are used

$$\gamma = \frac{c_p}{c_v}, \quad c_p - c_v = R. \quad (2.11)$$

For air at standard conditions, the ratio of specific heats,  $\gamma$ , is 1.4.

The viscosity is approximated using Sutherland formula

$$\mu = \mu_0 \left( \frac{T}{T_0} \right)^{2/3} \frac{T_0 + S}{T + S}. \quad (2.12)$$

For air,  $\mu_0 = 1.716 \times 10^{-5} \text{Pa} \cdot \text{sec.}$ ,  $T_0 = 273\text{K}$ , and  $S = 111.0\text{K}$ .

## B. Reynolds-averaged Navier-Stokes equations

For many practical problems, fluid motions are turbulent. As stated before, turbulent flows are characterized in part by small scale unsteady random fluctuations. Reynolds decomposition of turbulent flow introduces an averaging procedure such that a variable can be decoupled into the sum of the mean value and the fluctuating value

$$a = \bar{a} + a', \quad (2.13)$$

where  $\bar{a}$  is the mean value and  $a'$  is the fluctuating value. The mean value  $\bar{a}$  is defined as

$$\bar{a} = \frac{1}{\Delta t} \int_0^{\Delta t} a(x, \tau) d\tau. \quad (2.14)$$

The time interval  $\Delta t$  should be large enough compared to the time scale of fluctuation. An illustrative example in Wilcox [27], p. 32, suggests that the integration time of 20 seconds can be adequate for a flow at 10 m/sec in a 5 cm diameter pipe.

For the compressible turbulent flow, density also fluctuates. Density fluctuation can be eliminated from the averaged equations by employing density-weighted or Favre average on select variables rather than Reynolds average. Favre average is defined as

$$a = \tilde{a} + a'', \quad \tilde{a} \equiv \frac{\overline{\rho a}}{\bar{\rho}}. \quad (2.15)$$



The time average of fluctuation term is by definition

$$\overline{a'} = 0 \quad (2.16)$$

whereas for the density-weighted

$$\overline{\rho a''} = 0. \quad (2.17)$$

Some useful results satisfied by the averaging include

$$\overline{a + b} = \bar{a} + \bar{b}, \quad (2.18)$$

$$\overline{c u} = c \bar{u}, \quad (2.19)$$

where  $c$  is a constant,

$$\overline{\frac{\partial u}{\partial t}} = \frac{\partial \bar{u}}{\partial t}, \quad \overline{\frac{\partial u}{\partial x_i}} = \frac{\partial \bar{u}}{\partial x_i} \quad (2.20)$$

$$\overline{ab} = \bar{a}\bar{b}. \quad (2.21)$$

The compressible Reynolds-averaged Navier-Stokes (RANS) equations are obtained by introducing Reynolds and Favre decompositions into the Navier-Stokes equation given in Eqs. (2.1)-(2.3) and taking the time average. Specifically, velocity components, enthalpy and energy, which appear in Eqs. (2.1)-(2.3) as the products multiplied by density, are Favre averaged whereas the other variables are Reynolds averaged

$$u_i = \tilde{u}_i + u_i'', \quad E = \tilde{E} + E'', \quad H = \tilde{H} + H'', \quad (2.22)$$

$$\rho = \bar{\rho} + \rho', \quad p = \bar{p} + p', \quad (2.23)$$

$$t_{ij} = \bar{t}_{ij} + t'_{ij}, \quad q = \bar{q} + q'. \quad (2.24)$$

The selective use of Favre average causes the resultant equations not to include the density fluctuation term [28], pp. 603-606.

The definitions of the energy and enthalpy require careful attention since the

mean values include the turbulence kinetic energy. The total energy and enthalpy of mean flow are denoted by  $\hat{E}$  and  $\hat{H}$ , respectively, and they are related to the mean values

$$\tilde{E} = \tilde{e} + \frac{1}{2} \tilde{u}_i \tilde{u}_i + k = \hat{E} + k \quad (2.25)$$

$$\tilde{H} = \tilde{h} + \frac{1}{2} \tilde{u}_i \tilde{u}_i + k = \hat{H} + k \quad (2.26)$$

where the density-weighted average turbulence kinetic energy is defined as

$$k = \frac{1}{2} \frac{\overline{\rho u_i'' u_i''}}{\bar{\rho}}. \quad (2.27)$$

The Reynolds-averaged Navier-Stokes equations [28], p. 603, are then given by

$$\frac{\partial \bar{\rho}}{\partial t} + \frac{\partial \bar{\rho} \tilde{u}_j}{\partial x_j} = 0 \quad (2.28)$$

$$\frac{\partial \bar{\rho} \tilde{u}_i}{\partial t} + \frac{\partial \bar{\rho} \tilde{u}_i \tilde{u}_j}{\partial x_j} = -\frac{\partial \bar{p}}{\partial x_i} + \frac{\partial}{\partial x_j} (\bar{t}_{ij} - \overline{\rho u_i'' u_j''}) \quad (2.29)$$

$$\frac{\partial \bar{\rho} \hat{E}}{\partial t} + \frac{\partial \bar{\rho} \hat{H} \tilde{u}_j}{\partial x_j} = \frac{\partial}{\partial x_j} (\tilde{u}_i (\bar{t}_{ij} - \overline{\rho u_i'' u_j''}) - \bar{q}_j - \overline{\rho h'' u_j''}) \quad (2.30)$$

where

$$\bar{t}_{ij} = \mu \left( \frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i} - \frac{2}{3} \frac{\partial \bar{u}_k}{\partial x_k} \delta_{ij} \right) \quad (2.31)$$

$$\bar{q}_j = -\frac{c_p \mu}{Pr} \frac{\partial \bar{T}}{\partial x_j}. \quad (2.32)$$

RANS introduces new unknowns, which are the Reynolds stresses  $-\overline{\rho u_i'' u_j''}$  and the Reynolds heat fluxes  $\overline{\rho h'' u_j''}$ . Since the number of equations is less than the number of unknowns, the under-determined system of equations requires a closure.

Using Boussinesq approximation, the Reynolds stresses are linked to the mean flow in a form similar to the viscous stress tensor of Eq. (2.31). Specifically, it is

assumed that the Reynolds stress is related to the gradient of mean velocity by

$$-\overline{\rho u_i'' u_j''} = \mu_T \left( \frac{\partial \tilde{u}_i}{\partial x_j} + \frac{\partial \tilde{u}_j}{\partial x_i} - \frac{2}{3} \frac{\partial \tilde{u}_k}{\partial x_k} \delta_{ij} \right) - \frac{2}{3} \bar{\rho} k \delta_{ij}. \quad (2.33)$$

where  $\mu_T$  is the eddy viscosity determined from a turbulence model. Similarly, the Reynolds heat flux is related to the gradient of mean temperature as

$$\overline{\rho h'' u_j''} = -\frac{c_p \mu_T}{Pr_T} \frac{\partial \bar{T}}{\partial x_j} \quad (2.34)$$

where  $Pr_T$  is the turbulent Prandtl number and  $Pr_T = 0.9$  for air at standard conditions. In this work, the turbulence transportation equations based on the  $k - \omega$  model is used to determine the eddy viscosity,  $\mu_T$ . The last term of Eq. (2.33) which includes  $k$  arises from the need that Eq. (2.33) must to recover the definition of  $k$  in Eq. (2.27) when Eq (2.33) contracts, *i.e.*,  $i = j$ .

Using Eq. (2.33) and Eq. (2.34), Eq. (2.28) can be rewritten as

$$\frac{\partial \bar{p}}{\partial t} + \frac{\partial \bar{\rho} \tilde{u}_j}{\partial x_j} = 0 \quad (2.35)$$

$$\frac{\partial \bar{\rho} \tilde{u}_i}{\partial t} + \frac{\partial \bar{\rho} \tilde{u}_i \tilde{u}_j}{\partial x_j} = -\frac{\partial \bar{p}}{\partial x_i} + \frac{\partial \bar{\tau}_{ij}}{\partial x_j} \quad (2.36)$$

$$\frac{\partial \bar{\rho} \hat{E}}{\partial t} + \frac{\partial \bar{\rho} \hat{H} \tilde{u}_j}{\partial x_j} = \frac{\partial}{\partial x_j} (\tilde{u}_i \bar{\tau}_{ij} - \bar{q}_j) \quad (2.37)$$

where

$$\bar{\tau}_{ij} = (\mu + \mu_T) \left( \frac{\partial \tilde{u}_i}{\partial x_j} + \frac{\partial \tilde{u}_j}{\partial x_i} - \frac{2}{3} \frac{\partial \tilde{u}_k}{\partial x_k} \delta_{ij} \right) \quad (2.38)$$

$$\bar{q}_j = c_p \left( \frac{\mu}{Pr} + \frac{\mu_T}{Pr_T} \right) \frac{\partial \bar{T}}{\partial x_j}. \quad (2.39)$$

### 1. Nondimensional form

The Reynolds-averaged Navier-Stokes equations presented in the previous section are dependent on the choice of the measurement system ( SI, CGS, *etc.*). The comparison

of results obtained in the dimensional form can be difficult if different units or scales are used. The results presented in nondimensional variables such as the Mach number and Reynolds number can often provide more meaningful information than the results in dimensional variables. Another benefit of the nondimensional form is that the magnitude of the variables can be within similar range as long as their reference values are chosen carefully. For example, at standard sea level condition in SI units, the magnitude of pressure is approximately  $10^5$  times larger than that of density. Using non-dimensional variables given in Table I, the magnitudes of pressure and density are of the same order.

The non-dimensionalization used in this work is not unique because the choices of reference variables and the reference values are not unique. The non-dimensional form of the Reynolds-averaged Navier-Stokes equations employed in this work is obtained using the reference length  $L$ , speed of sound  $c_\infty$ , density  $\rho_\infty$  and dynamic viscosity  $\mu_\infty$ .

In the non-dimensional form, the Reynolds-averaged Navier-Stokes equations can be written as

$$\frac{\partial \rho^*}{\partial t^*} + \frac{\partial \rho^* u_j^*}{\partial x_j^*} = 0 \quad (2.40)$$

$$\frac{\partial \rho^* u_i^*}{\partial t^*} + \frac{\partial \rho^* u_i^* u_j^*}{\partial x_j^*} = -\frac{\partial p^*}{\partial x_i^*} + \frac{\partial \tau^*}{\partial x_j^*} \quad (2.41)$$

$$\frac{\partial \rho^* E^*}{\partial t^*} + \frac{\partial \rho^* H^* u_j^*}{\partial x_j^*} = \frac{\partial u_i^* \tau_{ij}^*}{\partial x_j^*} - \frac{\partial q_j^*}{\partial x_j^*} \quad (2.42)$$

where

$$\tau_{ij}^* = \frac{\mu^* + \mu_t^*}{Re} \left( \frac{\partial u_i^*}{\partial x_j^*} + \frac{\partial u_j^*}{\partial x_i^*} - \frac{2}{3} \frac{\partial u_k^*}{\partial x_k^*} \delta_{ij} \right) \quad (2.43)$$

$$q_j^* = -\frac{\gamma}{Re} \left( \frac{\mu^*}{Pr} + \frac{\mu_t^*}{Pr_t} \frac{\partial e^*}{\partial x_j^*} \right). \quad (2.44)$$

Table I. Nondimensional variables.

$x_i^* = x_i/L$	$u_i^* = \tilde{u}_i/c_\infty$	$t^* = tc_\infty/L$	$\rho^* = \bar{\rho}/\rho_\infty$	$p^* = \bar{p}/(\rho_\infty c_\infty^2)$
$E^* = \hat{E}/c_\infty^2$	$H^* = \hat{H}/c_\infty^2$	$\mu^* = \mu/\mu_\infty$	$\mu_t^* = \mu_t/\mu_\infty$	$e^* = \bar{e}/c_\infty^2$

The Reynolds number based on the reference variables in Table I is defined as

$$Re = \frac{\rho_\infty c_\infty L}{\mu_\infty}. \quad (2.45)$$

### C. Turbulence model

Most of the flows of interest in engineering applications are turbulent. One method to solve turbulent flows is Direct Numerical Simulation (DNS). This method is still too expensive for practical engineering purposes despite ever increasing computational power [29], pp. 338-340, [27], pp. 381-386. An alternative cost effective approach is turbulence modeling. One of the advantages of turbulence models is that solutions can be obtained on relatively coarse meshes. Consequently, the required computational cost is significantly less than that of DNS.

The current work employs turbulence modeling. Specifically, the  $k - \omega$  model is used. The turbulence model is used to determine the eddy viscosities. Once the distribution of eddy viscosities is determined, the RANS equations can be solved in the same manner as a laminar flow case. The turbulence effects are captured in the mean flow equations through the effective viscosity and conductivity.

Similar to the physical viscosity, the eddy viscosity  $\nu_T$  has dimensions of  $L^2/T$  where  $L$  is the dimension of length and  $T$  is time. Using a representative turbulence time, length and velocity scale,  $t_T$ ,  $v_T$  and  $l_T$  respectively, the eddy viscosity  $\nu_T$  can

be defined as

$$\nu_T = v_T l_T = \frac{l_T^2}{t_T} = v_T^2 t_T. \quad (2.46)$$

The turbulence model considered herein defines the eddy viscosity in terms of the appropriate scales by using the model equations. The  $k - \omega$  model defines the eddy viscosity in terms of the turbulence kinetic energy  $k = v_T^2$  and the specific dissipation rate  $\omega \approx 1/t_T$ . Before the detailed description of the model is presented, some of the limitations of the model are noted in the following.

The two equation model relies on the previously mentioned Boussinesq hypothesis. The hypothesis assumes that the Reynolds stresses are linearly related to the strain-rate of mean flow. Furthermore, it assumes that the proportionality between the stresses and the velocity gradient can be expressed using an eddy viscosity. The end result is that the predicted eddy viscosity is isotropic whereas many types of flows, such as contracting flows, wall bounded flows, exhibit anisotropic turbulences. In the case of the flat plate boundary layer flows, the measured stresses exhibit the anisotropic ratio of 4:2:3 for  $\bar{u}'^2:\bar{v}'^2:\bar{w}'^2$  [27], p. 40.

The deficiency of the hypothesis is also pronounced when a sudden change of mean strain rate occurs and when flows experience increased rate of strain. The limitations are expected because there is no mathematical theory that suggests that the Reynolds stresses and local strain-rate are directly related. The hypothesis assumes that the production and dissipation are locally balanced. When the balance is disturbed, the turbulence effect can be delayed, meaning that the downstream Reynolds stresses are strongly dependent on the upstream condition. This “history effect” is not recognized by the  $k - \omega$  model.

Wall curvature is also known to introduce increased strain rate, which makes the Boussinesq hypothesis invalid [30]. Experiments show that turbulence is enhanced on

concave walls when compared to flat plate flows. On the other hand, the Reynolds stresses on convex walls are reduced. The Reynolds stress equations must include the production term due to the Coriolis effect when expressed in a rotational frame of reference [31]. The production term disappears when the Reynolds stress equations are contracted to the kinetic energy equation. The  $k - \omega$  model, which depends on the kinetic energy equation, therefore, cannot account for the curvature or rotational effect.

Kim and Rhode [32] studied a modified law-of-the-wall model based on the variation of turbulence length scale due to wall curvature and reported improved results on swirling flows on curved walls. Wilcox and Chambers [33] proposed a modified  $k$ -equation to introduce a curvature effect into the  $k - \omega$  model. Modifications to the  $\epsilon$  equation in the  $k - \epsilon$  model, which is equivalent to the modification of the  $\omega$  equation, can be found in Launder et al. [34] and Hellsten [31]. Despite the known effects, the investigation of all of the limitations of the  $k - \omega$  model is beyond the scope of the present work.

The turbulence model used in the present work is based on incompressible flow. The incompressible turbulence model can be employed for compressible flow at reasonable high Mach numbers flow [28], pp. 603-606. This is based on Morkovin's hypothesis that states the effect of density fluctuation on the turbulence remains small for Mach numbers below 5 [35].

Despite the limitations and the assumptions made, the turbulence models have been applied successfully on many engineering problems [30, 36, 37, 38, 31]. In the following, the details the  $k - \omega$  model are presented.

### 1. The $k$ - $\omega$ model

The  $k - \omega$  model is a two-equation model. The model determines the eddy viscosity using two differential equations, one for the turbulent kinetic energy  $k$  and the other one for the specific dissipation rate  $\omega$ . The  $k - \omega$  model was originally developed by Wilcox [27]. The modified model used in the present work is due to Menter [39]. The modification involves the blending of the  $k - \epsilon$  model to the original  $k - \omega$  model. A well known deficiency in the original  $k - \omega$  model is the solution sensitivity to the freestream conditions [27, 40, 41, 31]. The blending strategy relieves the  $k - \omega$  model from this sensitivity.

The  $k$  transport equation in the  $k - \omega$  model is based on the turbulent energy budget equation. The budget equation [42], p. 63, is given by

$$\begin{aligned} \frac{Dk}{Dt} = & \tau_{ij} S_{ij} - \left[ 2\nu \overline{s_{ij} s_{ij}} - \frac{\partial}{\partial x_j} \left( \overline{\nu u'_i \frac{\partial u'_j}{\partial x_i}} \right) \right] \\ & + \frac{\partial}{\partial x_j} \left[ \nu \frac{\partial k}{\partial x_j} - \left( \frac{1}{2} \overline{u'_i u'_i u'_j} + \frac{1}{\rho} \overline{p' u'_j} \right) \right] \end{aligned} \quad (2.47)$$

where

$$S_{ij} = \frac{1}{2} \left( \frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right), \quad s_{ij} = \frac{1}{2} \left( \frac{\partial u'_i}{\partial x_j} + \frac{\partial u'_j}{\partial x_i} \right). \quad (2.48)$$

The first term of the right-hand side of Eq. (2.47) is the production term. This term is always positive because both  $\tau_{ij}$  and  $S_{ij}$  are symmetrical by definition. The second group of terms in the first bracket is the dissipative term. As a group, this term differs from the true dissipation term. However, the difference is shown to be quite small [27]. Besides the unknown  $k$ , the budget equation contains extra unknowns that need modeling.

The  $k - \omega$  model equations are given by

$$\frac{D\rho k}{Dt} = P - \beta^* \rho \omega k + \frac{\partial}{\partial x_j} \left[ (\mu + \sigma_k \mu_T) \frac{\partial k}{\partial x_j} \right] \quad (2.49)$$



$$\frac{D\rho\omega}{Dt} = \gamma_\omega \frac{\omega}{k} P - \beta\rho\omega^2 + \frac{\partial}{\partial x_j} \left[ (\mu + \sigma_\omega \mu_T) \frac{\partial \omega}{\partial x_j} \right] + (1 - F_1) \frac{2\rho\sigma_{\omega 2}}{\omega} \frac{\partial k}{\partial x_j} \frac{\partial \omega}{\partial x_j} \quad (2.50)$$

where  $P$  is the Boussinesq approximation of the production of the turbulence kinetic energy,

$$P = \mu_T \left( \frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3} \frac{\partial u_k}{\partial x_k} \delta_{ij} \right) - \frac{2}{3} \rho k \delta_{ij} \frac{\partial u_i}{\partial x_j}. \quad (2.51)$$

The dissipation term in the  $k$  budget equation is defined as

$$\epsilon \equiv \beta^* \rho \omega k \approx 2\nu \overline{s_{ij}s_{ij}} - \frac{\partial}{\partial x_j} \left( \overline{\nu u'_i \frac{\partial u'_j}{\partial x_i}} \right). \quad (2.52)$$

The last two terms in Eq. (2.47) are referred to as the transport by turbulent kinetic energy and the transport by pressure fluctuation. They are approximated as a diffusion of  $k$ ,

$$\sigma_k \mu_T \frac{\partial k}{\partial x_j} \approx - \left( \frac{1}{2} \overline{u'_i u'_i u'_j} + \frac{1}{\rho} \overline{p' u'_j} \right). \quad (2.53)$$

The transport equation for  $\omega$  is a postulated equation and it is modeled similarly [27], pp. 119-121. The last term of Eq. (2.50) arises from the transformed  $k - \epsilon$  model and it is referred to as a cross-diffusion term. The model coefficients are determined by blending the  $k - \omega$  constants and  $k - \epsilon$  constants, which are

$$[\sigma_k, \sigma_\omega, \beta]^T = F_1 [\sigma_{k1}, \sigma_{w1}, \beta_1]^T + (1 - F_1) [\sigma_{k2}, \sigma_{w2}, \beta_2]^T \quad (2.54)$$

where

$$\begin{aligned} [\sigma_{k1}, \sigma_{w1}, \beta_1]^T &= [0.5, 0.5, 0.0750]^T \\ [\sigma_{k2}, \sigma_{w2}, \beta_2]^T &= [1.0, 0.856, 0.0828]^T. \end{aligned} \quad (2.55)$$

The blending coefficient  $F_1$  is designed to approach 0 at wall and 1 in the freestream. The blending coefficient,  $F_1$ , is defined as

$$F_1 = \tanh(\Gamma^4) \quad (2.56)$$

where the argument  $\Gamma$  is given by

$$\Gamma = \min \left[ \max \left( \frac{\sqrt{k}}{\beta^* \omega d}, \frac{500\nu}{\omega d^2} \right), \frac{4\rho\sigma_{\omega_2} k}{\text{CD}_{k\omega} d^2} \right]. \quad (2.57)$$

$d$  is the distance to the wall and  $\text{CD}_{k\omega}$  is defined as

$$\text{CD}_{k\omega} = \max \left[ \frac{2\rho\sigma_{\omega_2}}{\omega} \frac{\partial k}{\partial x_j} \frac{\partial \omega}{\partial x_j}, 10^{-20} \right]. \quad (2.58)$$

Finally, the eddy viscosity  $\nu_t$  is defined as

$$\nu_T = \alpha^* \frac{k}{\omega} \quad (2.59)$$

where  $\alpha^* = 1$ .

## CHAPTER III

### NUMERICAL METHOD

The first part of this chapter presents the hybrid mesh generation method <sup>1</sup>. This is followed by the governing equations of the fluid motion. The rest of the chapter is devoted to describing of the solution methods for flow simulation.

The flow model is based on the Reynolds-averaged Navier-Stokes (RANS) equations and the  $k - \omega$  model. The discretization of the governing equations is based on the FVM. The spatial terms and the temporal terms are discretized independently. The solution algorithms are developed for unstructured meshes.

#### A. Hybrid mesh generation

##### 1. Introduction

The hybrid mesh generation technique developed in the present work is designed for turbomachinery applications. The present method takes advantage of the geometry features which are specific to turbomachinery cascades. Consequently, the cross sections of the flow domain at different radial location are discretized using topologically identical meshes. The method presented herein generates the 3-D mesh as a sequence of 2-D layers stacked along the span of the airfoil.

The 2-D grid is a combination of a structured O-grid, next to the airfoil surface, and an unstructured grid exterior to the O-grid. The O-grid is used to resolve the viscous region in the vicinity of the blade. The remainder of the 2-D domain is delimited by the outer boundary of the O-grid, the periodic boundaries, and the inlet

---

<sup>1</sup>Part of this chapter has been previously published in the *AIAA Journal of Propulsion and Power* [43] and copyrighted by the present author and Dr. Paul Cizmas.

and outlet boundaries. This 2-D domain is filled by triangular cells. The unstructured mesh is used to make the mesh generator applicable to a complex geometry for which conventional structured meshes may fail to produce an acceptable result.

A 2-D mesh is first generated at a select spanwise location. This 2-D mesh is a combination of a structured cells and unstructured cells. The 2-D “source ” mesh is then projected to “target ” layers at different spanwise locations. As many target layers as necessary are generated in order to capture blade geometry variation from hub to tip. Edges are then added between the “twin ” nodes of adjacent target layers in order to create volume cells. Consequently, the 3-D cells are either prisms or hexahedra, depending on whether they correspond to triangle or quadrilateral 2-D cells.

To improve mesh quality, the grid is smoothed by relocating the interior and periodic boundary nodes to their optimal points. The smoothing procedure employs the optimization of mesh quality using a steepest decent method. The O-grid region is not smoothed. The outer boundary of the O-grid serves as a fixed boundary for the smoothing process.

The spanwise variation of the airfoil shape results in a deformation of the elements of the mapped meshes. This deformation reduces the quality of the mesh. The mesh quality is further reduced by the fixed periodic boundaries that limit the redistribution of the nodes close to the boundaries. Unlike typical boundaries in solid modeling, the periodic boundaries of turbomachinery cascade do not have to be fixed. The only restriction is that the circumferential distance between two periodic nodes on the periodic boundaries must be constant. The method proposed herein allows that the nodes of the periodic boundaries move. The position of the nodes on the periodic boundaries results from the smoothing of the neighboring internal nodes. This additional degree of freedom for the nodes on the periodic boundary yields a

better quality mesh.

A major advantage of the technique developed in this study is that the complexity of a 3-D algorithm is reduced to that of a 2-D algorithm. The structured nature of the mesh in the spanwise direction simplifies mesh generation. In addition, flow computation benefits from the simple partition of the mesh. Because the connectivity is identical among the layers, the communication across them is simplified. This makes the mapped mesh attractive to parallel flow computation.

## 2. Mapping

The initial step in the mapped mesh generation is the definition of the boundary ribs, which are the series of ring-like closed boundaries to guide the direction of mapping. The blade surface is cross-sectioned at a predetermined interval along the spanwise direction. An equal number of nodes is placed around the airfoil at each spanwise location. The outer boundaries, *i.e.*, the inlet, outlet, and periodic boundaries, are defined at every radial location. An equal number of nodes is used for the outer boundaries at each spanwise location.

An O-grid mesh is generated to discretize the viscous region around the blade at each spanwise location. The hexahedral volume cells in the O-grid region are constructed by connecting the topologically identical quadrilateral cells of adjacent layers. These hexahedral cells are fixed permanently for the remainder of the mesh generation steps. The outer boundary of the O-grid block defines the inner boundary of the unstructured mesh.

The mesh of the source layer is generated first. The interior of the domain is bounded by the outer boundaries (inlet, outlet, and periodic) and the outer boundaries of the O-grid. This interior domain is tessellated to produce a mesh with triangular cells. The routine “Triangle” [44] is used herein to generate the 2-D unstructured

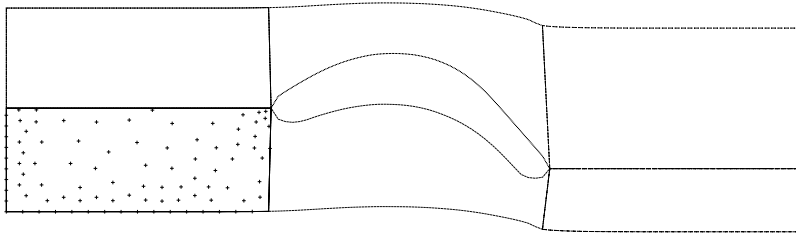


Fig. 1. Boundaries of background meshes with nodes in the lower inlet corner.

mesh. This mesh serves as the “source” to be mapped on the “target” layers.

The domain to be discretized by the unstructured mesh is divided into six blocks, as shown in Fig. 1. Structured meshes are generated for each block. These additional meshes are called the background meshes. Similar background meshes are constructed on the target layers so that a cell to cell mapping is established between the source and the target background meshes. The source nodes are then projected on the target layer into the corresponding cell of the background mesh. Bilinear interpolation is performed to find the mapped location of the node within the cell.

### 3. Mesh smoothing

A mapped mesh may not be acceptable for flow computation due to the lack of explicit quality control in the mapping procedure. Unless the variation from source to target layer is small, the mapping can produce low quality cells or even tangled elements. One of the most effective techniques for increasing the quality of a 3-D mesh is the smoothing method that is based on the optimization of mesh quality parameters. The computational cost of this technique, however, can be very high. The majority of the computational cost is due to the computation of quality measures. Gradient computation can also contribute significantly to the computational cost in

the steepest descent method.

The present study uses a 2-D quality measure rather than a 3-D quality measure. The underlying assumption is that the variation from one layer to the adjacent layer is small. This is true for turbomachinery airfoils, which usually have a continuous variation of the cross-section in the spanwise direction. Consequently, the overall quality of a prism cell, which has two triangular faces on the two adjacent layers, is dominated by the quality of the triangular cells. The mesh generation essentially becomes a series of 2-D mesh operations of area mesh smoothing. This approach not only reduces the computational cost significantly, but also simplifies the implementation.

#### a. Sub-mesh

A sub-mesh is an entity used for local sub-mesh smoothing. A sub-mesh is defined by a set of cells that share a common node. Typical convex and star sub-meshes are shown in Figs. 2a and 2b. All corners of a convex sub-mesh are convex. At least one corner is concave in a star sub-mesh. The type of the sub-mesh is important in the smoothing process. Mesh smoothing is applied iteratively to sub-meshes. Given a sub-mesh as an input, local mesh optimization is then reduced to finding the optimal location of the common node while the other nodes on the boundary of the sub-mesh are stationary.

For a node on the periodic boundary, a sub-mesh is defined as shown in Fig. 2c. The figure shows a sub-mesh for node  $N_0$ , which lies on the periodic boundary. The sub-mesh for node  $N_0$  can be assembled by using the boundary nodes  $N_5$  and  $N_6$  of the sub-mesh of node  $N_{0'}$ , the pair to node  $N_0$ . Once the optimized location of  $N_0$  is found, the location of its pair node  $N_{0'}$  is updated using the new location of  $N_0$  plus the pitch. For inlet and outlet boundaries the sub-mesh is generated by mirroring the half-moon sub-mesh with respect to the boundary face.

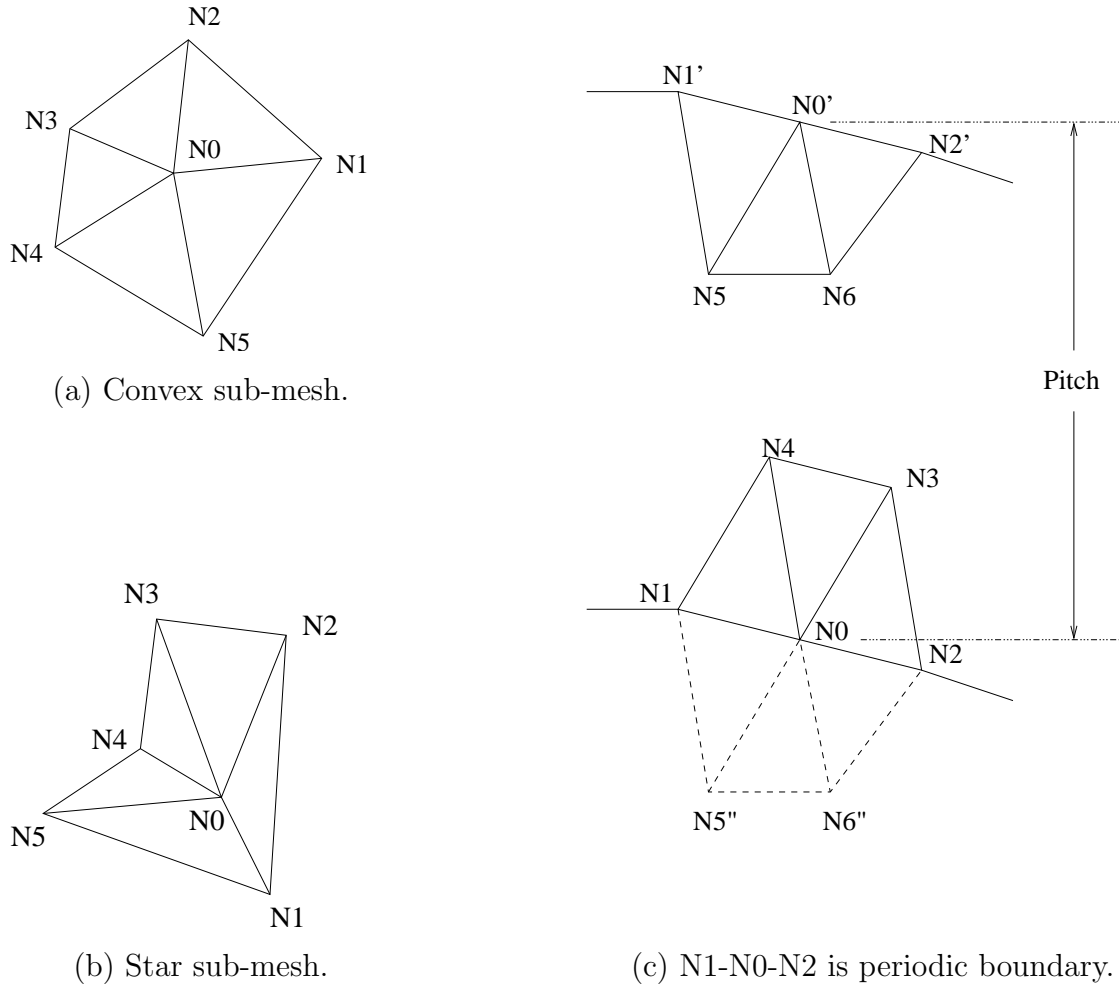


Fig. 2. Sub-mesh for node  $N_0$ .

Figure 3 shows the types of sub-mesh boundaries. The boundary of a sub-mesh can be simple, non-simple (tangled), or inverted simple. The boundary is considered positive if the nodes on the boundary are ordered in a counterclockwise direction. For the non-simple case, the edge segments cross each other and the cells overlap. The resulting mesh is unusable. The inverted simple sub-mesh, shown in Fig. 3d, occurs in extreme cases. This mesh is unusable.



b. Centroid smoothing

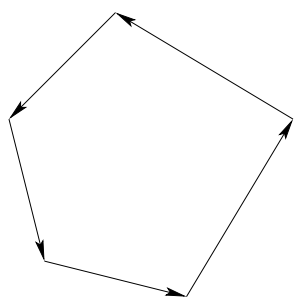
Centroid smoothing places the common node,  $N_0$ , at the centroid of the valid region. The valid region for the common node is defined as the area where the node can be placed such that all the resultant cells of the sub-mesh have positive areas. Tangled edges are caused by placing the common node outside of the valid region. For the non-simple case, a valid region cannot be defined.

The valid region depends on the type of sub-mesh. The common node of a convex sub-mesh can be placed anywhere inside the boundary in order to produce a valid triangulation. Unlike the case of the convex sub-mesh, only part of the interior of a star sub-mesh is a valid region. The valid region of a star sub-mesh is determined by the kernel, which is defined next.

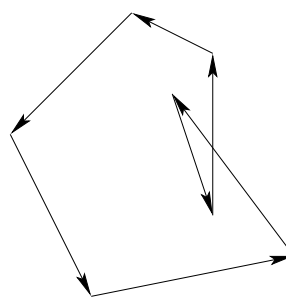
The shadow is defined as the set of points from which a line can be drawn to any point in the interior without crossing the boundary. A shadow is associated with each concave corner. For a sub-mesh with multiple concave corners, the kernel is defined as the intersection of the shadows [45]. For a star sub-mesh with only one concave corner, the kernel is identical to the shadow.

To construct a shadow, half-spaces are defined with respect to each boundary segment that forms the concave corner. Figure 4a shows the half-space formed by the line running through the nodes  $N_1$  and  $N_2$ . The second half-space is defined by the nodes  $N_2$  and  $N_3$ , as shown in Fig. 4b. The intersection of the two half-spaces defines the shadow of the concave corner at  $N_2$  as shown in Fig. 4c. There is a second concave corner in the sub-mesh shown in Fig. 4, so that the shadow associated with the second corner has to be determined to obtain the kernel. The kernel of the sub-mesh is shown in Fig. 5a.

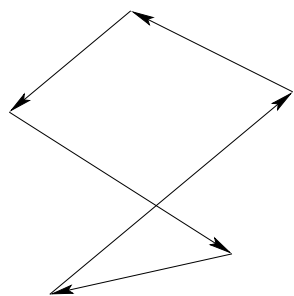
Near a concave region with high curvature, such as at the airfoil leading or trailing



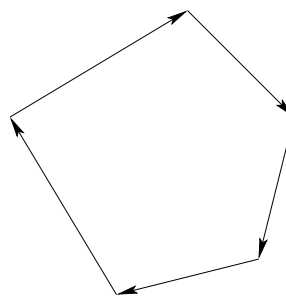
(a) Simple.



(b) Non-simple (Concave).



(c) Non-simple (Reflective).



(d) Inverted.

Fig. 3. Types of sub-mesh boundary.

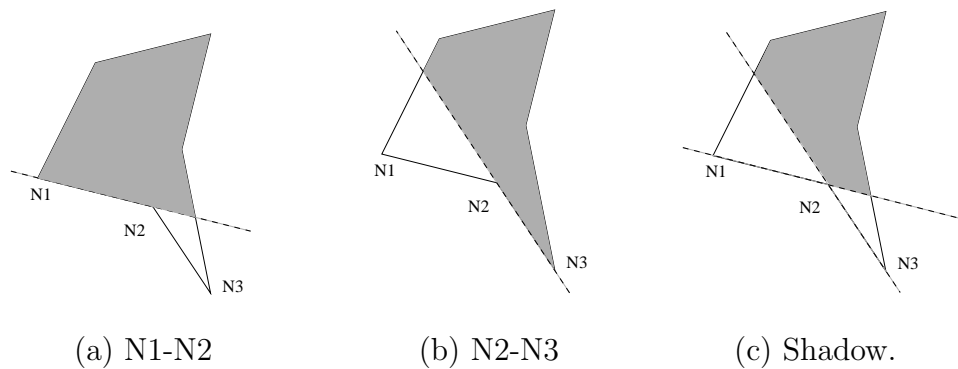


Fig. 4. Shadow at concave corner: intersection of half-spaces a) and b).

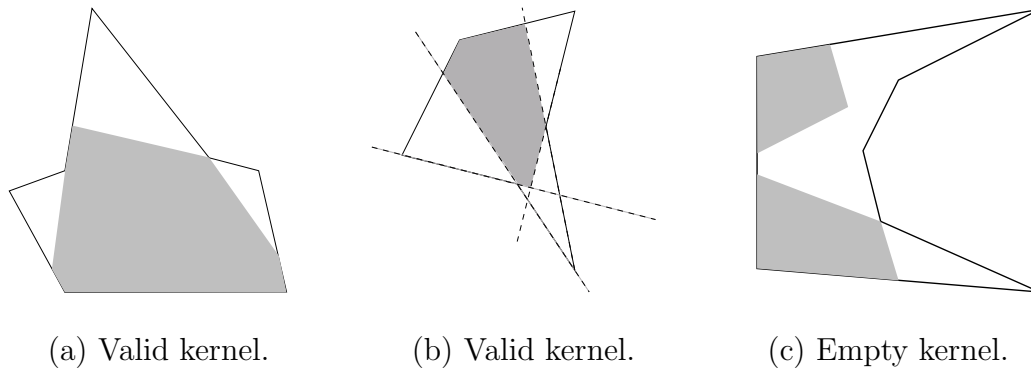


Fig. 5. Types of kernel.

edge, an intersection of shadows may not be found. In this case, the kernel is empty as shown in Fig. 5(c). In such a case, the valid location of the common node cannot be determined locally. In most cases, the smoothing of the neighboring sub-meshes changes the shape of the empty kernel sub-mesh so that a valid region can be found in later iterations. The prolonged presence of the empty kernel during the iterations, however, indicates the limitations of mesh smoothing. A valid triangulation can be obtained for the case of the empty kernel shown in Fig. 5(c) by introducing a new node at one of the two shadows and placing the existing common node in the other shadow. Further details on node insertion are given in the next section.

c. Optimization-based smoothing

Optimization-based smoothing is a technique based on the steepest descent optimization method [13]. This smoothing method operates on the sub-mesh and computes the new location of the internal node so as to increase the local minimum quality metric. The quality metric used throughout this paper is based on the ratio of the triangular cell area to the sum of the squared length of each of its edges. For a triangle cell with nodes at  $\mathbf{X}^1$ ,  $\mathbf{X}^2$ , and  $\mathbf{X}^3$ , the metric  $\tau$  is defined by

$$\tau = C \frac{(\mathbf{E}_{12} \times \mathbf{E}_{13}) \cdot \mathbf{e}_n}{\mathbf{E}_{12} \cdot \mathbf{E}_{12} + \mathbf{E}_{13} \cdot \mathbf{E}_{13} + \mathbf{E}_{23} \cdot \mathbf{E}_{23}}, \quad (3.1)$$

where  $C$  is a constant and  $\mathbf{E}_{ij}$  is the edge vector from  $\mathbf{X}^i$  to  $\mathbf{X}^j$ . The unit vector,  $\mathbf{e}_n$ , is normal to the triangle cell. Consequently, the vector  $\mathbf{E}_{12} \times \mathbf{E}_{13}$  has the same direction and orientation as the unit vector  $\mathbf{e}_n$  if the nodes of the cell  $T_{123}$  are arranged in a counterclockwise order. For an inverted cell, the numerator should be negative. The metric  $\tau$  has bounds of

$$-1 \leq \tau(T) \leq 1 \quad (3.2)$$

by setting  $C = 2\sqrt{3}$ . The metric for an equilateral triangle is either -1 or 1 depending on the orientation.

With a sub-mesh about a node  $N^i$ , the new location  $\hat{\mathbf{X}}^i$  of the node  $N^i$  is determined by

$$\hat{\mathbf{X}}^i = \mathbf{X}^i + \gamma^i \mathbf{G}^i, \quad (3.3)$$

where  $\mathbf{G}^i$  is the gradient of the metric at node  $N^i$  and  $\gamma^i$  is a factor to control the magnitude of the node displacement along  $\mathbf{G}^i$ . The new location of the node changes the metrics of the cells. The new minimum of the quality metric should be greater

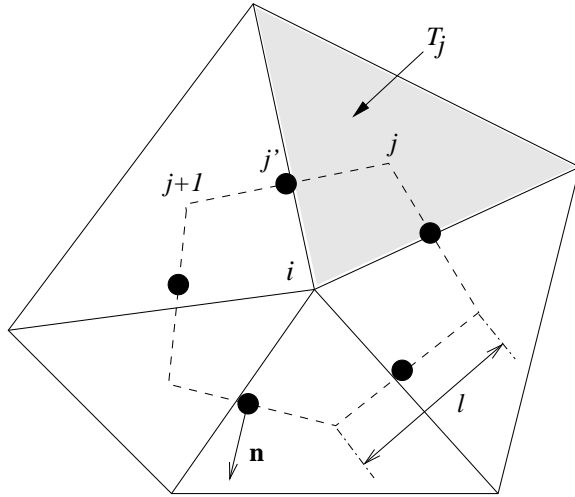


Fig. 6. Centroid dual cell.

than the old minimum, that is

$$\min(\hat{\tau}_j(\hat{\mathbf{X}}^i)) > \min(\tau_j(\mathbf{X}^i)). \quad (3.4)$$

Note that the subscripts are associated with the cells and the superscripts are associated with the nodes.

The major task in the optimization is the gradient computation, which requires repeated computations of the quality measures. A cost-effective way to compute the gradient is desired. The gradient computation based on the divergence theorem is used to approximate the uphill direction of the quality metric at the common node, however, the method is applicable only for simple sub-mesh boundaries.

A closed path connecting the centroids of adjacent cells in a counterclockwise direction is defined. This closed path generates a centroid dual cell, as shown in Fig. 6. The divergence theorem applied for  $\tau$  on the centroid dual cells

$$\int_A \nabla \tau dA = \oint_{\partial A} \tau \mathbf{n} dl, \quad (3.5)$$

is used to approximate the gradient at the common node,  $N^i$ :

$$\nabla\tau^i \approx \mathbf{G}^i = \frac{1}{A} \sum_{j=1}^{m^i} \tau_{j'} \mathbf{n}_{j'} l_{j'}, \quad (3.6)$$

where  $A$  is the interior area of the integral path,  $j'$  denotes the average between  $j$  and  $j+1$ , and  $m^i$  is the number of cells of the sub-mesh  $i$ . Herein, the quality metric for each cell is assumed to be associated with the centroid of the cell. The gradient is first order accurate and thus exact for a linearly varying  $\tau$ . The procedure for calculating the gradient is similar to flux summation in the finite volume method [24].

If the centroid path cannot be defined, as it happens with tangled and inverted sub-mesh cases, the gradient is computed using the perturbed  $\tau$  [12]. For each cell  $T_j$  of the sub-mesh, the gradient is approximated by

$$\nabla\tau_j \approx \frac{\tau_j^{\delta x} - \tau_j}{\delta x} \mathbf{e}_x + \frac{\tau_j^{\delta y} - \tau_j}{\delta y} \mathbf{e}_y \quad (3.7)$$

where  $\mathbf{e}_x$  and  $\mathbf{e}_y$  are the unit vectors in the  $x$  and  $y$  directions. The perturbed metrics,  $\tau^{\delta x}$  and  $\tau^{\delta y}$ , are computed with the location of the common node perturbed by  $\delta x$  and  $\delta y$ , respectively. The magnitudes of  $\delta x$  and  $\delta y$  are adjusted to the local scale, *i.e.*, 1% of the breadth of the sub-mesh. The gradient at the common node  $\mathbf{G}^i$  is set to

$$\mathbf{G}^i = \nabla\tau_{j^*} \quad (3.8)$$

where cell  $j^*$  has the minimum quality metric value

$$\tau_{j^*} = \min\{\tau_j\}, \quad j = 1, \dots, m^i. \quad (3.9)$$

Once the uphill direction,  $\mathbf{G}^i$ , of the quality metric is determined, the displacement of the common node along the uphill direction is specified by  $\gamma^i \mathbf{G}^i$ . Two procedures to compute  $\gamma^i$  are presented herein. The selection of the procedure depends

on the type of the sub-mesh.

For a simple sub-mesh case, centroid smoothing sets the initial location of the common node at the centroid of the valid region. The centroid is not the optimal point in most cases, but fairly close to it. Thus the search for the optimal node location is limited to a small region near the current position.

The search for the new node location is accomplished as follows. A line is drawn from the current node location in the uphill direction given by Eq. 3.6. The line intersects the boundary of the valid region. The length between the current location of the node and the intersection point determine the maximum  $d\mathbf{X}^i$ . An initial  $\gamma^i$  can be set to a value such that  $\gamma^i \mathbf{G}^i$  becomes a fraction of the maximum  $d\mathbf{X}^i$ . In the present implementation the value of the fraction is 10%.

If the initial translation of the node along  $\mathbf{G}^i$  does not improve the minimum  $\tau$ , the search distance is reduced by halving  $\gamma^i$ . The procedure repeats until the new minimum is greater than the current minimum. The number of trials, however, is limited to a fixed number of iterations.

A different methodology is used to calculate  $\gamma^i$  for non-simple or inverted sub-meshes. The presence of a non-simple or inverted sub-mesh voids all validity of the mesh. Therefore, it is critical to find the proper new location of the common node, which will transform such a sub-mesh to a valid simple sub-mesh. Furthermore, when a sub-mesh is not simple, the variations of the quality metrics appear to be extremely sensitive to the displacement magnitude of the common node.

A new quality metric for a cell  $T_j$  can be approximated using Taylor series [12]

$$\begin{aligned} \tau_j(\hat{\mathbf{X}}^i) &= \tau_j(\mathbf{X}^i + \gamma^i \mathbf{G}^i) \\ &= \tau_j(\mathbf{X}^i) + \gamma^i \mathbf{G}^i \cdot \left. \frac{\partial \tau_j}{\partial \mathbf{X}} \right|_{\mathbf{X}^i} + H.O.T. \end{aligned} \quad (3.10)$$

where  $\hat{\mathbf{X}}^i$  denotes the new location of node  $N^i$ . Substituting the gradient term by the gradient approximation  $\mathbf{G}_j$  for a cell  $T_j$ , where  $\mathbf{G}_j \equiv \nabla\tau_j$ , Eq. 3.10 becomes

$$\tau_j(\hat{\mathbf{X}}^i) \approx \tau_j(\mathbf{X}^i) + \gamma^i \mathbf{G}^i \cdot \mathbf{G}_j. \quad (3.11)$$

The value of the gradient  $\nabla\tau_j$  is approximated by the perturbed quality metric, as shown in Eq. 3.7. For the cell  $T_{j^*}$  with the current minimum  $\tau_{j^*}$ , Eq. 3.11 can be written as

$$\tau_{j^*}(\hat{\mathbf{X}}^i) \approx \tau_{j^*}(\mathbf{X}^i) + \gamma^i \mathbf{G}^i \cdot \mathbf{G}^i \quad (3.12)$$

where the second  $\mathbf{G}^i$  term recalls the definition of the gradient for the cell with the minimum quality measure, as shown in Eq. 3.8.

Equation 3.12 shows that the metric of the cell with the old minimum  $\tau_{j^*}$  will always improve because the inner product of  $\mathbf{G}^i$  and  $\mathbf{G}^i$  is positive. The product  $\mathbf{G}^i \cdot \mathbf{G}_j$  in Eq. 3.11, however, can be negative. When this occurs, the new  $\tau_j$  decreases for a positive  $\gamma^i$ . Therefore, when the product is negative, the decreasing metric should be restricted by being equal to or greater than the improved value of the old minimum metric. For a sub-mesh with  $m^i$  cells, this condition implies

$$\tau_{j^*}(\hat{\mathbf{X}}^i) \leq \tau(\hat{\mathbf{X}}^i), \quad 1 \leq j \leq m^i, j \neq j^*, \quad (3.13)$$

which can be expanded to

$$\tau_{j^*} + \gamma^i \mathbf{G}^i \cdot \mathbf{G}^i \leq \tau_j + \gamma^i \mathbf{G}^i \cdot \mathbf{G}_j. \quad (3.14)$$

Finally, rearranging Eq. 3.14, the minimum  $\gamma^i$  is

$$\gamma^i = \min \left\{ \frac{\tau_j - \tau_{j^*}}{\mathbf{G}^i \cdot \mathbf{G}^i - \mathbf{G}^i \cdot \mathbf{G}_j} \right\}, \quad 1 \leq j \leq m^i, j \neq j^*. \quad (3.15)$$

where  $\mathbf{G}^i \cdot \mathbf{G}_j$  is negative. For negative  $\mathbf{G}^i \cdot \mathbf{G}_j$ , Eq. 3.15 yields a positive  $\gamma^i$ . When



$\mathbf{G}^i \cdot \mathbf{G}_j$  is positive, Eq. 3.11 shows that a positive  $\gamma^i$  is sufficient for the increase of the quality metric.

Note that the minimum  $\gamma^i$  can be close to zero in some extreme cases. In such a case, the magnitude of  $\gamma^i \mathbf{G}^i$  is compared to the sub-mesh length scale. If the magnitude of the displacement given by the  $\gamma^i \mathbf{G}^i$  is smaller than a predefined threshold, the next smallest  $\gamma^i$  can be used instead [12].

#### 4. Edge swapping and node insertion

As shown in the previous section, there are limitations to using mesh smoothing for star sub-meshes. Specifically, when an empty kernel case occurs, mesh smoothing alone cannot guarantee a valid triangulation. Edge swapping and node insertion are thus introduced to overcome the limitation of the mesh smoothing and to further improve mesh quality.

Edge swapping is used to improve the quality of the unstructured mesh. Unlike the mesh smoothing methods, edge swapping alters the connectivity. The present study uses a slightly modified version of Lawson’s [46] edge swapping method. The Lawson’s edge swapping method compares the minimum angles of two adjacent triangle cells before and after the common edge is swapped. The edge is swapped only if the new minimum angle is greater than the old minimum angle. Otherwise, the current edge connectivity is kept.

The 2-D edge swapping technique is extended to the mapped mesh considered in this study in the following manner. The mapped mesh has a single set of connectivity information shared by all the layers. Specifically, given an edge from the connectivity table, the corresponding edge can be found on every layer. Thus, two neighboring triangle cells that share the edge can also be formed on every layer. These two neighboring triangle cells form a “quad-tube” when they are stacked in the spanwise

direction. The decision to swap the edge is then based on whether swapping improves the minimum measure among the quad-cells in the “quad-tube”.

Herein Lawson’s method is modified by replacing the maximization of the minimum angle criteria with the quality measure  $\tau$ . By using the quality measure  $\tau$ , edge swapping becomes consistent with mesh smoothing in the sense that both procedures maximize the minimum quality measure. Note that the quality measure,  $\tau$ , includes information on the mesh angle values. The criteria based on the quality measure  $\tau$  are more general than the criteria based on the maximization of the minimum angle, since the former also include information on the cell area and edge length.

Mesh smoothing improves the mesh greatly, but the improvement is limited by the connectivity restriction between layers. Edge swapping, however, modifies the connectivity of the smoothed meshes. A swapped edge alters the sub-meshes associated with all four nodes of the “quad-cell”, and further mesh smoothing may become necessary. Because mesh smoothing and edge swapping are coupled, they can be applied alternatively until (1) edge swapping is no longer necessary and (2) the node movement and/or the variations in the quality measure are smaller than some given limits.

When a source mesh is generated using a certain airfoil cross section, the connectivity of the resultant unstructured mesh is optimal with respect to the criteria of the mesh generator. For example, if Delaunay triangulation is used, the connectivity corresponds to the best max-min angle possible for the given distribution of nodes. When nodes are forced to move to new locations, as occurs with the mapping and smoothing procedures, the connectivity is no longer optimal. Edge swapping reduces the dependency of the overall mesh quality on the choice of the source mesh. As a result, the influence of the spanwise location of the source layer on the final global mesh quality is diminished.

To enhance mesh quality, node insertion is also employed. Recall that for the empty kernel case, introducing a new node or Steiner point can remove the inverted cells. Furthermore, node insertion at a later stage of the mesh generation process makes it possible to start from a coarse source mesh. This reduces the work load for mesh smoothing and edge swapping. Once the mesh is “converged”, additional nodes can be inserted where the resolution of the mesh does not meet the predefined length scale.

The number of newly added nodes is controlled by calculating a quality measure or “score” for each cell and then sorting the cells based on these scores. Then a predetermined number of nodes is placed on the worst cells first. As is the case with edge swapping, node insertion also alters the connectivity of the mapped mesh. The sorting procedure examines the “score” for each triangle cell in a “triangle-tube” that spans from hub to tip.

While the circumcircle center of a candidate triangle cell is often the preferred site for a new node [47], herein, for simplicity, the centroid of a triangle cell is used for a new node location. Only a fraction of the total number of new nodes are added at each step. The whole procedure repeats until all the intended number of nodes are added, and the mesh smoothing and edge swapping are completed.

## B. Flow solver

The numerical methods for solving the Navier-Stokes equations are discussed in this section. The governing equations are discretized using the Finite Volume Method (FVM) which takes advantage of an integral formulation of the conservation equations.

The solver is developed for unstructured meshes of mixed elements in a cell-vertex

approach. In this method, the conservative variables are assumed to be stored at the vertices of the mesh. The state variables stored at a vertex represent the control volume averaged values about the vertex, where the control volume about the vertex is defined by the median dual cell.

The flux integral on the surface of a cell, which can be any arbitrary polyhedral, is approximated by the sum of numerical quadratures on the faces of the cell. The inviscid flux is resolved using the Godunov method [48]. The gradient of state variables is computed using the least-squares method [25]. The computed gradients are used for the reconstruction and the viscous flux computation. Higher-order spatial accuracy is achieved using a piece-wise linear reconstruction [25].

The piece-wise linear reconstruction can cause solution oscillations near strong discontinuities. The spurious solutions can be avoided using limiter functions. An original multi-dimensional limiter function for unstructured meshes is credited to Barth and Jespersen [25]. The Barth limiter, however, is sensitive to small variations of the solution in the smooth solution region [26]. The Barth limiter is highly dissipative [49]. Venkatakrishnan's limiter is employed herein to avoid the activation of limiting due to the noise in the smooth solution region and to damp the oscillations at strong discontinuities. [26].

Once the flux integral is numerically solved, the discrete form of the governing equations are integrated in time as a system of ordinary differential equations. In this work, the time integration of the resultant ODE's is performed by an explicit scheme with optimal coefficients [50]. This time integration uses a multi-stage method that is slightly different from Runge-Kutta method. This integration scheme is designed for storage efficiency. [51] The explicit method requires smaller time steps compared to the implicit method. The time step limitation is known as the Courant-Friedrichs-Lewy (CFL) condition [52]. Despite this limitation, the method is well suited for

unsteady simulations for which time accuracy is of importance. In addition, the explicit method is easier to implement than the implicit method. For steady solutions, the ODE's are integrated as an unsteady solution until the rate of change of the conservation variables becomes negligible. To enhance the convergence rate, local time stepping [51] and implicit residual smoothing [53] are employed.

The numerical methods mentioned above will be presented in detail in the following discussion. This discussion will be divided into three parts: the spatial discretization, the temporal discretization and the boundary conditions. The flux evaluation, gradient computation, and piece-wise reconstruction will be presented in the spatial discretization section. This will be followed by the temporal discretization method, which includes time integration and implicit residual smoothing.

## 1. Spatial discretization

### a. Introduction

This section reviews the issues related to the flow solver implementation for an unstructured mesh. To perform the numerical computation of the Navier-Stokes equations, the discretized form of the equations is solved for a finite set of the conservative state variables,  $\mathbf{u}_i = [\rho, \rho u, \rho v, \rho w, \rho E]_i^T$ . In the FVM, adopted in this work, a mesh is constructed over the domain of interest which results in non-overlapping polyhedrons defining the control volumes. The governing equations are then solved simultaneously for each control volume. The mesh is constructed by tessellating the computational sites or nodes in such a way that the actual geometry is well approximated and the density of the nodes increases in the regions of high solution gradients.

Mesh local clustering is often necessary to resolve the geometry of complex boundaries and is used to achieve local mesh adaptation in regions of high solu-

tion gradients. Localized mesh enrichment is easier for an unstructured mesh than a structured mesh due to its flexible data structure. A disadvantage of using an unstructured mesh is that it requires explicit connectivity information. Structured meshes do not need connectivity information since they utilize the inherent connectivity information in their data structure. While the inherent connectivity definition simplifies structured mesh solvers and reduces their storage requirement, it is this strongly coupled relationship between the data structure and the mesh that makes a structured mesh difficult to use for complex geometry. A traditional structured mesh does not permit local addition or removal of mesh elements without disrupting its data structure. A structured mesh, however, can always be represented as an unstructured mesh. Therefore, despite the added cost associated with connectivity information, the flow solver herein uses an unstructured mesh for its generality.

The unknown state variables can be defined either at cell center or at nodes. Depending on where the average state variables are stored, a FVM implementation can be classified as a cell-centered or a cell-vertex approach. The present work employs the cell-vertex approach, in which the volume averaged state variables are stored at the vertices of meshes [24].

The computational cost in the FVM can be roughly approximated by the total number of unknowns. For a typical volume mesh composed of tetrahedral cells, the ratio of the number of cells to the number of nodes ranges approximately from 5 to 6 [18]. This implies that the overall cost for the cell-vertex method is smaller than the cost for the cell-centered method.

Unlike a cell-centered method, the control volumes about vertices in a cell-vertex method are not naturally defined by the mesh. A cell-vertex method typically employs dual cells to define the control volumes about nodes. Figure 7 shows a two-dimensional example of dual cells: median and centroid. The first-order stencil of a node is the

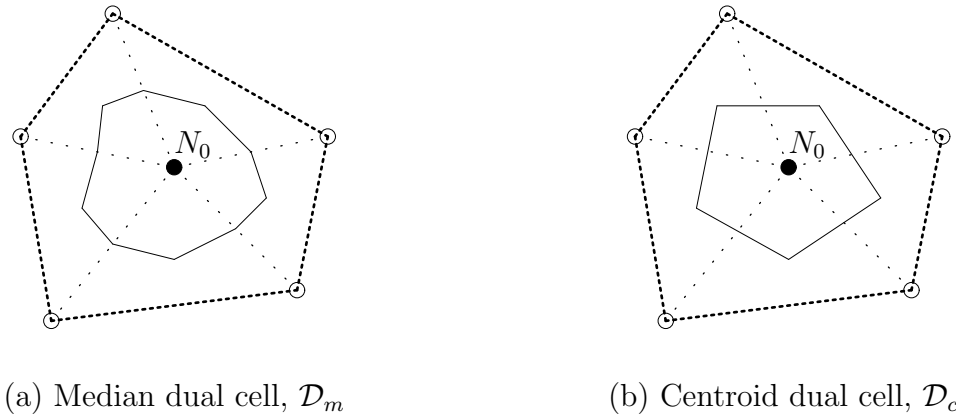


Fig. 7. Two-dimensional example of median and centroid dual cell for a node  $N_0$ . Solid lines denote the boundary of dual cells and thick dotted lines denote the boundary of the first-order stencil  $\mathcal{S}_{N_0}^1$  of node  $N_0$ .

collection of cells which share the common node. In Fig. 7, the first-order stencils for node  $N_0$  are denoted by thick dotted lines and the dual cells are defined by solid lines. Fig. 7 also shows that the areas of the dual cells are the sum of the fractional areas of the triangular cells in the first-order neighbor.

The boundaries of the median and centroid dual cells run from the centroid of a cell to the centroid of the adjacent cell. For the median dual cell, the boundary edges pass through the mid point of the edge shared by two neighboring cells resulting in two line segments per edge. For the centroid dual cell, the boundary edges run directly between centroids.

For a three dimensional case, a median-dual cell is formed by the surfaces crossing the mid-edges and the centroid of cells. Examples of median-dual cells are shown in Fig. 8. The volume contributions from a hexahedral cell and prism cell to a vertex  $i$  are shown in Fig. 8(a)-(b), respectively. A dual-cell can be formed with a set of neighbor cells of arbitrary polyhedrons. Because of this generality of the cell definition, the method is well suited for the mixed element meshes [54].

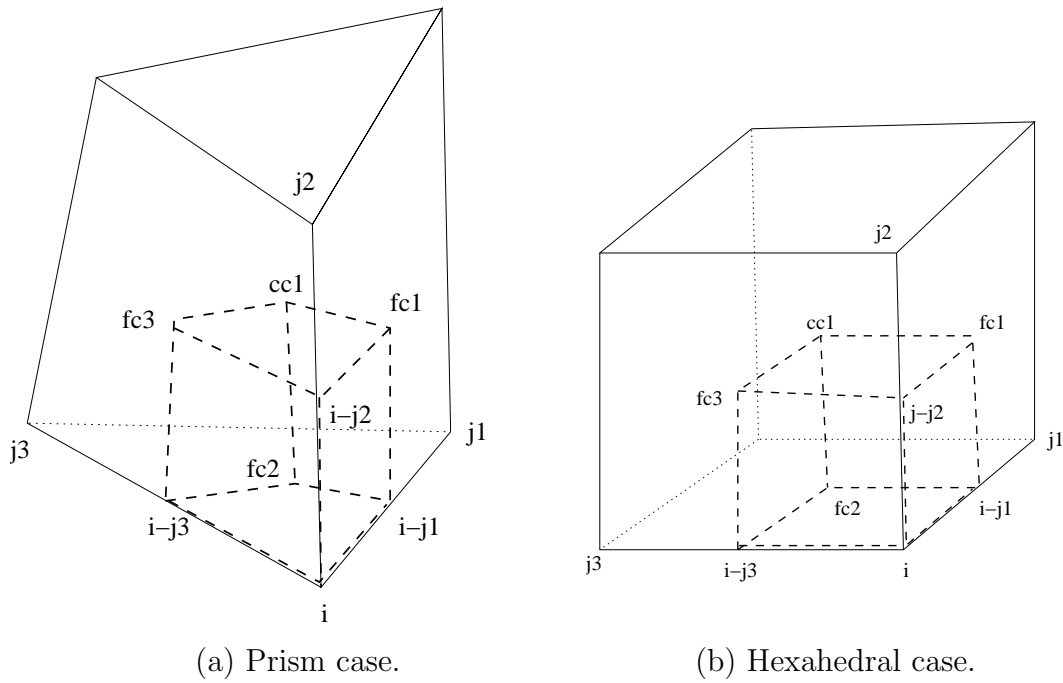


Fig. 8. Median-dual cells. The volume contributions to a median-dual cell for a vertex  $i$  are shown. Sub-cells are defined by the surface crossing the cell centroid  $cc$ , face centroid  $fc$  and mid-edge points  $i-j$ .

The area of the cell interface and the volume of a dual cell are computed in the following manner. For simplicity, a two-dimensional case is considered first. The dual cells shown in Fig. 7 define the control volume of node  $N_0$  in the cell-vertex method. For a centroid dual cell, there is an one-to-one relationship between the boundary segments of the dual cell and the edges. The one-to-one relationship permits association of the length of the dual cell boundary segment and the outward normal vector to an edge.

For a median dual cell, each edge is crossed by two-segments of the dual cell boundary. For simplicity, the two normal vectors can be averaged into a single normal vector. Similarly, the length of the two-segments can be summed up such that the two-segments are approximated by a single segment. The area of the dual cell is the sum of the quadrilateral contributions from the node sharing cells.

In the 3-D case, an area vector can be associated with an edge. The area vector



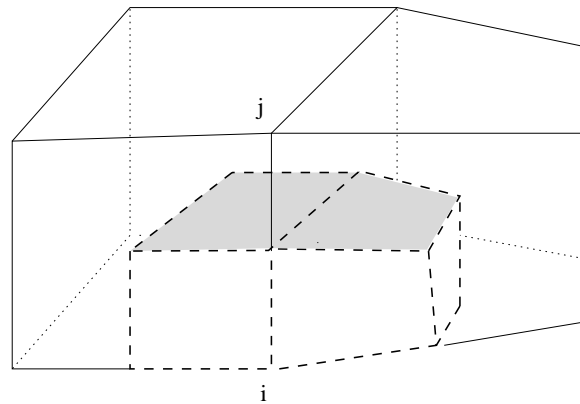


Fig. 9. Face area contributions (shaded area) from two hexahedral cells to an edge  $e_{ij}$ .

associated with an edge is computed by accumulating the area contribution from the cells which share the edge. Figure 9 shows two shaded quadrilateral faces. These two areas are the contributions from two adjacent hexahedral cells to edge,  $e_{ij}$ . The area contribution from a cell is always a quadrilateral regardless of the type of donor cell. Since the vertices of the quadrilateral are not always coplanar, a quadrilateral face is always decomposed into two triangles faces to compute the total area of the quadrilateral face.

The volume of a cell can be computed in a similar way. The volume contribution from a node-sharing cell is always a hexahedral. A hexahedral cell can be subdivided into 6 tetrahedral sub-cells. A segment of pseudo-code for area and volume computation is given below. The procedure assumes the following connectivity data are available: cell to face, face to edge and edge to nodes.

```

area = 0 /* area vector */
volume =0 /*dual cell volume */
do cell c
  do face f(c)
    do edge e(f(c))

```

```

    v1 = vector(me,cc)          /* Mid-Edge -> Cell Centroid */
    v2 = vector(me,fc)          /* Mid-Edge -> Face Centroid */
    atri = cross_product(v1,v2)
    area(e) = area(e) + atri    /* atri=triangle contribution */
10  vol(n1(e)) = atri*|e|/2    /* e=(n1,n2) */
    vol(n2(e)) = atri*|e|/2    /* |e| = edge length */
    od
  od
od
normal(e) = area(e)/|area(e)| /* normal vector */

```

The area contribution `atri` is a triangle defined by a cell centroid, face centroid, and the midpoint of an edge. A face of a median dual cell consists of an arbitrary number of faces which typically do not have a common normal vector. The normal vector `normal` on a face is approximated by the sum of the area-weighted vectors.

An operation similar to the area vector accumulation can be performed with an edge loop only, eliminating the need for the cell-face loop. Collectively, this type of operation is called an edge-based method, which is a popular technique to take advantage of the data structure of the unstructured mesh [25, 1, 55]. The present work uses the edge-based method.

The area vector is defined in the direction of the edge vector such that the inner product of the area vector and the edge vector is positive. Since an edge is defined in terms of two vertices, the direction of the edge vector is defined positively from the first node of the edge to the second node. This convention simplifies the implementations of edge-based procedures which allows a 3-D procedure to be applied in an 1-D manner. By temporarily rotating an edge in the area normal direction, one can associate the first and the second vertices of an edge as the left and right vertices. This technique,

called “grid-aligned method” [56], is used in the inviscid flux computation that will be discussed later in this chapter.

b. Integral formulation

The Navier-Stokes equations are discretized using the FVM. The main difference between Finite Difference Method (FDM) and FVM is the form of the equations each method is based on: FDM is developed from the differential form of the conservation equations whereas FVM is based on the integral form. For a generic scalar conservation equation, the differential form is

$$\frac{Du}{Dt} + F(u)_{i,i} = 0 \quad (3.16)$$

where  $(\cdot)_{,i} \equiv \partial(\cdot)/\partial x_i$ , and the integral form is

$$\frac{D}{Dt} \int_{\Omega} u d\Omega + \int_{\partial\Omega} F(u)_i n_i d\Omega = 0. \quad (3.17)$$

The weak form of the generic scalar conservation equation is

$$\frac{D}{Dt} \int_{\Omega} \phi u d\Omega - \int_{\Omega} \phi_{,i} F(u)_i d\Omega = \int_{\Omega} \phi F(u)_i n_i d\Omega. \quad (3.18)$$

Eq. (3.18) reduces to Eq. (3.17) if the test function is constant, *i.e.*,  $\phi_{,i} = 0$ . The integral form can be viewed as a weak formulation of the Finite Element Method with a constant test function.

c. The Navier-Stokes equations in the rotational frame of reference

The Navier-Stokes equations in the rotational frame of reference include the additional terms due to the rotation of the axes [57], pp. 16-18. The integral form of the equations

in terms of absolute variables [58, 36] are

$$\frac{\partial}{\partial t} \int_{\Omega} \mathbf{U} d\Omega + \oint_S \mathbf{F} \cdot \mathbf{n} dS = \int_{\Omega} \mathbf{G} d\Omega. \quad (3.19)$$

$\mathbf{U}$  is the state vector of the absolute conservative variables

$$\mathbf{U} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \end{pmatrix}. \quad (3.20)$$

The flux vector,  $\mathbf{F}$ , can be split into the convective part,  $\mathbf{F}_c$ , and the viscous part,  $\mathbf{F}_v$ ,

$$\mathbf{F} = \mathbf{F}_c + \mathbf{F}_v. \quad (3.21)$$

The convective part of the flux is

$$\mathbf{F}_c = \begin{pmatrix} \rho V \\ \rho u V + n_x p \\ \rho v V + n_y p \\ \rho w V + n_z p \\ \rho H V + V_g p \end{pmatrix} \quad (3.22)$$

where  $V$  is the contravariant velocity and  $V_g$  is the rotational velocity component in the normal direction,  $\mathbf{n}$ . Taking into account the velocity of the moving frame rotating at an angular velocity of  $\boldsymbol{\Omega}$ ,

$$V = (\mathbf{v} - \mathbf{v}_g) \cdot \mathbf{n} \quad (3.23)$$

where  $\mathbf{v}_g$  is the rotational velocity of the frame

$$\mathbf{v}_g = \mathbf{r} \times \boldsymbol{\Omega}. \quad (3.24)$$

The normal component of the frame velocity is

$$V_g = \mathbf{v}_g \cdot \mathbf{n} = (\mathbf{r} \times \boldsymbol{\Omega}) \cdot \mathbf{n}. \quad (3.25)$$

The source term due to the rotation in Eq. (3.19) is given by

$$\mathbf{G} = \begin{pmatrix} 0 \\ (\rho u \times \boldsymbol{\Omega}) \cdot \mathbf{e}_x \\ (\rho v \times \boldsymbol{\Omega}) \cdot \mathbf{e}_y \\ (\rho w \times \boldsymbol{\Omega}) \cdot \mathbf{e}_z \\ 0 \end{pmatrix} \quad (3.26)$$

where  $\mathbf{e}_x$ ,  $\mathbf{e}_y$  and  $\mathbf{e}_z$  are the unit vectors in Cartesian coordinates. Assuming the rotation about  $x$ -axis only, the rotational velocity becomes

$$\boldsymbol{\Omega} = \begin{pmatrix} \omega_x \\ 0 \\ 0 \end{pmatrix}, \quad (3.27)$$

which simplifies Eq. (3.26) to

$$\mathbf{G} = \begin{pmatrix} 0 \\ 0 \\ \rho \omega_x w \\ -\rho \omega_x v \\ 0 \end{pmatrix}. \quad (3.28)$$

The viscous flux is invariant in the rotational frame

$$\mathbf{F}_v = \begin{pmatrix} 0 \\ n_x\tau_{xx} + n_y\tau_{xy} + n_z\tau_{xz} \\ n_x\tau_{yx} + n_y\tau_{yy} + n_z\tau_{yz} \\ n_x\tau_{zx} + n_y\tau_{zy} + n_z\tau_{zz} \\ n_x\Theta_x + n_y\Theta_y + n_z\Theta_z \end{pmatrix} \quad (3.29)$$

where the viscous stress work and the heat conduction term,  $\Theta$  are

$$\begin{aligned} \Theta_x &= u\tau_{xx} + v\tau_{xy} + w\tau_{xz} + \lambda\frac{\partial T}{\partial x} \\ \Theta_y &= u\tau_{yx} + v\tau_{yy} + w\tau_{yz} + \lambda\frac{\partial T}{\partial y} \\ \Theta_z &= u\tau_{zx} + v\tau_{zy} + w\tau_{zz} + \lambda\frac{\partial T}{\partial z}. \end{aligned} \quad (3.30)$$

The thermal conductivity coefficient,  $\lambda$  in Eq. (3.30) is

$$\lambda = c_p \frac{\mu}{Pr} \quad (3.31)$$

and  $Pr$  is Prandtl number and  $Pr = 0.72$  for air. The components of the viscous stresses are

$$\begin{aligned} \tau_{xx} &= \frac{2}{3}\mu\left(2\frac{\partial u}{\partial x} - \frac{\partial v}{\partial y} - \frac{\partial w}{\partial z}\right) \\ \tau_{yy} &= \frac{2}{3}\mu\left(2\frac{\partial v}{\partial y} - \frac{\partial u}{\partial x} - \frac{\partial w}{\partial z}\right) \\ \tau_{zz} &= \frac{2}{3}\mu\left(2\frac{\partial w}{\partial z} - \frac{\partial u}{\partial x} - \frac{\partial v}{\partial y}\right) \\ \tau_{xy} &= \tau_{yx} = \mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) \\ \tau_{yz} &= \tau_{zy} = \mu\left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right) \\ \tau_{zx} &= \tau_{xz} = \mu\left(\frac{\partial w}{\partial x} + \frac{\partial u}{\partial z}\right). \end{aligned} \quad (3.32)$$

#### d. Semi-discrete form of the Navier-Stokes equations

A semi-discrete form of the Navier-Stokes equations is presented. The term *semi-discrete* herein implies that only the part of the Navier-Stokes equations in a con-

tinuous form is approximated by the numerical approximations. Specifically, the conservative variables are represented as an average value at a point and the spatial operators, such as the integrations in space, are approximated by numerical quadratures.

An inspection of Eq. (3.19) shows that the temporal change of conserved variables  $\mathbf{U}$  integrated over a volume  $\Omega$  is due to the surface integral of flux  $\mathbf{F}$  and the volume integral of the source  $\mathbf{G}$ . Once the surface and volume integrations are numerically approximated, Eq. (3.19), which is a system of partial differential equations (PDE), can be transformed to a system of ordinary differential equations (ODE) in time [23, 59]. The advantage of the approach is that the spatial and the temporal approximations of Eq. (3.19) can be made independently.

Given a polyhedral control volume,  $\mathbf{c}_i$ , the cell average of the conservative variables is defined as a volume-averaged variable

$$\mathbf{u}_i \equiv \frac{\int_{\Omega_i} \mathbf{U} d\Omega}{\Omega_i} \quad (3.33)$$

where  $\Omega_i$  is the volume of cell  $\mathbf{c}_i$ . It is assumed that the averaged variables,  $\mathbf{u}_i$ , are stored at the cell center, which is the vertex of the mesh in the cell-vertex method. The volume average of the source term is approximated by

$$\mathbf{g}_i \approx \frac{\int_{\Omega_i} \mathbf{G} d\Omega}{\Omega_i} \quad (3.34)$$

where the source term is evaluated at the vertex.

Let  $\mathbf{n}_j$  be the outward normal vector, and let  $A_j$  be the area of a cell face. Then, the surface integration of the flux vector for control volume,  $\mathbf{c}_i$ , can be approximated by

$$\oint_{\partial\Omega_i} \mathbf{F} \cdot \mathbf{n} dS \approx \sum_j^{m_i} \mathbf{f}_j \cdot \mathbf{n}_j A_j \quad (3.35)$$

where  $m_i$  is the number of faces defining the control volume,  $c_i$ .  $\mathbf{f}_j$  is a numerical flux which approximates the physical flux  $\mathbf{F}$ . Note that the numerical flux must be evaluated on the cell faces.

As noted in the previous section, the surface area  $A_j$  of a face  $\mathbf{s}_j$  is the sum of area contributions from the  $n_j$  number of cells which share a common edge

$$A_j \equiv \sum_k^{n_j} \hat{A}_k \quad (3.36)$$

where  $\hat{A}_k$  is the quadrilateral area contribution from the edge-sharing cells. The normal vector for the face  $\mathbf{s}_j$  is based on the area-weighted vector sums

$$\mathbf{n}_j \equiv \frac{\sum_k^{n_j} \hat{\mathbf{n}}_k \hat{A}_k}{A_j} \quad (3.37)$$

where  $\hat{\mathbf{n}}_k$  is the normal vector on the contributed faces. Using Eq. (3.37), the sum of the flux quadratures on the contributed surfaces can be approximated by a single flux evaluation on the combined face,  $\mathbf{n}_j A_j$ ,

$$\sum_k^{n_j} \mathbf{f}_k \cdot \hat{\mathbf{n}}_k \hat{A}_k \approx \mathbf{f}_j \cdot \mathbf{n}_j A_j \quad (3.38)$$

where the error in the approximation is negligible for a first-order and second-order solution. When the solution is assumed to vary quadratically or higher, the summation of fluxes on sub-surfaces  $\hat{A}_k$  cannot be substituted by a single quadrature as in the right hand side of Eq. (3.38) [60, 61]. Once the surface and volume integrations are approximated, Eq. (3.19) becomes

$$\frac{\partial}{\partial t}(\mathbf{u}_i \Omega_i) + \sum_{j=1}^{m_i} \mathbf{f}_j \cdot \mathbf{n}_j A_j = \mathbf{g}_i \Omega_i. \quad (3.39)$$

So far, an approximation has not been made for the differentiation in time.

Mesh deformation in time is not considered in this work. The temporal change



of a cell volume is zero for a rigid mesh, which further simplifies Eq. (3.39) to

$$\frac{\partial}{\partial t}(\mathbf{u}_i) = -\frac{\sum_{j=1}^{m_i} \mathbf{f}_j \cdot \mathbf{n}_j A_j + \mathbf{g}_i \Omega_i}{\Omega_i}. \quad (3.40)$$

This simplification cannot be made for a deforming mesh. In such a case, a temporal change of cell volume and the movement of surface of control volume must be considered. Geometric Conservation Law formalizes the procedures a conservative method must satisfy when dealing with a deforming mesh [62, 49].

The numerical method used for convective flux computation is presented next.

#### e. Inviscid flux

As the Reynolds number increases, the viscous effects are confined to the regions close to the wall, and the role of the inviscid flux dominates the remainder of the flow field. Therefore, the accurate resolution of the inviscid flux is crucial in the numerical simulation of the Navier-Stokes equations. In this work, the Godunov method, which is an upwind method for hyperbolic equations, is employed to resolve the inviscid flux. Specifically, the Riemann problem in the Godunov method is solved using the Roe's approximate Riemann solver [19, 20].

The nonlinear nature of the Riemann problem makes the exact solution rather expensive. Since the solutions are required repeatedly, a cost effective approximated solver is typically employed. The use of the approximated solver is further justified by the inherent numerical error in the Godunov method due to the averaging at the projection stage.

Instead of solving the nonlinear problem, Roe's approximate Riemann solver solves a linearized version of the problem. The Roe's solver is one of the most widely used method because of its proven accuracy and low numerical dissipation. The latter property of Roe's method is known to play a important role in resolving of

the boundary layer, for which excessive numerical dissipation can ruin the solution accuracy.

Roe's Riemann solver is based on 1-D physics and it does not produce a true multi-dimensional solution. For the multi-dimension problem, the 1-D method is extended using the grid aligned method [56], pp. 573-579. In this approach, the multi-dimensional physics can be approximated with the 1-D method by rotating the local coordinates temporarily such that they align with the local mesh interface. The technique is widely used despite shortcomings such as the misinterpretation of a misaligned discontinuity.

The inviscid flux from Roe's approximate Riemann solver [19, 20] is defined in terms of the two states across a median-dual cell face,  $\mathbf{u}_L$  and  $\mathbf{u}_R$ ,

$$\mathbf{f}_c = \frac{1}{2} \left[ \mathbf{F}_c(\mathbf{u}_L) + \mathbf{F}_c(\mathbf{u}_R) - |\tilde{\mathbf{A}}| \Delta \mathbf{u}_{R,L} \right] \quad (3.41)$$

where  $|\tilde{\mathbf{A}}|$  is the flux Jacobian with respect to the conservative variables and  $\Delta(\cdot) = (\cdot)_R - (\cdot)_L$  is the difference between the right and left states. The tilde symbol in  $|\tilde{\mathbf{A}}|$  indicates that the Jacobian is evaluated using averaged state variables that are constant and therefore the solution is based on a linearized equation. Roe's density weighted averages defined as [19]

$$\begin{aligned} \tilde{\rho} &= \sqrt{\rho_R \rho_L} \\ \tilde{u} &= (u_L \sqrt{\rho_L} + u_R \sqrt{\rho_R}) / (\sqrt{\rho_L} + \sqrt{\rho_R}) \\ \tilde{v} &= (v_L \sqrt{\rho_L} + v_R \sqrt{\rho_R}) / (\sqrt{\rho_L} + \sqrt{\rho_R}) \\ \tilde{w} &= (w_L \sqrt{\rho_L} + w_R \sqrt{\rho_R}) / (\sqrt{\rho_L} + \sqrt{\rho_R}) \\ \tilde{H} &= (H_L \sqrt{\rho_L} + H_R \sqrt{\rho_R}) / (\sqrt{\rho_L} + \sqrt{\rho_R}) \\ \tilde{V} &= \tilde{u} n_x + \tilde{u} n_y + \tilde{u} n_z - V_g. \end{aligned} \quad (3.42)$$

It can be shown that the flux Jacobian with respect to primitive variables is sparse.

Repeating the matrix-vector multiplication for the dissipation term is therefore inefficient. The dissipation terms can be pre-multiplied by  $|\tilde{\mathbf{A}}|\Delta\mathbf{u}_{R,L}$  and expanded as a vector sum for the computational efficiency [23], pp. 106-107,

$$\begin{aligned}
|\tilde{\mathbf{A}}|\Delta\mathbf{u}_{R,L} = & |\tilde{V}| \left( \Delta\rho - \frac{\Delta\rho}{\tilde{c}^2} \right) \begin{bmatrix} 1 \\ \tilde{u} \\ \tilde{v} \\ \tilde{w} \\ (\tilde{u}^2 + \tilde{v}^2 + \tilde{w}^2)/2 \end{bmatrix} \\
& + |\tilde{V}|\tilde{\rho} \begin{bmatrix} 0 \\ \Delta u - \Delta\tilde{V}n_x \\ \Delta v - \Delta\tilde{V}n_y \\ \Delta w - \Delta\tilde{V}n_z \\ \tilde{u}\Delta u + \tilde{v}\Delta v + \tilde{w}\Delta w - \tilde{V}\Delta\tilde{V} \end{bmatrix} \\
& + |\tilde{V} - \tilde{c}| \left( \frac{\Delta p - \tilde{\rho}\tilde{c}\Delta\tilde{V}}{2\tilde{c}^2} \right) \begin{bmatrix} 1 \\ \tilde{u} - \tilde{c}n_x \\ \tilde{v} - \tilde{c}n_y \\ \tilde{w} - \tilde{c}n_z \\ \tilde{H} - \tilde{c}\tilde{V} \end{bmatrix} \\
& + |\tilde{V} + \tilde{c}| \left( \frac{\Delta p + \tilde{\rho}\tilde{c}\Delta\tilde{V}}{2\tilde{c}^2} \right) \begin{bmatrix} 1 \\ \tilde{u} + \tilde{c}n_x \\ \tilde{v} + \tilde{c}n_y \\ \tilde{w} + \tilde{c}n_z \\ \tilde{H} + \tilde{c}\tilde{V} \end{bmatrix}.
\end{aligned} \tag{3.43}$$

Roe's approximated Riemann solver admits an expansion shock because vanishing viscosity does not provide enough diffusion. The instability of the Roe Riemann

solver in particular cases is also well documented [63, 64, 65]. The present work uses Harten’s entropy fix [66]. The fix modifies the definition of the eigenvalue such that the eigenvalue never reaches 0. Whereas the numerical dissipation provided by Roe’s method is proportional to the magnitude of eigenvalue, the Harten’s fix adds non-vanishing dissipation to remove the expansion shock and the instability problem at the expense being more dissipative.

#### f. Gradient computation

This section discusses the computation of the gradient operator on an unstructured mesh. The viscous flux is defined in terms of the gradients of the velocity components and the temperature. The diffusion terms in the  $k - \omega$  turbulence model also require the gradient of the  $k$  and  $\omega$  field. For these reasons, it is important to employ an accurate method for gradient computation.

The finite difference method is the one of the most widely used techniques to compute the gradient. The method relies on the spatially ordered placement of data points such as a 5-point stencil (north, east, west, south, and center) in a 2-D case. The stencils for an unstructured mesh are not ordered as the stencils of a structured mesh. If an extra neighbor is added in the 5-point stencil example mentioned above, the stencil pattern falls outside of the regular pattern used in the finite difference scheme. One attempt to compute the gradient on an unstructured mesh using finite difference techniques is found in Campbell et al. [67]. The suggested discrete operators are limited to two dimensional unstructured meshes and they are based on trigonometric functions, which appear to be too complex for general applications. It is therefore easy to conclude that a different strategy should be taken to compute the gradient on an unstructured mesh.

The gradient computation method should be efficient both in speed and storage

and the method should be accurate. The Green-Gauss method and the least-squares method are the two most widely used techniques to compute the gradient on unstructured meshes because they meet the properties mentioned above. Both the Green-Gauss and the least-squares methods are point-wise methods and can be programmed using edge-based operations.

The Green-Gauss method estimates the gradient of a generic scalar variable  $\Phi$  by a discrete form of the divergence theorem which states

$$\int_{\Omega} \nabla \Phi d\Omega = \oint_{\partial\Omega} \Phi \mathbf{n} d\Omega. \quad (3.44)$$

The gradient of  $\nabla \Phi$  at node  $\mathbf{n}_i$  can be approximated as

$$\nabla \Phi_i = \frac{1}{\Omega_i} \sum_{j \in \mathcal{S}_{\mathbf{n}_i}^1} \mathbf{h}(\Phi_i, \Phi_j, \mathbf{n}_{ij}, A_{ij}), \quad (3.45)$$

where  $\Omega_i$  is the volume of a dual cell associated with node  $\mathbf{n}_i$  and  $\mathcal{S}_{\mathbf{n}_i}^1$  is the first-order stencil with respect to the node  $\mathbf{n}_i$ . Function  $\mathbf{h}$  in Eq. (3.45) is defined as

$$\mathbf{h} = \begin{cases} \frac{1}{2}(\Phi_i + \Phi_j)\mathbf{n}_{ij} & \text{if } \mathbf{e}(\mathbf{n}_i, \mathbf{n}_j) \text{ is an internal edge,} \\ \frac{1}{2}(\Phi_i + \Phi_j)\mathbf{n}'_{ij} + \frac{1}{2}(\frac{5}{6}\Phi_i + \frac{1}{6}\Phi_j)\mathbf{n}_{ij}^B & \text{if } \mathbf{e}(\mathbf{n}_i, \mathbf{n}_j) \text{ is a boundary edge.} \end{cases} \quad (3.46)$$

$\mathbf{n}'_{ij}$  and  $\mathbf{n}_{ij}^B$  denote the normal vectors at the quadrature points on the dual cell surfaces which are associated with a boundary edge  $\mathbf{e}(\mathbf{n}_0, \mathbf{n}_i)$ , as shown in Fig. 10. In this figure, circles denote the quadrature points  $\mathbf{x}_{q_{ij}}$  for  $\mathbf{n}_i$ . Note that edges  $\mathbf{e}(\mathbf{n}_4, \mathbf{n}_i)$  and  $\mathbf{e}(\mathbf{n}_i, \mathbf{n}_1)$  are boundary edges. Internal edges,  $\mathbf{e}(\mathbf{n}_i, \mathbf{n}_2)$  and  $\mathbf{e}(\mathbf{n}_i, \mathbf{n}_3)$ , are associated with single quadrature points  $\mathbf{x}_{q_{i2}}$  and  $\mathbf{x}_{q_{i3}}$ , respectively. Each boundary edge  $\mathbf{e}(\mathbf{n}_i, \mathbf{n}_j)$  is associated with two quadrature points,  $\mathbf{x}_{q_{ij}}^B$  and  $\mathbf{x}'_{q_{ij}}$  for  $j = 1, 4$ . Note that Eq. (3.45) is exact for linearly varying  $\Phi$  within a dual cell.

The Green-Gauss method has a slight advantage over the least-squares method

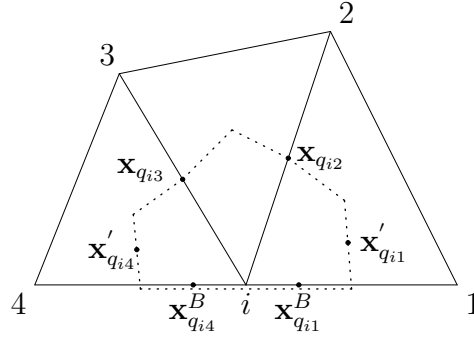


Fig. 10. First-order stencil for 2-D Green-Gauss gradient computation.

in terms of speed and storage requirements. Both methods compute the gradient accurately when  $\Phi$  varies linearly [24]. The Green-Gauss method, however, fails to produce accurate results at the nodes on the interface between different types of elements [22, 23]. For a first-order stencil that consists of a mixed type of cells, the surface integral can be taken about the boundary of the first-order stencil rather than the boundary of a dual cell. This modification of the integral path increases the truncation error since the volume of a first-order stencil is larger than the the volume of a dual cell. Furthermore, extra storage of connectivity data is required to describe the wider integral path. It is possible to devise a method that limits the use of a wide integral path only when it is necessary, and maintains the smaller stencil otherwise. This spruce up of the method, however, increases the complexity of the solution algorithm. For this reason, the present work uses the Green-Gauss method only if a mesh contains tetrahedral cells only. Otherwise, the least-squares method is used. Next, the least-squares method is presented.

A generic scalar variable,  $\Phi$ , at  $\mathbf{x}$  can be expressed in a Taylor series as a function of  $\Phi_i$  at  $\mathbf{x}_i$  as

$$\Phi = \Phi_i + (\mathbf{x} - \mathbf{x}_i) \cdot \nabla \Phi_i + H.O.T. \quad (3.47)$$

where  $H.O.T.$  is the truncation error. Suppose the node  $\mathbf{n}_j$  positioned at  $\mathbf{x}_j$  is one

of the nodes in the first order stencil  $\mathcal{S}_{\mathbf{n}_i}^1$  of  $\mathbf{n}_i$  and that the variation of  $\Phi$  is at most linear within the first order stencil. Then, Eq. (3.47) can be rearranged as

$$(\mathbf{x}_j - \mathbf{x}_i) \cdot \nabla \Phi_i = \Phi_j - \Phi_i, \quad \mathbf{x}_j \in \mathcal{S}_{\mathbf{n}_i}^1 \quad (3.48)$$

where the higher order term in Eq. (3.47) vanishes for a linear  $\Phi$ . Suppose there are  $n_i$  first-order neighboring nodes,  $\mathbf{n}_j$ , of  $\mathbf{n}_i$  in  $\mathcal{S}_{\mathbf{n}_i}^1$ . Since Eq. (3.48) holds for every node  $\mathbf{n}_j$ , Eq. (3.48) can be written as a system of  $n_i$  number of equations,

$$\begin{bmatrix} w_1 \Delta x_{1-i} & w_1 \Delta y_{1-i} & w_1 \Delta z_{1-i} \\ \vdots & \vdots & \vdots \\ w_{n_i} \Delta x_{n_i-i} & w_{n_i} \Delta y_{n_i-i} & w_{n_i} \Delta z_{n_i-i} \end{bmatrix} \nabla \Phi_i = \begin{bmatrix} w_1 (\phi_1 - \phi_i) \\ \vdots \\ w_{n_i} (\phi_{n_i} - \phi_i) \end{bmatrix} \quad (3.49)$$

where  $w_j$  is an arbitrary weighting and  $\Delta x_{j-i}$ ,  $\Delta y_{j-i}$ , and  $\Delta z_{j-i}$  are the x, y, and, z-component of  $\mathbf{x}_j - \mathbf{x}_i$ , respectively. The number of neighboring nodes, which is the same as the number of incident edges to  $\mathbf{n}_i$ , is most likely greater than 3 in a 3-D mesh. The number of neighboring nodes is always greater than or equal to 3 because a volume element requires at least 4 vertices. Therefore, the number of equations are generally greater than the number of unknowns. When this occurs, Eq. (3.49) becomes an over-determined system of equations and it is not possible to find an unique solution that is exact for all equations simultaneously.

In symbolic form, Eq. (3.49) is

$$\mathbf{L} \nabla \Phi = \mathbf{f} \quad (3.50)$$

where

$$\mathbf{L} = \begin{bmatrix} \mathbf{L}_1 & \mathbf{L}_2 & \mathbf{L}_3 \end{bmatrix} \quad (3.51)$$

$$\mathbf{f} = \begin{bmatrix} w_1(\phi_1 - \phi_0) \\ \vdots \\ \vdots \\ w_n(\phi_n - \phi_0) \end{bmatrix}. \quad (3.52)$$

The solution that minimizes

$$\|\mathbf{L}\nabla\Phi - \mathbf{f}\|_2 \quad (3.53)$$

can be computed as the solution of a least-squares problem. For Eq. (3.50), the least-squares problem is defined by pre-multiplying Eq. (3.50) by  $\mathbf{L}^T$ , which is the transpose of  $\mathbf{L}$ , to Eq. (3.50)

$$\mathbf{L}^T\mathbf{L}\nabla\Phi = \mathbf{L}^T\mathbf{f}. \quad (3.54)$$

The gradient  $\nabla\Phi$  is then

$$\nabla\Phi = \mathbf{P}\mathbf{f} \quad (3.55)$$

where

$$\mathbf{P} = (\mathbf{L}^T\mathbf{L})^{-1}\mathbf{L}^T \quad (3.56)$$

such that

$$\mathbf{P}\mathbf{L} = (\mathbf{L}^T\mathbf{L})^{-1}\mathbf{L}^T\mathbf{L} = \mathbf{I}. \quad (3.57)$$

The matrix  $\mathbf{P}$  is

$$\mathbf{P} = \frac{1}{a} \begin{bmatrix} \ell_{22}\ell_{33} - \ell_{23}\ell_{23} & -(\ell_{12}\ell_{33} - \ell_{13}\ell_{23}) & \ell_{12}\ell_{23} - \ell_{13}\ell_{22} \\ -(\ell_{12}\ell_{33} - \ell_{13}\ell_{23}) & \ell_{11}\ell_{33} - \ell_{13}\ell_{13} & -(\ell_{11}\ell_{23} - \ell_{13}\ell_{12}) \\ \ell_{12}\ell_{23} - \ell_{13}\ell_{22} & -(\ell_{11}\ell_{23} - \ell_{13}\ell_{12}) & \ell_{11}\ell_{22} - \ell_{12}\ell_{12} \end{bmatrix} \begin{bmatrix} \mathbf{L}_1^T \\ \mathbf{L}_2^T \\ \mathbf{L}_3^T \end{bmatrix} \quad (3.58)$$



where  $a$  is the determinant of  $\mathbf{L}^T \mathbf{L}$

$$a = \ell_{11}\ell_{22}\ell_{33} + 2\ell_{12}\ell_{23}\ell_{13} - \ell_{11}\ell_{23}^2 - \ell_{22}\ell_{13}^2 - \ell_{33}\ell_{12}^2 \quad (3.59)$$

and  $\ell_{ij} = \mathbf{L}_i \cdot \mathbf{L}_j$ .

The weighting coefficients  $w_j$  in Eq. (3.49) are chosen. They can be functions of the geometry and/or solution [24]. Using the distance between vertices, the weights can be defined as

$$w_j = \|\mathbf{x}_j - \mathbf{x}_i\|^{-t} \quad (3.60)$$

for values of  $t = 0, 1, 2$ . Herein, the weighting coefficients are only functions of the geometry, and  $t = 1$  is used in the present study.

If one uses geometrical weights as in Eq. (3.60),  $\mathbf{L}$  is a function of the distances between nodes and it is not dependent on the data  $\Phi$ . For a rigid mesh, not only is  $\mathbf{L}^T$  constant, but so is  $\mathbf{P}$ . The gradient can be computed efficiently if  $\mathbf{P}$  is pre-computed.

The gradient obtained using Eq. (3.55) can suffer from numerical errors for an extremely stretched mesh. Because of the high aspect ratio, the row vectors of the orthogonalization can fail to span in all three directions. In a case of failure, more accurate methods, such as QR decomposition or singular value decomposition, should be used [68, 69].

During the code development, the gradient methods were tested independently from the main solution algorithm. The tests were performed in the following manner. A simple scalar field, whose gradient was analytical, was first prescribed on a mesh. The Green-Gauss method and the least-squares method were subjected to the prescribed field to estimate the gradient. The numerically computed and analytically determined values at the vertices were then compared. The errors of the both methods were within machine-zero for the field defined by a first order polynomial. The

least-squares method in double precision accuracy produced the gradient result which was within the round-off error range even for a mesh with cell aspect ratio as high as 50000.

g. Piecewise linear reconstruction

Piecewise linear reconstruction is used to achieve a higher-order spatial accuracy. For linear accuracy, constant states are assumed within the cells and thus the left and right states across the element face are the constant states from both sides. In the piecewise linear reconstruction method the state variables at a cell are assumed to vary linearly within a cell.

While the reconstruction yields the interface states  $\Phi$  at higher-order spatial accuracy, it can create an oscillatory solution at extrema. Limiters are used to avoid oscillations [70, 24]. In a geometric sense, a limiter function acts as a slope controller of the numerical gradients,

$$\Phi(\mathbf{x}) = \Phi_0(\mathbf{x}_i) + \Psi_{n_i} \nabla \Phi_{n_i} \cdot (\mathbf{x} - \mathbf{x}_i) , \quad (3.61)$$

where the limiter  $\Psi_{n_i} \in [0, 1]$  and  $\mathbf{x} \in \mathcal{S}_{n_i}^1$ . To avoid creating a new extrema, the reconstructed  $\Phi(\mathbf{x})$  should be bounded by the local extrema

$$\Phi^{\min} \leq \Phi(\mathbf{x}) \leq \Phi^{\max} , \quad (3.62)$$

where

$$\Phi^{\min} = \min_{j \in \mathcal{S}_{n_i}^1} [\Phi_i, \Phi_j] , \quad (3.63)$$

$$\Phi^{\max} = \max_{j \in \mathcal{S}_{n_i}^1} [\Phi_i, \Phi_j] . \quad (3.64)$$

The monotonicity of  $\Phi$  can be satisfied by enforcing Eq. (3.62) at the quadrature points  $\mathbf{x}_{q_{ij}}$  rather than at all  $\mathbf{x} \in \mathcal{S}_{n_i}^1$  [25]. The limiter function  $\Psi_{n_i}$  proposed by

Barth [25] is given by

$$\Psi_{\mathbf{n}_i} = \min_j \Psi_{\mathbf{n}_i, \mathbf{n}_j} ,$$

where the intermediate result  $\Psi_{\mathbf{n}_i, \mathbf{n}_j}$  is obtained at the quadrature point associated with  $\mathbf{e}(\mathbf{n}_i, \mathbf{n}_j)$ ,

$$\Psi_{\mathbf{n}_i, \mathbf{n}_j} = \begin{cases} \min(1, \frac{\Phi^{\min} - \Phi_i}{\Phi(\mathbf{x}_{q_{ij}}) - \Phi}) & , \Phi(\mathbf{x}_{q_{ij}}) - \Phi_i < 0 , \\ \min(1, \frac{\Phi^{\max} - \Phi_i}{\Phi(\mathbf{x}_{q_{ij}}) - \Phi}) & , \Phi(\mathbf{x}_{q_{ij}}) - \Phi_i > 0 , \\ 1 & , \Phi(\mathbf{x}_{q_{ij}}) - \Phi_i = 0 . \end{cases} \quad (3.65)$$

The states on the shared face are determined by retaining the first order term of a Taylor series,

$$\begin{aligned} \Phi_{\mathbf{x}_{q_{ij}}} &= \Phi_i + \frac{1}{2} \Psi_{\mathbf{n}_i} \nabla \Phi_i \cdot (\mathbf{x}_j - \mathbf{x}_i) \\ \Phi_{\mathbf{x}_{q_{ji}}} &= \Phi_j - \frac{1}{2} \Psi_{\mathbf{n}_j} \nabla \Phi_j \cdot (\mathbf{x}_j - \mathbf{x}_i). \end{aligned} \quad (3.66)$$

Barth's limiter given in Eq. (3.65) is very diffusive [49, 18]. The sensitivity of the limiter function in the smooth region is known to stall the convergence [71]. A limiter function developed by Venkatakrishnan is designed to be less sensitive to the small variation of solution in the smooth region and is widely used due to its superior convergence properties [71]. The limiter function is defined as

$$\Psi_i = \min_j \begin{cases} \frac{1}{\Delta_2} \left[ \frac{(\Delta_{1,\max}^2 + \epsilon^2) \Delta_2 + 2 \Delta_2^2 \Delta_{1,\max}}{\Delta_{1,\max}^2 + 2 \Delta_2^2 + \Delta_{1,\max} \Delta_2 + \epsilon^2} \right] & : \Delta_2 > 0 \\ \frac{1}{\Delta_2} \left[ \frac{(\Delta_{1,\min}^2 + \epsilon^2) \Delta_2 + 2 \Delta_2^2 \Delta_{1,\min}}{\Delta_{1,\min}^2 + 2 \Delta_2^2 + \Delta_{1,\min} \Delta_2 + \epsilon^2} \right] & : \Delta_2 < 0 \\ 1 & : \Delta_2 = 0 \end{cases} \quad (3.67)$$

where

$$\begin{aligned} \Delta_2 &= \frac{1}{2} (\nabla \Phi_i \cdot \mathbf{e}_{ij}) \\ \Phi_{\max} &= \max(\Phi_i, \max\{\Phi_j\}) \\ \Phi_{\min} &= \min(\Phi_i, \min\{\Phi_j\}) \\ \Delta_{1,\max} &= \Phi_{\max} - \Phi_i \\ \Delta_{1,\min} &= \Phi_{\min} - \Phi_i. \end{aligned} \quad (3.68)$$

A small number,  $\epsilon^2$  prevents the division by zero when the gradient is very small. In the implementation,  $\epsilon^2$  is set to be a function of the local length scale,

$$\epsilon^2 = (c\delta h)^3 \quad (3.69)$$

where  $c$  is a constant and  $\delta h$  is the local mesh length scale. The present work uses  $c = 5$  and  $(\delta h)^3 = \Omega_i$ .

#### h. Viscous flux

The gradients computed by the Green-Gauss method and the least-squares method are available at the vertices. The viscous flux, on the other hand, is evaluated at the quadrature points which are mid-edge points.

One of the most popular methods to compute the viscous flux is based on the Galerkin finite element method [72]. In this method, the viscous term is evaluated using the Hessian matrix at the cell center. This eliminates the need for the viscous flux evaluation on the cell faces. The method is an attractive choice if the mesh is made of simplex elements only. The method, however, does not naturally extend to a mesh with non-simplex elements, such as quadrilateral elements in 2-D, and prism, pyramid, and hexahedral elements in 3-D [23]. This gradient computation method is not pursued since the present work uses a mixed elements mesh.

One simple method to compute the gradient at the mid-edge point is to use the arithmetic average of the gradients at the two ends of an edge [73]. The approach can recycle the computed gradients from the linear reconstruction step and thus save the computational cost. The arithmetic average of the gradients at the mid-point of the edge  $\mathbf{e}_{ij}$  is

$$\nabla\Phi_{(i+j)/2} \approx \frac{1}{2} (\nabla\Phi_i + \nabla\Phi_j) \quad (3.70)$$

where  $(i + j)/2$  denotes the mid-edge point. The stencil for  $\nabla\Phi_{(i+j)/2}$  is an union of the first-order stencils  $\mathcal{S}_{\mathbf{n}_i}^1, \mathcal{S}_{\mathbf{n}_j}^1$  of the nodes  $\mathbf{n}_i$  and  $\mathbf{n}_j$ . The resultant stencil contains an increased amount of data, which in turn reduces the weightings of  $\Phi_i$  and  $\Phi_j$  on the averaged gradient,  $\nabla\Phi_{(i+j)/2}$ . The unfavored weightings cause the local terms to eventually decouple from the averaged gradient and lead to severely reduced accuracy [22].

Accuracy of the averaged gradient can be substantially improved by increasing the weightings of  $\nabla\Phi_i$  and  $\nabla\Phi_j$  using the directional derivative. The directional derivative along the edge  $\mathbf{e}_{ij}$  is defined as

$$\frac{\partial\Phi}{\partial l}|_{(i+j)/2} \approx \frac{\Phi_j - \Phi_i}{|\mathbf{x}_j - \mathbf{x}_i|}. \quad (3.71)$$

The final form of the modified average gradient [21, 22] is obtained by 1) subtracting the gradient weight along an edge from the average gradient and 2) adding the directional derivative,

$$\nabla\Phi_{(i+j)/2} = \nabla\Phi_{(i+j)/2}^* - (\nabla\Phi_{(i+j)/2}^* \cdot \hat{\mathbf{e}}_{ij})\hat{\mathbf{e}}_{ij} + \frac{\Phi_j - \Phi_i}{|\mathbf{x}_j - \mathbf{x}_i|}\hat{\mathbf{e}}_{ij} \quad (3.72)$$

where

$$\nabla\Phi_{(i+j)/2}^* \equiv \frac{1}{2}(\nabla\Phi_i + \nabla\Phi_j), \quad (3.73)$$

$\hat{\mathbf{e}}_{ij}$  is the unit vector in the direction from node  $\mathbf{n}_i$  to  $\mathbf{n}_j$ . In the present work, Eq. (3.72) is used to compute the gradient of each component of a velocity vector and of a temperature. The computed gradients are then used to evaluate the viscous flux on dual-cell faces.

## 2. Temporal discretization

The residual  $\mathbf{R}_i$  of a cell  $c_i$  is defined as the right hand side of Eq. (3.40),

$$\mathbf{R}_i = - \sum_j^{m_i} \mathbf{f}_j \cdot \mathbf{n}_j A_j + \mathbf{g}_i \Omega_i. \quad (3.74)$$

The explicit time integration of Eq. (3.40) can be defined as a system of ordinary differential equations in time in terms of the residual,

$$\mathbf{u}_i^{n+1} = \mathbf{u}_i^n + \Delta t_i^n \mathbf{R}_i^n / \Omega_i \quad (3.75)$$

where

$$\Delta t_i^n = t_i^{n+1} - t_i^n. \quad (3.76)$$

The steady solution is obtained by solving the equations until the flow reaches a steady condition and  $\mathbf{R}$  becomes negligible. Time accuracy is not relevant for a steady flow computation so that the computation is allowed to evolve in time using a local time step,  $\Delta t_i$ . The local time step can speed up the convergence rate.

For a time accurate unsteady solution, a global time step should be used. Therefore, the stable time step for the unsteady case is

$$\Delta t_i^n \equiv \min\{\Delta t_j^n : j = 1, 2, \dots, m\} \quad (3.77)$$

where  $m$  is the number of nodes.

The forward Euler integration in Eq. (3.75) is not stable. A multi-stage integra-

tion is used instead. The following four stages scheme is used herein

$$\begin{aligned}
\mathbf{u}_i^{(0)} &= \mathbf{u}_i^n \\
\mathbf{u}_i^{(1)} &= \mathbf{u}_i^{(0)} + \alpha_1 \Delta t_i \mathbf{R}_i(\mathbf{u}_i^{(0)}, \nabla \mathbf{u}_i^{(0)}) / \Omega_i \\
\mathbf{u}_i^{(2)} &= \mathbf{u}_i^{(0)} + \alpha_2 \Delta t_i \mathbf{R}_i(\mathbf{u}_i^{(1)}, \nabla \mathbf{u}_i^{(0)}) / \Omega_i \\
\mathbf{u}_i^{(3)} &= \mathbf{u}_i^{(0)} + \alpha_3 \Delta t_i \mathbf{R}_i(\mathbf{u}_i^{(2)}, \nabla \mathbf{u}_i^{(2)}) / \Omega_i \\
\mathbf{u}_i^{(4)} &= \mathbf{u}_i^{(0)} + \alpha_4 \Delta t_i \mathbf{R}_i(\mathbf{u}_i^{(3)}, \nabla \mathbf{u}_i^{(2)}) / \Omega_i \\
\mathbf{u}_i^{n+1} &= \mathbf{u}_i^{(4)}
\end{aligned} \tag{3.78}$$

where the stage coefficients  $\alpha_i$  are

$$\begin{aligned}
\alpha_1 &= 0.1668 \\
\alpha_2 &= 0.3028 \\
\alpha_3 &= 0.5276 \\
\alpha_4 &= 1.0.
\end{aligned} \tag{3.79}$$

The viscous fluxes are evaluated at odd stages only for computational efficiency.

The time step,  $\Delta t_i$ , must satisfy the CFL condition. The time step is defined as

$$\Delta t_i = \sigma \frac{\Omega_i}{(\lambda_c^x + \lambda_c^y + \lambda_c^z)_i + C(\lambda_v^x + \lambda_v^y + \lambda_v^z)_i} \tag{3.80}$$

where  $\sigma$  is the CFL number and  $C = 4$ . The convective spectral radii [55] are

$$\begin{aligned}
\lambda_c^x &= (|u| + c) \Delta A^x \\
\lambda_c^y &= (|v| + c) \Delta A^y \\
\lambda_c^z &= (|w| + c) \Delta A^z
\end{aligned} \tag{3.81}$$

and the viscous spectral radii [23] are

$$\begin{aligned}\lambda_v^x &= \max\left(\frac{4}{3\rho}, \frac{\gamma}{\rho}\right) \left(\frac{\mu_L}{Pr_L} + \frac{\mu_T}{Pr_T}\right) \frac{(\Delta A^x)^2}{V_i} \\ \lambda_v^y &= \max\left(\frac{4}{3\rho}, \frac{\gamma}{\rho}\right) \left(\frac{\mu_L}{Pr_L} + \frac{\mu_T}{Pr_T}\right) \frac{(\Delta A^y)^2}{V_i} \\ \lambda_v^z &= \max\left(\frac{4}{3\rho}, \frac{\gamma}{\rho}\right) \left(\frac{\mu_L}{Pr_L} + \frac{\mu_T}{Pr_T}\right) \frac{(\Delta A^z)^2}{V_i}.\end{aligned}\tag{3.82}$$

The variables  $\Delta A^x$ ,  $\Delta A^y$  and  $\Delta A^z$  are the projections of the volume  $\Omega_i$  on the  $y-z$ ,  $z-x$  and the  $x-y$  planes [23]. They are defined as

$$\begin{aligned}\Delta A^x &= \frac{1}{2} \sum_{j=1}^{m_i} |n_x \Delta A_j| \\ \Delta A^y &= \frac{1}{2} \sum_{j=1}^{m_i} |n_y \Delta A_j| \\ \Delta A^z &= \frac{1}{2} \sum_{j=1}^{m_i} |n_z \Delta A_j|.\end{aligned}\tag{3.83}$$

#### a. Implicit residual smoothing

The time step can become very small and slow down the convergence, especially for low speed flows and sonic conditions. Implicit residual smoothing adds an implicit flavor into the explicit method by blending the residuals of the neighboring nodes [53]. This allows the solution to march in time at a larger time step than a non-smoothed explicit method.

The modified residual,  $\mathbf{R}'_i$ , is defined for node  $\mathbf{n}_i$  as [53]

$$\mathbf{R}'_i = \frac{\mathbf{R}_i + \epsilon \sum_{j=1}^{m_i} \mathbf{R}_j}{1 + \epsilon \sum_{j=1}^{m_i} 1}\tag{3.84}$$

where  $\mathbf{R}_i$  and  $\mathbf{R}_j$  are the current residuals at nodes  $\mathbf{n}_i$  and  $\mathbf{n}_j$ , respectively.  $m_i$  is the number of neighboring nodes in the first-order stencil for  $\mathbf{n}_i$ . The equations are typically solved by a Jacobi iteration with  $\epsilon = 0.5$ , which makes the system diagonally dominant so that only a few iterations are required to reach an acceptable convergence.



### 3. Boundary conditions

This section presents the boundary conditions. The boundary conditions implemented herein include solid wall, inlet, outlet, symmetric plane, and periodic boundary conditions. In the present work, the inlet and outlet boundary conditions and the solid wall boundary conditions for the continuity and the energy equations are applied “weakly” [74] as conditions on the flux at boundary surfaces. Instead of being applied directly to state variables, the boundary conditions are used to find the intermediate state variables on the boundary faces. The intermediate state variables are then used to evaluate the boundary flux.

On the solid wall, the contravariant velocity,  $V$ , in Eq. (3.22) is zero. This causes the boundary flux for the continuity equation to vanish. The non-slip boundary condition for the Navier-Stokes equations makes the computation of momentum equations on the wall unnecessary since the velocity on the wall can be directly determined from the specified wall velocity. The energy flux on wall boundaries reduces to the sum of  $pV_g$  in Eq. (3.22) and  $\nabla q \cdot \mathbf{n}_B$  in Eq. (3.29). For an adiabatic wall condition, the boundary flux for the energy equation further simplifies to  $pV_g$ . If a grid is stationary, the boundary flux for energy equation vanishes.

For subsonic inflow, the intermediate state variables on inlet faces are determined by one interior condition and four upstream conditions. For subsonic outflow, the intermediate state variables are determined by four interior conditions and the prescribed static pressure at outlet. For supersonic in/outflow, the intermediate state variables on boundary faces are determined by upstream conditions only. Specifically, the intermediate variables at the inlet are determined by the five freestream conditions and the intermediate variables at the outlet are determined by the five interior state variables.

The subsonic inlet boundary conditions are specified by 1) one Riemann invariant based on the interior conditions and 2) four upstream conditions that include the total pressure,  $p^*$ , the total temperature,  $T^*$ , and two inlet flow angles,  $\alpha$  and  $\beta$ . These specified conditions are used to compute the intermediate state variables on the boundary faces. The intermediate states are found in the following way. The Riemann invariant based on the interior conditions is

$$R^- = \mathbf{v}_i \cdot \mathbf{n}_{B_i} - \frac{2c_i}{\gamma - 1} \quad (3.85)$$

where  $\mathbf{n}_B$  is a unit normal vector on the boundary and  $c$  is the speed of sound. The total pressure and total temperature from the four upstream conditions are used to compute the intermediate entropy and total enthalpy

$$s = \frac{p^*}{(\rho^*)^\gamma}, \quad H = \frac{\gamma RT^*}{\gamma - 1}. \quad (3.86)$$

The Riemann invariant,  $R^+$ , can be expressed as

$$R^+ = R^- + \frac{4}{\gamma - 1}c \quad (3.87)$$

where  $c$  is an intermediate speed of sound that is not yet defined. In order to remove  $c$  from Eq. (3.87),  $c$  can be expressed in terms of the total enthalpy, tangential velocity component to the boundary face, and the Riemann invariants,

$$c = (\gamma - 1) \sqrt{H - \frac{u_t^2}{2} - \frac{1}{8}(R^+ + R^-)} \quad (3.88)$$

where

$$u_t = |\mathbf{v}_i| - \mathbf{v}_i \cdot \mathbf{n}_{B_i} \quad (3.89)$$

and the following relations are used

$$u_B = \frac{R^+ + R^-}{2} \quad (3.90)$$

$$c = \frac{\gamma - 1}{4} (R^+ - R^-). \quad (3.91)$$

$u_B$  and  $c$  are the intermediate normal velocity and the speed of sound. Substituting Eq. (3.88) into Eq. (3.87) results in a quadratic equation for  $R^+$  and a solution for the quadratic equation is

$$R^+ = \frac{1}{\gamma + 1} \left[ (\gamma - 3)R^- + 4\sqrt{H - \frac{u_t^2}{2} - \frac{\gamma - 1}{2}(R^-)^2} \right]. \quad (3.92)$$

Once  $R^+$  is computed, the intermediate normal velocity and the intermediate speed of sound can be computed using Eq. (3.90). The intermediate density is computed as

$$\rho = \left[ \frac{c^2}{\gamma s} \right]^{\frac{1}{\gamma - 1}}. \quad (3.93)$$

Finally, the components of the intermediate velocity are defined as

$$|u| = \sqrt{u_B^2 + u_t^2} \quad (3.94)$$

$$u = |u| \cos(\alpha) \quad (3.95)$$

$$v = |u| \sin(\alpha) \cos(\beta) \quad (3.96)$$

$$w = |u| \sin(\alpha) \sin(\beta) \quad (3.97)$$

where  $\alpha$  and  $\beta$  are the two prescribed inlet flow angles. Once the intermediate velocity and two thermodynamic states,  $s$  and  $\rho$ , are defined on the inlet boundary face, these intermediate values are used to evaluate the flux across the inlet faces.

The subsonic outflow boundary conditions consist of 1) the static pressure  $p_b$  specified on the outlet face and 2) the four conditions from the interior. Using the specified  $p_b$  and the entropy, tangential velocity, speed of sound, and total pressure

from interior conditions, the intermediate state variables can be defined in the following way. The intermediate pressure is equal to the specified backpressure

$$p = p_b. \quad (3.98)$$

The intermediate entropy and tangential velocity are assumed same as the interior values

$$s = s_i, \quad u_t = |\mathbf{v}_i| - \mathbf{v}_i \cdot \mathbf{n}_{B_i}. \quad (3.99)$$

The Riemann invariants are

$$R^+ = \mathbf{v}_i \cdot \mathbf{n}_{B_i} + \frac{2c_i}{\gamma - 1} \quad (3.100)$$

$$R^- = R^+ - \frac{4}{\gamma - 1} \sqrt{\gamma (p^*)^{\frac{\gamma-1}{\gamma}} s^{\frac{1}{\gamma}}}. \quad (3.101)$$

The intermediate density and the intermediate velocity can then be computed using the Riemann invariants as shown in Eq. (3.90)-(3.94). The computed intermediate state variables are used to evaluate the flux across the outlet faces.

The flux through a symmetric plane is zero and the normal velocity on the symmetric boundary is also zero. Because the normal velocity is zero, the tangential component of the normal velocity gradient must also vanish

$$t_i v_{i,j} n_j = 0 \quad (3.102)$$

where  $t_i$  is a unit vector in tangential direction and  $n_j$  is a unit normal vector. Since the flow is symmetrical with respect to the plane, the normal component of the tangential velocity gradient should vanish on the wall

$$n_i v_{i,j} t_j = 0. \quad (3.103)$$

The normal components of gradients of scalar variables along the symmetric boundary must vanish

$$n_i \phi_{,i} = 0. \quad (3.104)$$

The symmetric boundary conditions can be implemented in the cell-vertex method by 1) correcting the gradient components as is done in Eq. (3.102)-(3.104), 2) skipping the flux computations on the symmetry boundary faces such that zero flux condition on the boundary is enforced and 3) subtracting the normal components of the residual vectors for the momentum equations to ensure the zero normal velocity on the boundary

$$R_i = R_i - n_i R_j n_j, \quad 2 \leq i \leq 4, \quad 1 \leq j \leq 5 \quad (3.105)$$

where  $R_i$  is the residual vector.

The periodic boundary conditions are implemented for both translational periodic boundaries and rotational periodic boundaries. For a set of translational periodic boundaries, a vertex on the “master ” boundary is related to its pair vertex on the “slave ” boundary by

$$\mathbf{x}_{i_S} = \mathbf{x}_{i_M} + \mathbf{X}_{M \rightarrow S} \quad (3.106)$$

where  $\mathbf{X}_{i_S}$  and  $\mathbf{X}_{i_M}$  are the vectors that define the location of a slave and a master vertex on periodic boundaries and  $\mathbf{X}_{M \rightarrow S}$  is a vector that defines a translation from a master boundary to its slave boundary. The state variables at a master vertex and its slave vertex are equal by definition

$$\mathbf{u}_{i_M} = \mathbf{u}_{i_S}. \quad (3.107)$$

In the present work, fluxes on periodic boundaries are not explicitly computed

$$\mathbf{R}_{i_M} = - \sum_{\substack{j=1 \\ j \neq j_M^*}}^{m_{i_M}} \mathbf{f}_j \cdot \mathbf{n}_j A_j + \mathbf{g}_i \Omega_i \quad (3.108)$$

$$\mathbf{R}_{i_S} = - \sum_{\substack{j=1 \\ j \neq j_S^*}}^{m_{i_S}} \mathbf{f}_j \cdot \mathbf{n}_j A_j + \mathbf{g}_i \Omega_i \quad (3.109)$$

$$(3.110)$$

where  $\mathbf{n}_{j_M^*} A_{j_M^*}$  and  $\mathbf{n}_{j_S^*} A_{j_S^*}$  denote the boundary faces areas on the master and slave periodic boundaries, respectively. This implies that the flux sum to compute the residuals for periodic boundary vertices are partially completed. The resultant partial residuals for a master vertex and its slave vertex can be summed up at the master vertex

$$\mathbf{R}_{i_M}^* = \mathbf{R}_{i_M} + \mathbf{R}_{i_S} \quad (3.111)$$

such that the total residual  $\mathbf{R}_{i_M}^*$  represents the residual for the combined control volume,  $\Omega_{i_M}^*$  where

$$\Omega_{i_M}^* = \Omega_{i_M} + \Omega_{i_S}. \quad (3.112)$$

The ( $m$ )-stage equation of Eq. (3.78) can be written for the state variables of the master vertex as

$$\mathbf{u}_{i_M}^{(m)} = \mathbf{u}_{i_M}^{(m-1)} + \frac{\alpha_m \Delta t_{i_M}^*}{\Omega_{i_M}^*} \mathbf{R}_{i_M}^* \quad (3.113)$$

where  $t_{i_M}^*$  is a modified time step that takes into account of the combined control volume. Specifically, the 2-D projection of the control volume in Eq. (3.83) must be defined by the surface area vectors of the combined control volume. Volume term  $\Omega_i$  in Eq. (3.80) and Eq. (3.82) must be replaced by  $\Omega_{i_M}^*$ . Once the multi-stage time integration is completed for  $\mathbf{u}_{i_M}$ , the state variables of its slave vertex are updated

$$\mathbf{u}_{i_S}^n = \mathbf{u}_{i_M}^n. \quad (3.114)$$

For a set of rotational periodic boundaries, a vertex on 'master' boundary is related to its pair vertex on 'slave' boundary by

$$\mathbf{x}_{i_S} = \mathbf{T}_3 \mathbf{x}_{i_M}. \quad (3.115)$$

Assuming a rotation about  $x$ -axis only,  $\mathbf{T}_3$  is defined as

$$\mathbf{T}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & \sin \theta \\ 0 & -\sin \theta & \cos \theta \end{bmatrix} \quad (3.116)$$

where  $\theta$  is the angle measured on  $y - z$  plane from the master boundary to the slave boundary. The state variables at a master vertex and its slave vertex are related by

$$\mathbf{u}_{i_M} = \mathbf{T} \mathbf{u}_{i_S} \quad (3.117)$$

where

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \cos \theta & \sin \theta & 0 \\ 0 & 0 & -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \quad (3.118)$$

The total residual at  $\mathbf{n}_{i_M}$  can be computed from the partial residual at the master node,  $\mathbf{n}_{i_M}$ , and the slave node,  $\mathbf{n}_{i_S}$

$$\mathbf{R}_{i_M}^* = \mathbf{R}_{i_M} + \mathbf{T}^{-1} \mathbf{R}_{i_S} \quad (3.119)$$

where  $\mathbf{R}_{i_M}^*$  is the total residual accumulated at  $\mathbf{n}_{i_M}$ . Only  $\mathbf{u}_{i_M}$ , which is the state variables at  $\mathbf{n}_{i_M}$  on master periodic boundary, is directly computed using the total

residual,  $\mathbf{R}_{i_M}^*$ . The  $(m)$ -stage equation of Eq. (3.78) for  $\mathbf{u}_{i_M}$  is

$$\mathbf{u}_{i_M}^{(m)} = \mathbf{u}_{i_M}^{(m-1)} + \frac{\alpha_m \Delta t_{i_M}^*}{\Omega_{i_M}^*} (\mathbf{R}_{i_M} + \mathbf{T}^{-1} \mathbf{R}_{i_S}). \quad (3.120)$$

After each time step, the updated state variables at the master vertex are copied back to the state variables of its slave nodes

$$\mathbf{u}_{i_S}^{(n)} = \mathbf{T} \mathbf{u}_{i_M}^{(n)}. \quad (3.121)$$

### C. Implementation of turbulence model

The  $k - \omega$  turbulence model, Eq. (2.49)- (2.50), can be written in the integral form as

$$\frac{\partial}{\partial t} \int_{\Omega} \mathbf{W} d\Omega + \oint_{\partial\Omega} [\mathbf{\Gamma}_c - \mathbf{\Gamma}_v] \cdot \mathbf{n} dS = \int_{\Omega} \mathbf{\Pi} d\Omega \quad (3.122)$$

where

$$\begin{aligned} \mathbf{W} &= \begin{pmatrix} \rho k \\ \rho \omega \end{pmatrix} \\ \mathbf{\Gamma}_c &= \begin{pmatrix} \rho k (\mathbf{v} - \mathbf{v}_g) \\ \rho \omega (\mathbf{v} - \mathbf{v}_g) \end{pmatrix} \\ \mathbf{\Gamma}_v &= \begin{pmatrix} (\mu + \sigma_k \mu_T) \nabla k \\ (\mu + \sigma_\omega \mu_T) \nabla \omega \end{pmatrix} \\ \mathbf{\Pi} &= \begin{pmatrix} P_k - D_k \\ P_\omega - D_\omega + CD_\omega \end{pmatrix}. \end{aligned} \quad (3.123)$$



$\Gamma_c$  and  $\Gamma_v$  are the turbulent inviscid flux and viscous flux, respectively.  $P$  is the turbulent production as defined in Eq. (2.51) and

$$D_k = \beta^* \rho \omega k \quad (3.124)$$

$$P_\omega = \gamma_\omega \frac{\omega}{k} P_k \quad (3.125)$$

$$D_\omega = \beta \rho \omega^2 \quad (3.126)$$

$$CD_\omega = (1 - F_1) \frac{2\rho\sigma\omega_2}{\omega} \nabla k \cdot \nabla \omega. \quad (3.127)$$

$\mu_T$  is the turbulent viscosity defined as

$$\mu_T = \alpha^* \frac{\rho k}{\omega}. \quad (3.128)$$

The turbulence model Eq. (3.122) have the same form as the conservation form of the Navier-Stokes Eq. (3.19). Consequently, the model equations can be solved with the same numerical algorithms. The inviscid flux  $\Gamma_c$  is computed with the Roe's Riemann solver and the viscous flux  $\Gamma_v$  is computed with the average gradient.

The source term  $\Pi$  can be dominant in the  $k - \omega$  model. The rapid change of the source term causes the instability of the solution algorithms and the slow convergence of solution. A remedy to the source term dominance is to treat the source term implicitly [75, 23]. Consider a discrete form of the model equations in Eq. (3.122).

$$\mathbf{w}_i^{n+1} = \mathbf{w}_i^n - \frac{\Delta t_i}{\Omega_i} \left[ \sum_{j=1}^{m_i} (\Gamma_c^n - \Gamma_v^n)_j S_j - \Pi_i^n \Omega_i \right] \quad (3.129)$$

where  $\mathbf{w}$  is the volume average of  $\mathbf{W}$  over a control volume  $\mathbf{c}_i$ . The source term  $\Pi_i^n$  can be evaluated at the next time level

$$\mathbf{I} \cdot (\mathbf{w}_i^{n+1} - \mathbf{w}_i^n) = -\frac{\Delta t_i}{\Omega_i} \left[ \sum_{j=1}^{m_i} (\Gamma_c^n - \Gamma_v^n)_j S_j - \Pi_i^{n+1} \Omega_i \right] \quad (3.130)$$

where  $\mathbf{I}$  is the identity matrix.

The source term  $\mathbf{\Pi}_i^{n+1}$  in Eq. (3.130) can be approximated as

$$\mathbf{\Pi}_i^{n+1} \approx \mathbf{\Pi}_i^n + \left( \frac{\partial \mathbf{\Pi}}{\partial \mathbf{w}} \right)_i \Delta \mathbf{w}_i^n. \quad (3.131)$$

After plugging in Eq. (3.131) into Eq. (3.130) and rearranging terms, Eq. (3.130) becomes

$$\left[ \frac{\mathbf{I}}{\Delta t_i} - \left( \frac{\partial \mathbf{\Pi}}{\partial \mathbf{w}} \right)_i \right] \cdot (\mathbf{w}_i^{n+1} - \mathbf{w}_i^n) = -\frac{1}{\Omega_i} \mathbf{R}_i \quad (3.132)$$

where  $\mathbf{R}_i$  is the residual and it is defined as

$$\mathbf{R}_i \equiv \sum_{j=1}^{m_i} (\mathbf{\Gamma}_c^n - \mathbf{\Gamma}_v^n)_j S_j - \mathbf{\Pi}_i^n \Omega_i. \quad (3.133)$$

Eq. (3.132) can be rearranged for  $\mathbf{w}_i^{n+1}$ ,

$$\mathbf{w}_i^{n+1} = \mathbf{w}_i^n + \frac{1}{\Omega_i} \left[ \frac{\mathbf{I}}{\Delta t_i} - \left( \frac{\partial \mathbf{\Pi}}{\partial \mathbf{w}} \right)_i \right]^{-1} \cdot \mathbf{R}_i. \quad (3.134)$$

The ( $m$ )-stage in Eq. (3.78) is then defined as

$$\mathbf{w}_i^{(m)} = \mathbf{w}_i^{(0)} + \frac{\alpha_k}{\Omega_i} \left[ \frac{\mathbf{I}}{\Delta t_i} - \left( \frac{\partial \mathbf{\Pi}}{\partial \mathbf{w}} \right)_i^{(m-1)} \right]^{-1} \cdot \mathbf{R}_i^{(m-1)}. \quad (3.135)$$

The diagonal terms of the Jacobian matrix  $\partial \mathbf{\Pi} / \partial \mathbf{w}$  are approximated as

$$\frac{\partial}{\partial k} (P_k - D_k) \approx -\frac{D_k}{k} \quad (3.136)$$

$$\frac{\partial}{\partial \omega} (P_\omega - D_\omega + CD_\omega) \approx -\frac{|CD_\omega| + 2D_\omega}{\omega}. \quad (3.137)$$

Note that the off-diagonal terms are not computed.

Using the approximations of the Jacobian matrix terms, Eq. (3.135) can be rewritten as

$$\mathbf{w}_i^{(m)} = \mathbf{w}_i^{(0)} + \frac{\alpha_k}{\Omega_i} \mathbf{S} \cdot \mathbf{R}_i^{(m-1)}. \quad (3.138)$$

The matrix  $S$  is

$$\mathbf{S} = \frac{\Delta t_i}{s_1 s_2} \begin{pmatrix} s_2 & 0 \\ 0 & s_1 \end{pmatrix}, \quad (3.139)$$

where

$$\begin{aligned} s_1 &= 1 - D_k/k \\ s_2 &= 1 - (|CD_\omega| + 2D_\omega)/\omega. \end{aligned} \quad (3.140)$$

Despite the implicit treatment of the source terms, the model equations can still produce nonphysically large eddy viscosity in the region close to stagnation points and during the initial iteration steps. Limiting the values of  $P_k$  and  $\omega$  can reduce the instability caused by the occasional spikes [76, 77, 78, 37, 31].

In the present work, the ceiling value for  $\omega$  is defined by the Menter's wall boundary condition [75]

$$\omega_{\max} = \frac{60\nu_w}{\beta\Delta y_1^2} \quad (3.141)$$

The value of  $\omega$  is set to the maximum value defined as the above if  $\omega$  exceeds the limit. The production of the turbulent kinetic energy is limited by

$$P_k = \min(P_k, 20D_k). \quad (3.142)$$

In the following, the wall boundary conditions for the  $k - \omega$  model are presented. The turbulence kinetic energy is zero on the wall because of the vanishing fluctuating velocity. On the other hand, the wall boundary condition of  $\omega$  is not as simple as for  $k$  and the boundary condition should be implemented carefully. For smooth wall boundary, Wilcox [27], p. 343, suggests that exact solution of  $\omega$  should be imposed on the first few points from the wall in the viscous sublayer. The suggested value for  $\omega$  is given by

$$\omega_w \approx \frac{N_w\nu_w}{y^2}, \quad y^+ < 2.5 \quad (3.143)$$

where  $N_w$  is a model dependent constant and  $\nu_w$  is the kinematic viscosity on the

wall. This approach is not, however, suitable for an unstructured flow solver due to the extended region of boundary condition updates.

An alternative approach is the rough-surface wall approximation [27], pp. 175-177, which is given by

$$\omega_w = \frac{2500\nu_w}{k_s^2} \quad (3.144)$$

where  $k_s$  is the sand grain height. The rough wall boundary condition simulates a smooth wall if the wall-unit roughness height  $k_s^+ = k_s\nu_w/u^* < 5$ . The rough-surface wall boundary condition can be implemented in a different way using the grid spacing  $\Delta y_1$  next to the wall [75]

$$\omega_w = \frac{60\nu_w}{\beta\Delta y_1^2}. \quad (3.145)$$

#### D. Parallel implementation

This section presents the strategy adopted for the parallel computation of the flow model. The present work employs the domain decomposition [79, 80, 38], the Single Program Multiple Data (SPMD) approach [81], and the Message Passing Interface (MPI) library [81, 82]. The combined strategy makes the solution algorithm portable to a wide range of computational environments. For example, the implemented parallel solver has been successfully tested on SGI Origin, IBM p690 and PC based Linux clusters. In the following, the details of the parallelization strategy are presented.

The domain decomposition is performed as a pre-process step. At this stage, the global mesh is partitioned into as many blocks as the number of available processors. The practical number of partitions is, however, limited by the increasing communication overhead. An example of one-to-two partition is shown in Fig. 11 (a), which also shows the shadowed portion of the block meshes is overlapped. The intersection of the partitioned meshes set is referred to as the buffer layer. During the mesh

partitioning, the buffer layers are padded around the interfaces between the adjacent blocks of the partitioned meshes. The buffer layers define the region where data are exchanged between blocks.

The buffer layers consist of the master and the slave data points. Fig. 11 (b) shows the slave data points as a solid line and the master data points as a dotted line. The two arrows in the far right side of Fig. 11 (b) show the data dependency between the block  $B_1$  and  $B_2$ . Specifically, the master data points of  $B_2^M$  are identical to the slave data points of the block  $B_1^S$ . The slave data are dependent on the master data ( $B_1^S \leftarrow B_2^M$ ) and the same applies between the  $B_1^M$  and  $B_2^S$  ( $B_1^M \rightarrow B_2^S$ ).

At each multi-stage time integration step, solutions are obtained at the active data points only, which exclude the slave data points. For  $B_1$  in the example, the solutions on the solid-lined  $B_1^S$  are not explicitly determined locally. Instead, the data on  $B_1^S$  are updated from the solution data passed from the block  $B_2$ . The data transfer in the opposite direction must also be performed. The message passing is carried out in a synchronized manner using a blocking communication in the MPI standard.

The buffer layer in the present study is designed to have a single layer opposed to the much more common double layers [38]. A single layer herein means that a slave data point in a block mesh can be reached from a master data point in the same block within an edge. The design goal behind this choice is to reduce the communication cost. If one safely assumes that the double-layered buffer set contains twice as many data points as the single-layer set, the amount of inter-boundary data exchange is halved with the single-layer approach.

Most of the solution algorithms are compatible with the single-layer buffer. An exception is the evaluation of the viscous flux, which requires the gradients on the slave data points refreshed. The need for the gradient data roughly doubles the

amount of the required data exchange and it appears to cancel the communication cost in the single-layer case. The slight reduction of the cost (approximately 10%) is still possible with the single-layer case since the viscous flux does not depend on the density gradient. For the latter case, the gradients at the active data points can be computed from the local state variables only and this eliminates the need for the gradients data passing. A significant reduction of the communication cost is possible with the single-layer buffer because the viscous flux is not computed at every multi-stage time integration. As noted in Eq. (3.78), for computational efficiency the viscous flux is evaluated at odd stages only. This amounts to approximately 30% reduction of the communication cost.

In the present work, the pre-processing step also produces a reverse mapping from the partitioned meshes to global mesh. The table is used to map the global mesh from the partitioned meshes such that the fragmented block solutions can be later assembled into a single block solution.

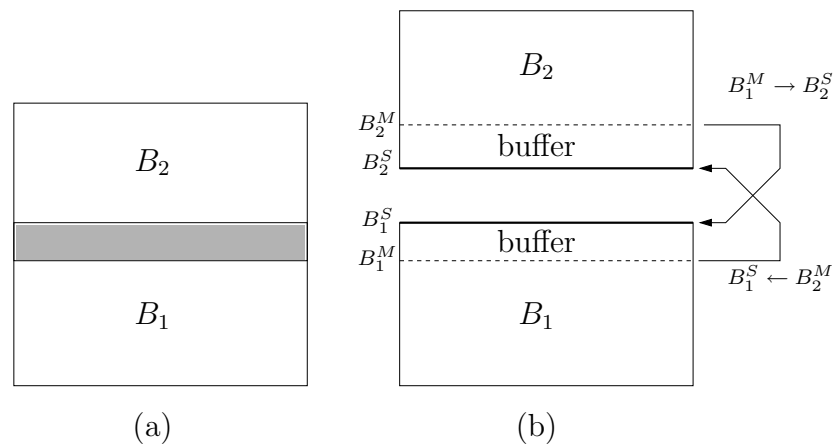


Fig. 11. Example of one-to-two mesh partitions. (a) The shaded area is the buffer layer ( $B_1 \cap B_2$ ) between the block mesh  $B_1$  and  $B_2$ . (b) The solid lines denote the slave data points ( $B_1^S$  and  $B_2^S$ ) and the dotted lines denote the master data points ( $B_1^M$  and  $B_2^M$ ).

## E. Graphical user interface

The implementation of a Graphical User Interface (GUI) is presented. The GUI was developed to enhance the usability of the flow solver. The major benefit of the GUI is that it detaches the users from the complex input file format and allows them to concentrate on the problems on hand.

The GUI is programmed in Tcl/Tk [83, 84, 85]. Tcl/Tk is an interpreted script language. Once a program is written in the script form which is machine independent, the program is portable to major operating systems such as Unix and Microsoft Windows.

The flow solver written in FORTRAN 90 [86, 87, 88] and the GUI written in Tcl/Tk are linked by a text file. The format of the text file is in FORTRAN name-list. The text file serves as an output of the GUI program and the input of the solver. Depending on the choices made through the GUI, the GUI records the current states of user input in the file. This text file is then subsequently read by the solver. These two independent processes appear to users as a single process, which results in an unified look.

In the following, the GUI windows are shown and the description of the panels are provided. Figure 12 shows an opening splash window, which greets the user and disappears after a preset time. The main control panel is shown in Figure 13. The main panel has two text boxes and eight sub-menu buttons. In the top text box, a case title can be typed in. The second text box is for a file name, which can be used to load the input file generated previously.

The sub-menu panels are accessed by pressing the appropriate buttons. The screen shots of the sub-menu are shown in Figures 14 - 20. In order to prevent the users from generating a faulty input file, the option buttons and input boxes which



are in conflict are designed to be inactive. An example is shown in Figure 15, which shows a grayed second row buttons when inviscid flow option is chosen.



Fig. 12. GUI: Opening splash.

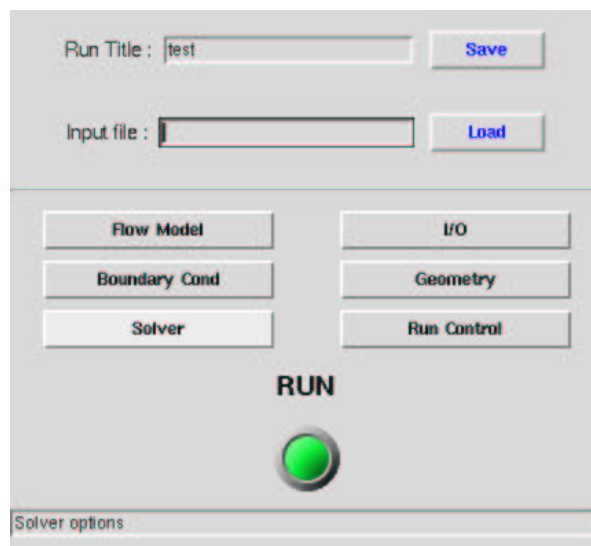


Fig. 13. GUI: Main panel.

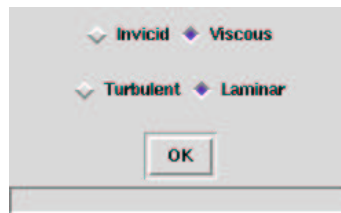


Fig. 14. GUI: Flow model panel.

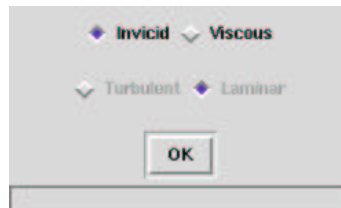


Fig. 15. GUI: Flow model panel. Inviscid option turns off laminar/turbulent options.

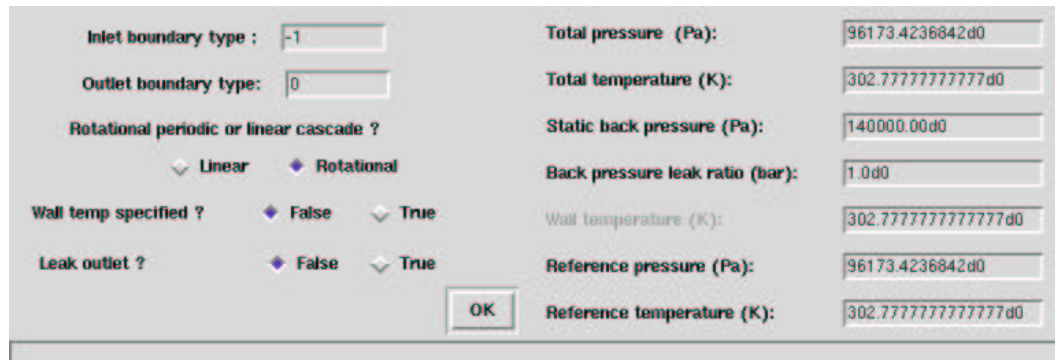


Fig. 16. GUI: Boundary condition panel.

Spatial accuracy :	<input checked="" type="radio"/> First order	<input type="radio"/> Higher order		
R-K integration stages :	<input type="radio"/> 2	<input checked="" type="radio"/> 3	<input type="radio"/> 4	<input type="radio"/> 5
Sub iteration residual smoothing :	<input type="text" value="2"/>			
Graident solution method ?	<input type="radio"/> Green-Gauss	<input checked="" type="radio"/> Least Squares		
CFL :	<input type="text" value="1.0d0"/>			
eps4 :	<input type="text" value="0.0d-0"/>			
dit_time :	<input type="text" value="1"/>			
<input type="button" value="OK"/>				

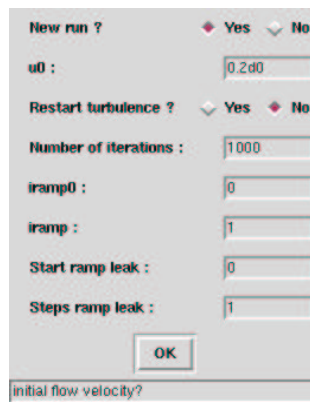
Fig. 17. GUI: Solver control panel.

Write TecPlot output?	<input type="radio"/> False	<input checked="" type="radio"/> True	Gridfile name :	<input type="text" value="test.mesh"/>
Write yplus output?	<input checked="" type="radio"/> False	<input type="radio"/> True	Rerun flowfield input data file:	<input type="text" value="test.0.out"/>
relative V (T) / absolute V (F)?	<input type="radio"/> False	<input checked="" type="radio"/> True	Rerun flowfield output data file:	<input type="text" value="test.1.out"/>
Echo ?	<input checked="" type="radio"/> False	<input type="radio"/> True	Turbulence data input file :	<input type="text" value="fixed.tur"/>
Run with debug ?	<input checked="" type="radio"/> False	<input type="radio"/> True	Temperature data input file:	<input type="text" value="fixed.true"/>
			Steps output compressor parameter :	<input type="text" value="10"/>
<input type="button" value="OK"/>			Steps output residual:	<input type="text" value="1"/>

Fig. 18. GUI: Input/Output control panel.

Scale :	<input type="text" value="1.0d-3"/>
Angular velocity (RPM) :	<input type="text" value="-68478.d0"/>
LE end of rotational body :	<input type="text" value="10.1600001513953d0"/>
<input type="button" value="OK"/>	
<small>Close panel and return to main.</small>	

Fig. 19. GUI: Geometry panel.



The image shows a GUI window titled "Execution control panel". It contains several controls for running a simulation:

- New run ?**: A radio button group with "Yes" selected and "No" unselected.
- u0 :**: A text input field containing "0.2d0".
- Restart turbulence ?**: A radio button group with "Yes" unselected and "No" selected.
- Number of iterations :**: A text input field containing "1000".
- iramp0 :**: A text input field containing "0".
- iramp :**: A text input field containing "1".
- Start ramp leak :**: A text input field containing "0".
- Steps ramp leak :**: A text input field containing "1".
- OK**: A button at the bottom center.

At the bottom left of the window, the text "initial flow velocity?" is visible.

Fig. 20. GUI: Execution control panel.

## CHAPTER IV

## RESULTS

This chapter presents the computational results. In the first section, the results of the hybrid mesh generator are presented.<sup>1</sup> The mesh generator is applied to a turbine blade with an extremely large stagger angle variation to demonstrate that the mesh generation method is capable of producing a hybrid mesh of an acceptable quality even at extreme conditions. The validation cases of the flow solver follow the mesh generation test.

The flow solver in the present study can handle inviscid, laminar, and turbulent flows. As discussed in the previous chapter, the flow solver computes the inviscid flux and the viscous flux separately. Moreover, the  $k-\omega$  turbulence model is implemented in a decoupled manner from the Navier-Stokes equations.

Only the convective flux must be computed for an inviscid flow case whereas both the convective and the viscous flux must be computed for a viscous flow case. A viscous case can be either laminar or turbulent. In the latter case, the  $k-\omega$  turbulence model is used to compute the eddy viscosity. In the laminar case, the eddy viscosity is simply zero. The laminar flow case, therefore, relies on a subset of the flow solver algorithms compared to the turbulent flow case. An inviscid case requires an even smaller subset of the solver algorithms than what is required for a laminar case.

Noting the hierarchical dependency of the solution algorithms, the flow solver is validated in the following three steps. In the first step, the flows are assumed inviscid and the tests are performed to ensure the proper implementation of the

---

<sup>1</sup>Part of this chapter has been previously published in the *AIAA Journal of Propulsion and Power* [43] and copyrighted by the present author and Dr. Paul Cizmas.

Euler solver, of which the important parts include the convective flux computation, the time integration algorithm, and the boundary conditions. The inviscid flow tests include a supersonic vortex flow case and a transonic flow over a bump. It is noted that an exact analytical solution exists for the supersonic vortex.

In the second step of the validation, a laminar boundary layer over a flat plate is solved to demonstrate the proper treatment of the viscous flux. The results are compared to the Blasius solution. In the third step of the validation, a turbulent boundary layer over a flat plate is solved to ensure the proper implementation of the turbulence model. This completes the preliminary test of the flow solver.

In the last section, the flow in a high speed centrifugal compressor is computed. The model geometry of the high speed centrifugal compressor includes not only the front wheel block, but also the backplate region. Based on the combined geometry, the axial loads on the moving wheel are computed at various operating points. The relationship between the leakage flow rates and the backpressure is also predicted.

#### A. Hybrid mesh generation for an axial turbine rotor

The hybrid mesh generation is tested on an axial turbine rotor blade shown in Fig. 21. The variation of the airfoil cross sections at 20% span increments is shown in Fig. 22. Mesh generation for this airfoil is challenging because of the large radial variation in shape and size. What makes it even more difficult is the extreme variation of the stagger angle, which is approximately 88 degrees.

The source mesh is generated at the hub. Figure 24 shows the source mesh where only the unstructured region is displayed. The unstructured portion of the 2-D layer contains 480 nodes, excluding the common nodes between the O-grid and the unstructured region. The mesh is rather coarse and additional nodes are to be

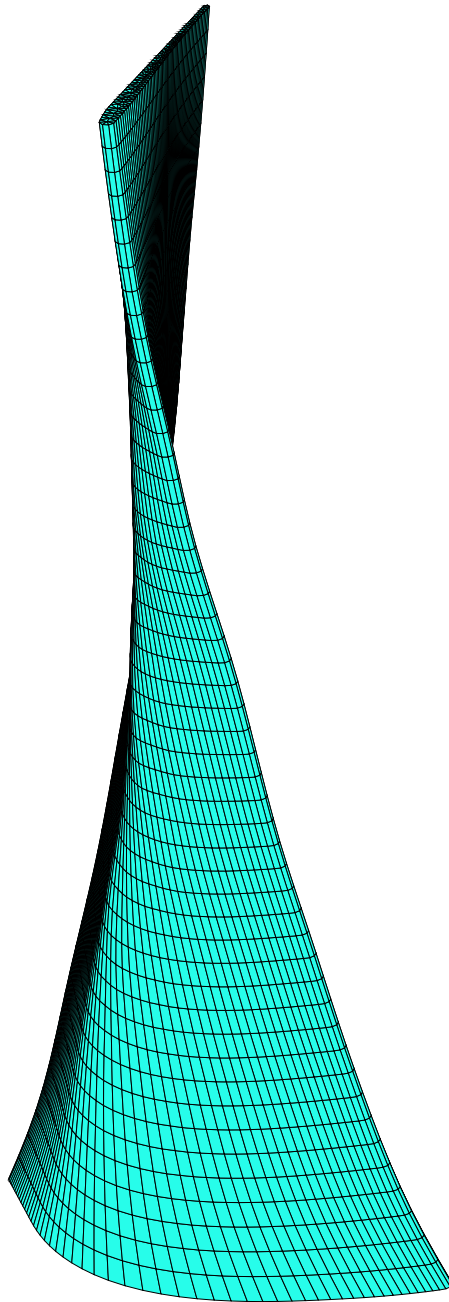


Fig. 21. Test case airfoil.

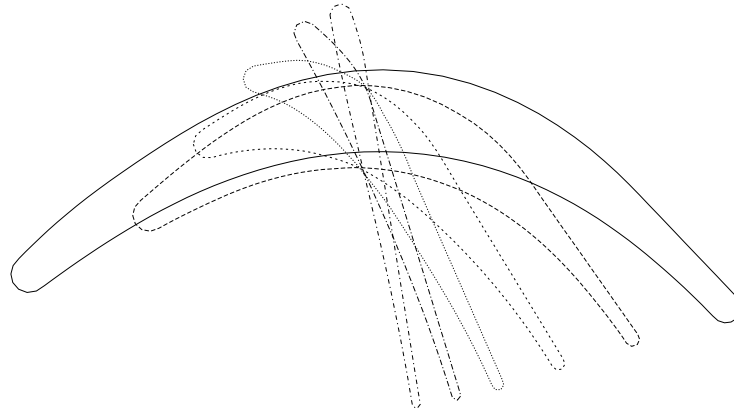


Fig. 22. Variation of airfoil cross sections.

added after the mapping and smoothing procedures are performed. The O-grid has 120 nodes along the airfoil and 10 nodes normal to the airfoil.

Mesh smoothing alone failed to eliminate the inverted cells on the layers close to the tip. These inverted cells resulted due to the significant change of the airfoil profile from hub to tip. Note that in this case the source layer was located at the hub. The optimal connectivity for the source mesh caused too much restriction of the mesh at the tip. Edge swapping was thus necessary to remove the constraints due to the biased connectivity of the source mesh.

The first mapped layer was generated at 10% span from the hub. After mapping and mesh smoothing, edge swapping was done for the hub and the layer at 10% span simultaneously. By advancing the layers by 10% span each time, the connectivity constraints were not as severe as in the initial attempt when the base mesh at the hub was mapped to the target mesh at the tip. This procedure of adding one new



layer each time and applying the edge swapping was repeated until the tip layer was reached. As a result, all the target meshes were successfully mapped and smoothed while connectivity was continuously updated. Note that the 10% span spacing was arbitrarily chosen and this spacing was not the actual spacing between the layers of the final mesh.

The variation in cell size at the interface between the O-grid and the unstructured region was controlled by forcing the unstructured cells on the interface to form equilateral triangles. On the tip layer, the node distribution was not dense enough due to the fact that the area the nodes had to cover increased significantly compared to the hub layer. The abrupt changes in cell size became problematic in the transition region from the O-grid to the unstructured region of the tip layer. To correct this problem, new nodes were added and smoothing and edge swapping were applied. The results are summarized in Tables II and III. Table II shows the improvement in the minimum angle, and Table III shows the improvement in the minimum quality measure. Because the mesh optimization was based on the quality metric  $\tau$  rather than on the minimum angle, the variation of  $\tau$  shown in Table III provides a better image of the mesh improvement than the values of the minimum angle shown in Table II. Note that the minimum angle for mesh smoothing is not defined due to the presence of the inverted cell.

The distribution of the quality measure  $\tau$  as a function of the spanwise location is shown in Fig. 23. Note that  $\tau$  is 1 for an equilateral and 0 for a collapsed triangle. A larger number of cells exhibited a low  $\tau$  on the hub and tip layers than on the layers near the mid-span. Note the similarity of the minimum  $\tau$  distribution for all the layers, which proves that edge swapping was effective in eliminating the dependency of the quality of the target meshes on the source mesh. Mesh smoothing maximized the minimum quality measure for every layer. Edge swapping corrected the initial

Table II. Minimum and maximum angle (MS, ES and NI denote mesh smoothing, edge swapping, and node insertion, respectively).

Case	Min. angle, degree	Max. angle, degree
MS	n/a	179.8
MS + ES	3.5	150.0
MS + ES + NI	11.9	147.3

Table III. Minimum of the quality metric  $\tau$  (MS, ES and NI denote mesh smoothing, edge swapping, and node insertion, respectively).

Case	Minimum $\tau$
Mapping only	-0.6994
MS	-0.0643
MS + ES	0.2137
MS + ES + NI	0.2512

connectivity to an unbiased one that satisfied all the layers.

Figures 25 through 30 show the grid at five spanwise locations. Node insertion removed the abrupt changes of cell size. By adding 400 new nodes, the total number of nodes in the unstructured mesh increased to 880. Additional mesh smoothing and edge swapping increased the minimum angle to 11.9 degrees from 3.5 degrees. The computational time necessary to build a 3-D hybrid mesh containing 106,080 nodes was approximately 5 minutes on a 600 MHz PC running Linux.

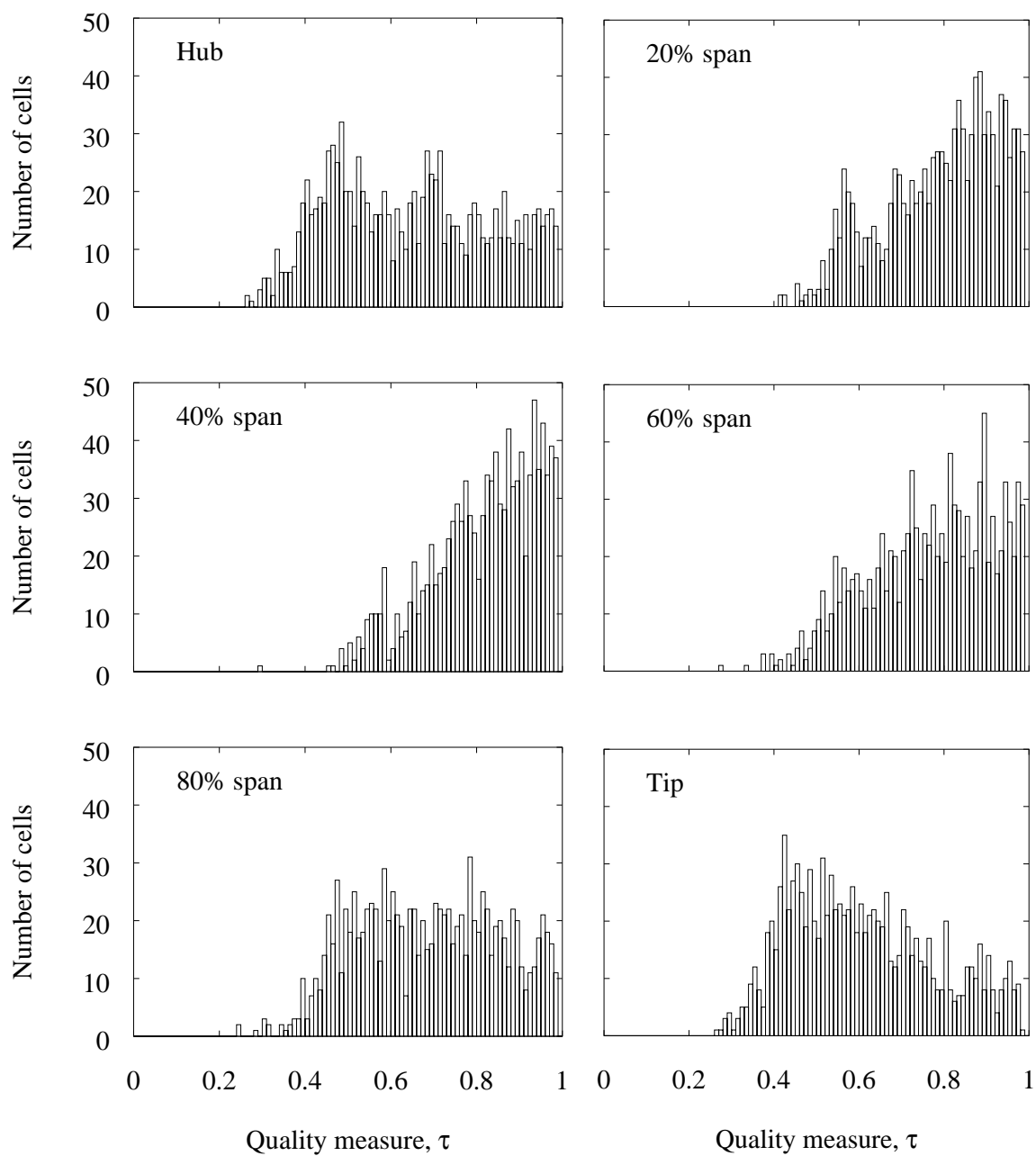


Fig. 23. Number of cells *vs.* quality measure  $\tau$ .

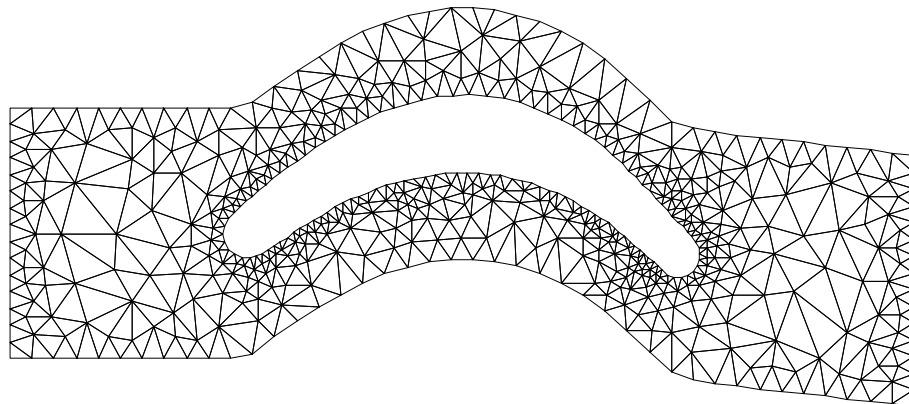


Fig. 24. Source mesh at the hub.

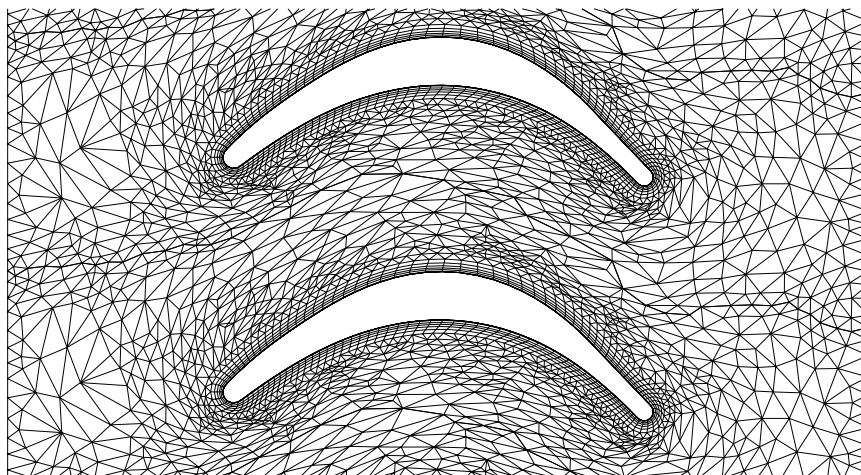


Fig. 25. Hub layer.

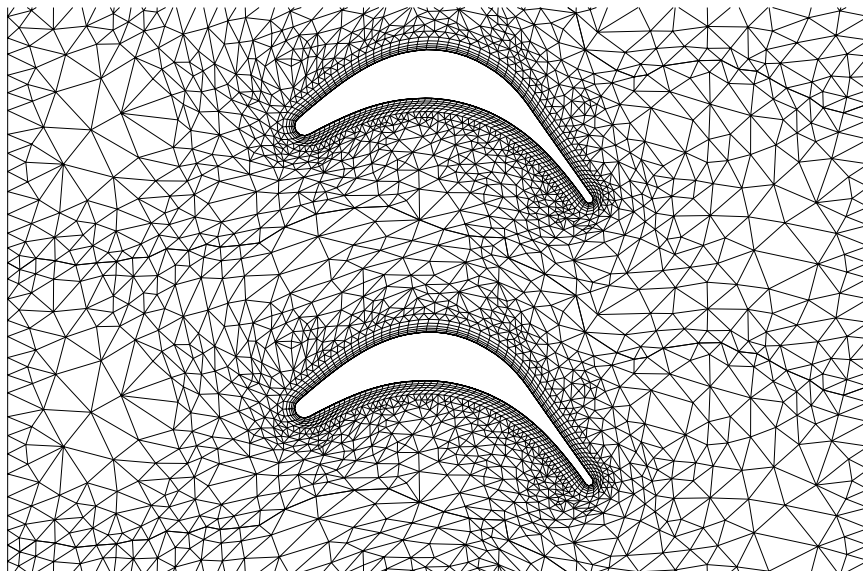


Fig. 26. Layer at 20% span from the hub.

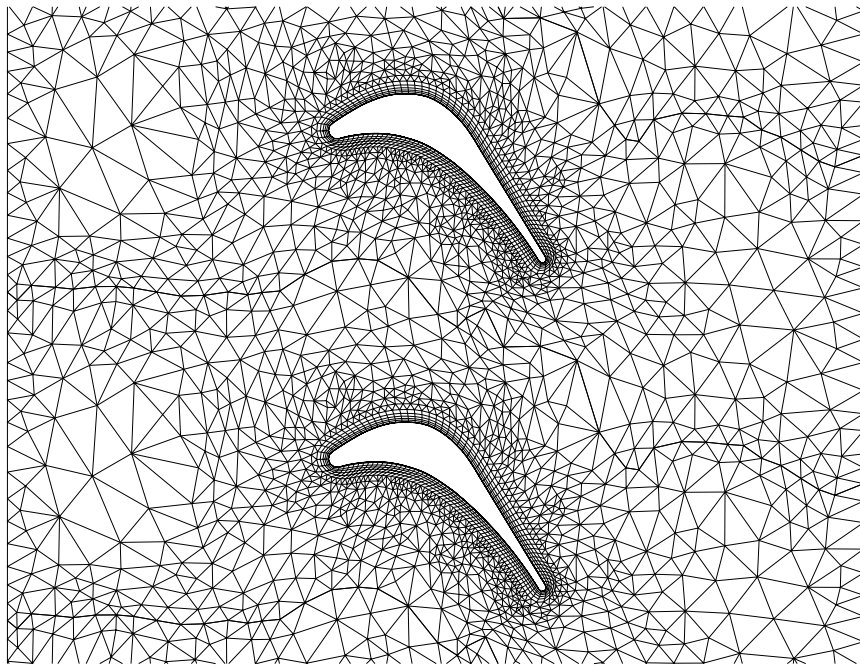


Fig. 27. Layer at 40% span from the hub.

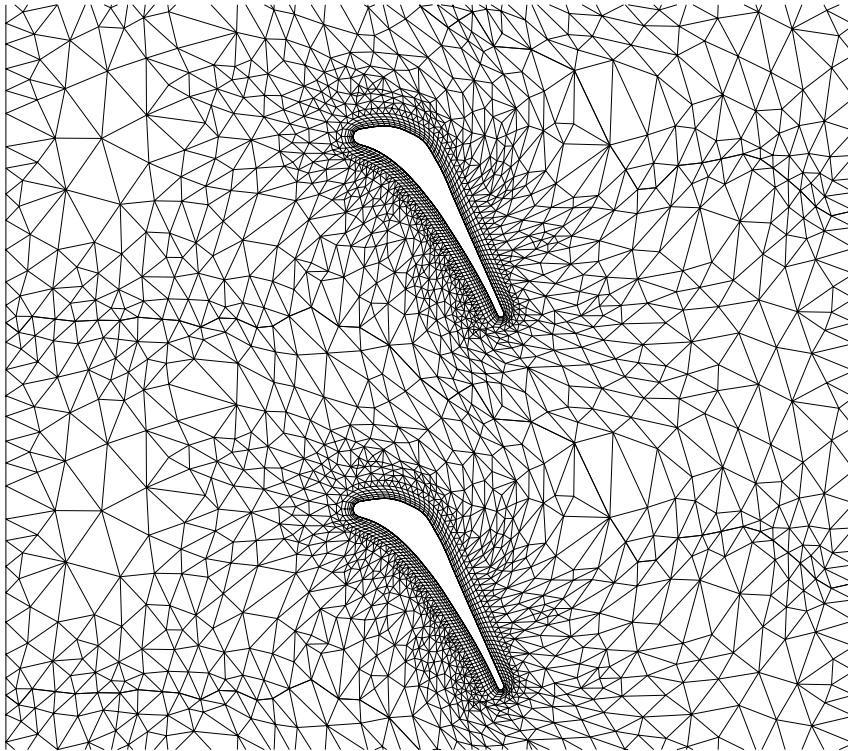


Fig. 28. Layer at 60% span from the hub.

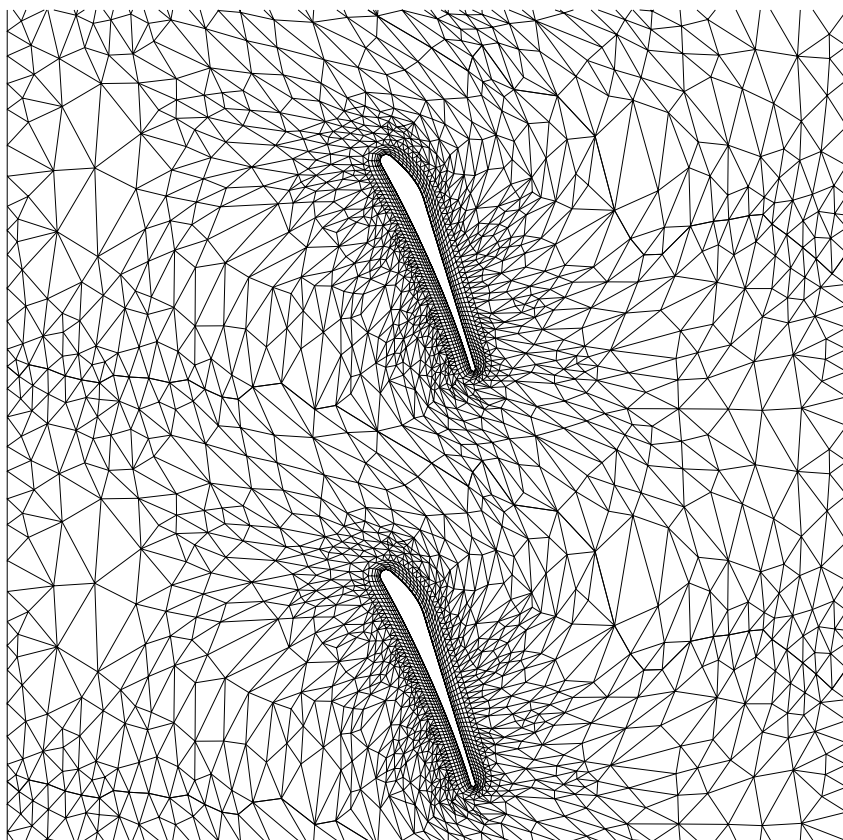


Fig. 29. Layer at 80% span from the hub.



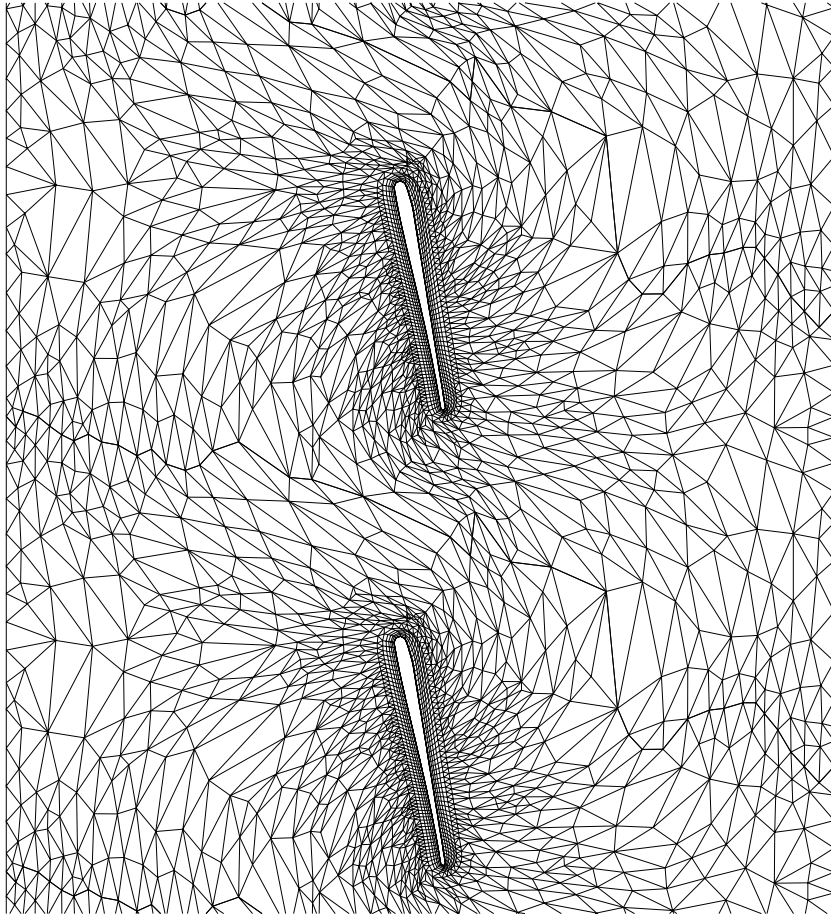


Fig. 30. Tip layer.

## B. Supersonic inviscid vortex flow in a circular channel

This section presents the computational result of the supersonic inviscid vortex flow. The vortex problem is an excellent code validation case for a compressible inviscid flow solver because an analytical solution is known for this case [60, 89]. The quality of the numerical solution can be easily quantified by the relative error using the computed solution and the closed-form solution.

The purposes of the test are twofold. Firstly, this test is intended to show the accuracy of the Euler solver. The numerical algorithms for the viscous and turbulence effects are not covered by the current test case. By validating a limited subset of the code prior to more complex test cases, the modularization of the code development can be achieved and one can assure that further additions for the viscous flows lie on a solid foundation.

Secondly, the other purpose for using this case is to test the implementation of the rotational frame of the reference. The supersonic vortex flow is computed with a fixed frame of reference and with a rotational frame of reference. The correct treatment of the rotational frame of reference plays an important role in computing turbomachinery flows.

The computational domain of the circular channel is defined by two concentric cylinders. An algebraic 3-D structured mesh is generated for the 90 degree section of the circular domain, as shown in Fig. 31. In order to create the volume mesh, 2-D meshes in the radial and circumferential plane are created first. The 2-D mesh from the circumferential plane is then extruded into the direction of the rotational axis to create a single layer volume mesh. The size of the mesh in terms of number of nodes was  $61 \times 31 \times 2$  in the circumferential, radial and meridional direction, respectively. The ratio of the radii of the outer wall to the inner wall is 1.5.

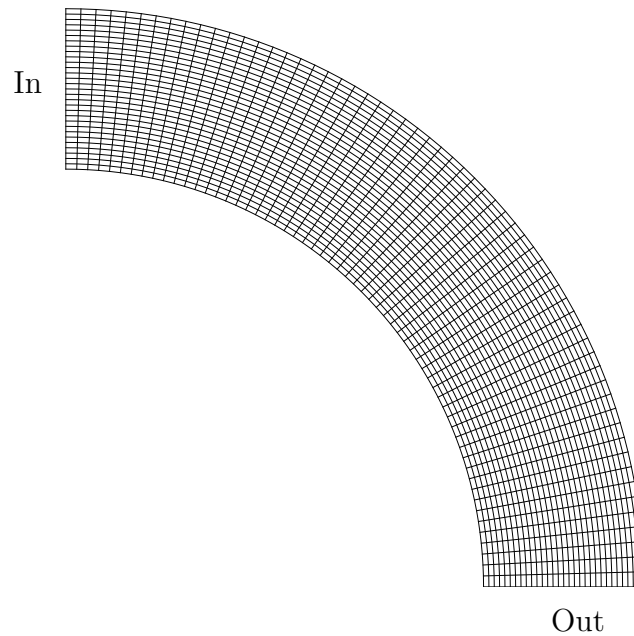


Fig. 31. Structured mesh for supersonic vortex case.

There are two notable features of the flow worth mentioning. Firstly, the supersonic vortex flow is shock-free and a higher-order solver is less likely to suffer from the reduced spatial accuracy. The reduced accuracy is a general symptom of the higher-order methods in the vicinities of shocks. When subjected to shocks, the limiter function of a higher-order solver is triggered and this causes the solution accuracy to be reduced to first order. Since the supersonic vortex flow does not form any shocks, a properly implemented higher-order solver should not degrade its spatial accuracy to first-order. For this reason, the numerical solutions are expected to be of the advertised spatial accuracy of the flow solver.

Secondly, the boundary conditions are considerably simpler to implement for the supersonic vortex case since the flow is supersonic both at the inlet and the outlet. In the supersonic cases, the downstream flow state is fully determined by the upstream condition at the adjacent location. Specifically, the far upstream condition

can be extrapolated to the interior side of the inlet boundary. The outlet boundary conditions are determined by the solution at the interior side of the outlet boundary. On the inner and outer walls, the slip boundary conditions are applied.

The exact solution of the supersonic compressible vortex [60, 89] depends on the radial location,  $r = \sqrt{x^2 + y^2}$ , and the inlet Mach number at the inner wall location,  $M_i$ . The subscript  $i$  in  $M_i$  denotes the inner arc. The radial and axial velocity components of the exact solution are zero and only the circumferential velocity is non-zero. The velocity fields in the Cartesian coordinates are defined as

$$u = U \sin \theta \quad (4.1)$$

$$v = -U \cos \theta \quad (4.2)$$

$$w = 0 \quad (4.3)$$

$$U = U_i \frac{r_i}{r} \quad (4.4)$$

$$\theta = \tan^{-1}(y/x). \quad (4.5)$$

It is convenient to define a temperature ratio

$$\frac{T}{T_i}(\gamma, M_i, r_i, r) = 1 + \frac{\gamma - 1}{2} M_i^2 (1 - \xi) \quad (4.6)$$

where  $\xi$  is the ratio of the radial locations squared

$$\xi = (r_i/r)^2. \quad (4.7)$$

The static pressure and the density follow the isentropic relationship and they are defined as

$$\frac{p}{p_i} = \left( \frac{T}{T_i} \right)^{\gamma/(\gamma-1)} \quad (4.8)$$

$$\frac{\rho}{\rho_i} = \left( \frac{T}{T_i} \right)^{1/(\gamma-1)}. \quad (4.9)$$

where  $\rho_i$  and  $p_i$  are the density and the pressure on the inner arc, respectively. Both the pressure and density are higher in the outer radial location than the inner region if  $\gamma > 1$ . The magnitude of the velocity is not dependent on the radial location.

The initial flow conditions are specified by adding a 10% pressure perturbation to the exact solution at few selected interior nodes. The local pressure disturbances increase the entropy locally. The concentrated entropy spikes travel downstream at the local convective velocities, as the entropy modes of the Euler equations should do. The steady solutions are obtained by continuing the iterations until all the disturbances reach the outlet and move away from the computational domain through the outlet boundary.

Figure 32 shows the pressure contour plot of the constant reconstruction case, which yields first-order spatial accuracy. The computation is made with a fixed mesh. The first-order solution suffers from excessive numerical diffusion as the uneven gaps between the contour lines indicate the deviation from the exact solution. The exact pressure is a function of radius. The gaps between adjacent contour lines, therefore, should remain the same along the arc.

The difference between the computed solution and the exact solution is reduced significantly when the linear reconstruction is employed. The pressure plot for the higher-order case is shown in Fig. 33. The higher-order solution is obtained on a rotational frame of reference. The mesh is assumed to rotate in the clock-wise direction. The rotational speed is set in such a way that the relative velocity on the inner wall is 50 percent of the absolute velocity. The linear reconstruction case produces a solution of superior quality compared to the constant reconstruction case.

The relative error of pressure for the constant reconstruction case is shown in Fig. 34. The relative error is defined as the normalized difference between the computed value and the exact solution. The difference is divided by the exact solution to

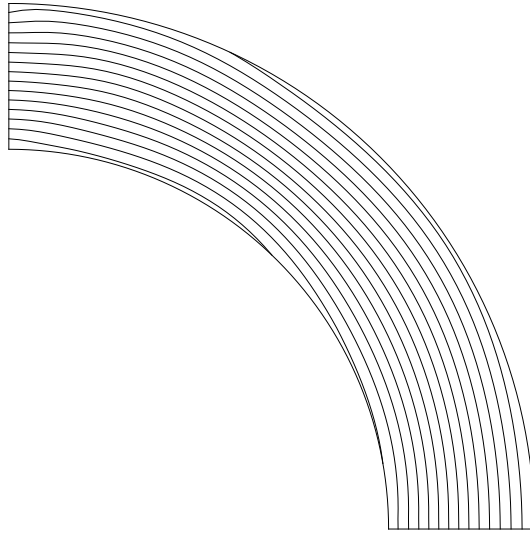


Fig. 32. Pressure contour of constant reconstruction case on a fixed frame of reference.

$$U_i = 2.25, \rho_i = 1, p_i = 1/\gamma, r_i = 1, \text{ and } r_o = 1.384.$$

normalize the error. The 1-norm of the relative error for the constant reconstruction case is 8.72%. The 2-norm of pressure error is less than  $5.12 \times 10^{-3}\%$  for the linear reconstruction case.

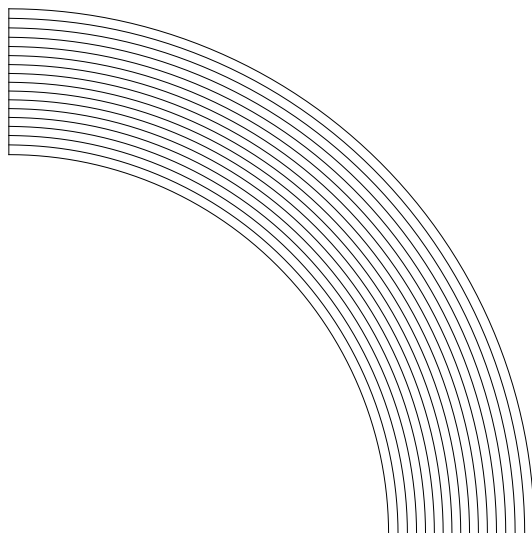


Fig. 33. Pressure contour of linear reconstruction case on a rotating frame of reference.

$U_i = 2.25$ ,  $\rho_i = 1$ ,  $p_i = 1/\gamma$ ,  $r_i = 1$ , and  $r_o = 1.384$ .

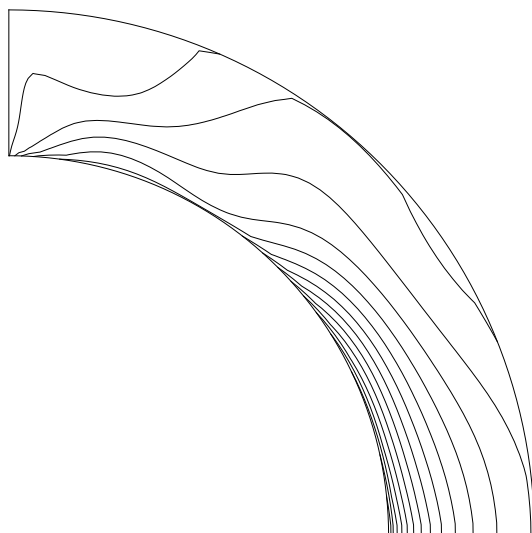


Fig. 34. Pressure error of constant reconstruction case.

### C. Transonic channel flow over a bump

This section presents the simulation of a transonic inviscid channel flow over a circular bump. The flow is mostly subsonic except for the supersonic region next to the bump. The purpose of the test is to evaluate whether the flow solver is capable of capturing the shock correctly.

Figure 35 shows the schematic of the flow over a bump and the symbols to designate the locations of interests. The flow runs from left to right in positive  $x$ -direction. The domain stretches from  $x = -2$  to  $x = 3$  and a circular bump of a unit width is placed at  $x = 0$ . The height of the bump is 0.042 and the channel height is 2.037. The circular bump is uniquely defined by the three points at  $(-0.5, 0)$ ,  $(0, 0.042)$ , and  $(0.5, 0)$ . Since the test flow is essentially two-dimensional, a 3-D mesh with periodic condition in  $z$  direction is used for the current 3-D solver.

The flow is super-critical, *i.e.*, inlet velocity is high enough that the subsonic flow becomes supersonic as it climbs over the bump. The supersonic region is attached to the bump and it does not extend to the upper wall of the channel such that the supersonic region is limited to a closed pocket about the bump. As the flow leaves the bump, it creates a shock on the downstream side of bump.

The subsonic inlet boundary conditions are determined by four upstream conditions and one condition from the interior. Specifically, the boundary conditions are determined by the total pressure, the total temperature, and two flow angles. The remaining condition is determined using an upstream traveling Riemann invariant from interior of the computational domain. The outlet boundary conditions are determined by four interior states and a static pressure from the downstream side. The flow is inviscid and thus the slip boundary condition is enforced on the solid walls.

The inlet Mach number is  $M_\infty = 0.85$  and the flow is in  $x$  direction. The static



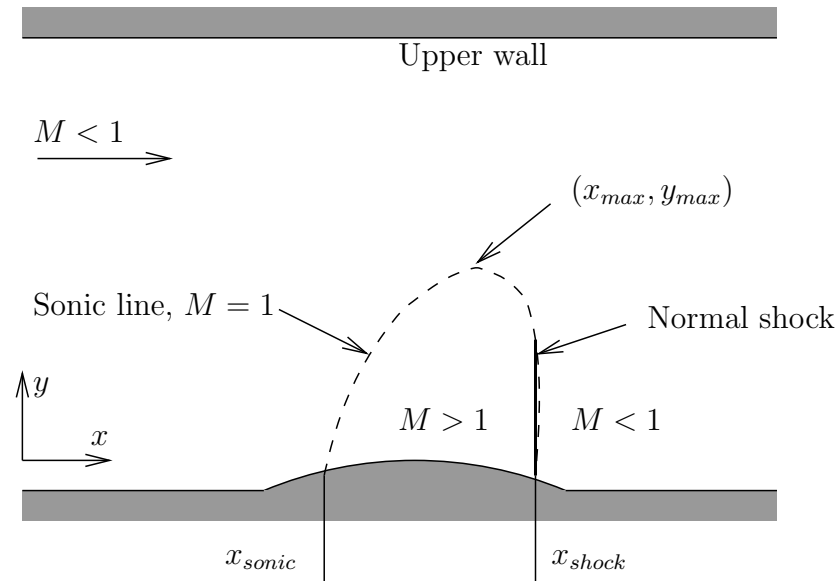


Fig. 35. Schematic of transonic flow over a bump.

pressure at the outlet is determined from the stagnation pressure. No stagnation pressure drop is assumed between the inlet and the outlet. Given the stagnation pressure at inlet,  $p^*$ , and the inlet Mach number,  $M_\infty$ , the outlet static pressure,  $p_o$ , is

$$p_o = p^* \left( 1 + \frac{\gamma - 1}{2} M_\infty^2 \right)^{\gamma/(1-\gamma)}. \quad (4.10)$$

The 3-D structured mesh has  $(71 \times 31 \times 2)$  nodes. The flow is considered periodic in the  $z$  direction to reduce the computational effort. The mesh points are packed close to the bump to capture the detailed flow features, as shown in Fig. 36.

The computed results are compared against various results by other researchers [90, 91, 92]. The results are summarized in Table IV. The locations and the Mach numbers associated with the sonic line about the supersonic pocket are also tabulated in the table. In Table IV,  $x_{sonic}$  and  $x_{shock}$  denote the  $x$  coordinates where the sonic line intersects the lower wall.  $x_{max}$  and  $y_{max}$  denote the peak of the sonic line.

Table IV also shows  $M_1$  and  $M_2$  which are the upstream and downstream Mach

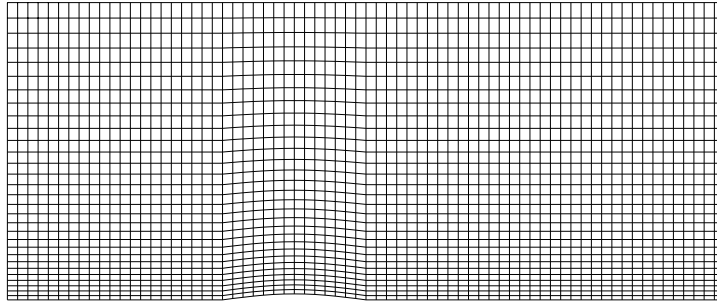


Fig. 36.  $(71 \times 31 \times 2)$  mesh for transonic flow over a bump.

numbers, respectively, across the normal shock.  $M_1$  and  $M_2$  are extracted from the numerical solution on the bump. The change of slope on the bump at the shock location,  $x_{shock}$ , is very small such that the isentropic normal shock relation is applicable. The Mach number downstream from the normal shock,  $M_{2*}$ , can be calculated as a function of the upstream Mach number,  $M_1$ , [93], p. 67,

$$M_{2*}^2 = \frac{2 + (\gamma - 1)M_1^2}{2\gamma M_1^2 - (\gamma - 1)}. \quad (4.11)$$

The error tabulated in the last column of Table IV is defined as the relative error between the computed Mach number  $M_2$  and the analytical value  $M_{2*}$ ,  $error = (M_2 - M_{2*})/M_{2*}$ .

As shown in Table IV, the current results compare well to the existing results in the literature. Figure 37 shows an iso-Mach contour plot, where one can see that the shock is captured clearly. Another clear feature observed in the figure is an iso-Mach line extending toward outlet. This iso-Mach line divides the outlet region into a high pressure upper region and a low pressure low region despite that a uniform backpressure is specified.

The flow near the upper wall experiences a slight compression and expansion due to the bump as displayed by the series of iso-Mach lines extended from the bump

Table IV. Comparison of transonic channel flow.

	$x_{sonic}$	$x_{shock}$	$x_{max}$	$y_{max}$	$M_1$	$M_2$	$M_{2*}$	$M_2\%Error$
Current work	-0.19	0.33	0.18	0.80	1.31	0.772	0.781	1.15%
Kermani & Plett	-0.18	0.31	0.16	0.79	1.26	0.822	0.807	1.86 %
Veulliot & Viviand	-0.19	0.33	0.20	0.82	n/a	n/a	n/a	n/a
Deconinck & Hirsch	-0.2	0.31	0.17	0.69	n/a	n/a	n/a	n/a
Lerat & Sides	-0.18	0.31	0.16	0.80	n/a	n/a	n/a	n/a
Manna	n/a	n/a	n/a	n/a	1.31	0.77	0.781	1.41%

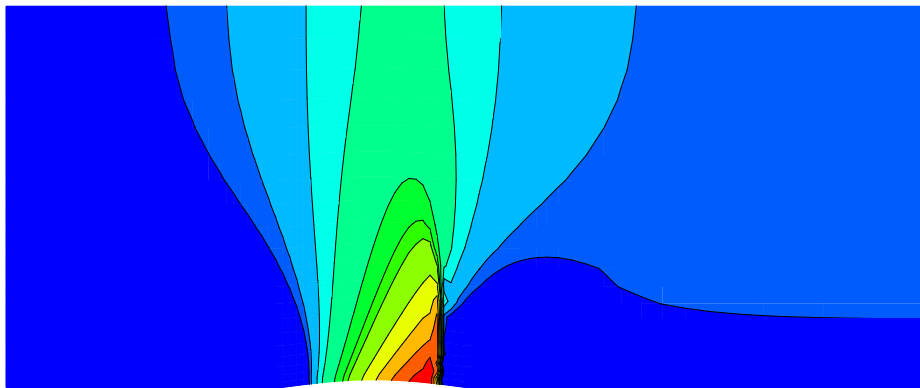


Fig. 37. Iso-Mach contour plot of transonic flow over a bump.

toward the upper walls. In contrast, the flow by the lower wall experiences the shock, which increases the entropy. The difference of Mach numbers arises from the energy loss, *i.e.*, total pressure drop across the shock.

The results from the supersonic vortex case in the previous section and the transonic channel flow test in the present section indicate that the solution algorithms are implemented right. This completes the validation cases for the inviscid flow cases. In the next section, the viscous flux computation will be tested.

#### D. Laminar boundary layer flow over a semi-infinite flat plate

This section presents the numerical test of a laminar flow over a semi-infinite flat plate. Having shown the accuracy of the solver for the inviscid flux computation in the previous two sections, this viscous flow test is intended to verify the accuracy of the numerical viscous flux.

The numerical solution of the full Navier-Stokes equations is compared to the Blasius boundary layer solution. The Blasius solution is based on an incompressible flow assumption. The freestream speed of the simulation, therefore, should be fairly low to minimize the compressibility effects. On the other hand, extremely slow flow speed not only reduces the convergence rate of a compressible flow solver, but also the accuracy of the solution. For these reasons, the freestream speed is set to  $U_\infty = 0.20$ , which lies within the incompressible flow regime, but it is not too low to affect the performance of the compressible solver.

In simulating the boundary layer flow, two different approaches can be used. First option is to consider the flow on the flat plate only. This requires one to impose a correct solution at the inlet boundary. To do so, one can preselect the Reynolds number  $Re_x$  at the inlet. The velocity profile of the Blasius solution is then scaled to the selected  $Re_x$  at the inlet of the domain. If a solution algorithm is properly implemented, the solution at downstream locations should be very similar to the Blasius solution.

In the second option, the domain includes both the freestream region upstream of the flat plate and the plate region. The constant freestream condition is imposed far upstream. In this work, the second approach is employed since the setup can be easily extended to the turbulent boundary layer test. The schematic of the setup is shown in Fig. 38.

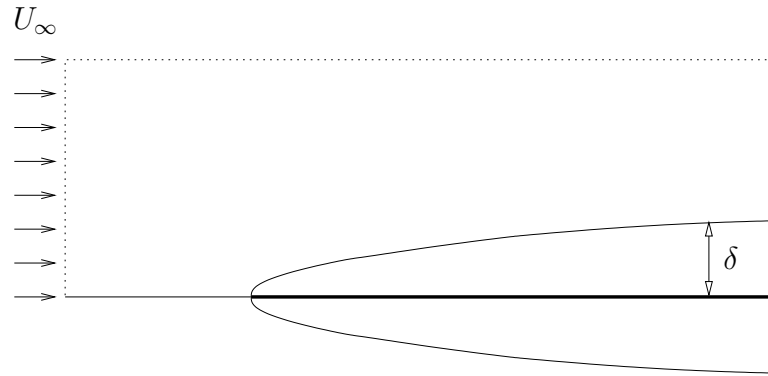


Fig. 38. Schematic of setup of Blasius boundary layer.

Constant freestream conditions are initially applied on the entire domain. The subsonic inflow and outflow boundary conditions are applied on the inlet and outlet boundaries, respectively. The symmetry plane boundary condition is applied along the lower boundary from the inlet up to the leading edge of the plate. The non-slip boundary condition is applied on the plate. Constant static pressure is specified on the upper boundary. The velocity component on the upper boundary in the direction normal to the plate is extrapolated from the interior of the computational domain. In other words, the upper boundary is treated as a subsonic outlet boundary. Zero pressure gradient normal to the wall and adiabatic wall conditions are also imposed on the plate.

The Blasius solution was computed for the comparison with the computed result of the Navier-Stokes equations. The Blasius boundary layer equation does not have a closed-form solution even though the Blasius boundary layer equation is simpler than the Navier-Stokes equation. Excluding the region close to the leading edge, the

boundary layer profile is self-similar. Introducing the similarity variable  $\eta$  defined as

$$\eta = y\sqrt{\frac{U_\infty}{\nu x}} \quad (4.12)$$

where  $U_\infty$  is the freestream flow speed,  $\nu$  is the kinematic viscosity, and  $x$  is the distance from the leading edge of the plate. Using the similarity variable, the Prandtl's boundary equation [94], pp. 232-234, without pressure gradient in the flow direction is reduced to the Blasius boundary equation,

$$\frac{d^3 f}{d\eta^3} + \frac{1}{2}f \frac{d^2 f}{d\eta^2} = 0, \quad (4.13)$$

which is a third order ordinary differential equation of a nondimensional stream function  $f = \psi(x, y)/\sqrt{U\nu x}$ , where  $\psi$  is the stream function. The boundary conditions for the nondimensional stream function are [94], p. 233,

$$f|_{\eta=0} = 0, \quad \frac{df}{d\eta}|_{\eta=0} = 0, \quad \frac{df}{d\eta}|_{\eta \rightarrow \infty} \rightarrow 1. \quad (4.14)$$

Once the values of the stream function,  $f$ , are calculated, the velocities can be found as

$$u = U \frac{df}{d\eta} \quad (4.15)$$

$$v = \frac{U}{2Re_x^{1/2}} \left( \eta \frac{df}{d\eta} - f \right). \quad (4.16)$$

Detailed solution procedures and tabulated solutions can be found in Schlichting [95] and White [94].

The boundary layer thickness is defined as the height at which u-velocity becomes 99% of the freestream value. From the tabulated results, this corresponds to  $\eta \approx 5.0$ . Using the definition of the similarity variable  $\eta$  in Eq. (4.12), the boundary thickness

$\delta$  can be estimated from the Blasius solution as

$$\delta = 5.0 \sqrt{\frac{\nu x}{U}}. \quad (4.17)$$

The height of the mesh used to solve the Navier-Stokes equations is set to approximately 5 times of the estimated boundary thickness  $\delta$ . A closeup view of the mesh used for the test is shown in Fig. 39. The mesh is constructed using  $(41 \times 41 \times 2)$  nodes. The length of the computational domain is set up in such a way that Reynolds number based on the distance from the leading edge to the end of the plate, freestream velocity and viscosity is approximately  $Re_x = 1.5 \cdot 10^5$ . The height of the domain is set to 5 times that of the estimated boundary thickness in Eq. (4.17). The 60% of the nodes in the normal direction to the wall are distributed from the wall at an equal spacing in the estimated boundary layer thickness and the remaining 40% of nodes are distributed in the remaining region at an increasing gap. A simple algebraic stretching is also used to cluster nodes near the leading edge in the parallel direction to the wall.

Both the first-order and higher-order solutions are computed. The velocity components in the  $x$ -direction are sampled at a location which corresponds to  $Re_x \approx 1.2 \times 10^5$ . The first-order solution can be seen in Fig. 40. The difference between the computed result and the Blasius solution is noticeable. The error is large in the region where  $u$  approaches the freestream flow speed. The variation of  $u$  with respect to  $\eta$  is close to linear from the wall up to  $\eta = 2$  and the error due to the constant reconstruction is not pronounced. The variation of  $u$  is no longer linear above  $\eta = 2$  and the constant reconstruction fails to capture the velocity variation.

In contrast to the first-order solution, the second-order solution shown in Fig. 41 shows an excellent agreement between the computed Navier-Stokes solution and the Blasius solution. A slight overshoot of velocity beyond  $\eta > 5$  is observed. The skin

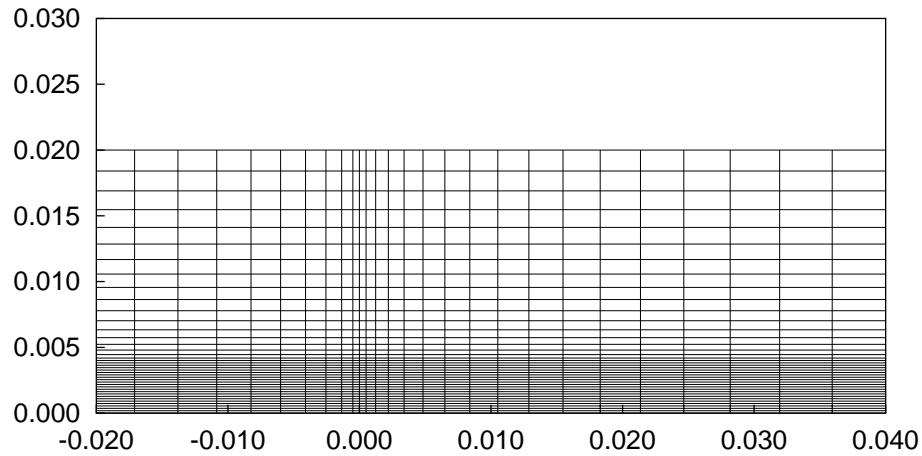


Fig. 39. Closeup view of mesh for Blasius boundary layer. The plate leading edge is placed at  $x = 0$ .

friction coefficients of the first-order case and the second-order case are also compared.

From the Blasius solution, the skin friction coefficient is

$$C_f = \frac{\tau_w}{\frac{1}{2}\rho U^2} = \frac{0.664}{\sqrt{Re_x}}. \quad (4.18)$$

The comparison between the skin friction coefficients of the Navier-Stokes solution and the Blasius solution is given in Figure 42. The results of the constant and linear reconstruction agree with the Blasius solution closely for most of the range of  $Re_x$  except at the leading edge region where the Blasius boundary layer equation hypothesis does not hold and the predicted  $C_f$  approaches infinity. As mentioned above, both the constant and linear reconstruction cases capture the slope of  $u$  variation next to the wall very close to the Blasius solution.



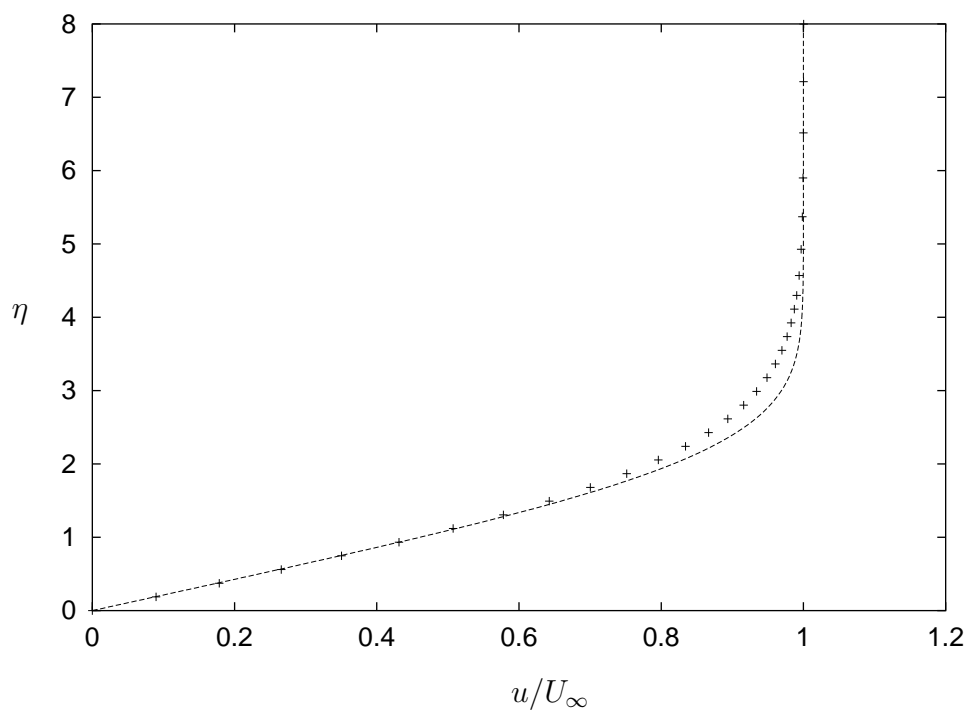


Fig. 40. Constant reconstruction case. U velocity profile of Blasius boundary layer,  $Re_x = 120000$ . Line denotes the Blasius solution and symbols denote the computed value.

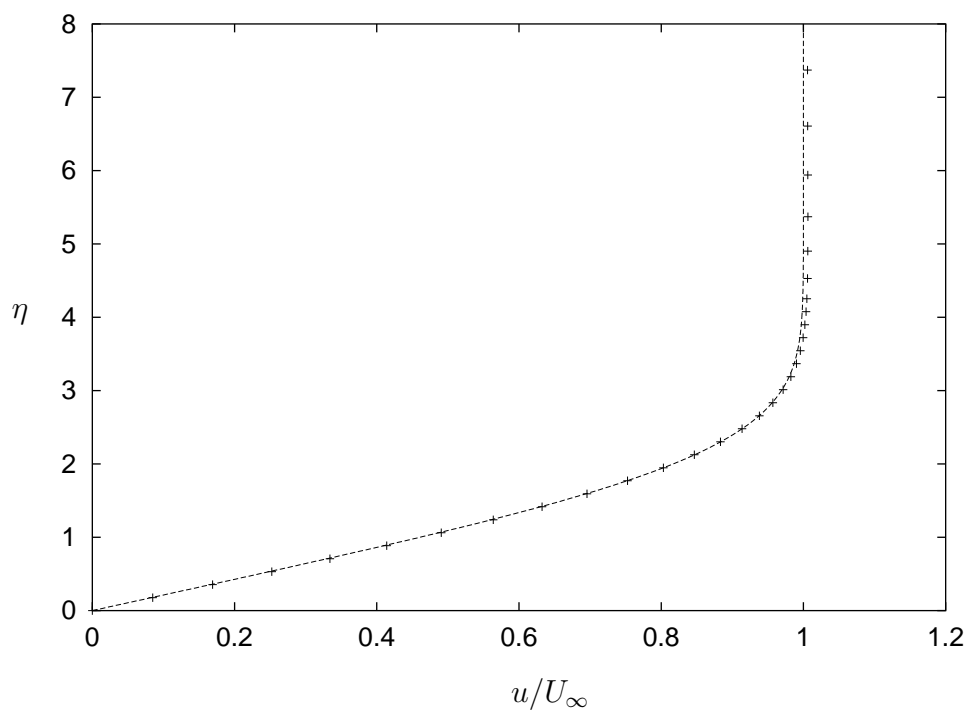


Fig. 41. Linear reconstruction case. U velocity profile of Blasius boundary layer,  $Re_x = 120000$ . Line denotes the Blasius solution and symbols denote the computed value.

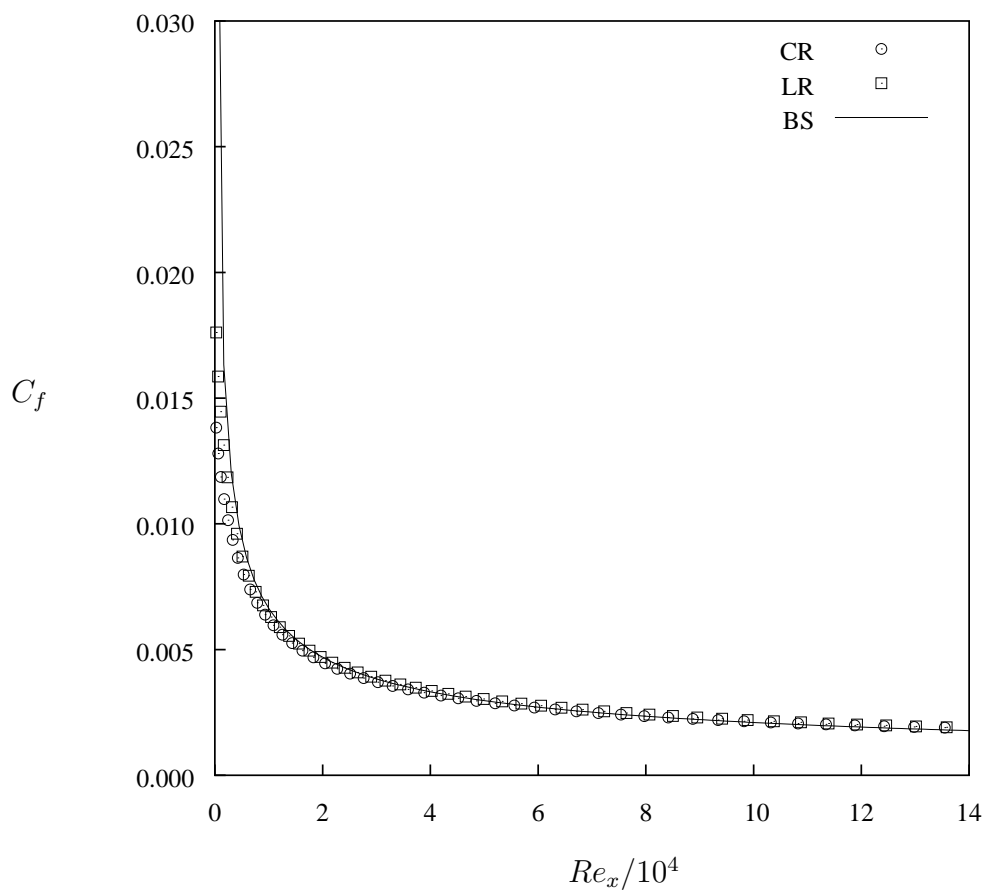


Fig. 42. Comparison of skin friction coefficients. CR=Constant Reconstruction, LR=Linear Reconstruction, and BS=Blasius Solution.

### E. Turbulent boundary layer flow over a semi-infinite flat plate

This section presents the numerical test of a turbulent flow over a semi-infinite flat plate. This turbulent flow test is used to verify the proper implementation of the  $k-\omega$  turbulence model [27, 75]. This test is well suited for code validation since turbulence models are well tailored for this type of problem [96]. The computed results are compared against experimental data from Wieghardt and Tillmann [97, 98].

The turbulent flow simulation is set up in a similar manner as in the previous laminar boundary layer flow. The mesh used in the simulation is constructed using  $(121 \times 91 \times 2)$  nodes as shown in Fig. 43. The distribution of nodes in the normal direction to the wall is determined using an exponential function in an attempt to obtain an even distribution of nodes in the log scale.

The length of the computational domain is set up in such a way that Reynolds number based on the distance from the leading edge to the end of the plate, freestream velocity and viscosity is approximately  $Re_x = 2 \cdot 10^7$ . The height of the domain is set to 10 times that of the estimated boundary thickness. The estimation is based on the Prandtl's one-seventh power-law profile [94] which is

$$\delta \approx \frac{0.16x}{Re_x^{1/7}} \quad (4.19)$$

where  $Re_x = 2 \cdot 10^7$  is used.

Mesh points are clustered close to the wall such that the  $y^+$  number is approximately 1. The  $y^+$  number is defined as

$$y^+ = \frac{u^* \Delta y_1}{\nu} \quad (4.20)$$

where  $\Delta y_1$  is the grid spacing between the wall to the first interior node in the direction

normal to the wall. The frictional velocity  $u^*$  is defined as

$$u^* = \sqrt{\frac{\tau_w}{\rho}} \quad (4.21)$$

where  $\tau_w$  is the wall shear stress, *i.e.*,  $\tau_w = \mu \partial u / \partial y|_{y=0}$ .

In the  $k - \omega$  model, non-zero wall boundary condition for  $\omega$  and the freestream conditions of both  $k$  and  $\omega$  must be specified. The freestream conditions can be defined in terms of turbulence intensity [95],

$$Tu = \frac{\sqrt{\frac{1}{3}(\overline{u'^2} + \overline{v'^2} + \overline{w'^2})}}{U_\infty} = \frac{\sqrt{\frac{2}{3}k}}{U_\infty} \quad (4.22)$$

and the eddy viscosity in the  $k - \omega$  model is defined as [27]

$$\nu_t = \frac{k}{\omega}. \quad (4.23)$$

The freestream velocity, the turbulence intensity and the freestream eddy viscosity are initially specified as  $U_\infty = 0.5$ ,  $Tu = 0.001$  and  $\nu_t = 0.001\nu$ . The freestream value of  $k$  is then computed by Eq. (4.22) with a known freestream velocity  $U_\infty$  and a specified turbulence intensity in the upstream region. Using Eq. (4.23), the freestream value of  $\omega$  can be readily computed.

The freestream conditions are specified on the upper boundary. The outlet boundary conditions are extrapolated from the interior. On the wall boundary, the velocity fluctuations are zero and the turbulent kinetic energy  $k$  is zero on the wall. The boundary condition for  $\omega$  is determined by the rough-surface boundary approximation.

Numerical tests by Hellsten [31, 99] and Thivet et al. [40] note the sensitivity of  $C_f$  to grid spacing. The skin friction coefficient,  $C_f$ , deviates slightly from the experimental data as the grid spacing near the wall varies. Thivet et al. [40] reports

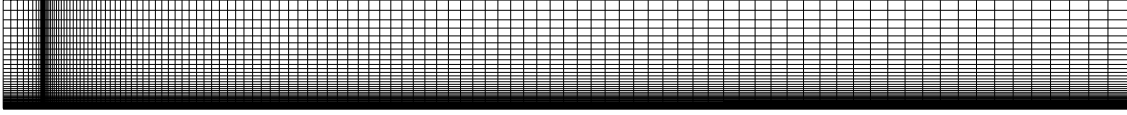


Fig. 43. Mesh for turbulent flow over a semi-infinite plate.

that the difference of computed  $C_f$  values using the smooth wall limit and the rough wall boundary condition becomes less than 1% if  $y_1^+ \approx 0.5k_s^+$ . The last condition can be satisfied by controlling the grid spacing  $\Delta y_1$  and the sand grain height  $k_s$ . The value of  $k_s$ , however, should be small enough that  $k_s^+$  is less than 5 if a rough-surface boundary condition is used for a smooth surface.

The tangential velocity of the  $k - \omega$  modeled solutions is shown in Fig. 44. The solid line represents the computed velocity and the plus symbols denote the experimental data reported by Wieghardt and Tillmann [97]. The dashed line shows the linear relationship between  $u^+$  and  $y^+$ . The experimental data was digitized by Yoder [100] and the digitized data is used in Fig. 44.

Figure 44 shows that the computed velocity profile in the linear sublayer,  $y^+ < 5$ , matches the experimental linear velocity profile well. The log-law region solution also compares fairly well to experimental data. The outer layer data are also in excellent agreement with the experimental data. A small mismatch of the peak  $u^+$  is due to the difference in the sampling location. The experimental data are sampled at  $Re_x = 1.0643 \times 10^6$  whereas the current results are sampled at  $Re_x = 1.0498 \times 10^6$ .

Figure 45 shows the distribution of skin friction coefficients with the  $k - \omega$  model. The experimental results are obtained from the Yoder's digitized data [100, 97]. The agreement between the computed and experimental data is favorable except in the low  $Re_x$  region.

This section presented the turbulent boundary layer flow. The results using the

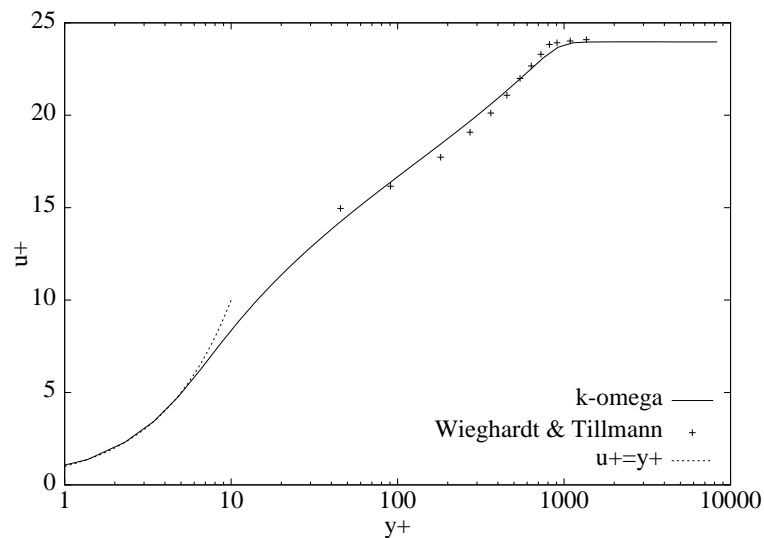


Fig. 44. Tangential velocity profile for the  $k - \omega$  model. The computed velocities are sampled at  $Re_x = 1.0498 \times 10^6$ . The experiment data by Wieghardt & Tillman result is sampled at  $Re_x = 1.0643 \times 10^6$ .

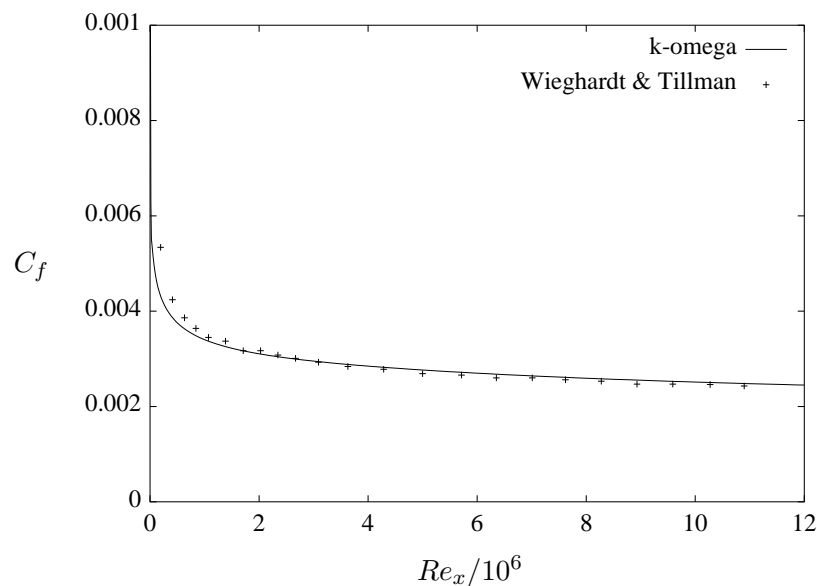


Fig. 45. Comparison of the computed skin friction coefficients,  $C_f$ , and the experiment data by Wieghardt and Tillmann. The computed result is based on the  $k - \omega$  model.

$k - \omega$  model matched the experimental results closely, insuring that the model was implemented correctly.

This concludes the code validation steps. The last four numerical tests, ranging from a supersonic vortex flow to a turbulent boundary layer flow, demonstrated that the solution algorithms were well implemented.

#### F. Honeywell high speed centrifugal compressor

This section presents the results of the parametric study of the axial thrust load of a centrifugal compressor. The compressor of a Honeywell turbocharger was used herein for the analysis. The values of the axial thrust were obtained through the numerical simulation of the flow in the impeller and the leakage flow through the gap between the back (unbladed) side of the wheel and the back plate. Because of the periodicity, the flow was calculated in only one passage of the impeller and the corresponding back side of the wheel. The numerical simulation provided the flow fields on both sides of the impeller. The integration of the pressure fields on the two sides of the wheel yielded the axial thrust load on the wheel.

The impeller inlet diameter,  $d_{1,s}$ , is 70 *mm* and the exit diameter,  $d_2$ , is 94.0 *mm*. The hub-to-tip ratio,  $d_{1,h}/d_2$ , is 0.245 and the impeller exit width ratio,  $b_2/d_2$ , is 0.069. The impeller tip clearance is constant along the blade and equal to 0.75 *mm*. The impeller has 6 full blades and 6 splitter blades. The compressor has a vaneless diffuser.

Two modifications were made with respect to the impeller geometry. First, to simplify the grid generation, the compressor was modeled having 12 full blades. Second, the impeller tip clearance was modeled to be 0.25 *mm*. The impeller is shown in Fig. 46. The entire computational domain is shown in Fig. 47 and a meridional



section through the grid on the back side of the impeller is shown in Fig. 48.



Fig. 46. Detail of the Honeywell centrifugal compressor impeller geometry.

The inlet flow in the compressor was axial. The inlet temperature was 302.4 deg Kelvin and the inlet stagnation pressure was 96,173  $Pa$ . In this numerical investigation, the angular velocity of wheel was varied between 68,478 and 96,815  $RPM$ . The pressure ratio,  $p_{inlet}/p_{exit}^*$ , was varied between 1.456 and 2.340. For these flow conditions, the flow in the compressor was transonic. The maximum Mach number varied with the wheel speed, as shown in Table V. The supersonic flow region was located near the trailing edge of the blade, in the tip clearance region.

An important parameter that determines the leakage mass flow rate was the pressure downstream of the leakage region. The value of this pressure was specified at the location shown in Fig. 49. Once the leakage pressure value was specified, the leakage mass flow resulted from the numerical simulation.

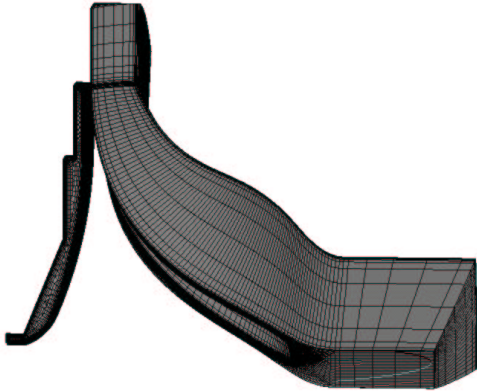


Fig. 47. Computational grid.

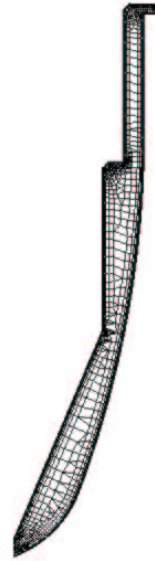


Fig. 48. Meridional section through the grid on the back side of the impeller.

Table V. Maximum Mach number variation as a function of wheel speed.

Angular velocity [ <i>RPM</i> ]	68478	83902	96815
Maximum Mach number	1.08	1.32	1.56

In certain circumstances, it may be beneficial to specify the mass flow rate as an input in the code and obtain the leakage pressure as a result of numerical simulation. It is not possible, however, to impose directly the mass flow rate as a boundary condition in a numerical algorithm that solves the Navier-Stokes equations. This limitation is due to the constraints imposed by the Riemann boundary conditions. An iterative process can be devised, however, such that the value of the leakage pressure is adjusted until the desired leakage mass flow rate value is obtained.

In order to verify the accuracy of the numerical results, one has to prove that the solution is independent of the grid size. Four meshes have been used to assess that

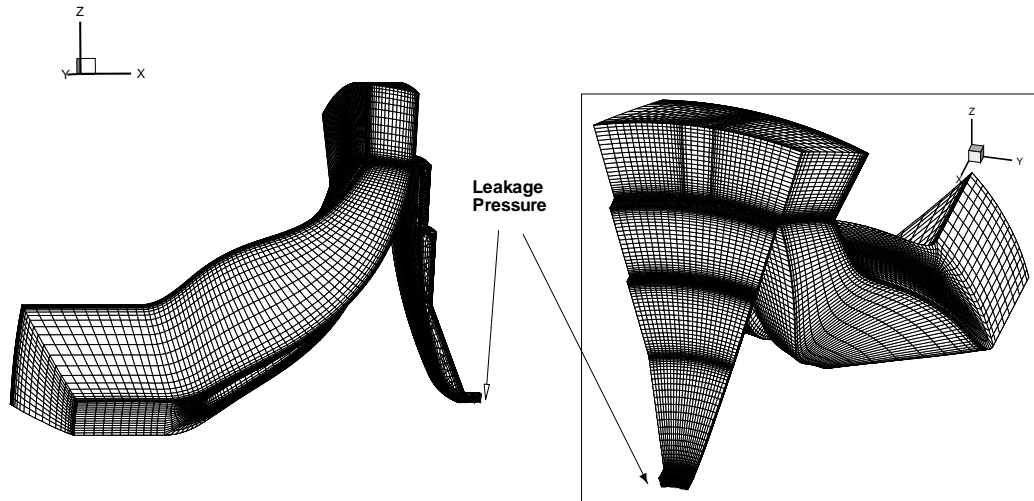


Fig. 49. Leakage pressure location.

the results are grid independent. The coarsest grid has approximately 183,000 nodes in the impeller mesh and  $3,100 \times 30$  nodes in the leakage flow mesh. The number of meridional planes has been kept constant and equal to 30 for all four grids. The  $y^+$  number for the coarsest grid is less than 3.5. The number of nodes of the finer meshes is shown in Table VI.

The outcome of the grid independence (or convergence) test is shown in Fig. 50[101]. The variation of both the axial thrust and the mass flow rate decreases significantly once the number of grid points is larger than 500,000. One can then conclude that the results are grid independent if the grid exceeds 500,000 points.

The parametric study of the axial thrust load requires a large computational effort since the flow needs to be simulated for numerous operating points. The number of iterations necessary to obtain a converged solution varies depending on the position of the operating point on the compressor map. A larger number of iterations is required for higher wheel speeds. In average, it takes approximately 15,000 iterations to obtain a converged solution with an (energy variable) error less than  $10^{-5}$ . In order

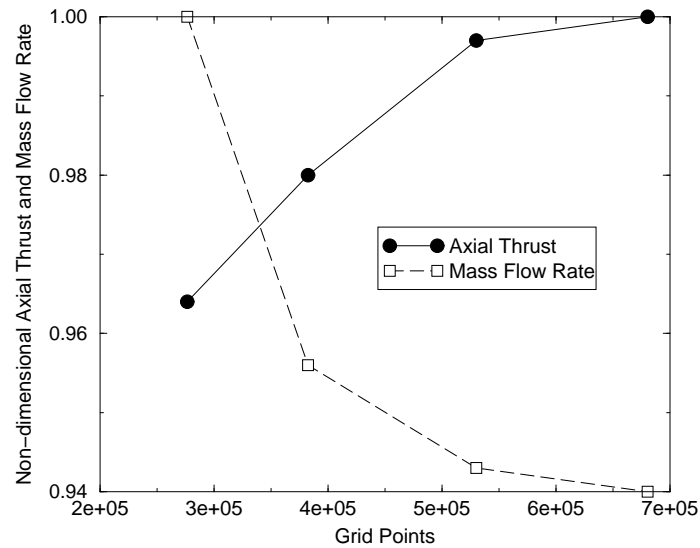


Fig. 50. Grid convergence test.

to reduce the computational effort, a computational grid with 198,280 grid points has been used. In addition to having a smaller number of grid points, the coarse grid used herein also converged faster because of its additional damping. The reduction of the computational effort, however, is penalized by an increase of the computational error. As shown in Fig. 50, there is a 5% difference between the axial thrust load predicted by a 198,280 grid points solution and the grid independent solution [101].

Figure 51 shows the computed operating points on compressor map. The lines with symbols denote the experimental data. The computed results are shown as the open triangles.

The variation of the axial thrust load was calculated for three angular velocities and two leakage pressure values. For each angular velocity, three operating points have been predicted. Consequently, the axial thrust has been calculated for 18 operating points.

The variation of the axial thrust load and leakage mass flow rate are shown in Table VII. Note that the axial thrust load is positive if it is pointing in the forward

Table VI. Size of computational grids.

Grid	Nodes			$y^+$
	Impeller Grid	Leakage Grid	Total	
1	183,471	$3100 \times 30$	276,471	3.5
2	265,471	$3890 \times 30$	382,171	3.5
3	396,483	$4450 \times 30$	529,983	2.5
4	501,583	$5950 \times 30$	680,083	2.5

direction. The variation of the axial thrust load as a function of the operating point and leakage pressure is shown in Fig. 52 [102]. For a given leakage pressure, the axial thrust load increases with the compressor pressure ratio. Consequently, all the factors that contribute to the increase of the compressor pressure ratio, will also contribute to the increase of the axial thrust. As a result, the axial thrust load increases with the wheel speed. For a constant wheel speed, the axial thrust load varies with the mass flow rate. The variation of the mass flow rate is governed by the exit static pressure.

The variation of the leakage flow as a function of the operating point is shown in Fig. 53 [102]. The leakage mass flow rate is more sensitive than the axial thrust load to the variation of the leakage mass flow rate.

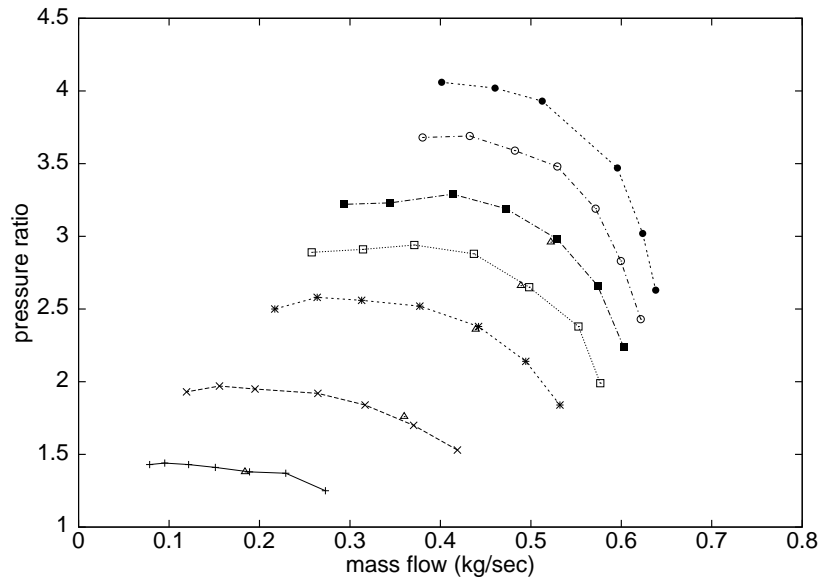


Fig. 51. Compressor map. Lines with symbols denote experimental data. Open triangles denote the computed results.

Table VII. Parameter variation.

$p_{inlet}/p_{exit}^*$	Angular velocity [RPM]	Leakage pressure [bar]	$\Pi_c^*$	Efficiency	Mass flow rate [kg/s]	Leakage mass flow rate [g/s]	Axial thrust [N]
1.456	68,478	1.0	1.774	0.805	0.3500	1.201	100.0
1.560	68,478	1.0	1.867	0.818	0.2915	1.434	113.6
1.622	68,478	1.0	1.933	0.815	0.2429	1.549	122.2
1.612	83,902	1.0	2.163	0.799	0.4715	0.953	133.3
1.820	83,902	1.0	2.367	0.796	0.4154	1.726	165.8
1.976	83,902	1.0	2.515	0.770	0.3322	2.044	186.3
1.768	96,815	1.0	2.599	0.768	0.5362	1.245	178.3
2.132	96,815	1.0	2.949	0.782	0.4906	2.033	222.4
2.340	96,815	1.0	3.224	0.724	0.3366	2.859	303.0
1.456	68,478	1.1	1.777	0.805	0.3505	0.827	117.3
1.560	68,478	1.1	1.864	0.815	0.2906	0.871	129.1
1.622	68,478	1.1	1.932	0.817	0.2470	1.236	138.3
1.612	83,902	1.1	2.162	0.795	0.4749	0.928	152.5
1.820	83,902	1.1	2.365	0.798	0.4118	1.614	180.5
1.976	83,902	1.1	2.515	0.769	0.3309	1.598	204.9
2.132	96,815	1.1	2.951	0.767	0.4903	1.651	244.2
2.340	96,815	1.1	3.142	0.715	0.3664	2.101	279.7

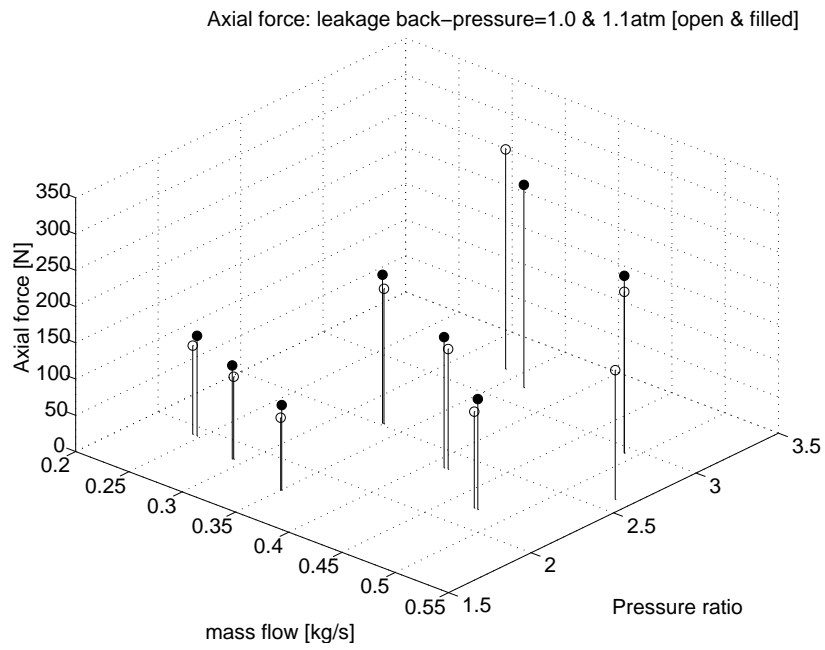


Fig. 52. Variation of axial thrust load with operating point.

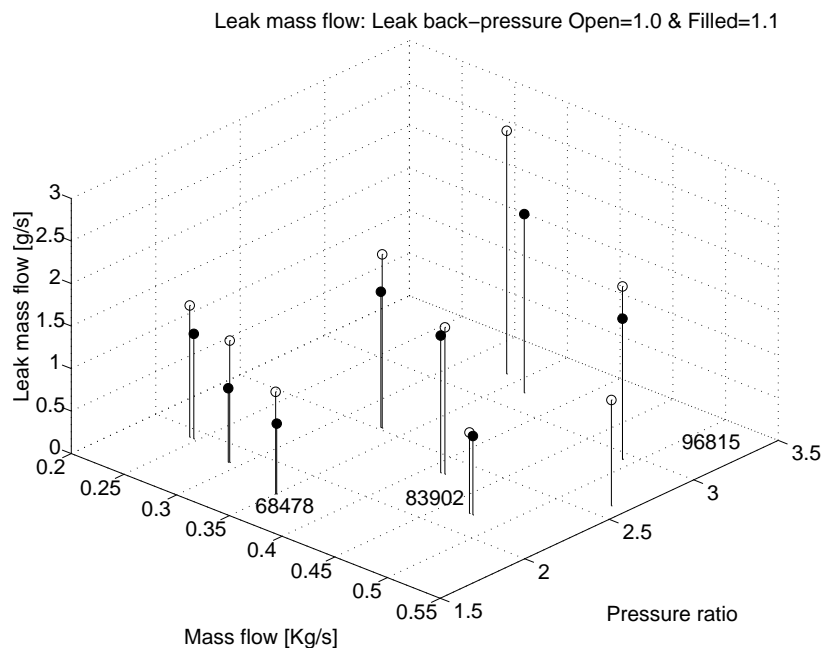


Fig. 53. Variation of leakage mass flow rate with operating point.

## CHAPTER V

## CONCLUSION AND FUTURE WORK

The hybrid mesh generation algorithm for turbomachinery airfoils took advantage of the flexibility of the 2-D unstructured mesh and the efficiency of a structured mesh. The grid generation method is robust and can efficiently handle challenging airfoils. The method is based on 2-D algorithms, which include a mapping technique and an optimization-based smoothing method. Further enhancement of grid quality was possible by edge swapping and node insertion. Edge swapping reduced the dependency of mesh quality on the choice of the source mesh. Mesh smoothing was extended to include nodes on the periodic boundaries by using the ghost-cell approach. This approach relaxes the restrictions imposed on the nodes of the periodic boundaries, and as a consequence, mesh quality was improved.

Lawson's method was extended in two ways. First, the maximization of the minimum angle was replaced by a more general quality measure. The purpose of this modification was to provide a unified quality measure for smoothing and edge swapping. Second, Lawson's method was extended from a 2-D to a 3-D algorithm by applying it to a "quad-tube". The decision to swap the edges of the tube was based on whether swapping improved the minimum quality measure of the quad-cells in the quad-tube. The methodology developed herein provides an integrated system for mapping, smoothing, edge swapping and node insertion.

The flow solver was developed using the finite volume method for a mixed elements unstructured mesh. The numerical flux was computed using the upwind method with the Roe Riemann solver for the convective flux and the averaged nodal gradient with directional derivative for the viscous flux. The Green-Gauss method and the least-squares method were employed to compute the gradients of the flow



state variables. The higher-order spatial accuracy was obtained by piece-wise linear reconstruction. The  $k - \omega$  turbulence model was implemented to account for turbulence effects. The edge-based data structures were used extensively to make the implemented solution algorithm well suited for a general mesh.

The supersonic vortex case was tested to demonstrate the accuracy of the higher-order scheme on the rotational frame of reference. The computed solution approximated well the analytical solution. The shock capturing capability was tested in the channel flow case. The accuracy of the flow solver was verified by the comparison of the results against published data. The solutions of the flow solver were validated by the boundary layer solutions in both a laminar and a turbulent flow case. A numerical test for a high speed Honeywell centrifugal compressor was conducted. A grid independent solution was achieved after a series of computations on grids of various spatial resolutions. Finally, the variation of the axial load was predicted for several compressor operating points.

A serial version of the flow solver was modified to run on multi-processor computers. The parallelization was implemented by using the single program multiple data (SPMD) approach where each processor was assigned a portion of the global mesh. The data exchange between the interfaces of the partitioned meshes was accomplished with MPI. The user friendliness of the flow solver was enhanced by the addition of a graphical user interface.

Possible future work directions are presented in the following paragraphs. There are much room for improvement to be made on the present work. The edge swapping method in the present hybrid mesh generation algorithm was limited to the interior edges only. The edge swapping, however, can also be applied to the periodic boundaries using a ghost-cell approach. The generalized edge swapping should further enhance the quality of a mesh and is suggested as a future study.

The implemented flow solution algorithm will benefit from the convergence acceleration schemes such as, preconditioning, and multi-grid.

The application of the parallel version of the solver is limited to a simple geometry due to the difficulty in domain decomposition. A general mesh partitioning tool such as “Metis” [103] is suggested for the future work.

## REFERENCES

- [1] Löhner, R., *Applied CFD Techniques*, Wiley, New York, 2001.
- [2] Mavriplis, D. J., “On Convergence Acceleration Techniques for Unstructured Meshes,” Tech. Rep. 98-45, ICASE, Hampton, VA, Oct. 1998.
- [3] Khawaja, A. and Kallinderis, Y., “Hybrid Grid Generation for Turbomachinery and Aerospace Applications,” *International Journal for Numerical Methods in Engineering*, Vol. 49, 2000, pp. 145–166.
- [4] Pirzadeh, S., “Three-dimensional Unstructured Viscous Grids by the Advancing-Layers Method,” *AIAA Journal*, Vol. 24, No. 1, 1996, pp. 43–49.
- [5] Owen, S., “A Survey of Unstructured Mesh Generation Technology,” *7th International Meshing Roundtable*, 1998, pp. 239–267.
- [6] Mingwu, L., Benzley, S., Sjaardema, G., and Tautges, T., “A Multiple Source and Target Sweeping Method for Generating All Hexahedral Finite Element Meshes,” *Proceedings of the 5th International Meshing Roundtable*, October 1996, pp. 217–225.
- [7] Staten, M., Canann, S., and Owen, S., “BMSWEEP: Locating Interior Nodes During Sweeping,” *Engineering with Computers*, Vol. 15, No. 3, 1999, pp. 212–218.
- [8] Marcum, D. L. and Whatherill, N. P., “Unstructured Grid Generation Using Iterative Point Insertion and Local Reconnection,” *AIAA Journal*, Vol. 33, 1995, pp. 1619–1625.

- [9] Batina, J., “Unsteady Euler Airfoil Solution Using Unstructured Dynamic Meshes,” *AIAA Journal*, Vol. 28, No. 8, 1990, pp. 1381–1388.
- [10] Bern, M. W. and Eppstein, D., “Mesh Generation and Optimal Triangulation,” Tech. Rep. CSL-92-1, Xerox Palo Alto Research Center, Palo Alto, CA, 1992.
- [11] Zhou, T. and Shimada, K., “An Angle-Based Approach to Two-Dimensional Mesh Smoothing,” *9th International Meshing Roundtable*, Sandia National Laboratories, New Orleans, Louisiana, October 2000, pp. 373–384.
- [12] Canann, S., Tristano, J., and Staten, M., “An Approach to Combined Laplacian and Optimization-Based Smoothing for Triangular, Quadrilateral, and Quad-Dominant Meshes,” *7th International Meshing Roundtable*, Sandia National Laboratories, Dearborn, MI, 1998, pp. 479–494.
- [13] Freitag, L. and Plassmann, P., “Local Optimization-based Simplicial Mesh Untangling and Improvement,” *International Journal of Numerical Methods in Engineering*, Vol. 49, No. 1-2, July 2000, pp. 109–125.
- [14] Amenta, A. B., Bern, M. W., and Eppstein, D., “Optimal Point Placement for Mesh Smoothing,” *J. Algorithms*, Vol. 30, No. 2, Feb 1999, pp. 302–322, Special issue for 8th SODA.
- [15] Feffermann, C. L., “Existence & Smoothness of the Navier-Stokes Equation,” [http://www.claymath.org/Millennium\\_Prize\\_Problems/](http://www.claymath.org/Millennium_Prize_Problems/), November 2003.
- [16] Mavriplis, D., “Three-dimensional multigrid for the Euler equations,” *AIAA Journal*, Vol. 30, 1992, pp. 1753–1761.
- [17] Potsdam, M., Intemann, G., Frink, N. T., Campbell, R., Smith, L., and Pirzadeh, S., “Wing Pylon Fillet Design Using Unstructured Mesh Euler

- Solvers,” AIAA Applied Aerodynamics Conference, AIAA Paper 93-3500, Monterey, CA, August 1993.
- [18] Venkatakrishnan, V., “A Perspective On Unstructured Grid Flow Solvers,” Tech. Rep. 95-3, ICASE, Hampton, VA, 1995.
- [19] Roe, P., “Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes,” *Journal of Computational Physics*, Vol. 43, 1981, pp. 357–372.
- [20] Roe, P., “Characteristic Based Schemes for the Euler Equations,” *Annual Review of Fluid Mechanics*, Vol. 18, 1986, pp. 337–365.
- [21] Weiss, J. M., Maruszewski, J. P., and Smith, W. A., “Implicit Solution of Preconditioned Navier-Stokes Equations Using Algebraic Multigrid,” *AIAA Journal*, Vol. 37, 1999, pp. 29–36.
- [22] Haselbacher, A. and Blazek, J., “Accurate and Efficient Discretization of Navier-Stokes Equations on Mixed Grids,” *AIAA Journal*, Vol. 38, No. 11, 2000, pp. 2094–2102.
- [23] Blazek, J., *Computational Fluid Dynamics: Principles and Applications*, Elsevier, New York, 2001.
- [24] Barth, T., “Aspects of Unstructured Grids and Finite-Volume Solvers for the Euler and Navier-Stokes Equations,” von Karman Institute (VKI) Lecture Series 1994-05, 1994.
- [25] Barth, T. J. and Jespersen, D. C., “The Design and Application of Upwind Schemes on Unstructured Meshes,” 27th AIAA Aerospace Sciences Meeting & Exhibit, AIAA Paper 89-0366, Reno, NV, January 1989.

- [26] Venkatakrishnan, V., “Convergence to Steady State Solutions of the Euler Equations on Unstructured Grids with Limiters,” 31st AIAA Aerospace Sciences Meeting & Exhibit, AIAA Paper 93-0880, Reno, NV, January 1993.
- [27] Wilcox, D. C., *Turbulence Modeling for CFD*, DCW Industries, Inc., La C anada, 1993.
- [28] Hirsch, C., *Numerical Computation of Internal and External Flows*, Vol. 2, Wiley, New York, 1988.
- [29] Pope, S., *Turbulent Flows*, Cambridge University Press, Cambridge, 2000.
- [30] Wilcox, D. C., “Simulation of Transition with a Two-Equation Turbulence Model,” *AIAA Journal*, Vol. 32, 1994, pp. 247–255.
- [31] Hellsten, A., “Some Improvements in Menter’s  $k$ - $\omega$  SST Turbulence Model,” 29th AIAA Fluid Dynamics Conference, AIAA Paper 1998-2554, Albuquerque, NM, June 1998.
- [32] Kim, N. and Rhode, D. L., “Swirling Streamline-Curvature Law of the Wall from a Novel Perturbation Analysis,” *Numerical Heat Transfer, Part B*, Vol. 36, 1999, pp. 331–350.
- [33] Wilcox, D. C. and Chambers, T. L., “Streamline Curvature Effects on Turbulent Boundary Layers,” 9th AIAA Fluid and Plasma Dynamics Conference & Exhibit, AIAA Paper 76-351, San Diego, CA, July 1976.
- [34] B. E. Launder, C. H. Priddin, B. I. S., “The Calculation of Turbulent Boundary Layers on Spinning and Curved Surfaces,” *Journal of Fluids Engineering*, Vol. 99, 1977, pp. 231–239.

- [35] Bradshaw, P., “Compressible Turbulent Shear Layers,” *Annual Review of Fluid Mechanics*, Vol. 9, 1977, pp. 33–54.
- [36] Tweedt, D. L., Chima, R. V., and Turkel, E., “Preconditioning for Numerical Simulation of Low Mach Number Three-Dimensional Viscous Turbomachinery Flow,” 28th AIAA Fluid Dynamics Conference and 4th AIAA Sear Flow Control Conference, AIAA Paper 97-1828, Snowmass, CO, June 1997.
- [37] Chima, R. V., “A K-omega turbulence model for quasi-three-dimensional turbomachinery,” 34th AIAA Aerospace Sciences Meeting & Exhibit, AIAA Paper 96-0248, Reno, NV, January 1996.
- [38] Yao, J., Jameson, A., Alonso, J. J., and Liu, F., “Development and Validation of a Massively Parallel Flow Solver for Turbomachinery Flows,” 38th AIAA Aerospace Sciences Meeting & Exhibit, AIAA Paper 00-0882, Reno, NV, January 2000.
- [39] Menter, F. R., “Two-Equation Eddy-Viscosity Turbulence Models for Engineering Applications,” *AIAA Journal*, Vol. 32, No. 8, August 1994, pp. 1598–1605.
- [40] Thivet, F., Daouk, M., and Knight, D. D., “Influence of the Wall Condition on  $k - \omega$  Turbulence Model Prediction,” *AIAA Journal*, Vol. 40, No. 1, 2002, pp. 179–181.
- [41] Menter, F. R., “Influence of Freestream Values on  $k - \omega$  Turbulence Model Prediction,” *AIAA Journal*, Vol. 30, No. 6, August 1992, pp. 1657–1659.
- [42] Tennekes, H. and Lumley, J. L., *A First Course in Turbulence*, The MIT Press, Cambridge, MA, 1972.

- [43] Kim, K. and Cizmas, P. G., “Three-Dimensional Hybrid Mesh Generation for Turbomachinery Airfoils,” *Journal of Propulsion and Power*, Vol. 18, No. 3, May 2002, pp. 536 – 543.
- [44] Shewchuk, J. R., “Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator,” *Applied Computational Geometry: Towards Geometric Engineering*, edited by M. C. Lin and D. Manocha, Vol. 1148 of *Lecture Notes in Computer Science*, Springer-Verlag, May 1996, pp. 203–222.
- [45] Preparata, F. P. and Shamos, M. I., *Computational Geometry*, Texts and Monographs in Computer Science, Springer-Verlag, 1985.
- [46] Lawson, C. L., “Software for  $C^1$  Surface Interpolation,” Tech. Rep. NASA-CR-155047, Jet Propulsion Lab., California Institute of Technology, Pasadena, CA, 1977.
- [47] Holmes, D. G. and Snyder, D. D., “The Generation of Unstructured Triangular Meshes Using Delaunay Triangulation,” *Numerical Grid Generation in Computational Fluid Mechanics*, Pineridge Press, Swansea, U.K., 1988, pp. 643–652.
- [48] Godunov, S. K., “A Difference Scheme for Numerical Computation Discontinuous Solution of Hydrodynamic Equations,” *Math. Sbornik*, Vol. 47, 1959, pp. 271–306, translated US Joint Publications Research Service, JPRS 7226, 1969.
- [49] Kim, K., *An Adaptive Mesh Method for the Simulation of Blade Vortex Interaction*, Master’s thesis, Texas A&M University, Aug. 1998.
- [50] van Leer, B., Tai, C. H., and Powell, K. G., “Design of Optimally Smoothing Multi-Stage Schemes for the Euler Equations,” *Communications in Applied*



*Numerical Mathematics*, Vol. 8, 1992, pp. 761–769.

- [51] Jameson, A., Schmidt, W., and Turkel, E., “Numerical Solutions of the Euler Equations by Finite Volume Methods Using Runge-Kutta Time-Stepping Schemes,” 14th AIAA Fluid and Plasma Dynamic Conference, AIAA Paper 81-1259, Palo Alto, CA, June 1981.
- [52] Courant, R., Friedrichs, K. O., and Lewy, H., “On the Partial Difference Equations of Mathematical Physics,” *Mathematische Annalen*, Vol. 100, 1928, pp. 32–74, translated IBM Journal, Vol. 11, 1967, pp. 215-234.
- [53] Jameson, A. and Baker, T. J., “Calculation of Inviscid Transonic Flow over a Complete Aircraft,” 24th AIAA Aerospace Sciences Meeting & Exhibit, AIAA Paper 86-0103, Reno, NV, January 1986.
- [54] Blazek, J., Irmisch, S., and Haselbacher, A., “Unstructured Mixed-Grid Navier-Stokes Solver for Turbomachinery Applications,” 37th AIAA Aerospace Sciences Meeting & Exhibit, AIAA Paper 99-0664, Reno, NV, January 1999.
- [55] Vijayan, P. and Kallinderis, Y., “A 3D Finite-Volume Scheme for the Euler Equations on Adaptive Tetrahedral Grids,” *Journal of Computational Physics*, Vol. 113, 1994, pp. 249–267.
- [56] Toro, E. F., *Riemann Solvers and Numerical Methods for Fluid Dynamics*, Springer-Verlag, Berlin, 2nd ed., 1999.
- [57] Hirsch, C., *Numerical Computation of Internal and External Flows*, Wiley, New York, 1988.
- [58] Chen, J. P., Ghosh, A. R., Sreenivas, K., and Whitfield, D. L., “Comparison of Computations Using Navier-Stokes Equations in Rotating and Fixed Coor-

- dinates for Flow Through Turbomachinery,” 35th AIAA Aerospace Sciences Meeting & Exhibit, AIAA Paper 97-0878, Reno, NV, January 1997.
- [59] Lomax, H., Pulliam, T. H., and Zingg, D. W., *Fundamentals of Computational Fluid Dynamics*, Vol. I, Springer Verlag, Berlin, 2001.
- [60] Aftosmis, M., Gaitonde, D., and Tavares, T. S., “Behavior of Linear Reconstruction Techniques on Unstructured Meshes,” *AIAA Journal*, Vol. 33, No. 11, 1995, pp. 2038–2049.
- [61] Essers, J. A., Delanaye, M., and Rogiest, P., “An Upwind-Biased Finite-Volume Technique for Solving Compressible Navier-Stokes Equations on Irregular Meshes. Applications to Supersonic Blunt-Body Flows and Shock-Boundary Layer Interactions,” 11th AIAA Computational Fluid Dynamics Conference, AIAA Paper 93-3377, Orlando, FL, July 1993.
- [62] Guillard, H. and Farhat, C., “On the Significance of the Geometric Conservation Law for Flow Computations on Moving Meshes,” *Computer Methods in Applied Mechanics and Engineering*, Vol. 190, 2000, pp. 1467–1482.
- [63] Xu, K., “Does Perfect Riemann Solver Exist?” 14th AIAA Computational Fluid Dynamics Conference, AIAA Paper 99-3344, Norfolk, VA, June 1999.
- [64] Gressier, J. and Moschetta, J.-M., “On the Pathological Behavior of Upwind Schemes,” 36th AIAA Aerospace Sciences Meeting & Exhibit, AIAA Paper 98-0110, Reno, NV, January 1998.
- [65] Kim, S., Kim, C., Rho, O., and Hong, S., “Cure for Shock Instability: Development of an Improved Roe Scheme,” 40th AIAA Aerospace Sciences Meeting & Exhibit, AIAA Paper 2002-0548, Reno, NV, January 2002.

- [66] Harten, A., “Self Adjusting Grid Methods for One Dimensional Hyperbolic Conservation Laws,” *Journal of Computational Physics*, Vol. 50, 1983, pp. 235–269.
- [67] Campbell, J. C., Hyman, J. M., and Shashkov, M. J., “Mimetic Finite Difference Operators for Second-Order Tensors on Unstructured Grids,” *Computers and Mathematics with Application*, Vol. 44, 2002, pp. 157–173.
- [68] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P., *Numerical Recipes in FORTRAN : the Art of Scientific Computing*, Cambridge University Press, Cambridge, England, 1992.
- [69] Golub, G. H. and van Loan, C. F., *Matrix Computations*, Johns Hopkins University Press, Baltimore, MD, 1993.
- [70] van Leer, B., “Towards the Ultimate Conservative Difference Scheme. V. A Second Order Sequel to Godunov’s Method,” *Journal of Computational Physics*, Vol. 32, 1979, pp. 101–136.
- [71] Venkatakrishnan, V., “Convergence to Steady-State Solutions of the Euler Equations on Unstructured Grids with Limiters,” *Journal of Computational Physics*, Vol. 118, 1995, pp. 120–130.
- [72] Barth, T. J., “Numerical Aspects of Computing Viscous High Reynolds Number Flows on Unstructured Meshes,” 29th AIAA Aerospace Sciences Meeting & Exhibit, AIAA Paper 91-0721, Reno, NV, January 1991.
- [73] Luo, H., Baum, J. D., Löhner, R., and Cabello, J., “Adaptive Edge-based Finite Elements Schemes for the Euler and Navier-Stokes Equations on Unstructured

- Grids,” 31st AIAA Aerospace Sciences Meeting & Exhibit, AIAA Paper 93-0336, Reno, NV, January 1993.
- [74] Whitaker, D. L., “Three-Dimensional Unstructured Grid Euler Computations Using a Fully-Implicit, Upwind Method,” 11th AIAA Computational Fluid Dynamics Conference, AIAA Paper 93-3337, Orlando, FL, July 1993.
- [75] Menter, F. R., “Zonal Two Equation  $k-\omega$  Turbulence Models for Aerodynamic Flows,” 24th AIAA Fluid Dynamics Conference, AIAA Paper 93-2906, Orlando, FL, July 1993.
- [76] Yodder, D. A., Georgiadids, N. J., and Orkwis, P. D., “Implementation of a Two-Equation  $k$ - $\omega$  Turbulence Model in NPARC,” 34th AIAA Aerospace Sciences Meeting & Exhibit, AIAA Paper 96-0383, Reno, NV, January 1996.
- [77] Liu, F. and Zheng, X., “A Strongly Coupled Time-Marching Method for Solving the Navier-Stokes and  $k-\omega$  Turbulence Model Equations with Multigrid,” *Journal of Computational Physics*, Vol. 128, No. 0211, August 1996, pp. 289–300.
- [78] Zheng, X. and Liu, F., “Staggered Upwind Method for Solving Navier-Stokes and  $k-\omega$  Turbulence Model Equations,” *AIAA Journal*, Vol. 33, No. 6, June 1995, pp. 991–998.
- [79] Martinelli, L. and Jameson, A., “Viscous Flow Solvers for Aero/Hydrodynamic Analysis and Design,” 29th AIAA Fluid Dynamics Conference, AIAA Paper 98-3003, Albuquerque, NM, June 1998.
- [80] Minyard, T. and Kallinderis, Y., “Applications of Grid Partitioning and Parallel Dynamics Load Balancing,” 35th AIAA Aerospace Sciences Meeting & Exhibit,

AIAA Paper 97-0879, Reno, NV, January 1997.

- [81] Snir, M., Otto, S., Huss-Lederman, S., Walker, D., and Dongarra, J., *MPI: The Complete Reference*, The MIT Press, Cambridge, MA, 1996.
- [82] Pacheco, P. S., *Parallel Programming with MPI*, Morgan Kaufmann, San Francisco, CA, 1997.
- [83] Welch, B. B., *Practical Programming in Tcl and Tk*, Prentice Hall, Upper Saddle River, NJ, 1995.
- [84] Harrison, M. and McLennan, M., *Effective Tcl/Tk Programming : Writing Better Programs with Tcl and Tk*, Addison-Wesley professional computing series, Addison-Wesley, Reading, MA, 1998.
- [85] Flynt, C., *Tcl/Tk for Real Programmers*, AP Professional, San Diego, CA, 1999.
- [86] Metcalf, M. and Reid, J. K., *Fortran 90/95 Explained*, Oxford University Press, Oxford, New York, 2nd ed., 1999.
- [87] Chapman, S. J., *Fortran 90/95 for Scientists and Engineers*, McGraw-Hill, Boston, 2nd ed., 1997.
- [88] "Fortran 90: A Conversion Course for Fortran 77 Programmers," Manchester and North High Performance Computing Training & Education Centre, <http://www.hpctec.mcc.ac.uk/hpctec/courses/Fortran90/F90course.html>, November 2003.
- [89] Coirier, W. J. and Jorgenson, P. C. E., "A Mixed volume grid approach for the Euler and Navier-Stokes equations," 34th AIAA Aerospace Sciences Meeting & Exhibit, AIAA Paper 96-0762, Reno, NV, January 1996.

- [90] Rizzi, A. and Viviand, H., “Numerical Methods for the Computation of Inviscid Transonic Flows with Shock Waves : a GAMM workshop,” *Notes on Numerical Fluid Mechanics*, Friedr. Vieweg & Sohn, Braunschweig, Wiesbaden, German, 1981.
- [91] Manna, M., “A Three Dimensional High Resolution Upwind Finite Volume Euler Solver,” von Karman Institute (VKI) Lecture Series 1992-180, April 1992.
- [92] Kermani, M., “Modified Entropy Correction Formula for the Roe Scheme,” 39th AIAA Aerospace Sciences Meeting & Exhibit, AIAA Paper 2001-0083, Reno, NV, January 2001.
- [93] Anderson, J. D., *Modern Compressible Flow with Historical Perspective*, McGraw-Hill, New York, 2nd ed., 1990.
- [94] White, F. M., *Viscous Fluid Flow*, McGraw-Hill, New York, 1991.
- [95] Schlichting, H., *Boundary-Layer Theory*, McGraw-Hill, New York, 1979.
- [96] Jiang, Y. T., Damodaran, M., and Lee, K. H., “High-Resolution Finite Volume Computation of Turbulent Transonic Flow Fast Airfoils,” *AIAA Journal*, Vol. 35, No. 7, July 1997, pp. 1134–1142.
- [97] Wieghardt, K. and Tillmann, W., “On the Turbulent Friction Layer for Rising Pressure,” NACA TM 1314, 1951.
- [98] Patel, V. C., Rodi, W., and Scheuerer, G., “Turbulence Models for Near-Wall and Low-Reynolds Number Flows: A Review,” *AIAA Journal*, Vol. 23, No. 9, September 1985, pp. 1308–1319.

- [99] Hellsten, A., “On the Solid-Wall Boundary Condition of  $\omega$  in the  $k - \omega$ -Type Turbulence Models,” Tech. Rep. B-50, Laboratory of Applied Thermodynamics, Helsinki University of Technology, Espoo, Finland, March 1998.
- [100] Yoder, D. A. and Georgiadis, N. J., “Implementation and Validation of the Chien k-epsilon Turbulence Model in the WIND Navier-Stokes Code,” 37th AIAA Aerospace Sciences Meeting & Exhibit, AIAA Paper 99-0745, Reno, NV, January 1999, See also, <http://www.grc.nasa.gov/www/valid/fpturb/fpturb01/fpturb01.html>, November 2003.
- [101] Han, Z. and Cizmas, P. G. A., “Prediction of Axial Thrust Load in Centrifugal Compressors,” *International Journal of Turbo & Jet-Engines*, Vol. 20, No. 1, 2003, pp. 1–16.
- [102] Cizmas, P. G. A. and Kim, K., “Parametric Study of Axial Thrust Load,” Tech. rep., Turbomachinery Research Consortium, Texas A&M University, 2003.
- [103] Karypis, G. and Kumar, V., “Multilevel Algorithms for Multi-Constraint Graph Partitioning,” Tech. Rep. 98-019, University of Minnesota, Department of Computer Science, 1998.

## VITA

Kyusup Kim was born on May 2, 1969 in Seoul, Republic of Korea (South Korea). He received his B.S. in Mechanical Engineering from Washington State University, Pullman, WA in 1994 and received his M.S. in the Aerospace Engineering at Texas A&M University in 1998. His research interests include adaptive unstructured mesh, mesh optimization and high resolution flow solver. The author can be reached via electronic mail at [qkim@aggienetwork.com](mailto:qkim@aggienetwork.com).