STABILIZED FINITE ELEMENTS FOR COMPRESSIBLE

TURBULENT NAVIER-STOKES

By

Jon Taylor Erwin

W. Kyle Anderson
Professor of Computational Engineering
(Chair)

Sagar Kapadia
Research Assistant Professor of
Computational Engineering
(Committee Member)

Li Wang
Research Assistant Professor of
Computational Engineering
(Committee Member)

Kidambi Sreenivas
Research Professor of Computational
Engineering
(Committee Member)

Clarence O. E. Burg
Associate Professor of Mathematics
(Committee Member)

STABILIZED FINITE ELEMENTS FOR COMPRESSIBLE TURBULENT NAVIER-STOKES

By

Jon Taylor Erwin

A Dissertation Submitted to the Faculty of the University of Tennessee at Chattanooga in Partial Fulfillment of the Requirements of the Degree of Doctor of Philosophy in Computational Engineering

The University of Tennessee at Chattanooga
Chattanooga, Tennessee

December 2013

ABSTRACT

In this research a stabilized finite element approach is utilized in the development of a high-order flow solver for compressible turbulent flows. The Reynolds averaged Navier-Stokes (RANS) equations and modified Spalart-Almaras (SA) turbulence model are discretized using the streamline/upwind Petrov-Galerkin (SUPG) scheme. A fully implicit methodology is used to obtain steady state solutions or to drive unsteady problems at each time step. Order of accuracy is assessed for inviscid and viscous flows in two and three dimensions via the method of manufactured solutions. Proper treatment of curved surface geometries is of vital importance in high-order methods, especially when high aspect ratio elements are used in viscous flow regions. In two dimensions, analytic surface representations are used to ensure proper surface point placement, and an algebraic mesh smoothing procedure is applied to prevent invalid elements in high aspect ratio meshes. In dealing with complex three-dimensional geometries, high-order curved surfaces are generated via a Computational Analysis PRogramming Interface (CAPRI), while the interior meshes are deformed through a linear elasticity solver. In addition, the effects of curved elements on solution accuracy are evaluated. Finally, several test cases in two and three dimensions are presented and compared with benchmark results and/or experimental data.

DEDICATION

This dissertation is dedicated to my loving and supportive wife, Tresa Erwin, and to our wonderful daughter, Layla Erwin.

ACKNOWLEDGEMENTS

I would like to thank my adviser, Dr. W. Kyle Anderson, for his support and encouragement in this endeavor. Sincere thanks also go to committee, Dr. Sagar Kapdia, Dr. Li Wang, Dr. Kidambi Sreenivas, and Dr. Clarence O. E. Burg for serving on my dissertation committee and for providing valuable insights throughout this research. I would also like to thank Dr. Timothy Swafford for being a constant source of guidance and encouragement throughout my time at the SimCenter. Additionally, I owe a debt of gratitude to the entire SimCenter faculty and staff for providing an open and friendly environment for research.

I would also like to thank my parents, Rex and Evonna Lynn and Dennis and Andrea Erwin, for all of their support an encouragement. Finally, I would like to thank my wife, Tresa, without whose patience and support none of this would have been possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

The development and application of Computational Fluid Dynamics (CFD) has evolved to the point where very complicated flow fields can be computed for many steady and unsteady scenarios. However, for many flows, success has been more limited by the severe computational resources required to resolve small, but important flow features with sufficient accuracy. Potentially significant advances for computing these flows can be achieved by using high-order spatial discretization, coupled with adaptive meshing capabilities. For example, considering a typical second-order scheme with truncation error of order h^2, by assuming the same constant of proportionality for the leading error term, a similar level of truncation error could be obtained using a third-order scheme on a mesh with only $N^{2/3}$ number of mesh points. To demonstrate the potential impact, 71 million mesh points have recently been used for simulations over a nose landing gear configuration using a second-order accurate scheme [1]. Using the (very) rough estimate provided above, similar accuracy could be obtained using only 171 thousand nodes. Note, however, that on a given mesh, the higher-order methods also have additional degrees of freedom and quadrature-related function evaluations that must be accounted for to obtain a refined estimate of any potential savings. While the above estimate is admittedly very optimistic and ignores the realities associated with non-uniformly distributed truncation errors and having sufficient geometry resolution, success

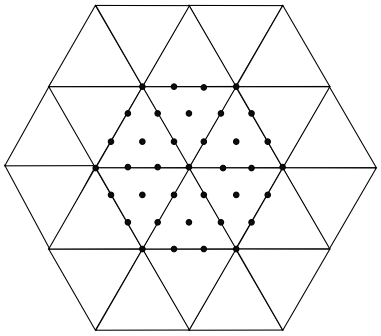
in developing high-order schemes to a level of maturity where they can be used for production-level simulations can have significant impact in many areas.

To this end, high-order discontinuous-Galerkin (DG) and Petrov-Galerkin (PG) algorithms have been under development for several years. Globally, a significant level-of-effort has been dedicated to the development of DG schemes [2–16], whereas much less effort has been dedicated to the development of PG schemes [16–36].

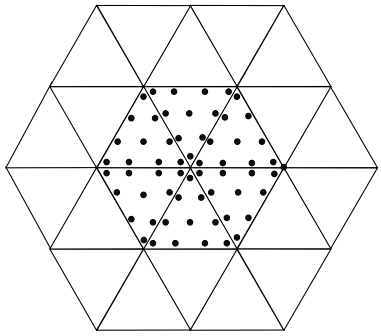
While the DG method is conceptually simple to understand and many researchers have obtained excellent results, the PG scheme offers some potential advantages when balancing the level of work and computational resources required to obtain a solution.

Motivation

Before describing the PG methodology, motivation for studying the PG approach is provided by estimating the overall work involved in obtaining a solution for both the PG and the DG schemes. To obtain the estimates, first consider a two-dimensional mesh that is assumed to be regular and does not include any boundaries. The unknowns for the PG scheme are assumed continuous across element boundaries, whereas for the DG scheme, unknowns are stored on a per-element basis and are assumed to be discontinuous across element boundaries. Depictions for a fourth-order accurate scheme are shown in figures I.1a and I.1b for the PG and DG schemes, respectively. Using the configurations depicted in figure I.1, estimates for the number of degrees of freedom and the number of non-zeros that would be required for a fully implicit algorithm can be obtained for each scheme. Note that while the figure depicts configurations for fourth-order accurate schemes, the estimates are derived below for arbitrary order.



(a) Stencil for fourth-order PG scheme



(b) Stencil for fourth-order DG scheme

Figure I.1 Stencils for PG and DG schemes

To obtain the estimates, note that for an algorithm of order p, an isolated triangle contains $n_e = p - 1$ mid-side degrees of freedom (DOF) along each edge (this does not include the vertices at each end of the edge), $n_i = \max(0, (p-1)(p-2)/2)$ degrees of freedom in the interior, and $n_t = (p+1)(p+2)/2$ total degrees of freedom. To determine the degrees of freedom for an entire mesh, the PG scheme requires one DOF for each vertex plus $n_e \times$ (number of edges), plus $n_i \times$ (number of triangles). Using an estimate that there are approximately twice as many edges and triangles as there are nodes, the total DOF for a PG scheme is given as

$$\text{DOF}_{PG} = N_V + n_e N_E + n_i N_T \approx N_V(1 + 2n_e + 2n_i) \tag{I.1}$$

where N_V is the number of vertices in the mesh, N_E is the number of edges in the mesh, and N_T is the number of triangles. Computing the DOF for the DG scheme is somewhat simpler because the number of degrees of freedom for each triangle is given as $3 + 3n_e + n_i$. Using the above approximation that the number of triangles in the mesh is twice the number of vertices, the DOF

for a DG scheme is given as

$$\mathrm{DOF}_{DG} = N_T(3 + 3n_e + n_i) \approx 2N_V(3 + 3n_e + n_i) \tag{I.2}$$

Estimating the number of non-zero entries that are needed for a fully-implicit implementation is achieved by noting that for a PG scheme, the number of connections for each vertex node in a topologically regular mesh is given as

$$C_v^{PG} = 7 + 6n_e + 6n_e + 6n_i \tag{I.3}$$

where the term connection refers to a dependency relationship between nodes. Similarly, each mid-edge node and interior node has the following number of connections

$$C_e^{PG} = 2n_t - n_e - 2 \tag{I.4}$$

$$C_i^{PG} = n_t \tag{I.5}$$

The total number of non-zero entries in a matrix representing the full linearization of the residual is determined by summing the connections for each entity multiplied by the total number of entities in the mesh. Again, using the approximate relations between the number of edges and triangles in a mesh, one obtains the following estimate

$$\begin{aligned} \mathrm{NNZ}_{PG} &= C_v^{PG} N_V + C_e^{PG} n_e N_E + C_i^{PG} n_i N_T \\ &\approx N_V \left(C_v^{PG} + 2C_e^{PG} n_e + 2C_i^{PG} n_i \right) \end{aligned} \tag{I.6}$$

For the DG scheme, estimating the number of non-zeros is again facilitated by first considering each triangle individually and subsequently multiplying by the number of triangles. Here, the number of connections for each vertex, mid-side node, and interior node, is given as

$$C_v^{DG} = n_t + 2n_t \tag{I.7}$$

$$C_e^{DG} = n_t + n_t \tag{I.8}$$

$$C_i^{DG} = n_t \tag{I.9}$$

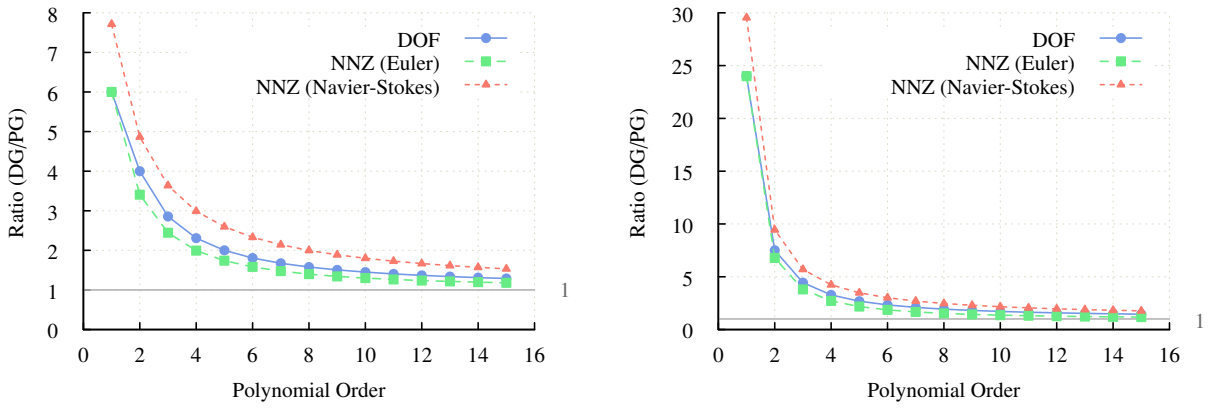
Because a triangle has three vertices and three edges, the total storage for the number of non-zeros in the mesh is given by

$$\begin{aligned} \mathrm{NNZ}_{DG} &= N_T \left(3C_v^{DG} + 3C_e^{DG} n_e + C_i^{DG} n_i\right) \\ &\approx 2N_V \left(3C_v^{DG} + 3C_e^{DG} n_e + C_i^{DG} n_i\right) \end{aligned} \tag{I.10}$$

In three dimensions, a similarly canonical mesh is not available. However, by examining several meshes for actual geometries, it is observed that there are approximately 13 edges that connect to any vertex, with 21–23 tetrahedral elements connecting at a common node. Therefore, to obtain the estimates for three dimensions, an icosahedron, which has 12 connecting edges and 20 connecting tetrahedrons, is used as a representative configuration.

Figure I.2 shows the ratio of the number of degrees of freedom for a DG scheme to that of a PG scheme, as well as the ratio of the number of non-zero entries for both inviscid and viscous flows. Note that when determining the ratio of non-zero entries in the matrix for inviscid

flows, the DG scheme only requires data immediately adjacent to the interface between triangles or tetrahedra, whereas for viscous flows, all the data in each adjoining element is required. As seen in the figure, for low orders of accuracy, the DG scheme requires significantly more degrees of freedom and non-zero matrix entries than the PG scheme. Of particular interest is the fact that for three-dimensional geometries, the DG scheme requires an order of magnitude more resources than the PG scheme for linear and quadratic elements. Numerical experiments for electromagnetic applications using both PG and DG formulations have verified these trends [37]. Note that in reference [37], the DG scheme has been shown to exhibit as much as 40% lower errors on a given mesh. However, the gains in accuracy are more than offset by the computing requirements.



(a) Two dimensional estimates

(b) Three dimensional estimates

Figure I.2 Ratio of Degrees of Freedom and Non-Zero Entries in Matrix for Full Linearization

Although the work estimates given above are quite relevant, they are not dispositive as to which scheme will ultimately gain acceptance. First, more favorable work estimates for the DG

scheme can be obtained on non-tetrahedral meshes. As an example, Table I.1 shows results for linear and quadratic elements obtained by computing the degrees of freedom and non-zero entries on a cubic volume subdivided into tetrahedral, hexahedral, and prismatic elements. In agreement with the estimates above, the DG scheme requires significantly more unknowns and matrix elements than the PG scheme for tetrahedrons. However, the DG scheme compares somewhat more favorably for hexahedra, with prismatic elements falling in between. Other factors that will ultimately determine the acceptance of these schemes include robustness, matrix conditioning, accuracy, and computational effort required to compute the residuals and matrix entries.

Table I.1 Relative work for PG and DG schemes for different element types and accuracy order

	Tetrahedron		Hexahedron		Prismatic	
	DOF Ratio	NNZ Ratio	DOF Ratio	NNZ Ratio	DOF Ratio	NNZ Ratio
Linear	22.16	19.8	7.53	5.74	11.35	9.42
Quadratic	7.19	6.20	2.92	2.14	4.02	3.15

As demonstrated in references [37] and [38] using the method of manufactured solutions, the accuracy of the PG scheme appears to be somewhat better than the DG scheme when measured against the number of degrees of freedom, while the DG scheme may have accuracy advantages when compared to the PG scheme on the same mesh. The advantage of the PG scheme is most significant for low-to-moderate orders of accuracy. In some applications, such as those that require uniformly high-order accuracy throughout the flow field, it appears that the schemes are comparable when balancing accuracy and work. However, for applications where significant portions of the

flow-field are discretized with low-to-moderate-order elements, the PG scheme should be considered as a means for obtaining numerical solutions; this would include adaptive meshing strategies where low-order elements constitute the initial mesh.

Because of the significant potential advantages of the Petrov-Galerkin scheme, an existing solver for inviscid flows, based on the work described in reference [37], will be extended to include viscous simulation capability for laminar and turbulent flows. In addition, the capability for handling multiple element types will be added, as the inviscid solver currently uses on tetrahedral elements.

Outline

In the remaining chapters, the extensions to the high-order streamline/upwind Petrov-Galerkin (SUPG) solver for viscous flows and mixed element types is detailed. Chapter II presents the methodology utilized in this research. This includes a brief presentation of the compressible Navier-Stokes equations with a modified SA turbulence model as well as a description of the SUPG discretization. In addition, strategies for generating curved high-order meshes in two and three dimensions are presented. To end the chapter, a computationally efficient methodology for calculating the distance to possibly curved surfaces is described. Chapter III presents the method of manufactured solutions as a means of code verification. Results are shown demonstrating proper order of accuracy characteristics for both two and three dimensional solutions. Next, Chapter IV explores some issues that arise when using high-order schemes on curved meshes. Results from several numerical test cases are presented in Chapter V. These results include time dependent and

steady state flows as well as inviscid, laminar, and turbulent results. Finally, conclusions are summarized in Chapter VI and recommendations for future work are made.

CHAPTER II

METHODOLOGY

Governing Equations

The compressible Reynolds Averaged Navier-Stokes equations coupled with the one equation of the modified Spalart and Allmaras turbulence model [15, 39] can be written in the conservative form as

$$\frac{\partial \boldsymbol{Q}(\boldsymbol{x},t)}{\partial t} + \nabla \cdot (\boldsymbol{F}_e(\boldsymbol{Q}) - \boldsymbol{F}_v(\boldsymbol{Q}, \nabla \boldsymbol{Q})) = \boldsymbol{S}(\boldsymbol{Q}, \nabla \boldsymbol{Q}) \qquad \text{in} \quad \Omega \tag{II.1}$$

where Ω is a bounded domain. The vector of conservative flow variables $\boldsymbol{Q}$, the inviscid and viscous Cartesian flux vectors, $\boldsymbol{F}_e$ and $\boldsymbol{F}_v$, are defined by

$$\boldsymbol{Q} = \begin{Bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \\ \rho \tilde{\nu} \end{Bmatrix} \qquad \boldsymbol{F}_e^x = \begin{Bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ (\rho E + p)u \\ \rho u \tilde{\nu} \end{Bmatrix} \qquad \boldsymbol{F}_e^y = \begin{Bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vw \\ (\rho E + p)v \\ \rho v \tilde{\nu} \end{Bmatrix} \qquad \boldsymbol{F}_e^z = \begin{Bmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + p \\ (\rho E + p)w \\ \rho w \tilde{\nu} \end{Bmatrix} \tag{II.2}$$

$$\boldsymbol{F}_v^x = \begin{Bmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ \tau_{xz} \\ u\tau_{xx} + v\tau_{xy} + w\tau_{xz} + \kappa\frac{\partial T}{\partial x} \\ \frac{1}{\sigma}\mu(1+\psi)\frac{\partial \tilde{\nu}}{\partial x} \end{Bmatrix} \quad \boldsymbol{F}_v^y = \begin{Bmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \\ \tau_{yz} \\ u\tau_{xy} + v\tau_{yy} + w\tau_{yz} + \kappa\frac{\partial T}{\partial y} \\ \frac{1}{\sigma}\mu(1+\psi)\frac{\partial \tilde{\nu}}{\partial y} \end{Bmatrix}$$

$$\boldsymbol{F}_v^z = \begin{Bmatrix} 0 \\ \tau_{xz} \\ \tau_{yz} \\ \tau_{zz} \\ u\tau_{xz} + v\tau_{yz} + w\tau_{zz} + \kappa\frac{\partial T}{\partial z} \\ \frac{1}{\sigma}\mu(1+\psi)\frac{\partial \tilde{\nu}}{\partial z} \end{Bmatrix} \tag{II.3}$$

where ρ, p, and E denote the fluid density, pressure, and specific total energy per unit mass, respectively. The vector $\boldsymbol{u} = (u, v, w)$ represents the Cartesian velocity vector and $\tilde{\nu}$ represents the turbulence working variable in the modified SA model. The pressure is determined by the equation of state for an ideal gas,

$$p = (\gamma - 1)\left(\rho E - \frac{1}{2}\rho(u^2 + v^2 + w^2)\right) \tag{II.4}$$

where γ is defined as the ratio of specific heats, which is 1.4 for air. Temperature and thermal conductivity are represented by T and κ, respectively, and are related to the total energy and velocity

as

$$\kappa T = \gamma \left(\frac{\mu}{Pr} + \frac{\mu_T}{Pr_T} \right) \left(E - \frac{1}{2} \left(u^2 + v^2 + w^2 \right) \right) \tag{II.5}$$

where Pr and Pr_T are the Prandtl and turbulent Prandtl number that are set to be 0.72 and 0.9 respectively. The fluid viscous stress tensor, τ, is defined for a Newtonian fluid as

$$\tau_{ij} = (\mu + \mu_T) \left(\frac{\partial \boldsymbol{u}_i}{\partial \boldsymbol{x}_j} + \frac{\partial \boldsymbol{u}_j}{\partial \boldsymbol{x}_i} - \frac{2}{3} \frac{\partial \boldsymbol{u}_k}{\partial \boldsymbol{x}_k} \delta_{ij} \right) \tag{II.6}$$

where δ_{ij} is the Kronecker delta and subscripts i, j, k refer to the Cartesian coordinate components for $\boldsymbol{x} = (x, y, z)$. In addition, μ refers to the fluid dynamic viscosity and is obtained via Sutherland's law, while μ_T denotes a turbulence eddy viscosity, which is obtained by

$$\mu_T = \begin{cases} \rho \tilde{\nu} f_{v1} & \text{if} \quad \tilde{\nu} \geq 0 \\ 0 & \text{if} \quad \tilde{\nu} < 0 \end{cases} \tag{II.7}$$

The source term, $\boldsymbol{S}$, in equation (II.1) has zero components for the continuity, momentum and energy equations, and takes the following form for the turbulence model equation [15, 39]

$$S_T = c_{b1} \tilde{S} \mu \psi - c_{w1} \rho f_w \left(\frac{\mu \psi}{\rho d} \right)^2 + \frac{1}{\sigma} c_{b2} \rho \nabla \tilde{\nu} \cdot \nabla \tilde{\nu} - \frac{\mu}{\rho \sigma} (1 + \psi) \nabla \rho \cdot \nabla \tilde{\nu} \tag{II.8}$$

The parameters for the production and destruction components of the modified SA turbulence model are given as

$$\tilde{S} = \begin{cases} S + \hat{S} & \text{if} \quad \hat{S} \geq -c_{v2}S \\ S + \dfrac{S(c_{v2}^2 + c_{v3}\hat{S})}{(c_{v3} - 2c_{v2})S - \hat{S}} & \text{if} \quad \hat{S} < -c_{v2}S \end{cases} \tag{II.9}$$

$$S = \sqrt{\boldsymbol{\omega} \cdot \boldsymbol{\omega}} \qquad \hat{S} = \frac{\mu\psi}{\rho\kappa_T^2 d^2} f_{v2} \qquad f_{v1} = \frac{\psi^3}{\psi^3 + c_{v1}^3} \qquad f_{v2} = 1 - \frac{\psi}{1 + \psi f_{v1}}$$

and

$$r = \frac{\mu\psi}{\rho\tilde{S}\kappa_T^2 d^2} \qquad g = r + c_{w2}(r^6 - r) \qquad f_w = g\left(\frac{1 + c_{w3}^6}{g^6 + c_{w3}^6}\right)^{1/6} \tag{II.10}$$

respectively. Here $\boldsymbol{\omega}$ denotes the vorticity vector, $\nabla \times \boldsymbol{u}$, while d refers to the distance to viscous wall at a specific location and must account for the curvature of the actual boundaries. The variable ψ in the equations is designed to remove the effects of a negative turbulence working variable on the robustness of the turbulence model as it is discretized by a high-order spatial discretization scheme. This variable ψ is given by

$$\psi = \begin{cases} 0.05 \ln(1 + e^{20\mathcal{X}}) & \text{if} \quad \mathcal{X} \leq 10 \\ \mathcal{X} & \text{if} \quad \mathcal{X} > 10 \end{cases} \tag{II.11}$$

where

$$\mathcal{X} = \frac{\rho\tilde{v}}{\mu} \tag{II.12}$$

such that it remains positive or becomes zero as the turbulence working variable goes negative, thus preventing the instability issue caused by unbounded turbulence eddy viscosity. The constants in

the modified SA model to close the main flow equations are given as

$$c_{b1} = 0.1355 \qquad \sigma = 2/3 \qquad c_{b2} = 0.622 \qquad \kappa_T = 0.41 \qquad c_{w1} = \frac{c_{b1}}{\kappa_T^2} + \frac{1 + c_{b2}}{\sigma}$$
$$c_{w2} = 0.3 \qquad c_{w3} = 2 \qquad c_{v1} = 7.1 \qquad c_{v2} = 0.7 \qquad c_{v3} = 0.9 \tag{II.13}$$

In the case of laminar flow, the governing equations reduce to the compressible Navier-Stokes equations, where the turbulence model equation vanishes and the turbulence eddy viscosity in the fluid viscous stress tensor and the thermal conduction terms is set to zero.

Discretization

The computational domain Ω is partitioned into a tessellation of non-overlapping elements, such that $\Omega = \bigcup_i \Omega_i$, where Ω_i refers to the volume of an element i in the computational mesh. The Galerkin finite-element approximation is expanded as a series of Lagrangian basis functions [40], ϕ_j, and solution coefficients for element i as

$$\boldsymbol{Q}_i(\boldsymbol{x}) = \sum_j \widehat{\boldsymbol{Q}}_{i_j} \phi_j(\boldsymbol{x}) \tag{II.14}$$

Here, the summation is over the nodes comprising element i, where $\widehat{\boldsymbol{Q}}_i$ is the value of the dependent variables at the nodes of the element. Because the set of basis functions is defined in a master element spanning between $\{0 < \xi, \eta, \zeta < 1\}$, a coordinate mapping from the reference to a physical element is required. The reference-to-physical transformation and the corresponding Jacobian J_i

associated with each element i are given by

$$\boldsymbol{x}_i = \sum_j \hat{\boldsymbol{x}}_{ij}\phi_j(\xi,\eta,\zeta) \qquad J_i = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} & \frac{\partial x}{\partial \zeta} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} & \frac{\partial y}{\partial \zeta} \\ \frac{\partial z}{\partial \xi} & \frac{\partial z}{\partial \eta} & \frac{\partial z}{\partial \zeta} \end{bmatrix} \tag{II.15}$$

where $\hat{\boldsymbol{x}}_i$ represent the element-wise geometric mapping coefficients.

In the streamline/upwind Petrov-Galerkin method [16–36] the system of equations is written as a weighted residual scheme,

$$\begin{aligned} 0 = & \iiint_\Omega \phi \left[\frac{\partial \boldsymbol{Q}}{\partial t} + \nabla \cdot \Big(\boldsymbol{F}_e(\boldsymbol{Q}) - \boldsymbol{F}_v(\boldsymbol{Q}, \nabla \boldsymbol{Q}) \Big) - \boldsymbol{S}(\boldsymbol{Q}, \nabla \boldsymbol{Q}) \right] \mathrm{d}\Omega \\ & + \sum_i \iiint_{\Omega_i} \left[\left(\frac{\partial \phi}{\partial x}[\boldsymbol{A}] + \frac{\partial \phi}{\partial y}[\boldsymbol{B}] + \frac{\partial \phi}{\partial z}[\boldsymbol{C}] \right) [\boldsymbol{\tau}] \left(\frac{\partial \boldsymbol{Q}}{\partial t} + \nabla \cdot \Big(\boldsymbol{F}_e(\boldsymbol{Q}) - \boldsymbol{F}_v(\boldsymbol{Q}, \nabla \boldsymbol{Q}) \Big) - \boldsymbol{S}(\boldsymbol{Q}, \nabla \boldsymbol{Q}) \right) \right] \mathrm{d}\Omega_i \end{aligned} \tag{II.16}$$

where $[\boldsymbol{A}]$, $[\boldsymbol{B}]$, and $[\boldsymbol{C}]$ are the inviscid flux Jacobians, and $[\boldsymbol{\tau}]$ is the stabilization matrix [17]. The weighting function, ϕ, is defined using the same basis functions as the dependent variables so that without the stabilizing term a Galerkin-type method would result. The stabilization term is added to compensate for a lack of dissipation in the stream-wise direction, thereby preventing oscillations that commonly plague the Galerkin method for convection-dominated flows. Note that in the stabilization term, the integration is strictly over the element interiors due to lack of differentiability of the basis functions at the element boundaries. For inviscid flows, $[\boldsymbol{\tau}]$ can be

obtained using the following definitions [41]:

$$[\boldsymbol{\tau}]^{-1} = \sum_i \left| \frac{\partial \phi_i}{\partial x}[\boldsymbol{A}] + \frac{\partial \phi_i}{\partial y}[\boldsymbol{B}] + \frac{\partial \phi_i}{\partial z}[\boldsymbol{C}] \right| \tag{II.17}$$

$$\left| \frac{\partial \phi_i}{\partial x}[\boldsymbol{A}] + \frac{\partial \phi_i}{\partial y}[\boldsymbol{B}] + \frac{\partial \phi_i}{\partial z}[\boldsymbol{C}] \right| = [\boldsymbol{T}][\boldsymbol{\Lambda}][\boldsymbol{T}]^{-1} \tag{II.18}$$

where $[\boldsymbol{T}]$ is the matrix of right eigenvectors and $[\boldsymbol{\Lambda}]$ is the diagonal matrix of eigenvalues of the left hand side of equation (II.18). Note that many alternative stabilization matrices can be derived using flux functions often used in finite-volume schemes [35]. Specifically, flux functions such as flux-vector splitting [42–44] can be written as a sum of contributions, $\boldsymbol{f}^+$ and $\boldsymbol{f}^-$, whose eigensystems have positive and negative eigenvalues, respectively. Using these definitions, the absolute value matrix in equations (II.17) and (II.18) can be replaced by the difference of $\frac{\partial \boldsymbol{f}^+}{\partial \boldsymbol{Q}}$ and $\frac{\partial \boldsymbol{f}^-}{\partial \boldsymbol{Q}}$. Potential advantages of this approach are that differentiability, positivity, and conservation of total enthalpy [35, 45, 46] can be maintained.

For viscous flows, additional terms are required as the Reynolds number is decreased and the viscous terms become dominant [16, 47, 48]. The reason for this modification is that the weighted residual formulation, given in equation (II.16), requires that second derivatives be evaluated for the viscous flux term that is multiplied by the stabilization matrix. The evaluation of this term results in a discretization that is one order less than the nominal order of the rest of the scheme. As a consequence, when the viscous terms become dominant, the stabilization matrix must behave as $O(h^2)$ instead of $O(h)$ for the matrix described above. Theoretical analysis for scalar equations results in applying a multiplicative scaling to the stabilization parameter based on

the local Peclet number (an excellent discussion can be found in reference [49]). In the current research however, attempts to extend this methodology to systems of equations in a manner that maintains the proper order of accuracy has not proven to be robust.

The form of stabilization matrix used here is very similar to that given in reference [47], and can be motivated by first considering a scalar convection-diffusion equation given as

$$a\frac{\partial u}{\partial x} - \frac{\partial}{\partial x}\left(\nu\frac{\partial u}{\partial x}\right) = a\frac{\partial u}{\partial x} - \frac{\partial}{\partial x}\left(f_v\right) = 0 \tag{II.19}$$

where a is the convection speed and ν is the diffusion coefficient. An inviscid stabilization term that corresponds to that described in equation (II.17) is given by

$$\tau^{-1} = \sum_i \left|\frac{\partial \phi_i}{\partial x} a\right| \tag{II.20}$$

Here, it is evident that τ varies as $O(h)$, which is appropriate for the inviscid case but does not have the proper limiting behavior for viscous dominated cases. To achieve the proper asymptotic behavior, a viscous modification can be added so that τ^{-1} is now given as

$$\tau^{-1} = \sum_i \left(\left|\frac{\partial \phi_i}{\partial x} a\right| + \frac{\partial \phi_i}{\partial x}\nu\frac{\partial \phi_i}{\partial x}\right) \tag{II.21}$$

Here, it is apparent that the two terms comprising τ^{-1} vary as $|a/L|$ and ν/L^2, respectively, so that a proportionality relationship for τ can be written as

$$\tau \propto \frac{L^2}{|aL| + \nu} \tag{II.22}$$

With this formulation, when the convection term dominates, τ exhibits the $O(h)$ property, whereas for viscous dominated cases, τ varies as $O(h^2)$, which is the desired behavior.

To extend this approach to systems of equations, first note that the viscous flux in equation (II.19) can be written as

$$f_v = \nu \frac{\partial u}{\partial x} \tag{II.23}$$

so that ν in equation (II.21) can be expressed as

$$\nu = \frac{\partial f_v}{\partial \left(\frac{\partial u}{\partial x}\right)} \tag{II.24}$$

Extending this methodology to systems is accomplished by noting that the viscous terms for the Navier-Stokes equations can be written in the form

$$\boldsymbol{F}_v^x = \boldsymbol{G}_{1j} \frac{\partial \boldsymbol{Q}}{\partial \boldsymbol{x}_j}, \quad \boldsymbol{F}_v^y = \boldsymbol{G}_{2j} \frac{\partial \boldsymbol{Q}}{\partial \boldsymbol{x}_j}, \quad \boldsymbol{F}_v^z = \boldsymbol{G}_{3j} \frac{\partial \boldsymbol{Q}}{\partial \boldsymbol{x}_j} \tag{II.25}$$

where, for example

$$\boldsymbol{G}_{11} = \frac{\partial \boldsymbol{F}_v^x}{\partial \left(\frac{\partial \boldsymbol{Q}}{\partial x}\right)}, \quad \boldsymbol{G}_{12} = \frac{\partial \boldsymbol{F}_v^x}{\partial \left(\frac{\partial \boldsymbol{Q}}{\partial y}\right)}, \quad \boldsymbol{G}_{13} = \frac{\partial \boldsymbol{F}_v^x}{\partial \left(\frac{\partial \boldsymbol{Q}}{\partial z}\right)} \tag{II.26}$$

The resulting form for the viscous contribution to the stabilization matrix is finally given as

$$[\boldsymbol{\tau_v}]^{-1} = \sum_i \left(\begin{bmatrix} \frac{\partial \phi_i}{\partial x} & \frac{\partial \phi_i}{\partial y} & \frac{\partial \phi_i}{\partial z} \end{bmatrix} \begin{bmatrix} G_{11} & G_{12} & G_{13} \\ G_{21} & G_{22} & G_{23} \\ G_{31} & G_{32} & G_{33} \end{bmatrix} \begin{bmatrix} \frac{\partial \phi_i}{\partial x} \\ \frac{\partial \phi_i}{\partial y} \\ \frac{\partial \phi_i}{\partial z} \end{bmatrix} \right) \tag{II.27}$$

This formulation has the advantage that a mesh spacing parameter is not required and the transition from an inviscid- to a viscous-dominated stabilization matrix occurs naturally and is consistent with the discretization of the governing equations.

For inviscid flows, boundary conditions are applied weakly by converting the volume integral involving the flux terms in the first integral in equation (II.16) into a surface integral using the divergence theorem as

$$\begin{aligned} 0 = \sum_i \iiint_{\Omega_i} \left[\phi \frac{\partial \boldsymbol{Q}}{\partial t} + \nabla \phi \cdot \Big(\boldsymbol{F}_e(\boldsymbol{Q}) - \boldsymbol{F}_v(\boldsymbol{Q}, \nabla \boldsymbol{Q}) \Big) - \phi \boldsymbol{S}(\boldsymbol{Q}, \nabla \boldsymbol{Q}) \right] \mathrm{d}\Omega_i \\ + \iint_{\partial \Omega_i \cap \partial \Omega} \phi \Big(\boldsymbol{F}_e(\boldsymbol{Q}) - \boldsymbol{F}_v(\boldsymbol{Q}, \nabla \boldsymbol{Q}) \Big) \cdot \hat{n} \, \mathrm{d}\Gamma \\ + \sum_i \iiint_{\Omega_i} \left[\left(\frac{\partial \phi}{\partial x}[\boldsymbol{A}] + \frac{\partial \phi}{\partial y}[\boldsymbol{B}] + \frac{\partial \phi}{\partial z}[\boldsymbol{C}] \right) [\boldsymbol{\tau}] \left(\frac{\partial \boldsymbol{Q}}{\partial t} + \nabla \cdot \Big(\boldsymbol{F}_e(\boldsymbol{Q}) - \boldsymbol{F}_v(\boldsymbol{Q}, \nabla \boldsymbol{Q}) \Big) - \boldsymbol{S}(\boldsymbol{Q}, \nabla \boldsymbol{Q}) \right) \right] \mathrm{d}\Omega_i \end{aligned} \tag{II.28}$$

and subsequently evaluating the flux at the wall using zero normal velocity. A similar procedure is used for viscous flows with the exception that the velocities and turbulence working variable are currently set to zero on no-slip walls and a constant temperature assumption is used.

After discretization, the system of nonlinear algebraic equations is solved using a Newton-type method where the linear system is solved at each step using the Generalized Minimal Residual (GMRES) method [50] with ILU(k) preconditioning [51]. In addition, the three-dimensional parallel flow solver uses the standard MPI message-passing library for inter-processor communication [52] and meshes are partitioned using the METIS mesh partitioner [53].

For time-dependent flows, time integration is performed via an implicit, second-order backward difference formula (BDF2). For steady state problems, a local time-stepping method based on CFL number is incorporated to alleviate the stiffness of the system in the initial stages of the calculation. In many turbulent flow cases [54, 55], a simple CFL strategy has proven adequate, though not ideal, for achieving steady state convergence. In order to maintain stability in the transient solution, a constant CFL of one was maintained while the L_2 norm of the turbulence working variable residual continued to rise. When the turbulence residual began to decrease, the CFL was increased linearly to some maximum value, typically in the range of 100–200. Here, an attempt has been made to automate this procedure following a similar approach as described in reference [56]. In particular, the CFL at each time step is increased or decreased by a factor related to the change in the L_2 residual norm. At time step n the CFL is given by

$$\mathrm{CFL}^n = \min\left(\mathrm{CFL}^{n-1} \cdot \beta^{\frac{\|\boldsymbol{R}^{n-1}\|_2 - \|\boldsymbol{R}^n\|_2}{\|\boldsymbol{R}^{n-1}\|_2}}, \mathrm{CFL}_{\max}\right) \tag{II.29}$$

for some value $\beta > 1$. In the present work a value of $\beta = 2$ was used exclusively.

Mesh Curving Strategy

In the present work, each case begins with a P_1 (linear) mesh, where all element boundaries are linear. Additional nodes are then added to each element to generate a higher-order mesh. For strictly flat surfaces, this poses no problem. For curved surfaces, however, care must be taken to snap any new surface nodes to the original geometry. For anisotropic boundary-layer elements, surface snapping poses a problem in and of itself. With high aspect ratio elements, surface snapping alone often creates negative volumes; a means of displacing the interior nodes of the mesh is needed.

Two Dimensions

In two dimensions, curved boundaries are recaptured using the analytic definition of each boundary. For each case involving curved boundaries, additional code is required to snap higher order nodes to the original surface.

Next, a means of displacing interior nodes is required. The methodology outlined by Allen [57] is chosen because it is computationally efficient and easily implemented. For each mesh point, denoted p, the distance to each of the *nsurf* curved surfaces, as well as to the far-field surface, are computed. The distance to each curved surface is defined as

$$S^{p,ns} = \left| \boldsymbol{x}^{p} - \boldsymbol{x}^{p,ns} \right| \tag{II.30}$$

where $1 \leq ns \leq nsurf$, $\boldsymbol{x}^p$ is the position of point p, and $\boldsymbol{x}^{p,ns}$ is the position of the point on surface ns closest to point p. Similarly, the far-field distance for each point is defined as

$$S_F^p = \left| \boldsymbol{x}^p - \boldsymbol{x}^p_{farfield} \right| \tag{II.31}$$

From these quantities, a normalized distance scale is defined between each mesh point and curved surface as

$$\psi^{p,ns} = \frac{S^{p,ns}}{S_F^p + S^{p,ns}} \tag{II.32}$$

To deal with multiple curved surfaces, the displacement of each mesh point is defined based on a weighted combination of the displacements of the nearest point on each surface. As such, it is necessary to define the quantities

$$\begin{aligned} S^p_{min} &= \min_{1 \leq ns \leq nsurf} S^{p,ns} \\ S^{p,ns}_{surf} &= \frac{S^p_{min}}{S^{p,ns}} \end{aligned} \tag{II.33}$$

Smooth surface weighting functions are then defined as

$$\begin{aligned} S^p_{Total} &= \sum_{ns=1}^{nsurf} \left(S^{p,ns}_{surf} \right)^{ssc} \\ \varphi^{p,ns} &= \frac{S^{p,ns}_{surf}}{S^p_{Total}} \end{aligned} \tag{II.34}$$

Finally, the displacement of each mesh point is computed as

$$\Delta \boldsymbol{x}^p = \sum_{ns}^{nsurf} \varphi^{p,ns} (1 - \psi^{p,ns})^{st} \Delta \boldsymbol{x}^{p,ns} \tag{II.35}$$

where *st* controls the decay of the displacements away from the curved surface(s). Typical values for *st* range from 2 to 5, with negligible differences in the resulting displacements. The multi-surface scaling exponent *ssc* is not utilized since multiple curving surfaces are not needed in the present work. For elements with interior nodes, the locations of these nodes are not determined using the above procedure. Instead, coefficients for hierarchical basis functions are first determined using the nodes on the boundary of the element, and the coordinates of the interior nodes are determined by evaluating the basis functions at the appropriate positions in the reference element. This procedure minimizes non-linearity in the mapping between the reference and physical spaces that otherwise occurs if the interior nodes are positioned independently.

To illustrate the current mesh curving strategy, a demonstration mesh, shown in figure II.1, is considered for a circular geometry with viscous spacing normal to the surface of 1×10^{-3}. Figure II.2 shows a close-up view of the viscous boundary layer at various stages.

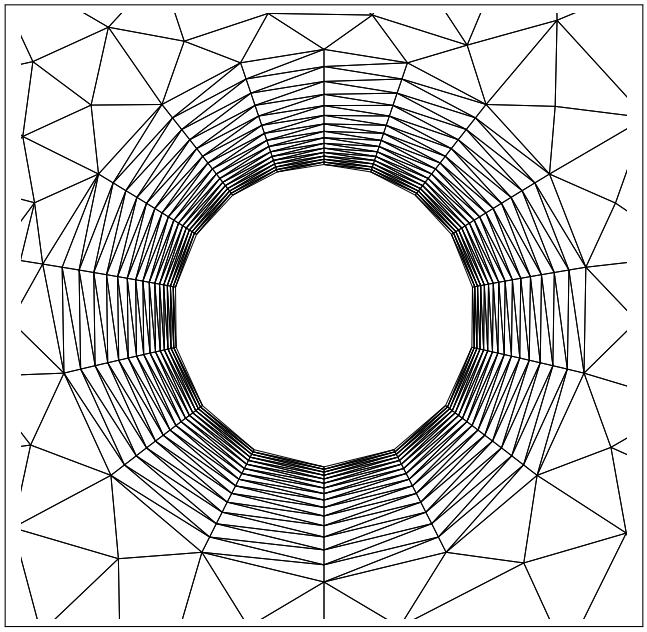
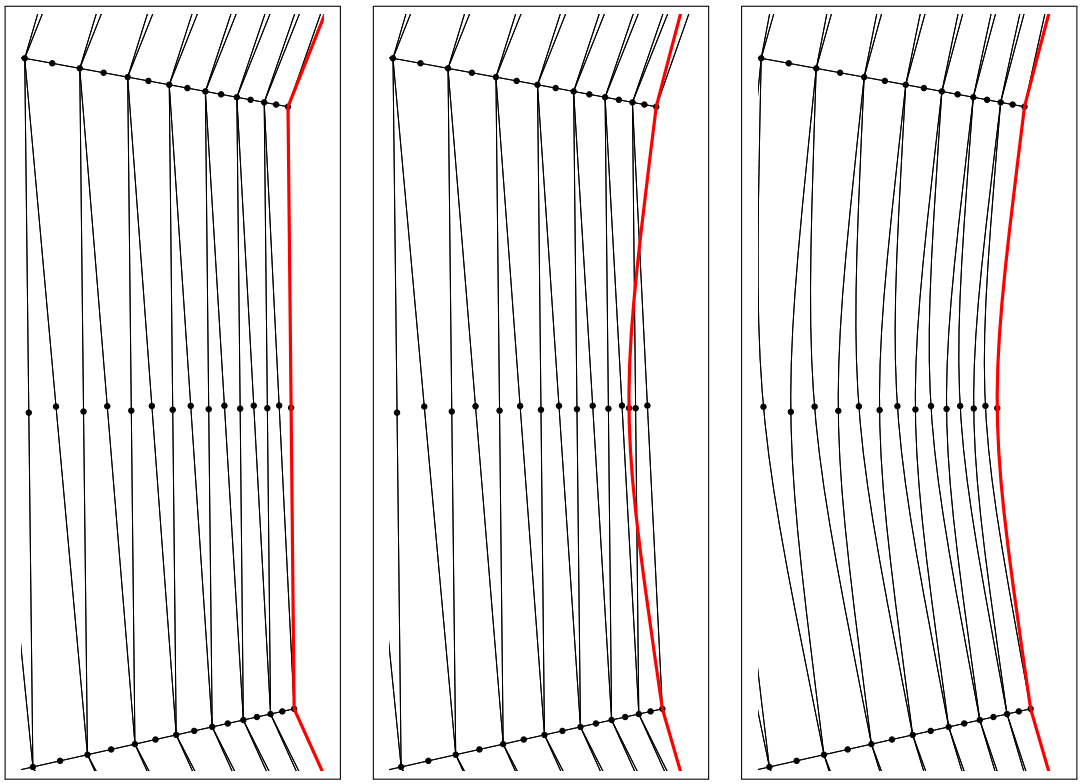


Figure II.1 Demonstration mesh for mesh curving strategy



(a) initial mesh (b) snapped mesh (c) final mesh

Figure II.2 Illustration of mesh curving strategy

Three Dimensions

While the strategy outlined above for capturing curved geometry is adequate for two dimensional cases, a more robust procedure is required for dealing with complex, three dimensional geometries. Developing new code on a case-by-case basis becomes impractical at best. To that end, three dimensional curved meshes are generated through a parametric definition provided by an external CAD engine. In particular, higher order points are projected onto the true geometry using CAPRI (Computational Analysis PRogramming Interface) [58]. CAPRI provides a programming interface for interrogating various commercial CAD engines. By providing access to the native parametric surface definition, CAPRI facilitates the projection of higher-order points onto a curved surface.

The mesh generating procedure begins with a CAD defined geometry. Similarly to the two dimensional case, a traditional linear mesh is generated on the geometry. Additional points are inserted into the linear mesh naively and then projected onto the CAD surface using CAPRI. To accommodate the properly curved surface elements, interior elements in the boundary layer are required to deform to avoid the generation of negative Jacobians. Here we make use of modified linear elasticity theory [59] which assumes that the computational mesh obeys the isotropic linear

elasticity relations, taken in the following form:

$$\begin{aligned}
&\frac{\partial}{\partial x}\left[d_{11}\frac{\partial \delta_x}{\partial x}+d_{12}\frac{\partial \delta_y}{\partial y}+d_{13}\frac{\partial \delta_z}{\partial z}\right]+\frac{\partial}{\partial y}\left[d_{44}\left(\frac{\partial \delta_x}{\partial y}+\frac{\partial \delta_y}{\partial x}\right)\right]+\frac{\partial}{\partial z}\left[d_{66}\left(\frac{\partial \delta_x}{\partial z}+\frac{\partial \delta_z}{\partial y}\right)\right]=0\\
&\frac{\partial}{\partial x}\left[d_{44}\left(\frac{\partial \delta_x}{\partial y}+\frac{\partial \delta_y}{\partial x}\right)\right]+\frac{\partial}{\partial y}\left[d_{21}\frac{\partial \delta_x}{\partial x}+d_{22}\frac{\partial \delta_y}{\partial y}+d_{23}\frac{\partial \delta_z}{\partial z}\right]+\frac{\partial}{\partial z}\left[d_{55}\left(\frac{\partial \delta_y}{\partial z}+\frac{\partial \delta_z}{\partial x}\right)\right]=0\\
&\frac{\partial}{\partial x}\left[d_{66}\left(\frac{\partial \delta_x}{\partial z}+\frac{\partial \delta_z}{\partial x}\right)\right]+\frac{\partial}{\partial y}\left[d_{55}\left(\frac{\partial \delta_y}{\partial z}+\frac{\partial \delta_z}{\partial y}\right)\right]+\frac{\partial}{\partial z}\left[d_{31}\frac{\partial \delta_x}{\partial x}+d_{32}\frac{\partial \delta_y}{\partial y}+d_{33}\frac{\partial \delta_z}{\partial z}\right]=0
\end{aligned} \tag{II.36}$$

where $\boldsymbol{\delta} = (\delta_x, \delta_y, \delta_z)$ denotes the nodal displacement vector in the Cartesian coordinate directions and the coefficients, d, are defined as

$$\begin{aligned}
d_{11} = d_{22} = d_{33} &= \frac{E(1-\upsilon)}{(1+\upsilon)(1-2\upsilon)}\\
d_{12} = d_{13} = d_{21} = d_{23} = d_{31} = d_{32} &= \frac{E\upsilon}{(1+\upsilon)(1-2\upsilon)}\\
d_{44} = d_{55} = d_{66} &= \frac{E}{2(1+\upsilon)}
\end{aligned} \tag{II.37}$$

where E represents Young's modulus and υ denotes Poisson's ratio. Physically, Young's modulus is a measure of the stiffness of a material, while Poisson's ratio is the ratio of transverse to axial strain. As such, the values E and υ may be used to influence the deformation of the interior mesh. In the present work, Poisson's ratio is set to a constant value of $\upsilon = 0.3$, while Young's modulus is set at each quadrature point to the inverse distance to the deforming surface. This strategy makes sense given the nature of the particular problem being solved: the stiffest regions of the interior mesh should be in the boundary layer near the solid wall. The overall mesh curving procedure is outlined in the flowchart in figure II.3.

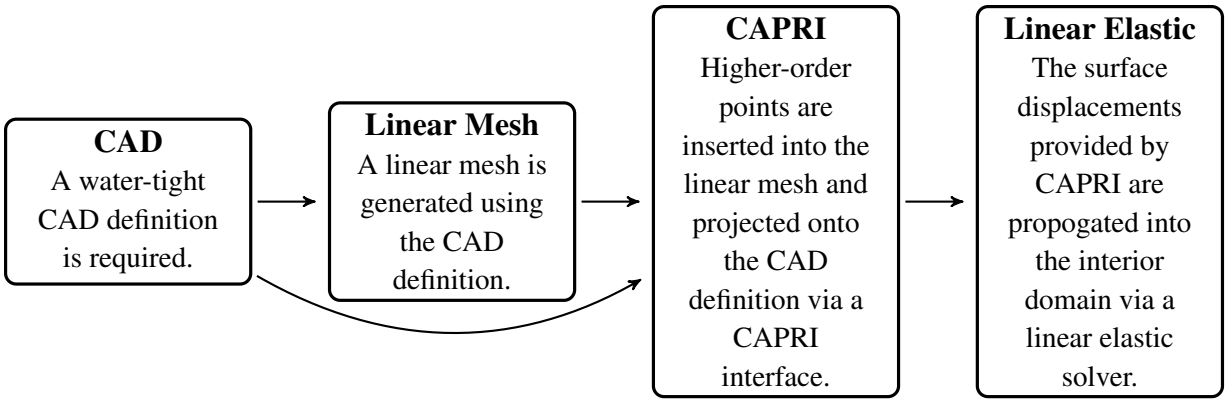


Figure II.3 Mesh curving procedure

To demonstrate both the effectiveness and the shortcomings of this procedure, consider an ONERA M6 swept wing mesh. Figure II.4 shows the projected geometry as well as the magnitude of the displacement vector in a mid-span slice of the interior mesh. It is clear that the displacement magnitude decays quickly as the distance from the wing increases. Additionally, the steps of the mesh curving strategy are clearly demonstrated by looking closely at the leading edge of the wing in figure II.5 as well as a cut of the interior mesh. In particular, figure II.5a shows the initial linear mesh, while figure II.5b shows the projected surface mesh as it clearly crosses into the boundary layer of the linear interior mesh. Finally, figure II.5c shows the valid final mesh.

Using the procedure described above, the majority of the interior mesh was deformed successfully, although a small number of invalid elements could not be eliminated near the tip of the swept wing. To circumvent this difficulty, the wing tip of the surface geometry was omitted from the surface projection in order to obtain a valid mesh.

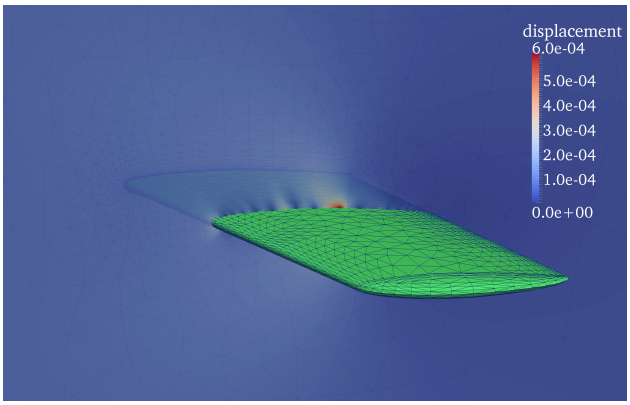
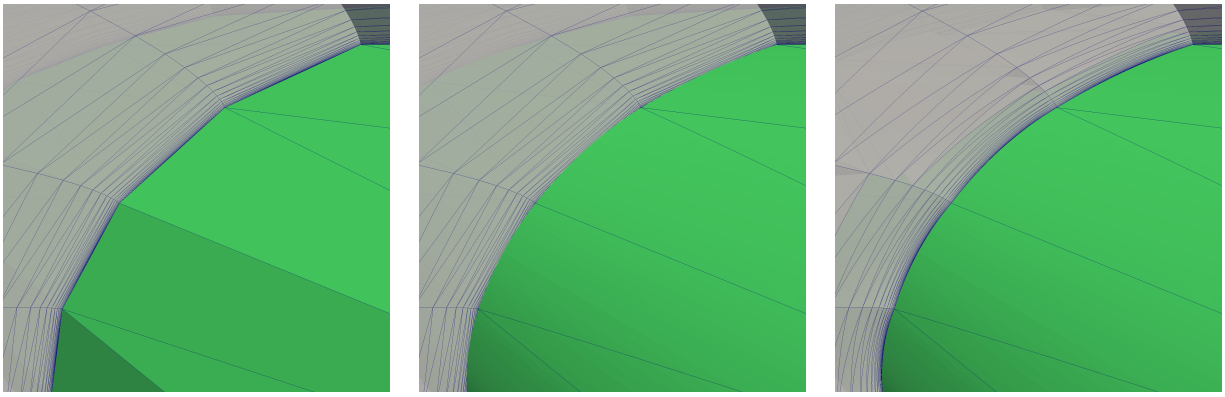


Figure II.4 Curved mesh displacements for ONERA M6 swept wing geometry



(a) Linear geometry.

(b) Mesh overlapping when only the surface is curved.

(c) Final mesh with curved interior elements.

Figure II.5 Demonstration of mesh curving on ONERA M6 swept wing geometry

Wall Distance

In the source term of the modified SA turbulence model, given in equations (II.8) to (II.10), the distance to the viscous wall is required at every point in the mesh. When dealing with higher order curved geometry, it is important that these quantities reflect the curved surface definition. As such, an efficient method for computing the distance to a curved three dimensional surface is needed.

Given a point in space, $\boldsymbol{x}^p$, the nearest point on a parametric surface, $\boldsymbol{S}(\xi, \eta)$, occurs when the vector between $\boldsymbol{x}^p$ and $\boldsymbol{S}(\xi, \eta)$ is orthogonal to the surface. This is represented by a system of two equations,

$$\begin{aligned} (\boldsymbol{S}(\xi, \eta) - \boldsymbol{x^p}) \cdot \boldsymbol{S}_\xi = 0 \\ (\boldsymbol{S}(\xi, \eta) - \boldsymbol{x^p}) \cdot \boldsymbol{S}_\eta = 0 \end{aligned} \tag{II.38}$$

where $\boldsymbol{S}_\xi = \frac{\partial \boldsymbol{S}(\xi,\eta)}{\partial \xi}$ and $\boldsymbol{S}_\eta = \frac{\partial \boldsymbol{S}(\xi,\eta)}{\partial \eta}$. Therefore, finding the distance between a point and a surface involves finding the roots, (ξ^*, η^*), of equation (II.38) via Newton's method. The distance between the point and the surface is then given as

$$d = \left| \boldsymbol{S}(\xi^*, \eta^*) - \boldsymbol{x}^p \right| \tag{II.39}$$

Since the surface elements are finite parametric surfaces, the root is only valid if it lies within the parametric range of the element. When the root lies outside the range of the element, the nearest point on the surface lies along an edge of the element. The procedure above is easily modified to

find the nearest point on a one-dimensional parametric edge. If the root on the edge lies outside the parametric range of the edge, the nearest point on the edge lies at one of the endpoints.

Since this work is dealing with discrete computational meshes, the distance from any point to a viscous wall involves finding the minimum distance amongst a large number of parametric surfaces. Solving a root-finding problem for every mesh point/surface element pair quickly becomes impractical. In this work, an octree data structure is used to alleviate this problem. Given n spatially defined objects, an octree is a data structure in which a three dimensional space is recursively subdivided into ever smaller containers such that $O(\log n)$ tree traversal algorithms may be utilized.

The octree begins as a cube encompassing all n objects. The cube is then subdivided into eight equal octants that act as containers; each octant may contain any of the objects that lie completely within its bounds. This process continues recursively until some stopping criteria is met. The octree drastically reduces the number of distance calculations that must be performed. For each mesh point, the cost of finding the distance to the surface is reduced from $O(n)$ to $O(\log n)$.

Even using the octree, many distance calculations must be performed between mesh points and the viscous surface. Due to the expense of these calculations for curved surface elements, surface points rather than surface elements are stored in the octree in this research. As such, for each node in the mesh, a nearest neighbor search is carried out to find the nearest surface node. From there, the elements and edges surrounding the nearest surface node are checked to find the true minimum distance to the viscous surface.

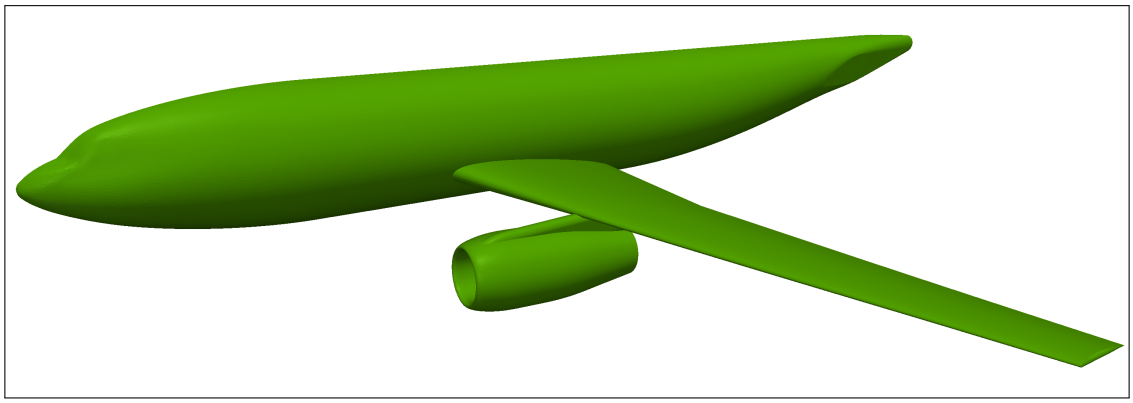
Since points are dimensionless, an appropriate stopping criteria is needed to prevent indefinite recursion of the octree. In this work, a minimum number of objects must lie in a container

before that container is considered for subdivision. This strategy prevents the octree from becoming too sparse as well. Through some brief experimentation, a minimum object count of 20 for each container has proven adequate.

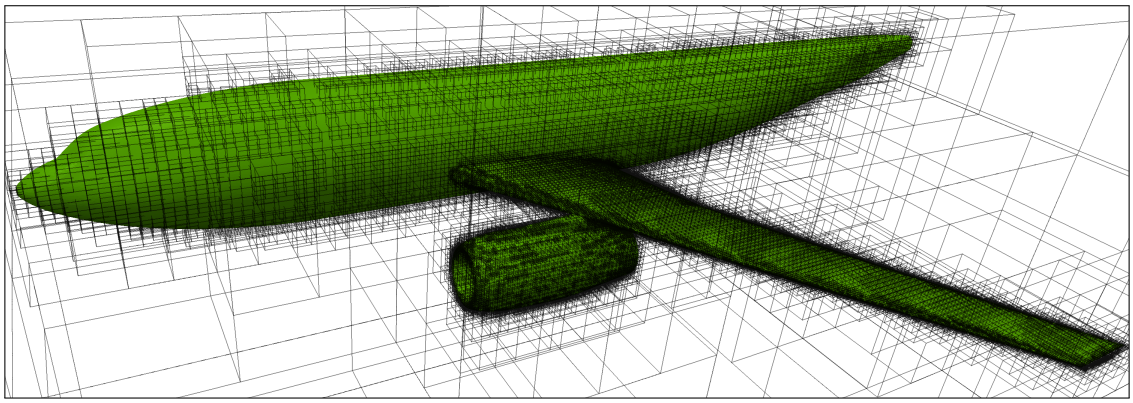
Once all points are inserted into the octree, each of the containers is contracted such that it forms a minimum bounding box of the points within it or any of its sub-containers. This simple process serves to minimize the number of containers that must be visited during nearest neighbor searches. To illustrate this procedure, a non-trivial geometry is presented in figure II.6. This geometry consists of 238,687 surface nodes and 476,429 surface triangles. The initial octree containing all of the surface nodes is shown in figure II.6b and consists of 23,880 containers in 13 levels. Finally, figure II.6c shows the final octree with contracted containers.

A potential problem arises from the choice to put the surface points, rather than the elements, into the octree. As illustrated in figure II.7 in two dimensions, the wrong node may be found during the nearest neighbor search. Here, the nearest point on the surface is clearly on the edge nearest node 101. The nearest surface node, however is node 41. During the subsequent search through the neighboring edges of node 41, the proper face will never be checked. To avoid this problem, a surface node is only considered during the nearest neighbor search if at least one face connected to that node is visible from the point in question. Visibility is checked by examining the dot product between the outward-pointing unit normal vector of the face (at the surface node) and the unit vector from the surface node to the mesh point. A positive dot product indicates visibility.

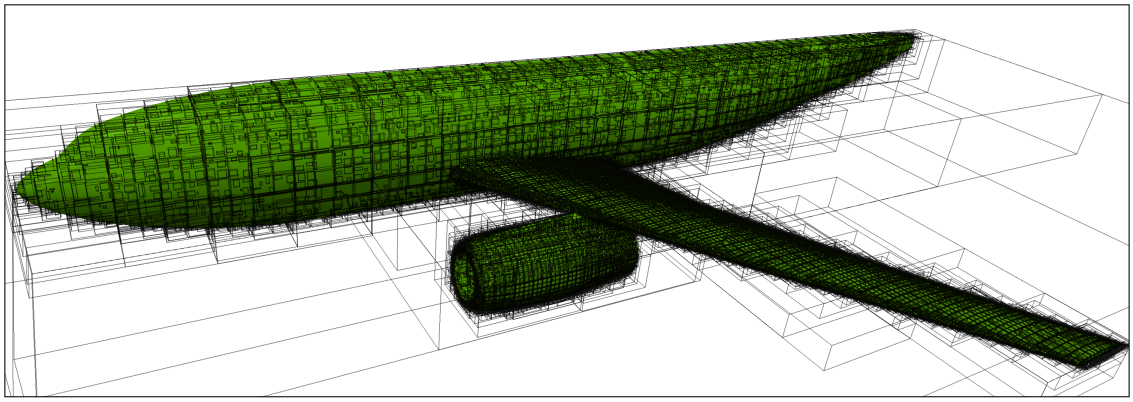
Ultimately, the entire process described above must be performed in a parallel computing environment. Initially, each process contains a portion of the computational mesh which may or



(a) Geometry definition



(b) Initial octree



(c) Contracted octree

Figure II.6 Demonstration of octree used for distance calculation

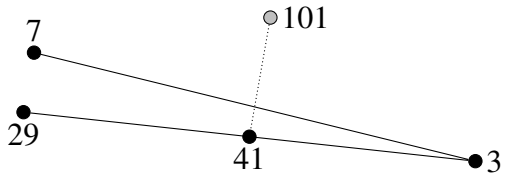


Figure II.7 Issue with nearest neighbor search

may not include any viscous surfaces. To facilitate parallel distance computation, a surface mesh is constructed on each process that contains only the surfaces of interest native to that process. These surface meshes are then distributed to all processes such that each process has access to the full surface mesh of interest. Each process then builds the complete octree for its copy of the surface mesh. Finally, each process uses the octree to calculate the distance to the surface for each node it owns. At this point, the octree is discarded since it is no longer needed.

CHAPTER III

CODE VERIFICATION

In the current work, the method of manufactured solutions is used extensively for code verification. The method of manufactured solutions is a general procedure for generating non-trivial exact solutions to a PDE or system of PDEs. Consider a PDE system in general form,

$$\mathcal{D}u = \mathcal{S} \tag{III.1}$$

where $\mathcal{D}$ is the differential operator, u is the solution, and $\mathcal{S}$ is the source term. In order to find an exact solution to this system, one chooses a source term and, using methods from applied mathematics, inverts the differential operator to solve for u. With the method of manufactured solutions, on the other hand, one chooses, or "manufactures," a solution that is substituted into the governing equations to obtain a source term $\mathcal{S}$. This is accomplished by simply applying the differential operator to the chosen solution.

Salari and Knupp [60] have outlined a comprehensive set of guidelines for choosing manufactured solutions:

1. The manufactured solutions should be composed of smooth analytic functions. Possible choices include polynomial, trigonometric, and exponential functions. This criteria ensures

that the solution can be easily computed at all spatial and temporal locations within the computational domain.

2. The solution should be general enough to exercise every term in the governing equations.

3. The solution should have a sufficient number of non-trivial derivatives. For instance, when verifying a code that is theoretically second order accurate, a linear solution would not provide a sufficient test since second order accuracy would be assured.

4. The solution should contain no singularities, discontinuities, or steep gradients. These would require unnecessarily high grid resolution in order to obtain a converged solution.

It is important to note that the manufactured solution need not be physically realistic since the code verification process is purely a mathematical exercise. In fact, non-physical solutions are much easier to generate, and are therefore recommended for the method of manufactured solutions [60]. The observed order between a coarse and fine mesh is calculated via

$$p = \frac{\log\left(\frac{\mathcal{E}_{\text{coarse}}}{\mathcal{E}_{\text{fine}}}\right)}{\log\left(\frac{h_{\text{coarse}}}{h_{\text{fine}}}\right)} \tag{III.2}$$

where $\mathcal{E}$ represents the L_1, L_2, or L_∞ norm of the solution error. For a mesh with N degrees of freedom, the mesh spacing is approximated as $h \cong \left(\frac{1}{N}\right)^{1/2}$ for two-dimensional flows and as $h \cong \left(\frac{1}{N}\right)^{1/3}$ for three-dimensional flows.

Two Dimensions

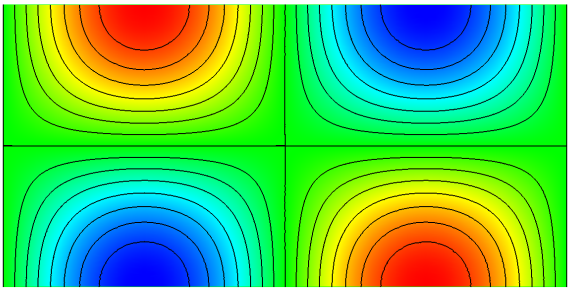
To assess the order of accuracy for the two-dimensional Petrov-Galerkin code, trigonometric functions, given in equation (III.3) and shown in figure III.1, are used to derive the forcing function.

$$\begin{aligned}
\rho(x, y) &= A_\rho(1 + \sin(\pi x)\cos(\pi x)\sin(\pi y)\cos(\pi y)) \\
u(x, y) &= A_u(1 + \sin(2\pi x)\cos(2\pi x)\sin(2\pi y)\cos(2\pi y)) \\
v(x, y) &= A_v(1 + \cos(2\pi x)\cos(2\pi x)\cos(2\pi y)\cos(2\pi y)) \\
T(x, y) &= A_T(1 + \sin(2\pi x)\sin(2\pi x)\sin(2\pi y)\sin(2\pi y))
\end{aligned} \tag{III.3}$$

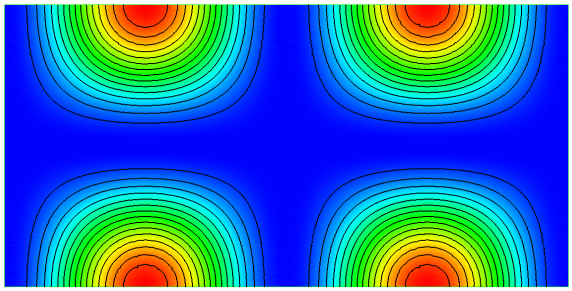
In evaluating the order of accuracy, a series of sequentially refined meshes consisting of 585, 2193, and 8481 nodes are used where the coarsest mesh is shown in figure III.2. As seen, the mesh consists of highly stretched triangles along the mid-section to represent spacings commonly used for viscous flows. The normal spacings at the center of the coarse, medium, and fine meshes are 1.2×10^{-4}, 6.0×10^{-5}, and 3.0×10^{-5}, respectively. It should be noted that similar experiments have been conducted using meshes with near-equilateral triangles with similar results.

Table III.1 demonstrates the order of accuracy obtained for the Euler equations. As seen, when linear (P_1) and cubic (P_3) polynomials are utilized, the design-order of accuracy is slightly exceeded, whereas for quadratic (P_2) polynomials it is slightly lower.

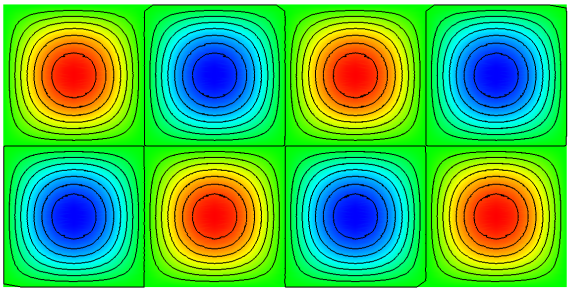
Recall that for the Petrov-Galerkin scheme, the stabilization matrix must be scaled properly as the Reynolds number is decreased and the viscous terms become dominant. Figure III.3 depicts the observed order of accuracy for schemes using linear, quadratic, and cubic polynomials over



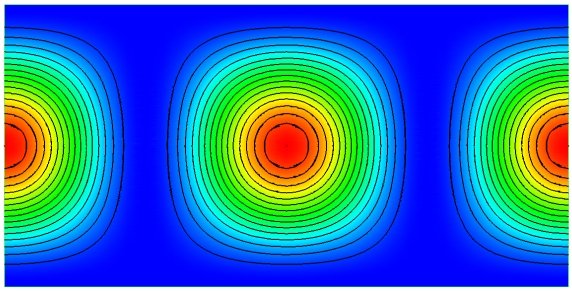
(a) Density, ρ



(b) Temperature, T



(c) x-velocity, u



(d) y-velocity, v

Figure III.1 Manufactured solution for compressible Euler and Navier-Stokes equations

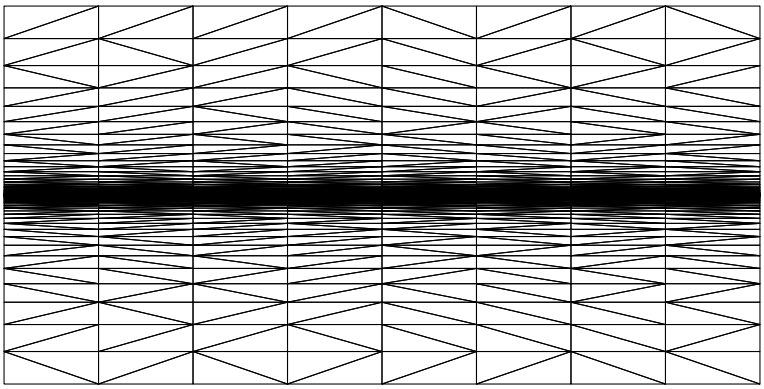
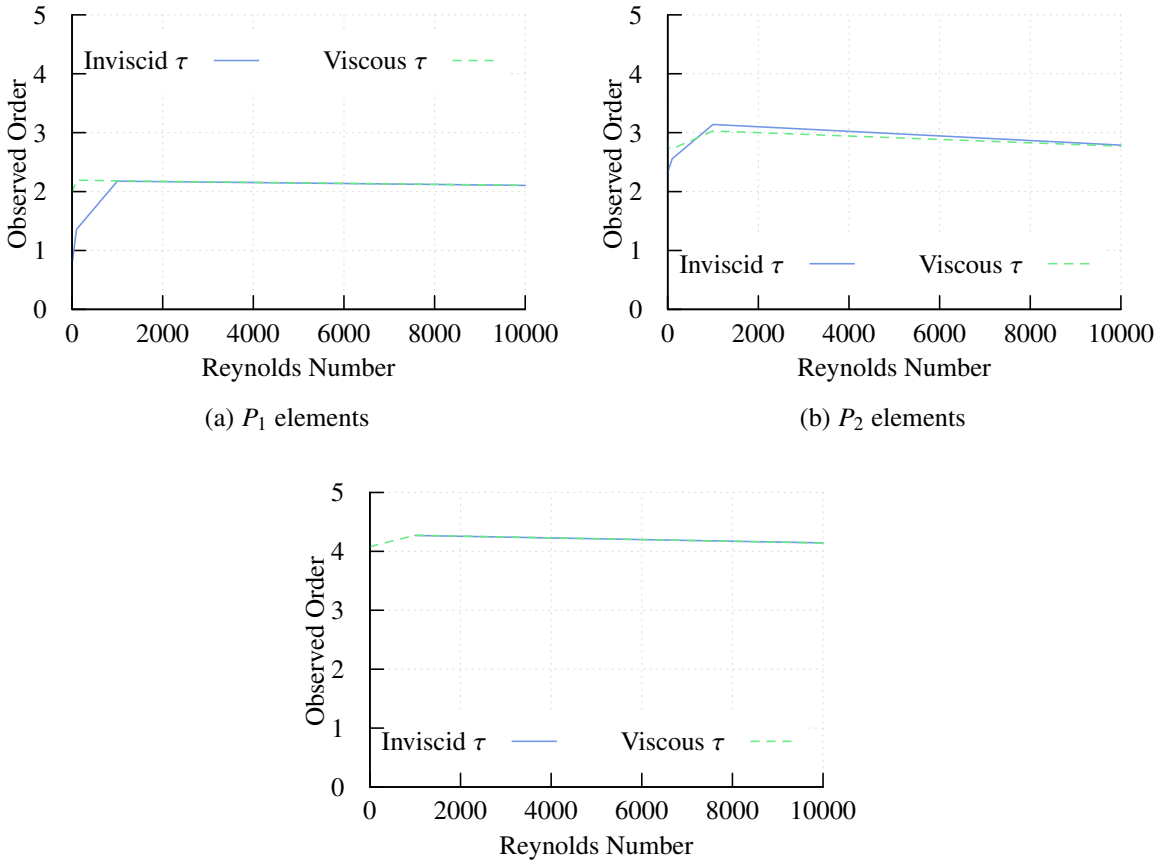


Figure III.2 Coarse mesh used for manufactured solution

Table III.1 Order of accuracy for the two-dimensional Euler equations

Nodes in Mesh		P_1 Elements		P_2 Elements		P_3 Elements	
		L_1	L_2	L_1	L_2	L_1	L_2
2,193/8,481	ρ:	2.04372345	2.08393604	2.81862585	2.76753487	4.21410239	4.17809007
	u:	2.04337131	2.06345087	2.66929471	2.69285056	4.18304794	4.18180332
	v:	2.00185491	2.01485236	2.48665495	2.40957259	4.19425997	4.18662066
	T:	2.16934669	2.17385020	3.01595459	2.95281295	4.21895592	4.21438761

a range of Reynolds numbers, with and without the scaling applied. It is seen that in all cases, failure to scale the stabilization matrix results in reduced order of accuracy as the Reynolds number approaches unity, although it agrees with the order obtained for the Euler equations for Reynolds numbers exceeding approximately 1000. Specifically, for linear elements at a Reynolds number of one, the obtained order of accuracy is 2.025 when the scaling is used and 0.848 when the scaling is not applied. Similarly, for quadratic elements, the obtained orders of accuracy are 2.867 and 2.285, respectively. For cubic elements, a stable solution is not obtained when neglecting the scaling, whereas an order of accuracy of 4.083 has been achieved with proper scaling.



(a) P_1 elements

(b) P_2 elements

(c) P_3 elements

Figure III.3 Observed order at varying Reynolds numbers

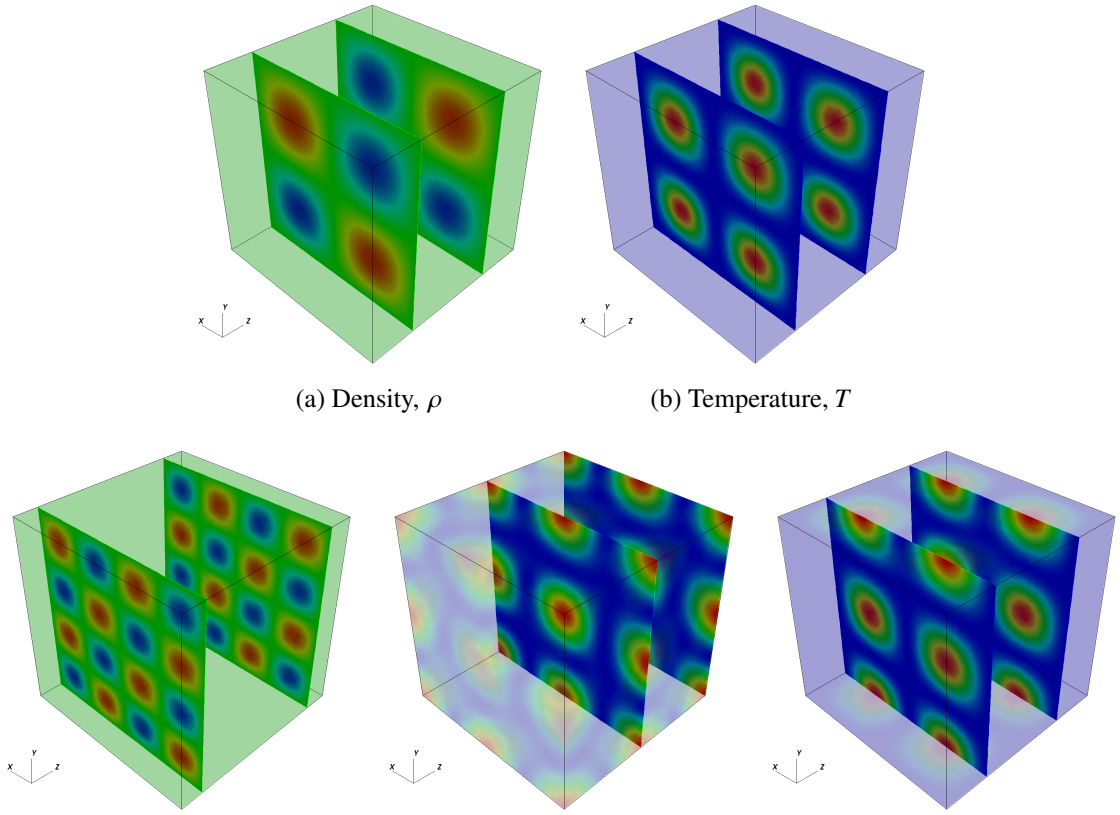
Three Dimensions

The order of accuracy for the three-dimensional solver is assessed in the same manner as the two-dimensional solver. Trigonometric functions similar to those in the previous section are utilized. In particular, the functions given in equation (III.4) and shown in figure III.4, are used to derive the forcing function.

$$
\begin{aligned}
\rho(x, y, z) &= A_\rho(1 + \sin(\pi x)\cos(\pi x)\sin(\pi y)\cos(\pi y)\sin(\pi z)\cos(\pi z)) \\
u(x, y, z) &= A_u(1 + \sin(2\pi x)\cos(2\pi x)\sin(2\pi y)\cos(2\pi y)\sin(2\pi z)\cos(2\pi z)) \\
v(x, y, z) &= A_v(1 + \cos(2\pi x)\cos(2\pi x)\cos(2\pi y)\cos(2\pi y)\cos(2\pi z)\cos(2\pi z)) \\
w(x, y, z) &= A_w(1 + \sin(2\pi x)\sin(2\pi x)\cos(2\pi y)\cos(2\pi y)\sin(2\pi z)\sin(2\pi z)) \\
T(x, y, z) &= A_T(1 + \sin(2\pi x)\sin(2\pi x)\sin(2\pi y)\sin(2\pi y)\sin(2\pi z)\sin(2\pi z))
\end{aligned}
\tag{III.4}
$$

Using the manufactured solution described in equation (III.4), an order of accuracy study is performed on a sequence of four tetrahedral meshes. The observed orders of accuracy for linear and quadratic elements are shown in Table III.2.

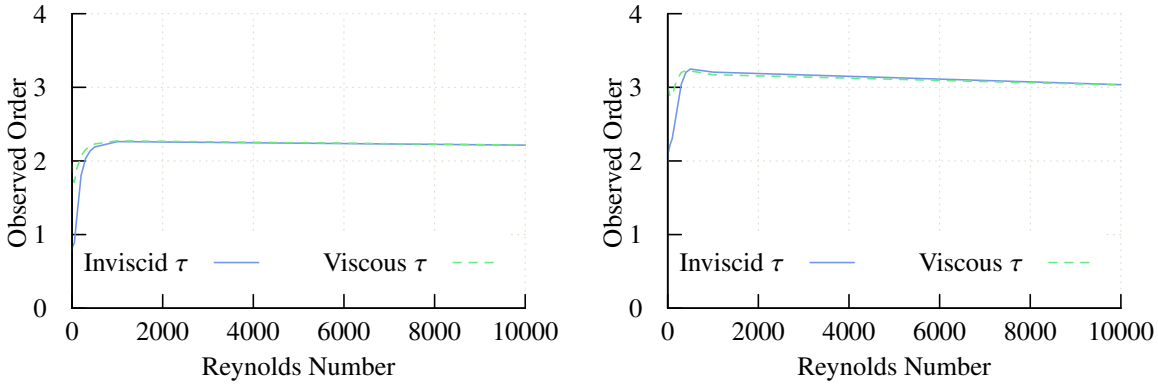
Figure III.5 shows the achieved order of accuracy for schemes using linear and quadratic polynomials over a range of Reynolds numbers, with and without the viscous scaling applied. As with the two dimensional results, failure to scale the stabilization matrix results in reduced order of accuracy at low Reynolds numbers, but agrees with that obtained for the Euler equations for high Reynolds numbers. For linear elements at a Reynolds number of one, the achieved order of accuracy is 1.958 when the scaling is used and 0.791 otherwise. Similarly, for quadratic elements, the obtained orders of accuracy are 2.996 and 1.986, respectively.



(a) Density, ρ (b) Temperature, T

(c) x-velocity, u (d) y-velocity, v (e) z-velocity, w

Figure III.4 Manufactured solution for compressible Euler equations



(a) P_1 elements (b) P_2 elements

Figure III.5 Observed order at varying Reynolds numbers

Table III.2 Order of accuracy for the three-dimensional Euler equations

Nodes in Mesh		P_1 Elements		P_2 Elements	
		L_1	L_2	L_1	L_2
2,930/18,676	ρ:	2.8254973503	2.9081784421	3.6002086308	3.6291123030
	u:	2.7171089951	2.8365915370	3.3875955556	3.4101041811
	v:	2.7637086966	2.8358031373	3.4066550847	3.4030405292
	w:	2.7047513828	2.7403120725	3.4496641072	3.4129722557
	T:	2.7276349293	2.7712926768	3.6618960118	3.6532938916
18,676/128,610	ρ:	2.3985001180	2.4203780776	3.5630517014	3.5411000748
	u:	2.4038197365	2.4484297214	3.4324111671	3.4206384785
	v:	2.3330247765	2.3593610505	3.4549756354	3.4219523571
	w:	2.3588440317	2.3779764038	3.5769450076	3.6000623938
	T:	2.3538178777	2.3693439273	3.5844259242	3.5374002503

CHAPTER IV

CURVED ELEMENTS

For viscous flows, a large percentage of nodes in the mesh may be contained within the boundary layer. For turbulent flows, experience with second-order finite-volume schemes indicates that as many as half of the mesh points may be located within this region. While the distribution of mesh points for higher-order algorithms may be somewhat different, a large percentage of nodes will still remain in the boundary layer as dictated by the physics of the flow field. Because of the typically small spacing normal to the surface, a high number of elements may consequently need to be curved to successfully accommodate the deformations required to accurately represent the geometry.

In references [61–63] it is demonstrated that when elements are curved, many additional degrees of freedom are required in the reference space to accurately represent a complete polynomial in physical space. Following the procedure described in reference [61], this can be demonstrated by first considering a linear mapping between the physical coordinates, (x, y), and the coordinates in the reference elements (r, s),

$$\begin{aligned} x_1(r,s) &= \alpha_1 + \alpha_2 r + \alpha_3 s \\ y_1(r,s) &= \beta_1 + \beta_2 r + \beta_3 s \end{aligned} \tag{IV.1}$$

Using a quadratic polynomial in physical space for the solution variables, direct substitution of the above equations for x and y results in a quadratic polynomial in the reference space as well

$$q_2(r,s) = \gamma_1 + \gamma_2 r + \gamma_3 s + \gamma_4 r^2 + \gamma_5 rs + \gamma_6 s^2 \tag{IV.2}$$

It is apparent that with a linear mapping, a quadratic polynomial can be accurately represented by a triangle with six degrees of freedom, which is typically used for third-order accurate schemes. However, if the edges of the triangle are curved, the mapping between physical space and the reference space becomes nonlinear, as shown in equation (IV.3) for a quadratic mapping

$$\begin{aligned} x_2(r,s) &= \alpha_1 + \alpha_2 r + \alpha_3 s + \alpha_4 r^2 + \alpha_5 rs + \alpha_6 s^2 \\ y_2(r,s) &= \beta_1 + \beta_2 r + \beta_3 s + \beta_4 r^2 + \beta_5 rs + \beta_6 s^2 \end{aligned} \tag{IV.3}$$

Substitution of these equations into a quadratic representation of the solution in physical space demonstrates that many more degrees of freedom are required in the reference space to faithfully represent the solution

$$q_4(r,s) = \gamma_1 + \gamma_2 r + \gamma_3 s + \gamma_4 r^2 + \gamma_5 rs + \gamma_6 s^2 + \Phi(r,s) \tag{IV.4}$$

where

$$\Phi(r,s) = \gamma_7 r^3 + \gamma_8 r^2 s + \gamma_9 rs^2 + \gamma_{10} s^3 + \gamma_{11} r^4 + \gamma_{12} r^3 s + \gamma_{13} r^2 s^2 + \gamma_{14} rs^3 + \gamma_{15} s^4 \tag{IV.5}$$

Comparing equation (IV.2) with equation (IV.4), it is apparent that when the boundaries of the element are curved, additional degrees of freedom are required in the mapped space to accurately represent the quadratic function in physical space.

A more rigorous approach to that given above is given in references [62, 63] where it is shown that a polynomial of degree n over a P-sided polygon with edges represented by m-degree polynomials requires the following degrees of freedom

$$N = \sum_{k=1}^{P} N_k - P \tag{IV.6}$$

where

$$N_k = \frac{1}{2}[(n+2)(n+1) - \mu_{mn}(n-m+2)(n-m+1)] \tag{IV.7}$$

and

$$\mu_{mn} = \begin{cases} 1 & \text{if} \quad m \leq n \\ 0 & \text{if} \quad m > n \end{cases} \tag{IV.8}$$

It is seen that to represent a quadratic function on a triangular element with quadratic sides requires 12 degrees of freedom, implying some terms in equation (IV.4) can be combined into bubble modes that are zero. A similar representation for a triangular element with three curved sides is given in reference [64].

The results above demonstrate that to obtain third-order accuracy over a quadratic-curved element could require similar storage and operation count as an element typically designed to achieve fifth-order accuracy. While this paints a rather pessimistic picture, it should be noted that,

in practice, the detrimental effect of curving the element boundaries is mitigated if the unrepresented terms in equation (IV.5) only produce errors below truncation error. As a result, the desired order of accuracy can still be achieved provided the edges are not excessively curved [65–67].

In order to evaluate the appropriateness of this claim for three dimensional elements, a straightforward test is designed. Here, uniformly spaced P_1 and P_2 hexahedral meshes are generated as shown in figure IV.1. Subsequently, the trigonometric function given by

$$f(\boldsymbol{x}) = \frac{1}{4}\left(1 + \sin\left(\frac{\pi x}{2}\right)\cos\left(\frac{\pi x}{2}\right)\sin\left(\frac{\pi y}{2}\right)\cos\left(\frac{\pi y}{2}\right)\sin\left(\frac{\pi z}{2}\right)\cos\left(\frac{\pi z}{2}\right)\right) \tag{IV.9}$$

is applied throughout the domain. A grid refinement study is then performed, whereby the function and its gradient are evaluated in an isoperimetric manner throughout the domain. Finally, the L_2 norm of the error is evaluated in order to determine order of accuracy. This procedure is applied in turn to similar meshes consisting entirely of tetrahedra, pyramids, and prisms.

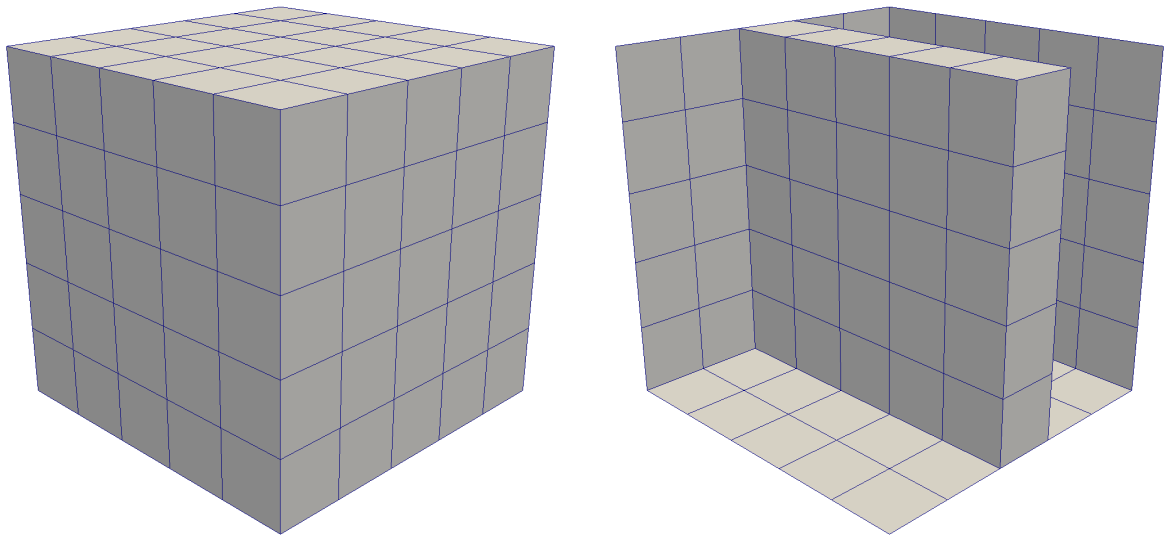


Figure IV.1 Uniform hexahedral mesh

Tables IV.1 and IV.2 show the order of accuracy results for straight, uniform meshes. These baseline results demonstrate that all element types achieve the appropriate order of accuracy of $P+1$ for function evaluations and P for gradient evaluations.

Table IV.1 Order of accuracy on P_1 uniform meshes

Nodes in Mesh	Tetrahedron		Pyramid		Prism		Hexahedron	
	$f(\boldsymbol{x})$	$\nabla f(\boldsymbol{x})$	$f(\boldsymbol{x})$	$\nabla f(\boldsymbol{x})$	$f(\boldsymbol{x})$	$\nabla f(\boldsymbol{x})$	$f(\boldsymbol{x})$	$\nabla f(\boldsymbol{x})$
216/512	1.91511	0.95737	1.92071	0.94616	1.92595	0.99488	1.93218	1.06936
512/1000	1.95313	0.97644	1.95622	0.96997	1.95907	0.99816	1.96251	1.04250
1000/2197	1.97254	0.98619	1.97435	0.98232	1.97601	0.99923	1.97803	1.02622
2197/4096	1.98361	0.99175	1.98468	0.98941	1.98567	0.99966	1.98688	1.01614
4096/8000	1.98963	0.99478	1.99031	0.99330	1.99094	0.99982	1.99170	1.01038

Table IV.2 Order of accuracy on P_2 uniform meshes

Nodes in Mesh	Tet		Pyramid		Prism		Hex	
	$f(\boldsymbol{x})$	$\nabla f(\boldsymbol{x})$	$f(\boldsymbol{x})$	$\nabla f(\boldsymbol{x})$	$f(\boldsymbol{x})$	$\nabla f(\boldsymbol{x})$	$f(\boldsymbol{x})$	$\nabla f(\boldsymbol{x})$
216/512	2.96777	1.96100	2.95342	1.95668	2.98692	1.98790	3.01914	2.00072
512/1000	2.98237	1.97848	2.97426	1.97595	2.99299	1.99347	3.01109	2.00036
1000/2197	2.98972	1.98739	2.98492	1.98588	2.99595	1.99622	3.00665	2.00020
2197/4096	2.99388	1.99247	2.99099	1.99155	2.99760	1.99776	3.00403	2.00011
4096/8000	2.99614	1.99524	2.99430	1.99465	2.99849	1.99859	3.00256	2.00007

With baseline results in hand, a transformation is applied to the uniform mesh as given by

$$\begin{aligned} x' &= x + \frac{1}{10}\sin(\pi x)\cos(\pi y)\cos(\pi z) \\ y' &= y + \frac{1}{10}\cos(\pi x)\sin(\pi y)\cos(\pi z) \\ z' &= z + \frac{1}{10}\cos(\pi x)\cos(\pi y)\sin(\pi z) \end{aligned} \tag{IV.10}$$

which produces the curved mesh shown in figure IV.2. The grid refinement study is repeated on the curved mesh, with results tabulated in Tables IV.3 and IV.4. The effect of the curved elements proves negligible in all cases, indicating that any unrepresented terms in the expansion of the basis functions fall below truncation error.

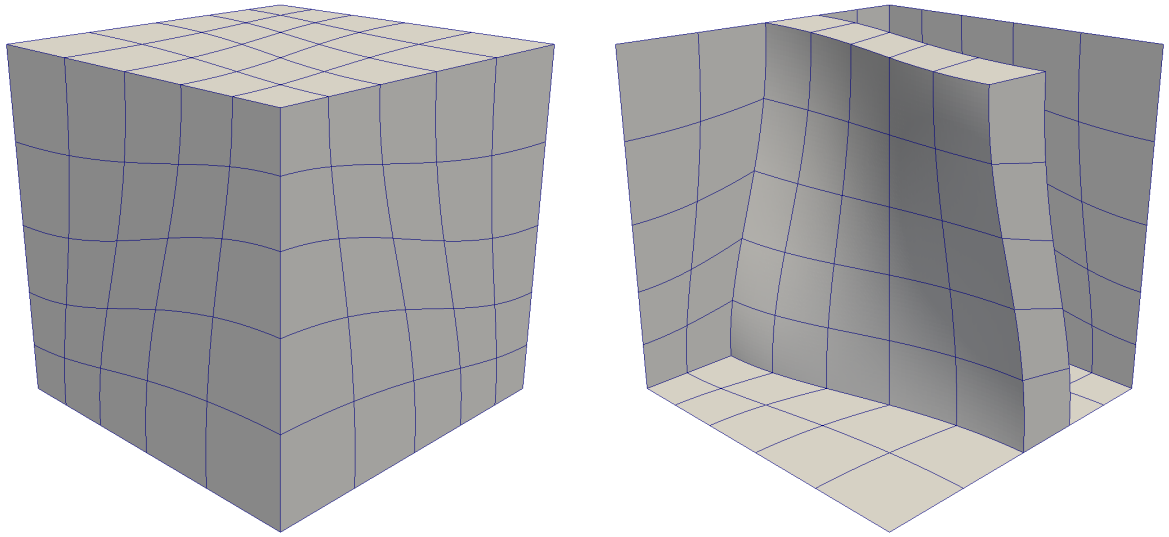


Figure IV.2 Curved hexahedral mesh

Table IV.3 Order of accuracy on P_1 curved meshes

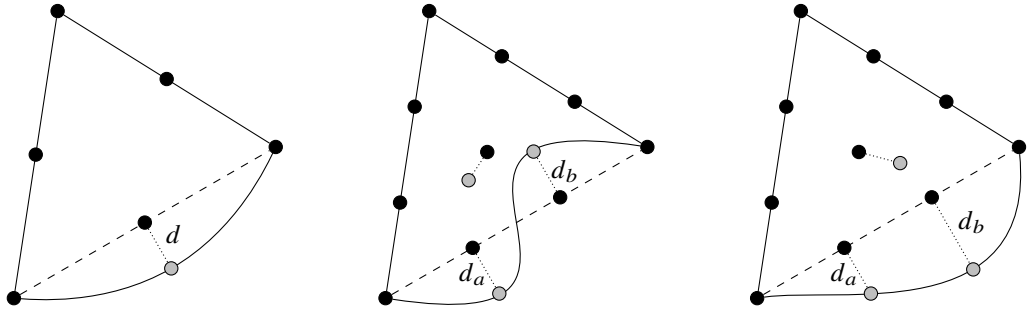
Nodes in Mesh	Tetrahedron		Pyramid		Prism		Hexahedron	
	$f(\boldsymbol{x})$	$\nabla f(\boldsymbol{x})$	$f(\boldsymbol{x})$	$\nabla f(\boldsymbol{x})$	$f(\boldsymbol{x})$	$\nabla f(\boldsymbol{x})$	$f(\boldsymbol{x})$	$\nabla f(\boldsymbol{x})$
216/512	1.86835	0.91981	1.86248	0.88915	1.88697	0.95599	1.89801	1.03627
512/1000	1.92395	0.95332	1.92954	0.93969	1.93480	0.97438	1.94133	1.02299
1000/2197	1.95442	0.97190	1.95798	0.96356	1.96094	0.98460	1.96490	1.01446
2197/4096	1.97241	0.98295	1.97457	0.97781	1.97636	0.99066	1.97877	1.00901
4096/8000	1.98242	0.98913	1.98380	0.98582	1.98494	0.99405	1.98649	1.00583

Table IV.4 Order of accuracy on P_2 curved meshes

Nodes in Mesh	Tetrahedron		Pyramid		Prism		Hexahedron	
	$f(\boldsymbol{x})$	$\nabla f(\boldsymbol{x})$	$f(\boldsymbol{x})$	$\nabla f(\boldsymbol{x})$	$f(\boldsymbol{x})$	$\nabla f(\boldsymbol{x})$	$f(\boldsymbol{x})$	$\nabla f(\boldsymbol{x})$
216/512	2.92392	1.90636	2.92492	1.93608	2.96478	1.96118	3.03805	1.99935
512/1000	2.95883	1.94681	2.93892	1.94081	2.98219	1.97903	3.02591	1.99935
1000/2197	2.97619	1.96840	2.96291	1.96374	2.99010	1.98787	3.01671	1.99953
2197/4096	2.98590	1.98097	2.97763	1.97800	2.99429	1.99282	3.01054	1.99968
4096/8000	2.99113	1.98791	2.98578	1.98597	2.99646	1.99548	3.00686	1.99979

Ciarlet [65,66] has quantified the requirements for maintaining the design order of accuracy on curved isoparametric elements. Specifically, referring to figure IV.3, the element in figure IV.3a has one node between the vertices and is referred to here as a "type 1" element. Similarly, a "type 2" element has two mid-edge nodes and one additional node in the center. (Note that in reference [65], these elements are denoted as types 2 and 3, respectively). It is shown in the references that the order of accuracy for element type 1 can be maintained provided that, as the mesh is refined, the distance between the straight-line segment and the actual position of the node on the curved boundary is reduced in a quadratic manner. Similarly, the requirement for maintaining accuracy for type 2 elements is that this distance must decrease in a cubic manner with mesh refinement.

To demonstrate the precision of these estimates, the manufactured solutions previously discussed in Chapter III are used with a sequence of meshes consisting of 322, 1239, 4837, and 19139 nodes. As seen in figure IV.4, each mesh is comprised of nominally equilateral triangles where type 1 elements are formed by perturbing one edge of every triangle by a predetermined percentage of the edge length. The ratio of the perturbation size divided by the edge length is referred to here as the "percent curvature," or simply "curvature." Note that while a type 1 element may have curvature on all three edges, only one edge is deformed in this test to allow for larger



(a) Element type 1 (b) Element type 2 (S-bend) (c) Element type 2 (H-bend)

Figure IV.3 Elements for higher-order accuracy

deformations. Note also that if the percent curvature is held constant during mesh refinement, the distance between the node at the mid-point of the straight-line segment and the actual position on the edge only varies linearly. Quadric and cubic variations are obtained by dividing the percent curvature by two and four, respectively, as the mesh is refined.

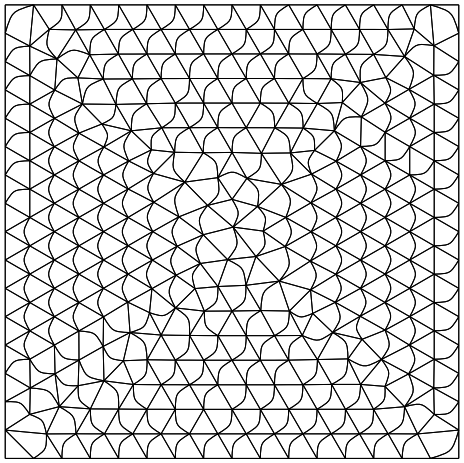


Figure IV.4 Grid Convergence on Meshes with Curved Elements

The solution variables are represented using nominally quadratic (P_2) and cubic (P_3) elements and the order of accuracy of each scheme is established and provided in Table IV.5. Here, the first column provides the number of nodes present in the two meshes used in determining the order of accuracy. The next two columns indicate the experimentally obtained order of accuracy when the distance is reduced linearly with mesh refinement, while the final two columns indicate results for a quadratic variation. As seen in the table, the design orders of accuracy are not obtained when the distance is reduced linearly, whereas a quadratic variation yields the desired order. It is important to note that while the analysis in reference [65] pertains to isoparametric elements, the results obtained using cubic elements in this experiment actually correspond to subparametric elements, but that the design order of accuracy is still maintained.

Table IV.5 Order of accuracy for type 1 elements

Nodes in Mesh	Variation with order h		Variation with order h^2	
	P_2	P_3	P_2	P_3
322/1,239	1.96	2.54	2.99	4.35
1,239/4,837	1.91	2.18	2.94	4.16
4,837/19,139	2.01	2.16	2.96	4.13

Similar results are shown in Table IV.6 for a series of elements of type 2, using cubic basis functions. In the table, an "S-bend" refers to an element similar to that shown in figure IV.3b, which is constructed by perturbing one node on the edge by a specified amount in the direction of a "left-facing" normal, whereas the other node is perturbed in the opposite direction. Shown in figure IV.3c, an "H-bend," which also forms a type 2 element, is constructed by perturbing both

nodes in the same direction but with different amplitudes. As with the results shown previously in Table IV.5, the results in Table IV.6 demonstrate that the design order of accuracy is obtained by satisfying the requirements for reducing the distance between the straight-line segment and the actual node locations. In contrast, violating these conditions yields suboptimal orders of accuracy.

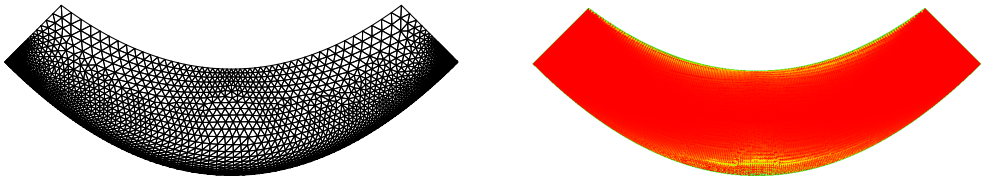
Table IV.6 Order of accuracy for type 2 elements

Nodes in Mesh	Variation with order h^2		Variation with order h^3	
	S-bend	H-bend	S-bend	H-bend
322/1,239	3.28	3.37	4.25	4.30
1,239/4,837	3.18	3.21	4.20	4.27
4,837/19,139	3.13	3.14	4.14	4.19

For a viscous flow, elements near the surface will often need to be curved to accommodate proper representation of the geometry without creating elements with negative areas/volumes. However, the results presented in Tables IV.5 and IV.6 clearly indicate that strict requirements must be met regarding the deformation of any element and, in particular, how the deformations must be reduced as the mesh is refined. It is important to note that in practice, surface perturbations are naturally decreased in a quadratic manner as the mesh is refined, thereby satisfying the requirements for type 1 elements. Unfortunately, while the results in Table IV.5 indicate that design order can inherently be achieved with these elements, the desired order of accuracy may be difficult to achieve for elements of type 2.

To examine this further, the method of manufactured solutions is again used on a series of meshes to determine the order of accuracy. In the tests that follow, highly stretched elements

are present near one of the boundaries, although an inviscid flow is assumed to eliminate any potential influence caused by the scaling of the stabilization matrix, or changes to other parameters such as the Reynolds number. The geometry considered is shown in figure IV.5a, where parabolas represent the upper and lower walls so that quadratic functions can precisely describe the geometry. To avoid crossing grid lines when only "snapping" surface points, the grid along the upper wall has normal spacing similar to that of an inviscid flow, whereas the lower wall has viscous-type spacing. The x-coordinate spans from -0.5 to $+0.5$ with normal spacing along the lower wall of 5.0×10^{-4}.



(a) Parabolic mesh (coarsest in series)

(b) y-displacements

Figure IV.5 Mesh and contours of y-displacement for parabolic mesh

As output by a typical mesh generation procedure, the vertices at the endpoints of surface edges are accurately positioned, whereas nodes in the middle of these edges require repositioning. The location of these nodes is altered in the direction normal to the edges until the actual surface intersection is found. These surface perturbations are subsequently propagated into the interior using the methodology described in Chapter II. Figure IV.5b shows contours of the displacements in the y-coordinate direction distributed throughout the mesh as a result of perturbing these points.

As seen, the largest y-displacements occur at the center of the mesh, decreasing at either end. Although not shown, displacements in the x-direction vary differently, with the largest deformations occurring toward the ends. For the sequence of grids consisting of 738, 2310, 8045, and 29760 nodes, the corresponding maximum perturbations at the surface are approximately 3.8×10^{-4}, 9.88×10^{-5}, 2.5×10^{-5}, and 6.25×10^{-6}, respectively. Note that the perturbations are reduced by a factor of four with each mesh refinement, which also corresponds to a reduction in the percent curvature by a factor of 2. The realized reductions in perturbation size thereby satisfy the quadratic behavior necessary for a type 1 element, but are not reduced rapidly enough for an element of type 2.

Observed orders of accuracy for the manufactured solutions on these parabolic meshes are given in Table IV.7. The table demonstrates that for linear and quadratic elements the theoretical order of accuracy is obtained whereas suboptimal convergence is achieved for cubic elements. Note that for linear elements, no repositioning of any nodes is required. Further, because quadratic elements are of type 1, the naturally occurring squared reduction of the perturbations at the surface allows the desired order to be maintained. In contrast, for cubic elements, the theoretical order of accuracy is not achieved because the perturbations at the wall, which are propagated into the interior, are only decreased in a quadratic manner as the mesh is refined instead of with a cubic reduction as required. If the edges are "re-lofted" by fitting a quadratic through the nodes, the resulting mesh is now comprised of type 1 elements. As seen in the fifth column of the table, the design order of accuracy is again achieved. While the procedure used here for re-lofting the nodes along the edge is not very general, the result again demonstrates that the design order can be achieved by using subparametric type 1 elements. Also, it is relatively simple to modify the mesh

curving algorithm so that the displacements of nodes in the interior of the mesh always yields type 1 elements. Finally, as shown in the last column, if only surface nodes are "snapped" to the actual geometry, the mapping of the interior elements from the reference space to the physical space remains linear and the design order of accuracy is also obtained. It should be noted that, while the P_3 solution with type 2 elements failed to achieve the theoretical order of accuracy, the errors observed are roughly one order of magnitude lower than for the P_2 solution.

Table IV.7 Order of accuracy study for parabolic mesh with viscous wall spacing

Nodes in Mesh	P_1	P_2	P_3	P_3 (re-lofting)	P_3 (surface)
738/2,310	2.51	3.46	3.63	5.35	5.44
2,310/8,045	2.20	3.08	3.35	4.45	4.50
8,045/29,760	2.14	3.04	3.16	4.84	4.75

The results presented above demonstrate that achieving higher than third-order accuracy for viscous flows requires some care when curving elements near the boundary. In particular, while type 1 elements may be used to represent the geometry in conjunction with higher-order polynomials for flow variables, the creation of type 2 elements needs to be minimized or avoided. However, in perturbing the interior mesh points to accommodate changes at the surface, it is easily seen how elements of type 2 can be created from elements with originally straight sides. First, because the two vertices at the ends of each surface edge are typically placed correctly by the grid generation procedure, any repositioning of mid-edge nodes naturally creates an oscillatory behavior for the perturbations that are subsequently propagated into the interior. Secondly, as seen in figure IV.6, even if elements of type 1 can accurately represent the surface, coarser elements

in the interior that may span surface undulations may be deformed into type 2 elements, which could potentially impact the order of accuracy of the scheme. It is therefore recommended that propagating surface perturbations into the interior be minimized as much as possible or that type 1 elements be used.

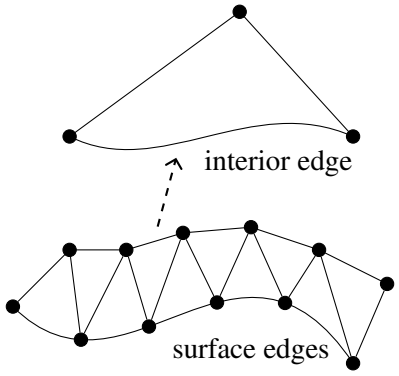


Figure IV.6 Creation of type 2 cells in the interior of the mesh

CHAPTER V

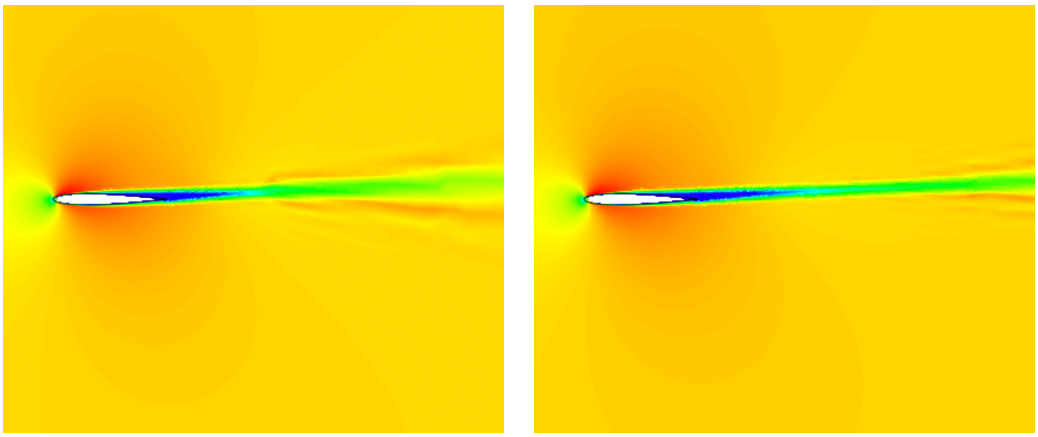
NUMERICAL RESULTS

Two Dimensions

Laminar Flow Over NACA 0012

The first demonstration case is for laminar flow over a NACA 0012 airfoil at a Mach number of 0.5, an angle of attack of 3 degrees, and a Reynolds number of 5000 based on the chord of the airfoil. A depiction of the flow field is shown in figure V.1, which shows Mach number contours computed using both linear elements and quadratic elements, on a mesh with 31,042 nodes. As seen in the figure, the wake captured using quadratic elements is less diffused and propagates intact much farther downstream than the wake obtained using linear elements. Although not quantitatively apparent from the figure, the flow also separates at approximately mid-chord of the airfoil.

Computations are conducted using three meshes consisting of 3296, 9594, and 31042 nodes, respectively. Figure V.2a shows pressure coefficients computed using linear and quadratic elements on the finest mesh, quadratic and cubic elements on the medium mesh, and cubic elements on coarsest mesh. As seen, all the pressure distributions agree very well. Skin friction coefficients, shown in figure V.2b also agree well between the schemes on different meshes, although some differences are apparent near the leading edge. By examining the skin friction over the first quarter of the airfoil (figure V.2c) it is seen that the solution obtained with linear elements

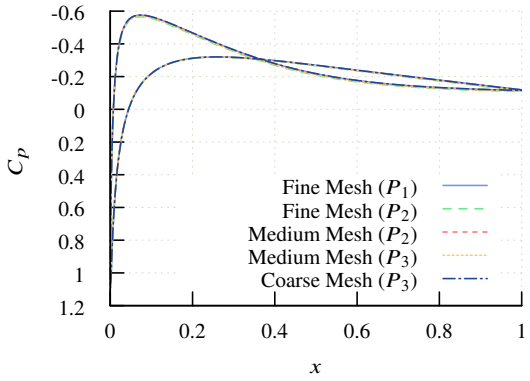


(a) P_1 elements (b) P_2 elements

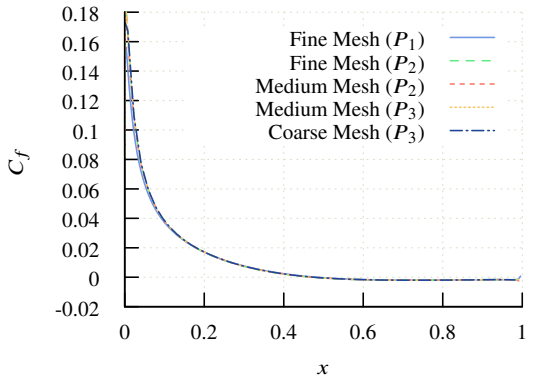
Figure V.1 Mach contours for NACA 0012, $M_\infty = 0.5$, $\alpha = 3°$, $Re = 5000$

clearly differs from the other solutions, which are all in excellent agreement with one another. Note that in figure V.2c the solutions obtained with cubic elements have been depicted using symbols to help discriminate between the other solutions. While further refinement is necessary to ensure that the skin frictions do not exhibit further change, it is apparent that the solutions obtained with cubic elements on the coarsest mesh, and quadratic elements on the medium mesh, are both somewhat more accurate than the solution obtained with linear elements on the finest mesh.

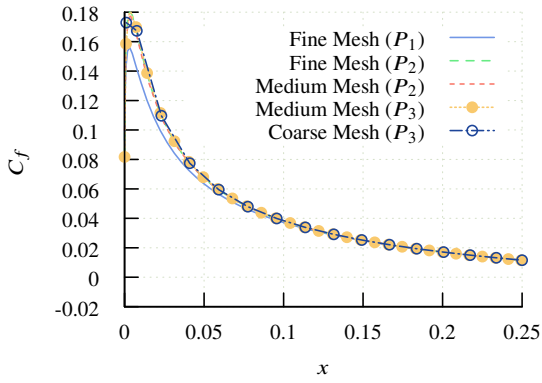
It is interesting to compare the computer resources required in obtaining the solution with quadratic elements on the medium mesh, cubic elements on the coarsest mesh, and linear elements on the finest mesh. To facilitate the comparison, Table V.1 provides detailed statistics for each mesh. Note that quadratic elements require additional degrees of freedom corresponding to the number of edges in the mesh, whereas for cubic elements, two unknowns are added for each edge, along with a subsequent unknown placed in the interior of the triangle. Because of the extra degrees of freedom required for quadratic elements over linear elements, the total number of unknowns in



(a) Pressure coefficient over airfoil



(b) Skin friction coefficient over airfoil



(c) Skin friction coefficient near leading edge

Figure V.2 Results for laminar flow over NACA 0012 airfoil: $M_\infty = 0.5$, $\alpha = 3°$, $Re = 5000$

the medium mesh is approximately 20% greater (38,054) than in the finest mesh with linear elements. For cubic elements on the coarsest mesh there are about 6% fewer (29,181) unknowns than for the finest mesh with linear elements. The work required to compute residuals is proportional to the number of triangles multiplied by the number of Gauss points used in each triangle. Using Table V.1 and assuming 3 Gauss points for linear elements, 7 for quadratic elements, and 12 for cubic elements, it is easily estimated that the residual computation for quadratic elements requires only about 70% of the work needed for linear elements whereas cubic elements only require roughly 40% of the work needed for linear elements. Further reductions can be achieved using 6 Gauss points for quadratic elements and 11 points for cubic elements without adversely effecting accuracy. For an explicit scheme, this would imply that significant savings are available by using higher-order elements. However, in the present implementation, a higher percentage of time is spent in assembling and solving the linear system so that the cost between the schemes is roughly equivalent. It is important to note several key points. First, while the computer resources required for obtaining the solution with linear elements on the finest mesh is similar to that required for the higher-order schemes on coarser meshes, the solution obtained with linear elements is clearly not as accurate as the other solutions. Further refinement is required for the linear elements to accurately assess the relative timings. Secondly, in this example there is no exact solution and therefore no precise measure of the errors so the comparisons are only qualitative. Finally, it should be noted that to achieve equal accuracy between linear and quadratic elements, the mesh spacing for the quadratic elements is larger and varies as the mesh spacing for linear elements raised to the $2/3$ power ($h_{p_2} \approx h_{p_1}^{2/3}$). Similarly, for cubic elements the mesh spacing required to achieve a specified

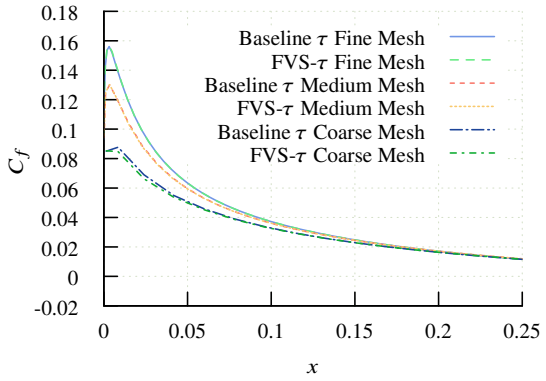
level of error varies as the square root of the mesh spacing for linear elements ($h_{p_3} \approx h_{p_1}^{1/2}$). Because of these nonlinear relationships, the relative savings depends on the level of error desired in the computations, with the largest savings occurring for lower errors.

Table V.1 Mesh statistics for NACA 0012

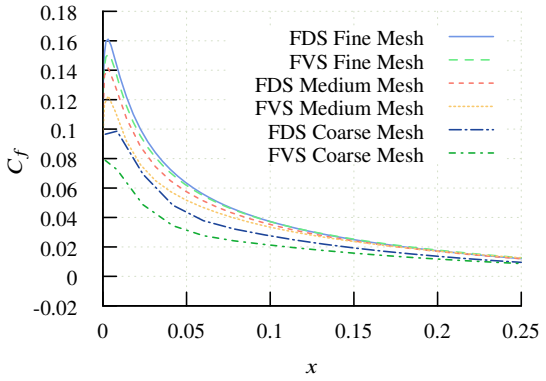
	Fine Mesh	Medium Mesh	Coarse Mesh
Nodes	31,042	9,594	3,296
Edges	92,482	28,460	9,727
Triangles	61,440	18,866	6,431

As noted in Chapter II, the stabilization matrices can be designed using flux formulas originally developed for finite-volume schemes. Figure V.3 depicts skin friction results obtained using the "standard" stabilization matrix given in equation (II.18), as well as skin frictions computed using stabilization matrices based on van Leer's flux-vector splitting [42, 43]. In figure V.3a, skin-friction results computed with linear elements are shown along the upper surface of the airfoil for all three meshes described above. It is seen that the skin friction changes as the mesh is refined, although the agreement between the results obtained with the different stabilization matrices is very good and indicates no significant difference in the level of dissipation. In contrast, seen in figure V.3b, similar results obtained with a finite-volume scheme exhibit significant differences, with the flux-vector splitting results being clearly more dissipative than those obtained with flux-difference splitting. For the finite-volume scheme, less dissipation is present for flux-difference splitting because of a vanishing eigenvalue corresponding to the velocity normal to the wall [68].

In contrast, for the Petrov-Galerkin scheme, the dissipation is proportional to the inverse of the flux Jacobian so one would not expect higher dissipation with flux-vector splitting.



(a) Skin friction coefficients computed with stabilization matrices based on flux-difference and flux-vector splitting



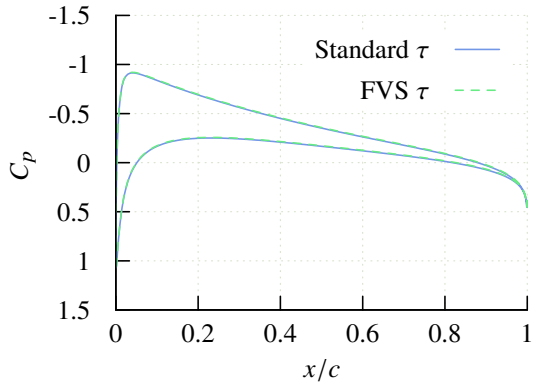
(b) Skin friction coefficients computed using a finite-volume scheme with both flux-difference splitting and flux-vector splitting

Figure V.3 Effect of stabilization matrices on skin friction coefficient

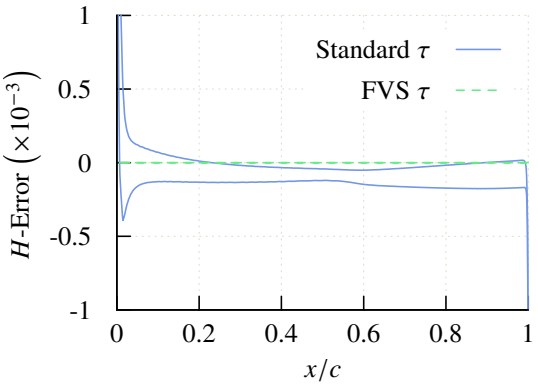
Inviscid Flow Over NACA 0012

The next case is for subsonic inviscid flow over a NACA 0012 airfoil at a free-stream Mach number of 0.5 and an angle of attack of two degrees. The intent of presenting this case is to simply demonstrate another potential advantage to considering alternate formulations of the stabilization matrix. Here, the formulation for the stabilization matrix is based on the flux-vector splitting formulas given by Hänel [69], which are modifications to van Leer's original splittings so that total enthalpy can be maintained throughout the flow-field. Figures V.4a and V.4b show the pressure distribution and total enthalpy errors, respectively, along the surface of the airfoil. As seen in figure V.4a, the pressure distributions obtained using the new stabilization matrix agrees

very well with results obtained using the standard definition. However, as seen in figure V.4b, the total enthalpy errors using the new formulation are essentially zero over the entire surface of the airfoil whereas the errors obtained using the standard definition are as high as 0.1 percent near the leading edge.



(a) Pressure coefficient computed with stabilization matrices based on flux-difference and flux-vector splitting



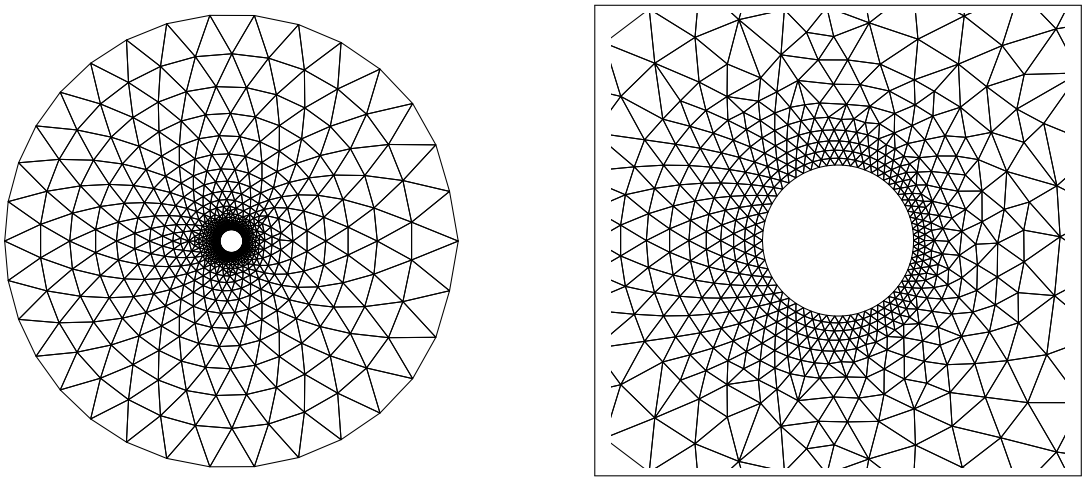
(b) Total enthalpy error computed with stabilization matrices based on flux-difference and flux-vector splitting

Figure V.4 Results for inviscid flow over NACA 0012 airfoil: $M_\infty = 0.5$, $\alpha = 2°$

Time-Accurate Cylinder

The next demonstration case compares computed velocity profiles with experimental data for time dependent flow over a circular cylinder. In the experimental data, described in references [70,71], the cylinder is impulsively started to a uniform translation velocity that corresponds to a Reynolds number of 40 based on the diameter of the cylinder. Simulations have been conducted using linear and quadratic elements on a sequence of meshes ranging in size from 858, 3408, and 13571 nodes, with the coarsest mesh depicted in figure V.5. Although not shown, the

finest mesh has 248 nodes around the circumference of the cylinder with normal spacing at the wall of 0.007. Subsequent coarser meshes have approximately one-half and one-quarter of the number of mesh points on the surface so that the coarsest mesh only has 62 points in this region. For all the meshes, no clustering is provided along the wake centerline. Although the experimental data corresponds to an incompressible flow, the current simulations are run at a free-stream Mach number of 0.2. All computations are done using second-order backward-time differencing with a time step of 0.01, non-dimensionalized by the reference length and the free-stream speed of sound. It should be noted that simulations have also been done using a time step of 0.05 with no observable differences in the computed profiles.



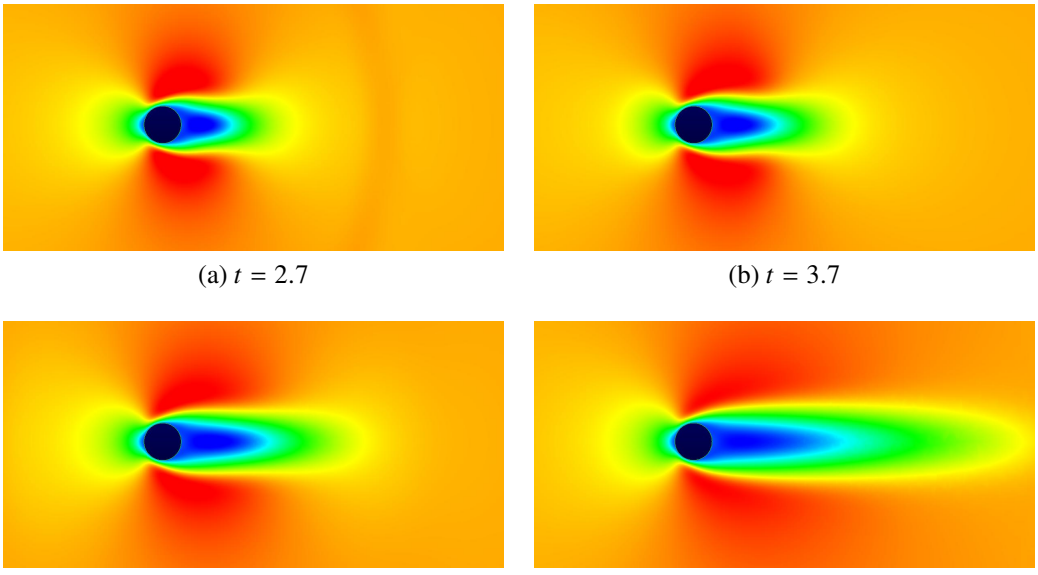
(a) Coarsest Mesh for Circular Cylinder

(b) Near-Field View of Mesh

Figure V.5 Mesh for Circular Cylinder

A depiction of the computed start-up process is shown in figure V.6, where contours of the x-component of velocity are shown at non-dimensional times where experimental data exists [70,

71]. As seen in the figures, the flow at each time is symmetric, with the wake growing continuously as time progresses.



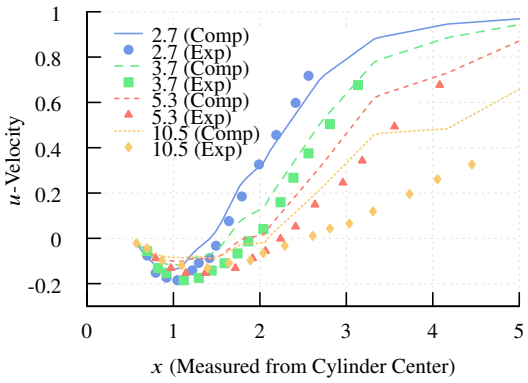
(a) $t = 2.7$ (b) $t = 3.7$

(c) $t = 5.3$ (d) $t = 10.5$

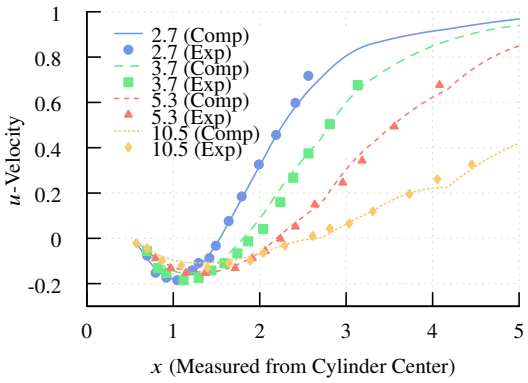
Figure V.6 Contours of the x-component of velocity for impulsively started cylinder

Figure V.7 shows a comparison of the computationally obtained x-component of velocity along the x-axis with those from the experiment. Results obtained using linear elements are shown on the left side of the figure, whereas similar results are shown on the right side for quadratic elements. It is seen in the figures that for linear elements, the results on the coarsest mesh are in poor agreement with experiment but continually improve as the mesh is refined until good agreement is obtained on the finest mesh with 13,571 degrees of freedom. In contrast, for quadratic elements good agreement is obtained on the coarsest mesh containing only 3,338 degrees of freedom. The solutions on the two finer meshes are very similar to those on the coarsest mesh but are slightly

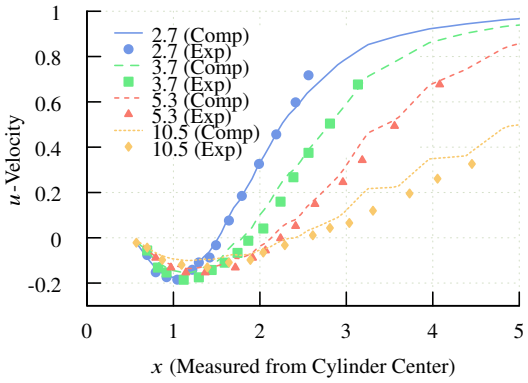
smoother aft of approximately two diameters downstream of the cylinder. Overall, the comparisons with experiment are very good and demonstrate that accurate solutions can be obtained using the higher-order elements on a much coarser mesh than required for linear elements.



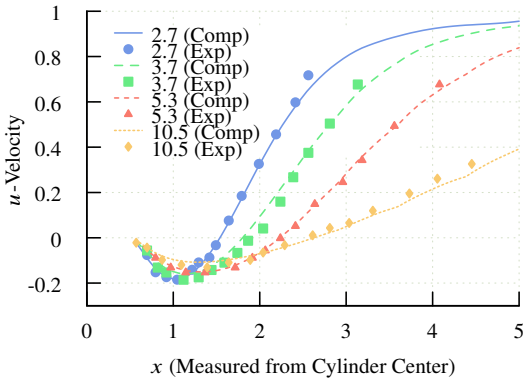
(a) Coarse mesh, linear elements



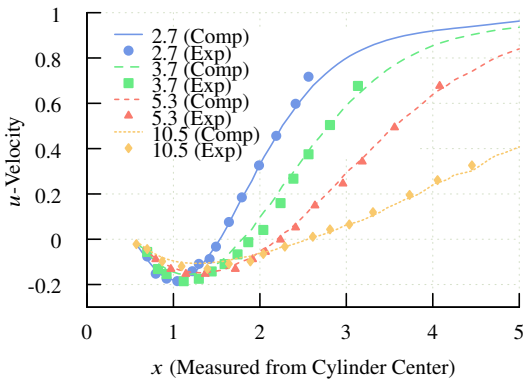
(b) Coarse mesh, quadratic elements



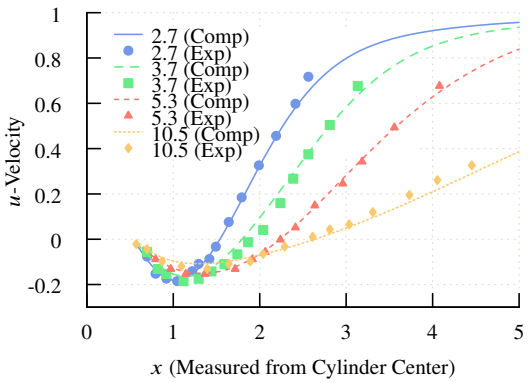
(c) Medium mesh, linear elements



(d) Medium mesh, quadratic elements



(e) Fine mesh, linear elements



(f) Fine mesh, quadratic elements

Figure V.7 Comparison of computed velocity profiles with experimental data

Three Dimensions

Delta Wing

Vortical flow over a low aspect ratio delta wing is analyzed by Thomas, Krist, and Anderson [72] using the finite-volume code CFL3D. In addition, experimental data is readily available for this geometry. The CFL3D and experimental results presented in reference [72] are digitized for comparison. In accordance with reference [72], laminar flow is specified at a Mach number of 0.3, an angle of attack of 20.5 degrees, and a Reynolds number of 0.95×10^6. Shown in figure V.8, the mesh has 562,221 nodes with viscous spacing of 5×10^{-5} normal to the surface.

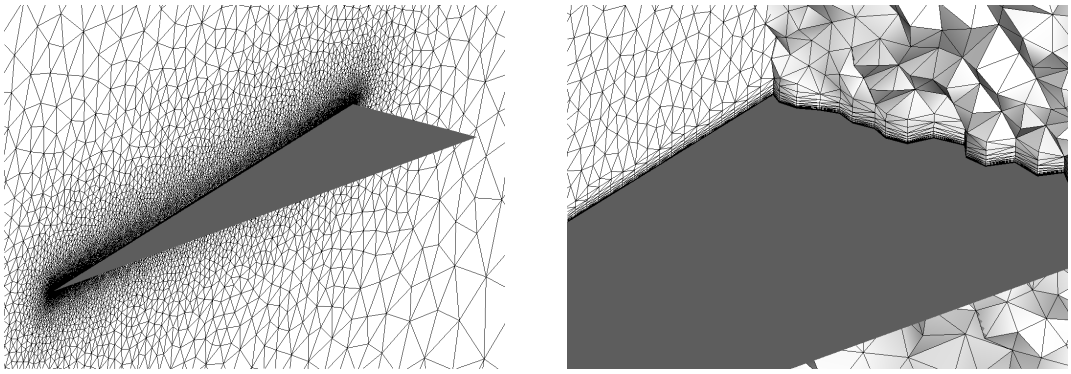


Figure V.8 Mesh for low aspect ratio delta wing

Figure V.9 shows total pressure contours obtained using linear elements, where vortices are seen emanating from the leading edge of the wing. Figure V.10 shows pressure coefficients at four locations along the chord of the geometry. Reasonably good agreement is seen between the current work and CFL3D as well as with experimental data. However, oscillatory behavior is observed near the leading edge on the aft portion of the wing. It should be noted that after the residual is reduced

to about 2×10^{-9}, unsteadiness in the solution is observed and reduction of the residual no longer occurs.

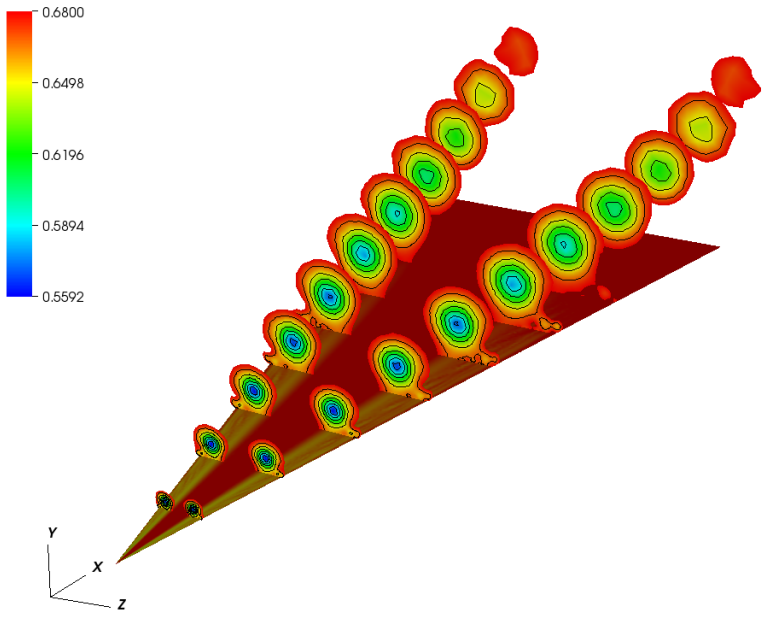
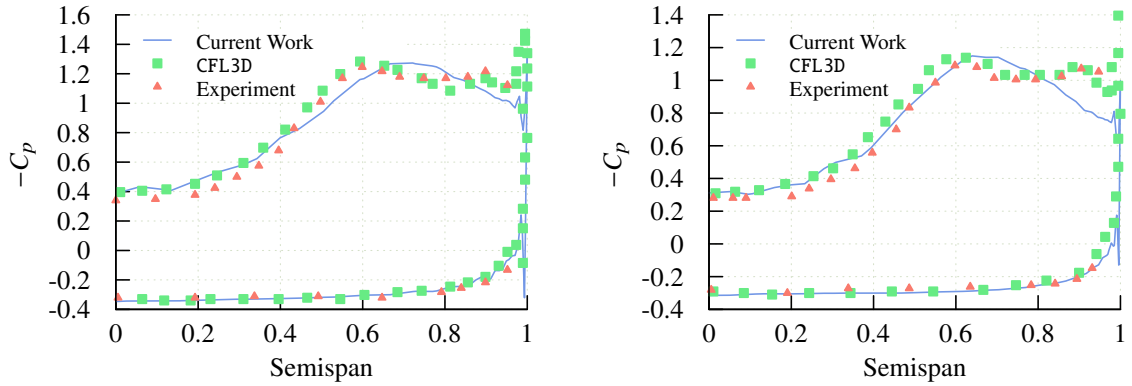


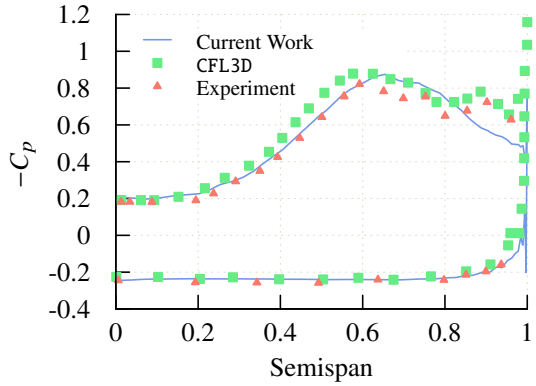
Figure V.9 Contours of total pressure

Results have also been obtained for the delta wing configuration using quadratic elements. For these simulations, a mesh with 200,918 nodes has been generated, with spacing at the wall three times larger than on the previous mesh and fewer triangles on the surface. The resulting pressure distributions are shown in figure V.11, where it is observed that generally good agreement is obtained with experimental data and with the results of reference [72]. As with the results obtained with linear elements, the solution near the leading edge exhibits some oscillations.

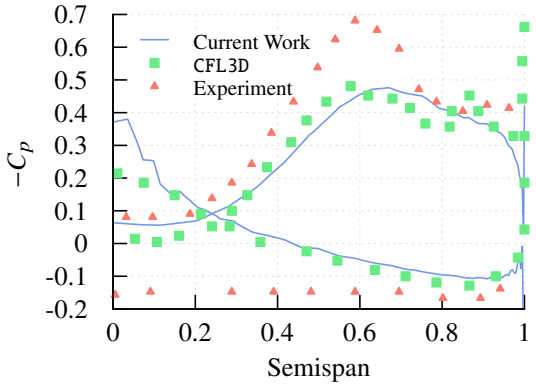


(a) $x/L = 0.3$

(b) $x/L = 0.5$

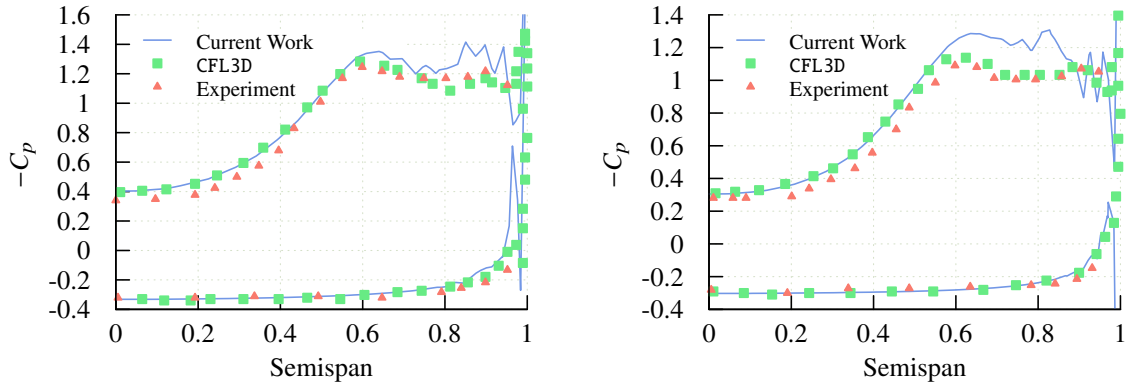


(c) $x/L = 0.7$



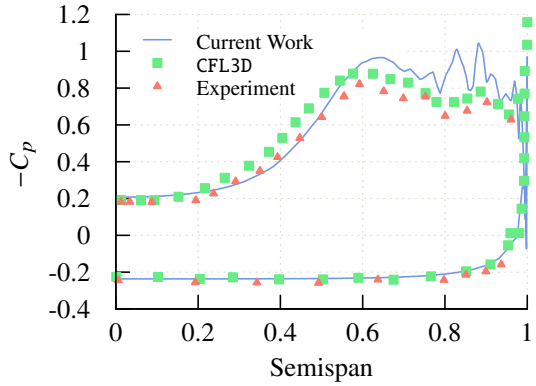
(d) $x/L = 0.9$

Figure V.10 C_p plots at various stations along delta wing, linear elements

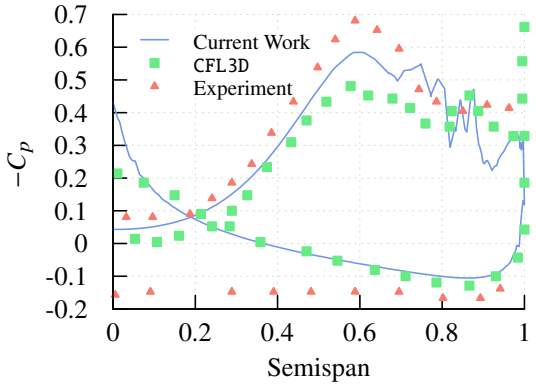


(a) $x/L = 0.3$

(b) $x/L = 0.5$



(c) $x/L = 0.7$



(d) $x/L = 0.9$

Figure V.11 C_p plots at various stations along delta wing, quadratic elements

Viscous flows over a three-dimensional circular cylinder

This test case considers three-dimensional viscous flow over a circular cylinder at a Reynolds number of 2580 based on the diameter of the cylinder and free-stream Mach number of 0.2 [73]. The geometry definition and the computational mesh is displayed in figure V.12. The x-axis is along the stream-wise flow direction and the z-axis is in the span-wise direction.

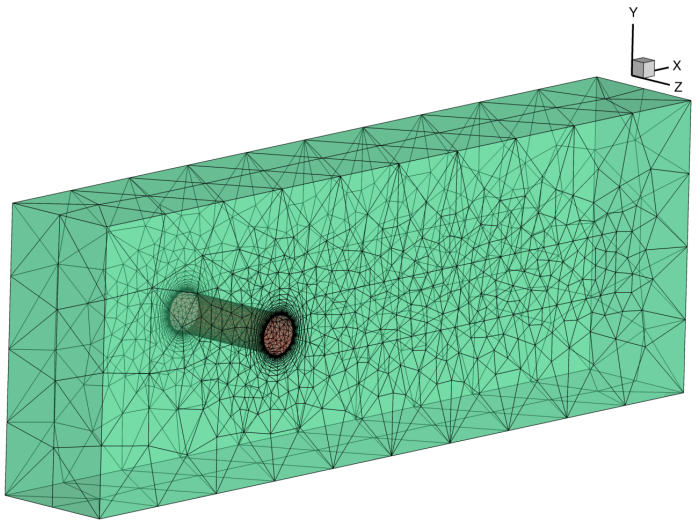
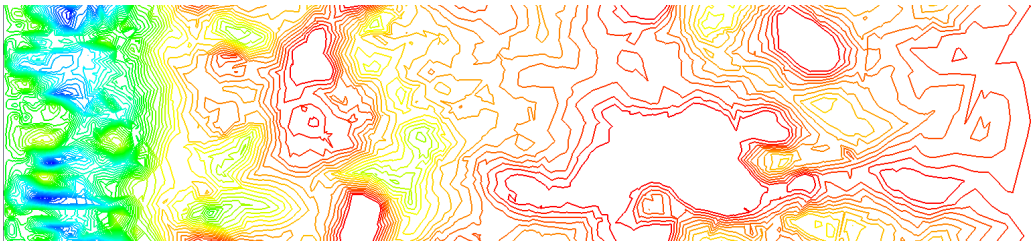


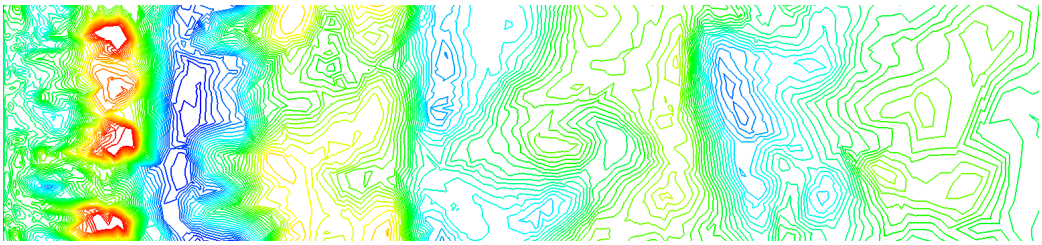
Figure V.12 Computational mesh (containing 68,629 tetrahedral elements) for the three-dimensional viscous flow over a circular cylinder

Flow-through boundary conditions are specified at the inlet and outlet. Periodic boundary conditions are applied on the side walls in the span-wise direction while symmetry is imposed for the top and bottom walls. The cylinder is modeled as a nonslip isothermal wall. The mesh on the cylinder surface is represented by quadratic polynomials and the interior mesh points are moved via the linear elasticity solver discussed in Chapter II to allow curved meshes in the viscous boundary layer. Figure V.13 shows contours of the instantaneous stream-wise, transverse, and

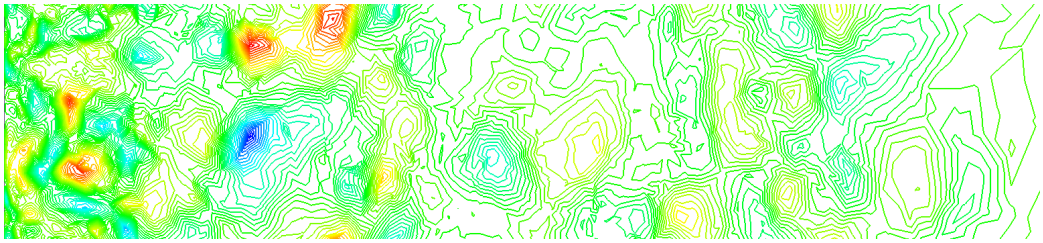
span-wise velocities in the wake of the cylinder using the third-order scheme and second-order backward difference time integration with a fixed time-step of 0.02. Unsteady recirculation is clearly observed as are alternating regions of positive and negative transverse velocity.



(a) Instantaneous stream-wise velocity



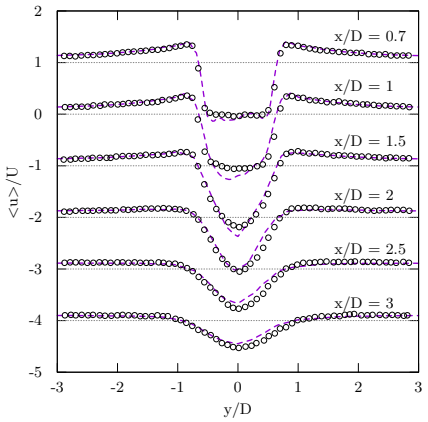
(b) Instantaneous transverse velocity



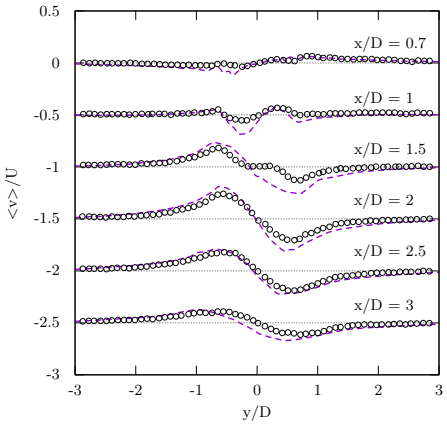
(c) Instantaneous span-wise velocity

Figure V.13 Instantaneous velocities in the x-z plane (y=0) in the wake of the cylinder. 60 contours are included from -0.2 to 0.2

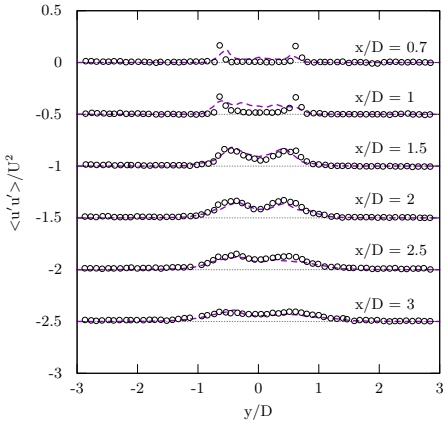
Figure V.14 compares, with experimental data [73], various normalized quantities at several stream locations in the wake region obtained with the third-order scheme. The current method shows favorable agreement with the experimental results.



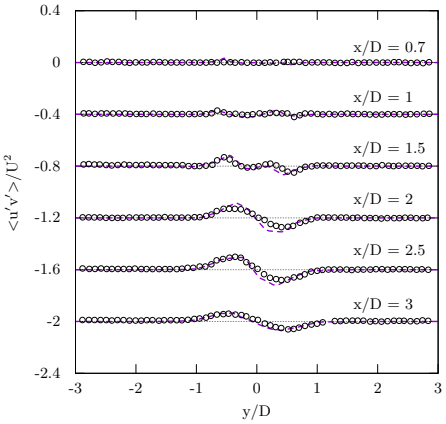
(a) Mean normalized stream-wise velocity



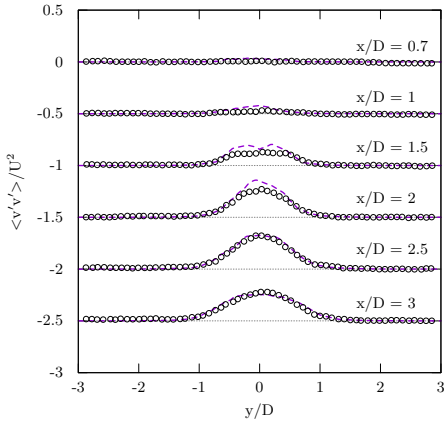
(b) Mean normalized transverse velocity



(c) Normalized $< u'u' >$



(d) Normalized $< u'v' >$



(e) Normalized $< v'v' >$

Figure V.14 Normalized quantities in the wake of the circular cylinder using 3rd order SUPG (dashed line) in comparison with experimental data (circles)

Turbulent Flow Over an ONERA M6 Swept Wing

This numerical example consists of turbulent flow over an ONERA M6 swept wing configuration at a Reynolds number of 11,270,000 based on the mean aerodynamic chord (MAC), Mach number of 0.3, and 3.06° angle of attack. The ONERA M6 mesh shown in figure V.15 consists of 257,241 tetrahedral elements with 44,676 nodes in the second-order scheme and 350,103 degrees of freedom in the third-order scheme. The spacing normal to the surface of the wing is 1×10^{-5}, yielding an estimated y^+ value of about 1.

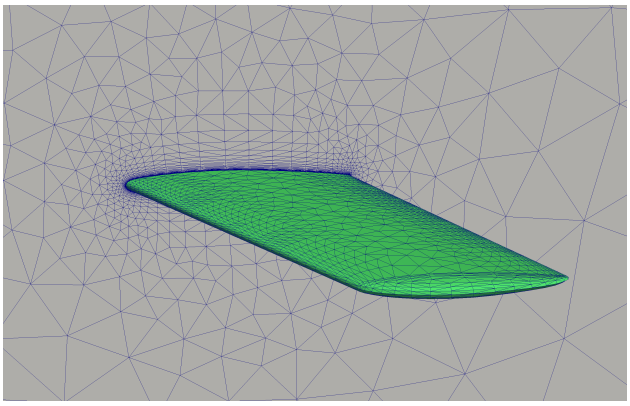


Figure V.15 ONERA M6 swept wing geometry

Second and third order SUPG discretizations are employed, with the modified SA turbulence model presented in Chapter II applied with discretization order consistent with the main flow equations. The implicit problem is solved using the ILU(1) preconditioned GMRES algorithm. Using the time stepping strategy described in Chapter II, steady state solutions are achieved for all discretizations with the L_2 norms of the density and turbulent working variable being reduced to 10^{-17} and 10^{-16}, respectively. The CFL evolution strategy described previously was applied effectively in this case as demonstrated by the convergence history plotted in figure V.16.

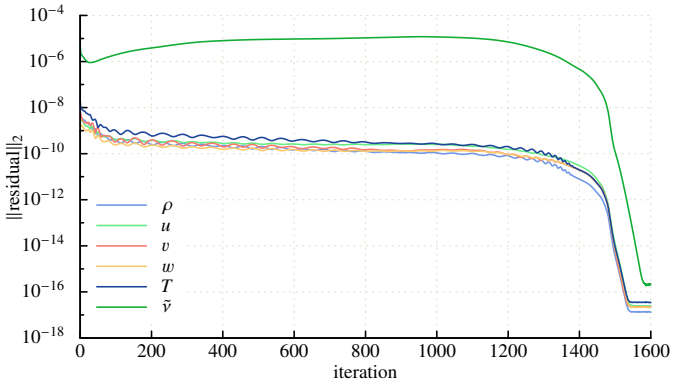
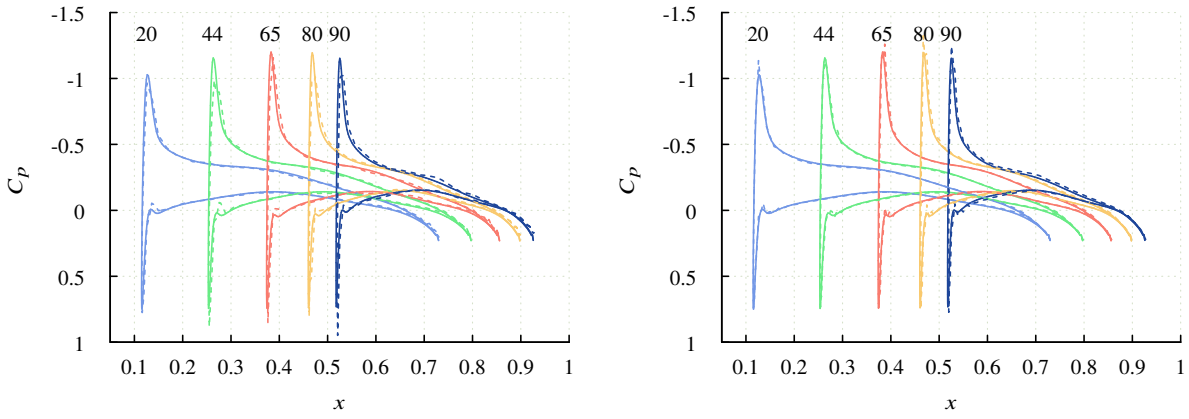


Figure V.16 Convergence history for P_2 SUPG solution on ONERA M6 swept wing geometry

The computed surface pressure coefficients obtained using the second- and third-order SUPG discretizations are shown in figures V.17a and V.17b for various locations along the semi-span. A CFL3D solution obtained from a substantially fine grid is shown to serve as a benchmark solution . Both the P_1 and P_2 solutions show good agreement with the benchmark CFL3D results.



(a) P_1 (b) P_2

Figure V.17 Pressure coefficients plotted at various span-wise locations on the turbulent ONERA M6. Solid lines represent the CFL3D solution while dashed lines represent the SUPG solutions. The span-wise locations (as % semi-span) are indicated above each curve

To demonstrate the importance of accurately representing curved meshes, figure V.18 shows the pressure coefficients at 65% semi-span obtained using a piecewise linear geometry representation for the third order SUPG scheme. Although convergence was achieved to machine precision, the linear geometry definition clearly has a negative impact on the accuracy of the obtained solution. The non-smooth surface geometry causes significant oscillations to appear in the pressure distribution. This further emphasizes the need for a robust strategy for obtaining curved higher-order meshes.

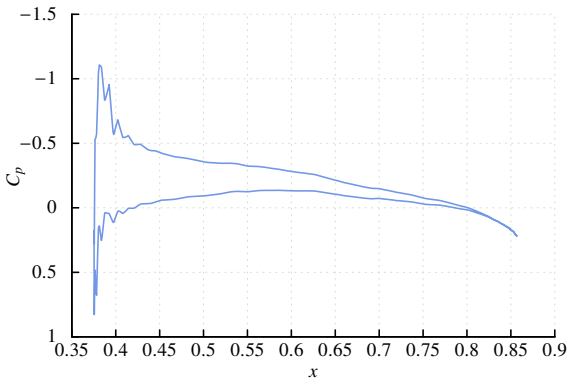
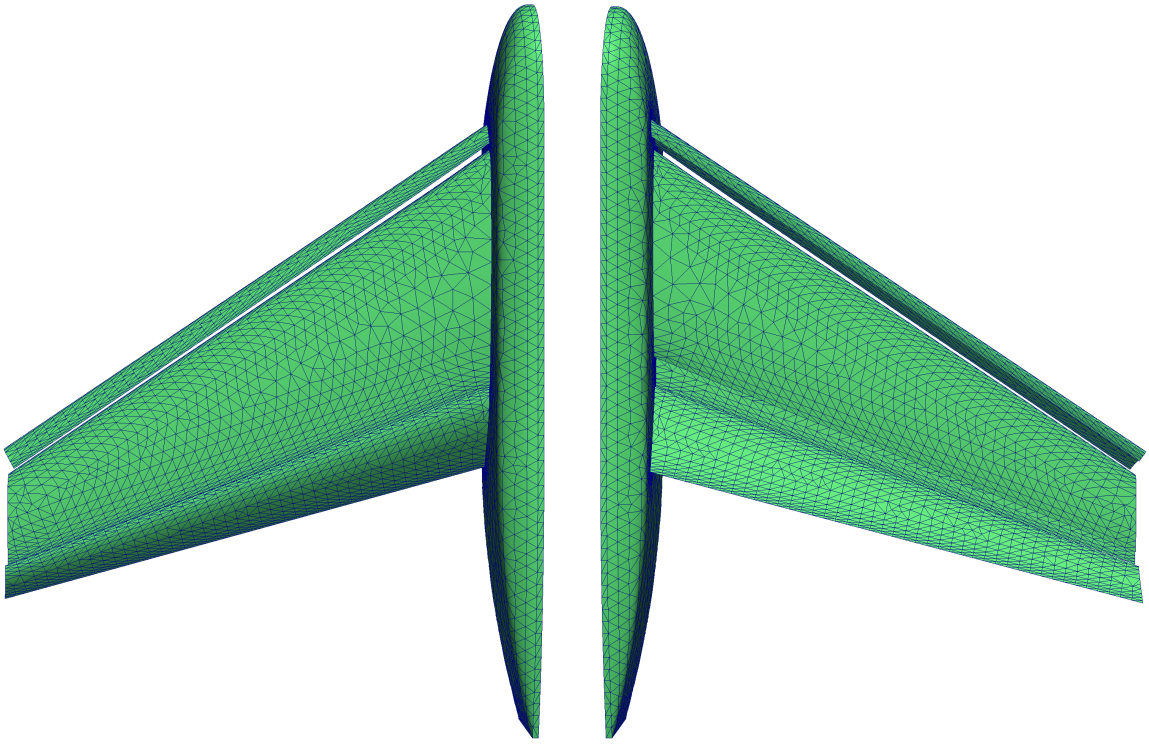


Figure V.18 Pressure coefficients at 65% semi-span for P_2 solution on the linear mesh

Turbulent Flow over a NASA Trap Wing

Finally, results are shown below for turbulent flow over a NASA trap wing configuration used in the 1stAIAA CFD High Lift Prediction Workshop [74]. In this paper, only configuration one, without brackets, is considered with a Reynolds number of 4,300,000 based on the mean aerodynamic chord (MAC), 12.99° angle of attack, and a Mach number of 0.2.

This test case was solved using the second and third order SUPG discretizations where the mesh contains 1,126,835 tetrahedral elements with the 194,370 nodes in the second-order scheme and 1,528,767 degrees of freedom in the third-order scheme. It should be noted that these meshes are much coarser than those provided for the High Lift Prediction Workshop.[1] Also, in contrast to the ONERA M6 case, curving the mesh using linear elasticity did not result in any negative Jacobians.



(a) Top view. (b) Bottom view.

Figure V.19 Surface mesh for trap wing geometry

[1] http://hiliftpw.larc.nasa.gov/index-workshop1.html

Contours of the computed turbulence working variable at three span-wise locations are shown in figure V.20 for the third-order scheme. The maximum turbulence working variable appears in the circulation region between the slat and main wing, with significant values also occurring in the circulation region between the main wing and the flap.

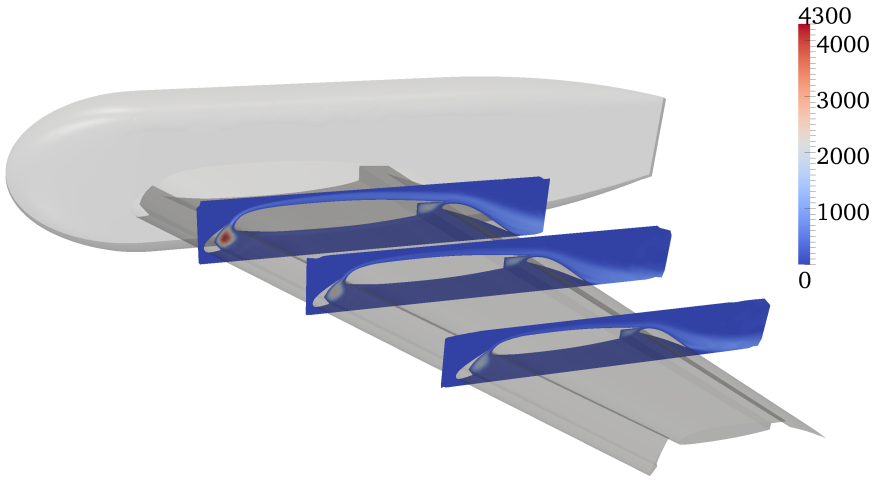
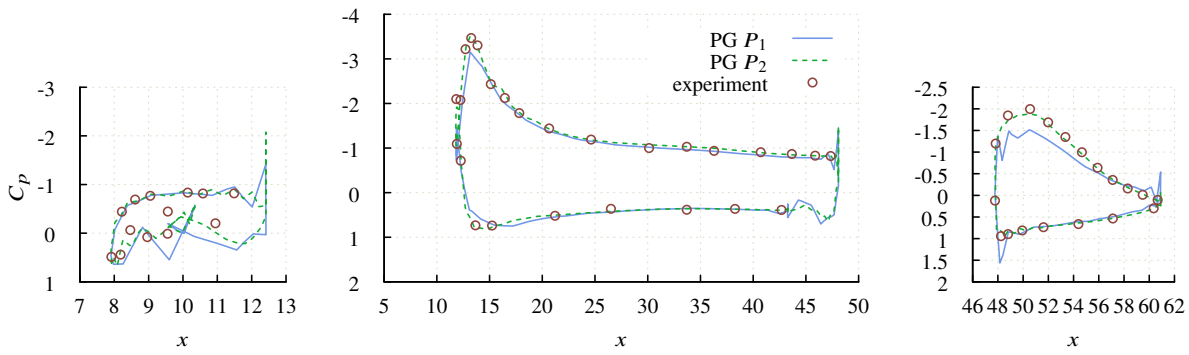


Figure V.20 Turbulence working variable at three stations along trap wing configuration

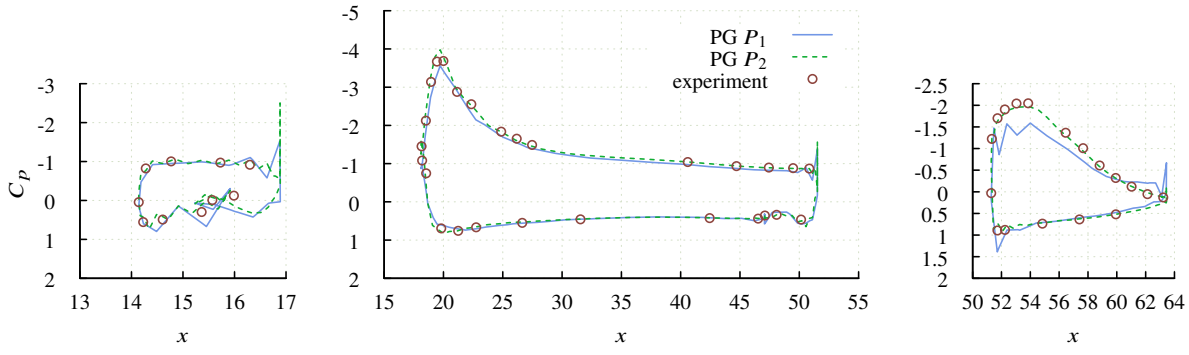
Figures V.21 to V.27 compare computed pressure coefficients with experimental wind tunnel data at various stations along the semi-span of the three elements. At all stations, the second-order solution under-predicts the negative pressure on the upper surface of all three wing elements. This under-prediction is most severe on the upper surface of the flap element whose discretization is very coarse with only about 10 points along the upper surface. The third order solution shows vast improvement in this regard. Both discretization orders suffer from oscillations in the pressure distributions, especially near trailing edges. Again, the third order solution displays improvement in this regard as well. However, both solutions exhibit oscillations in the pressure distribution

along the chord of the slat element; this is likely exacerbated by the coarseness of the mesh on that element.



(a) slat (b) main wing (c) flap

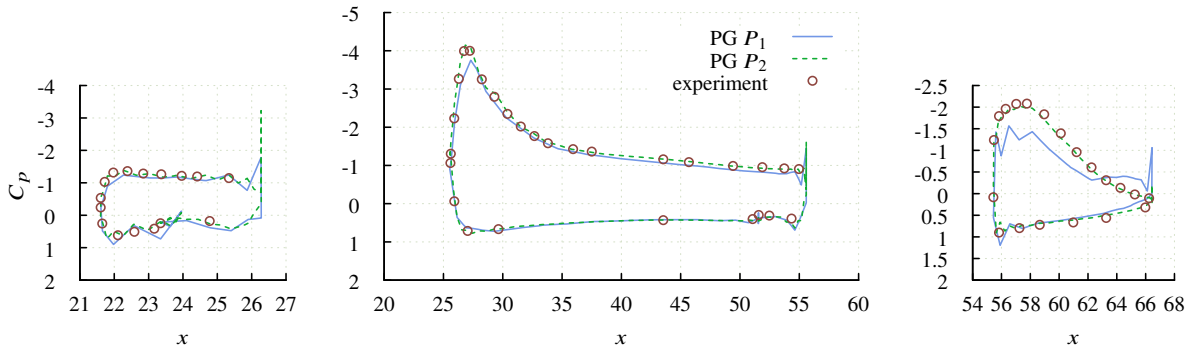
Figure V.21 Pressure coefficients at 17% semi-span on the NASA trap wing



(a) slat (b) main wing (c) flap

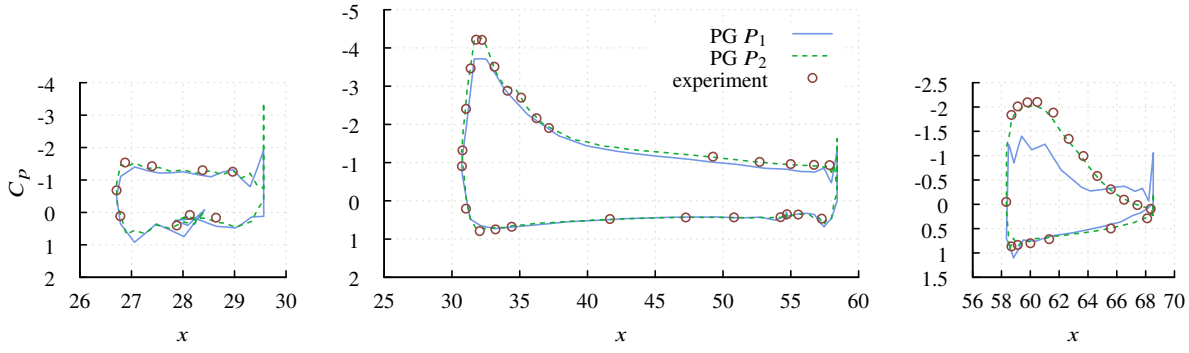
Figure V.22 Pressure coefficients at 28% semi-span on the NASA trap wing

It should be noted that this case was run using the previous time-stepping strategy where the CFL number was held constant until the residual for the turbulence model began to reduce. This was done simply because the newer strategy had not yet been implemented. For the second-order



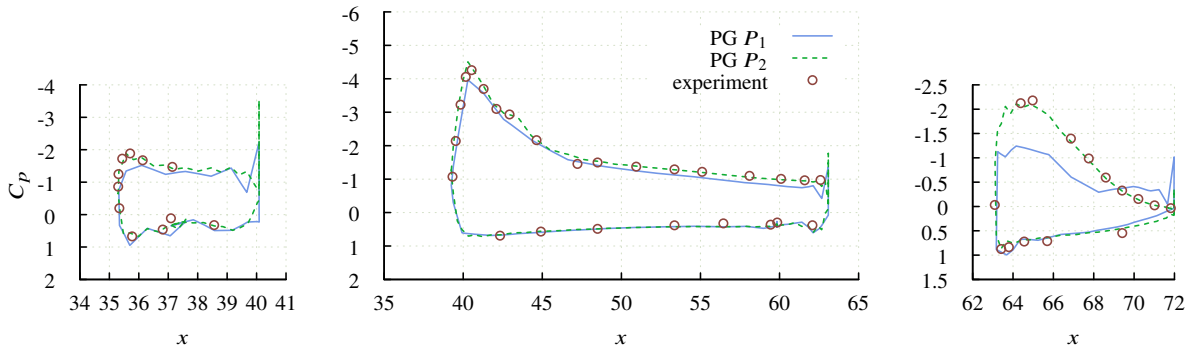
(a) slat (b) main wing (c) flap

Figure V.23 Pressure coefficients at 41% semi-span on the NASA trap wing



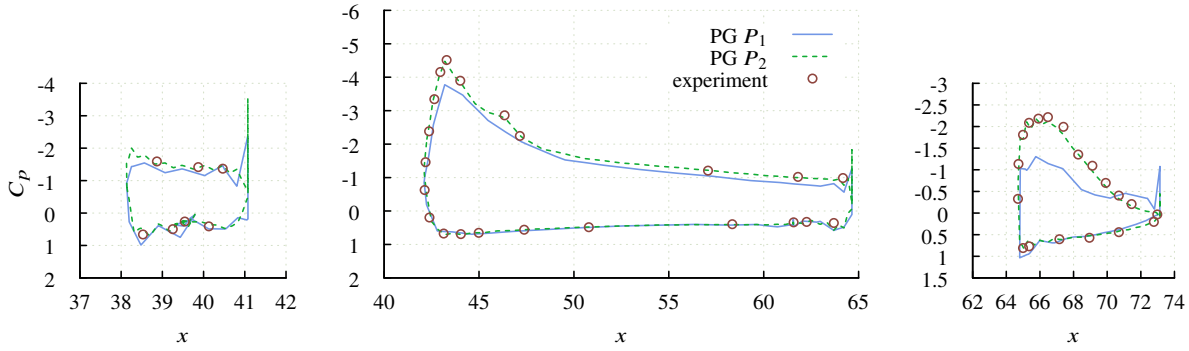
(a) slat (b) main wing (c) flap

Figure V.24 Pressure coefficients at 50% semi-span on the NASA trap wing



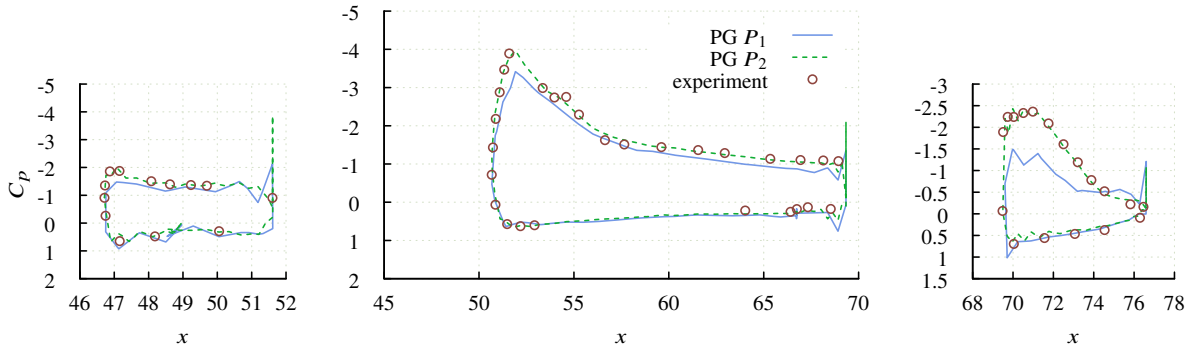
(a) slat (b) main wing (c) flap

Figure V.25 Pressure coefficients at 65% semi-span on the NASA trap wing



(a) slat (b) main wing (c) flap

Figure V.26 Pressure coefficients at 70% semi-span on the NASA trap wing



(a) slat (b) main wing (c) flap

Figure V.27 Pressure coefficients at 85% semi-span on the NASA trap wing

solution, convergence of the density and turbulence working variable were reduced to 10^{-15} and 10^{-14}, respectively. However, for the third-order accurate solution, the density residual has been reduced by four orders of magnitude whereas the residual for the turbulence working variable has only been reduced by two orders of magnitude over its initial value. In this case, it has been found helpful to start the solution at a lower Reynolds number to provide some extra damping.

CHAPTER VI

CONCLUSION

Summary

In this research, the viability of the streamline/upwind Petrov-Galerkin (SUPG) method for obtaining high-order solutions to the compressible Navier-Stokes equations with a modified SA turbulence model has been established. The SUPG discretization is presented as a practical high-order alternative to more popular (but potentially more computationally expensive) discontinuous Galerkin schemes. Using the method of manufactured solutions, the order of accuracy for inviscid and viscous flow solutions has been verified up to fourth order for two-dimensional flow and up to third order for three-dimensional flow. Alternate formulations for the stabilization matrix have been presented using flux functions commonly used in finite-volume schemes. Here, stabilization matrices based on flux-vector splitting as well as flux-difference splitting were shown to provide similar results. For viscous flows, additional terms are added to the stabilization matrix that ensure proper order of accuracy when the viscous terms become dominant at low Reynolds numbers.

The modified SA turbulence model requires that the distance to the viscous surface be known throughout the mesh. In the context of high-order methods, it is important that this distance reflect the true curved geometry. As such, an efficient method for calculating distances to a curved

surface was presented. An octree data structure was utilized to facilitate an efficient nearest neighbor search, while an iterative approach was used to find the distance between a point in space and a curved parametric surface.

The importance of curved surface meshes has been demonstrated, and strategies for producing such meshes have been outlined. In two dimensions, analytic surface definitions were used to accurately capture curved surface geometries, while the interior mesh was deformed using a straight-forward algebraic mesh smoothing approach. Due to the complexities of three-dimensional geometry, a more robust approach was required. Here, a CAPRI interface was used to project the surface mesh onto the CAD defined geometry. To avoid collapsed interior elements in anisotropic boundary layer regions, a linear elasticity solver was utilized to displace the interior mesh. When dealing with curved isoparametric elements, analysis of polynomial basis functions has shown that curved elements require additional degrees of freedom to achieve the proper order characteristics. It was observed through numerical experimentation, however, that the unrepresented terms in the polynomial expansion fall below truncation error for moderately curved elements. Additionally, it was demonstrated that achieving greater than third order accuracy for isoparametric curved elements requires that the curvature of interior elements decrease in a manner not typically achieved in practical application. However, it was shown that subparametric elements can be utilized to obtain the desired order.

Finally, numerical results were presented for several two and three dimensional test cases consisting of both steady and unsteady flows and involving inviscid, laminar, and turbulent flow solutions. The numerical results showed strong agreement with experimental data and/or benchmark results.

Recommendations for Future Work

While the viability of the high-order SUPG scheme for compressible turbulent flows has been established, opportunities for future work remain. As discussed in the introduction, the greatest cost benefit over other high-order schemes occurs at moderate orders. This predicates the need for adaptive *p*-refinement, where initial solutions will consist of low order elements exclusively and discretization is increased on an element-by-element basis as required. In addition, some issues regarding robustness need to be addressed. In particular, the procedure for generating curved three-dimensional meshes about complex geometries proved inadequate in a minority of cases. In this research, the problem was overcome by restricting the curvature of the surface mesh in problem areas. Although this strategy proved effective, such limitations are undesirable and warrant further research. Finally, failure to reach machine convergence in some steady-state cases indicates a need for further study. A more robust time-marching strategy for high-order methods should be explored.

REFERENCES

[1] Vatsa, V. N., Lockard, D. P., and Khorrami, M. R., "Application of FUN3D Solver for Aeroacoustics Simulation of a Nose Landing Gear Configuration," *17th AIAA/CEAS Aeroacoustics Conference*, Portland, OR, June 2011, AIAA Paper 2011-2820.

[2] Cockburn, B., Hou, S., and Shu, C.-W., "The Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws IV: the multidimensional case," *Mathematics of Computation*, Vol. 54, No. 190, 1990, pp. 545–581.

[3] Bassi, F. and Rebay, S., "A High-Order Accurate Discontinuous Finite Element Method for the Numerical Solution of the Compressible Navier-Stokes Equations," *Journal of Computational Physics*, Vol. 131, No. 2, 1997, pp. 267–279.

[4] Bassi, F. and Rebay, S., "High-Order Accurate Discontinuous Finite Element Solution of the 2D Euler Equations," *Journal of Computational Physics*, Vol. 138, No. 2, 1997, pp. 251–285.

[5] Darmofal, D. and Fidkowski, K., "Development of a Higher-Order Solver for Aerodynamic Applications," *42nd AIAA Aerospace Sciences Meeting and Exhibit*, American Institute of Aeronautics and Astronautics, Reno, NV, USA, Jan. 2004, AIAA paper 2004-436.

[6] Fidkowski, K. J., Oliver, T. A., Lu, J., and Darmofal, D., "p-Multigrid solution of high-order Discontinuous Galerkin Discretizations of the Compressible Navier-Stokes equations," *J. Comput. Phys.*, Vol. 207, 2005, pp. 92–113.

[7] Nastase, C. R. and Mavriplis, D. J., "High-order discontinuous Galerkin methods using an hp-multigrid approach," *Journal of Computational Physics*, Vol. 213, No. 1, 2006, pp. 330–357.

[8] Persson, P.-O. and Peraire, J., "An Efficient Low Memory Implicit DG Algorithm for Time Dependent Problems," *44th AIAA Aerospace Sciences Meeting and Exhibit*, American Institute of Aeronautics and Astronautics, Reno, NV, USA, Jan. 2006, AIAA paper 2006-113.

[9] Persson, P.-O. and Peraire, J., "Sub-Cell Shock Capturing for Discontinuous Galerkin Methods," *44th AIAA Aerospace Sciences Meeting and Exhibit*, American Institute of Aeronautics and Astronautics, Reno, NV, USA, Jan. 2006, AIAA paper 2006-112.

[10] Fidkowski, K. J. and Darmofal, D., "A Triangular Cut-cell Adaptive Method for High-order Discretizations of the Compressible Navier-Stokes Equations," *J. Comput. Phys.*, Vol. 225, No. 2, 2007, pp. 1653–1672.

[11] Wang, L., *Techniques for High-order Adaptive Discontinuous Galerkin Discretizations in Fluid Dynamics*, Ph.D. thesis, University of Wyoming, April 2009.

[12] Diosady, L. T. and Darmofal, D. L., "Preconditioning methods for discontinuous Galerkin solutions of the Navier-Stokes equations," *Journal of Computational Physics*, Vol. 228, No. 11, 2009, pp. 3917–3935.

[13] Mavriplis, D., Nastase, C., Shahbazi, K., Wang, L., and Burgess, N., "Progress in High-Order Discontinuous Galerkin Methods for Aerospace Applications," *47th AIAA Aerospace Sciences Meeting*, American Institute of Aeronautics and Astronautics, Orlando, FL, USA, Jan. 2009, AIAA-2009-601.

[14] Dumbser, M., "Arbitrary high order $P_N P_M$ schemes on unstructured meshes for the compressible Navier-Stokes equations," *Computers & Fluids*, Vol. 39, No. 1, 2010, pp. 60–76.

[15] Burgess, N. and Mavriplis, D., "Robust Computation of Turbulent Flows Using a Discontinuous Galerkin Method," *50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, American Institute of Aeronautics and Astronautics, Nashville, TN, USA, Jan. 2012, AIAA Paper 2012-457.

[16] Venkatakrishnan, V., Allmaras, S. R., Kamenetskii, D. S., and Johnson, F. T., "Higher order schemes for the compressible Navier-Stokes equations," *16th AIAA Computational Fluid Dynamics Conference*, American Institute of Aeronautics and Astronautics, Orlando, FL, USA, June 2003, AIAA-2003-3987.

[17] Brooks, A. N. and Hughes, T. J. R., "Streamline Upwind/Petrov-Galerkin Formulations for Convection Dominated Flows with Particular Emphasis on the Incompressible Navier-Stokes Equations," *Computer Methods in Applied Mechanics and Engineering*, Vol. 32, No. 1–3, Sept. 1982, pp. 199–259.

[18] Hughes, T. and Tezduyar, T., "Finite element methods for first-order hyperbolic systems with particular emphasis on the compressible euler equations," *Computer Methods in Applied Mechanics and Engineering*, Vol. 45, No. 1–3, 1984, pp. 217–284.

[19] Hughes, T. J. R., "Recent progress in the development and understanding of SUPG methods with special reference to the compressible Euler and Navier-Stokes equations," *International Journal for Numerical Methods in Fluids*, Vol. 7, No. 11, 1987, pp. 1261–1275.

[20] Tezduyar, T., "Stabilized Finite Element Formulations for Incompressible Flow Computations," Vol. 28 of *Advances in Applied Mechanics*, Elsevier, 1991, pp. 1–44.

[21] Franca, L. P. and Frey, S. L., "Stabilized finite element methods: II. The incompressible Navier-Stokes equations," *Computer Methods in Applied Mechanics and Engineering*, Vol. 99, No. 2–3, 1992, pp. 209–233.

[22] Beau, G. L., Ray, S., Aliabadi, S., and Tezduyar, T., "SUPG finite element computation of compressible flows with the entropy and conservation variables formulations," *Computer Methods in Applied Mechanics and Engineering*, Vol. 104, No. 3, 1993, pp. 397–422.

[23] Soulaïmani, A. and Fortin, M., "Finite element solution of compressible viscous flows using conservative variables," *Computer Methods in Applied Mechanics and Engineering*, Vol. 118, No. 3–4, 1994, pp. 319–350.

[24] Almeida, R. C. and Galeão, A. C., "An adaptive Petrov-Galerkin formulation for the compressible Euler and Navier-Stokes equations," *Computer Methods in Applied Mechanics and Engineering*, Vol. 129, No. 1–2, 1996, pp. 157–176.

[25] Bonhaus, D. L., *A Higher Order Accurate Finite Element Method for Viscous Compressible Flows*, Ph.D. thesis, Virginia Polytechnic Institute and State University, Nov. 1998.

[26] Wong, J. S., Darmofal, D. L., and Peraire, J., "The solution of the compressible Euler equations at low Mach numbers using a stabilized finite element algorithm," *Computer Methods in Applied Mechanics and Engineering*, Vol. 190, No. 43–44, 2001, pp. 5719–5737.

[27] Bassi, F., Crivellini, A., Rebay, S., and Savini, M., "Discontinuous Galerkin solution of the Reynolds-averaged Navier-Stokes and k-[omega] turbulence model equations," *Computers & Fluids*, Vol. 34, No. 4–5, 2005, pp. 507–540.

[28] Tezduyar, T. E. and Senga, M., "Stabilization and shock-capturing parameters in SUPG formulation of compressible flows," *Computer Methods in Applied Mechanics and Engineering*, Vol. 195, No. 13–16, 2006, pp. 1621–1632, A Tribute to Thomas J.R. Hughes on the Occasion of his 60th Birthday.

[29] Kirk, B. S. and Carey, G. F., "Development and validation of a SUPG finite element scheme for the compressible Navier-Stokes equations using a modified inviscid flux discretization," *International Journal for Numerical Methods in Fluids*, Vol. 57, No. 3, 2008, pp. 265–293.

[30] Yano, M. and Darmofal, D. L., "Massively Parallel Solver for the High-Order Galerkin Least Squares Method," *19th AIAA Computational Fluid Dynamics Conference*, American Institute of Aeronautics and Astronautics, San Antonio, TX, USA, June 2009, AIAA-2009-4135.

[31] Kirk, B. S., Bova, S. W., and Bond, R. B., "The Influence of Stabilization Parameters in the SUPG Finite Element Method for Hypersonic Flows," *48th AIAA Aerospace Sciences Meeting*, American Institute of Aeronautics and Astronautics, Orlando, FL, USA, Jan. 2010, AIAA-2010-1183.

[32] Hughes, T. J., Scovazzi, G., and Tezduyar, T. E., "Stabilized Methods for Compressible Flows," *Journal of Scientific Computing*, Vol. 43, No. 3, 2010, pp. 343–368.

[33] Chalot, F. and Normand, P.-E., "Higher-Order Stabilized Finite Elements in an Industrial Navier-Stokes Code," *ADIGMA — A European Initiative on the Development of Adaptive*

Higher-Order Variational Methods for Aerospace Applications, edited by N. Kroll, H. Bieler, H. Deconinck, V. Couaillier, H. van der Ven, and K. Sørensen, Vol. 113 of *Notes on Numerical Fluid Mechanics and Multidisciplinary Design*, chap. 11, Springer Berlin Heidelberg, 2010, pp. 145–165.

[34] Kirk, B. and Oliver, T., "Validation of SUPG Finite Element Simulations of Shock-wave/Turbulent Boundary Layer Interactions in Hypersonic Flows," *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, American Institute of Aeronautics and Astronautics, Grapevine, TX, USA, Jan. 2013, AIAA paper 2013-306.

[35] Erwin, J. T., Anderson, W. K., Kapadia, S., and Wang, L., "Three-Dimensional Stabilized Finite Elements for Compressible Navier-Stokes," *AIAA Journal*, Vol. 51, No. 6, June 2013, pp. 1404–1419.

[36] Erwin, J. T., Anderson, W. K., Wang, L., and Kapadia, S., "High-Order Finite-Element Method for Three-Dimensional Turbulent Navier-Stokes," *21st AIAA Computational Fluid Dynamics Conference*, American Institute of Aeronautics and Astronautics, San Diego, CA, USA, June 2013, AIAA-2013-2571.

[37] Anderson, W. K., Wang, L., Kapadia, S., Tanis, C., and Hilbert, B., "Petrov-Galerkin and Discontinuous-Galerkin Methods for Time-Domain and Frequency-Domain Electromagnetic Simulations," *J. Comput. Phys.*, Vol. 230, 2011, pp. 8360–8385.

[38] Glasby, R. S., Burgess, N. K., Anderson, W. K., Wang, L., Mavriplis, D. J., and Allmaras, S. R., "Comparison of SU/PG and DG Finite-Element Techniques for the Compressible Navier-Stokes Equations on Anisotropic Unstructured Meshes," *51st AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, American Institute of Aeronautics and Astronautics, Grapevine, TX, Jan. 2013, AIAA paper 2013-691.

[39] Moro, D., Nguyen, N.-C., and Peraire, J., "Navier-Stokes Solution Using Hybridizable Discontinuous Galerkin Methods," *20th AIAA Computational Fluid Dynamics Conference*, American Institute of Aeronautics and Astronautics, June 2011, AIAA-2011-3407.

[40] Solin, P., Segeth, K., and Dolezel, I., *High-Order Finite Element Methods*, Studies in Advanced Mathematics, Chapman and Hall, 2003.

[41] Barth, T. J., "Numerical methods for gasdynamic systems on unstructured meshes," *An Introduction to Recent Developments in Theory and Numerics for Conservation Laws*, edited by D. Kroner, M. Ohlberger, and C. Rhode, Vol. 5, Springer, New York, NY, USA, 1998, pp. 195–285.

[42] van Leer, B., "Flux-vector splitting for the Euler equations," *Eighth International Conference on Numerical Methods in Fluid Dynamics*, edited by E. Krause, Vol. 170 of *Lecture Notes in Physics*, Springer Berlin / Heidelberg, 1982, pp. 507–512.

[43] Anderson, W. K., Thomas, J. L., and van Leer, B., "Comparison of finite volume flux vector splittings for the Euler equations," *AIAA journal*, Vol. 24, No. 9, Sept. 1986, pp. 1453–1460.

[44] Steger, J. L. and Warming, R. F., "Flux vector splitting of the inviscid gasdynamic equations with application to finite-difference methods," *Journal of Computational Physics*, Vol. 40, No. 2, 1981, pp. 263–293.

[45] Gressier, J., Villedieu, P., and Moschetta, J.-M., "Positivity of Flux Vector Splitting Schemes," *Journal of Computational Physics*, Vol. 155, No. 1, 1999, pp. 199–220.

[46] Wang, C. and Liu, J., "Positivity property of second-order flux-splitting schemes for the compressible Euler equations," *Discrete and Continuous Dynamical Systems – Series B*, Vol. 3, No. 2, May 2003, pp. 201–228.

[47] Shakib, F., Hughes, T. J., and Johan, Z., "A new finite element formulation for computational fluid dynamics: X. The compressible Euler and Navier-Stokes equations," *Computer Methods in Applied Mechanics and Engineering*, Vol. 89, No. 1–3, 1991, pp. 141–219, Second World Congress on Computational Mechanics.

[48] Franca, L. P., Frey, S. L., and Hughes, T. J. R., "Stabilized finite element methods: I. Application to the advective-diffusive model," *Computer Methods in Applied Mechanics and Engineering*, Vol. 95, No. 2, 1992, pp. 253–276.

[49] Fries, T.-P. and Matthies, H. G., "A Review of Petrov-Galerkin Stabilization Approaches and an Extension to Meshfree Methods," Tech. Rep. Informatikbericht Nr.: 2004-01, Institute of Scientific Computing, Technical University Braunschweig, Brunswick, Germany, March 2004.

[50] Saad, Y. and Schultz, M. H., "GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems," *SIAM J. Sci. Stat. Comput.*, Vol. 7, No. 3, July 1986, pp. 856–869.

[51] Saad, Y., *Iterative methods for sparse linear systems*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2nd ed., 2003.

[52] Gropp, W., Lusk, E., and Skjellum, A., *Using MPI: Portable Parallel Programming with the Message Passing Interface*, MIT Press, Cambridge, MA, 1994.

[53] Karypis, G., "METIS, University of Minnesota, Department of Computer Science," http://www-users.cs.umn.edu/ karypis/metis.

[54] Wang, L., Anderson, W. K., and Erwin, J. T., "Solutions of High-order Methods for Three-dimensional Compressible Viscous Flows," *42nd AIAA Fluid Dynamics Conference and Exhibit*, American Institute of Aeronautics and Astronautics, New Orleans, LA, USA, June 2012, AIAA Paper 2012-2836.

[55] Wang, L., Anderson, W. K., Erwin, J. T., and Kapadia, S., "High-order Methods for Solutions of Three-dimensional Turbulent Flows," *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, American Institute of Aeronautics and Astronautics, Grapevine, TX, USA, Jan. 2013, AIAA Paper 2013-856.

[56] de Barros Ceze, M. A., *A Robust hp-Adaptation Method for Discontinuous Galerkin Discretizations Applied to Aerodynamic Flows*, Ph.D. thesis, University of Michigan, 2013.

[57] Allen, C. B., "Parallel Flow-Solver and Mesh Motion Scheme for Forward Flight Rotor Simulation," *24th AIAA Applied Aerodynamics Conference*, American Institute of Aeronautics and Astronautics, San Fransisco, CA, USA, June 2006, AIAA-2006-3476.

[58] Brock, W., Burdyshaw, C., Karman, S., Betro, V., Hilbert, B., Anderson, K., and Haimes, R., "Adjoint-Based Design Optimization Using CAD Parameterization Through CAPRI," *50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, American Institute of Aeronautics and Astronautics, Nashville, TN, USA, Jan. 2012, AIAA Paper 2012-968.

[59] Zienkiewicz, O. C., Taylor, R. L., and Zhu, J. Z., *The Finite Element Method: Its Basis and Fundamentals*, Butterworth-Heinemann, 30 Corporate Drive, Suite 400, Burlington, MA, USA, 2005.

[60] Salari, K. and Knupp, P., "Code Verification by the Method of Manufactured Solutions," Tech. Rep. SAND2000-1444, Sandia National Laboratories, Albuquerque, NM, USA, June 2000.

[61] Lee, N.-S. and Bathe, K.-J., "Effects of element distortions on the performance of isoparametric elements," *Numerical Methods in Engineering*, Vol. 36, No. 20, 1993, pp. 3553–3576.

[62] McLeod, R., "Node Requirements for High-order Approximation over Curved Finite Elements," *IMA Journal of Applied Mathematics*, Vol. 17, No. 2, 1976, pp. 249–254.

[63] McLeod, R., "High Order Transformation Methods for Curved Finite Elements," *IMA Journal of Applied Mathematics*, Vol. 21, No. 4, 1978, pp. 419–428.

[64] Šolín, P., Segeth, K., and žel, I., *Higher-order finite element methods*, Vol. 1, CRC Press, Boca Raton, FL, USA, 2004.

[65] Ciarlet, P. G., *The Finite Element Method for Elliptic Problems*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2nd ed., 2002.

[66] Ciarlet, P. G. and Raviart, P., "Interpolation theory over curved elements, with application to finite element methods," *Computer Methods in Applied Mechanics and Engineering*, Vol. 1, No. 2, 1972, pp. 217–249.

[67] Johnson, C., *Numerical Solution of Partial Differential Equations by the Finite Element Method*, Dover, New York, NY, USA, 2009.

[68] van Leer, B., Thomas, J. L., Roe, P. L., and Newsome, R. W., "A comparison of numerical flux formulas for the Euler and Navier-Stokes equations," *8th AIAA Computational Fluid Dynamics Conference*, American Institute of Aeronautics and Astronautics, Honolulu, HI, USA, June 1987, AIAA-1987-1104.

[69] Hänel, D., Schwane, R., and Seider, G., "On the accuracy of upwind schemes for the solution of the Navier-Stokes equations," *8th AIAA Computational Fluid Dynamics Conference*, American Institute of Aeronautics and Astronautics, Honolulu, HI, USA, June 1987, pp. 42–46, AIAA-1987-1105.

[70] Coutanceau, M. and Bouard, R., "Experimental determination of the main features of the viscous flow in the wake of a circular cylinder in uniform translation. Part 1. Steady flow," *Journal of Fluid Mechanics*, Vol. 79, No. 02, 1977, pp. 231–256.

[71] Coutanceau, M. and Bouard, R., "Experimental determination of the main features of the viscous flow in the wake of a circular cylinder in uniform translation. Part 2. Unsteady flow," *Journal of Fluid Mechanics*, Vol. 79, No. 2, 1977, pp. 257–272.

[72] Thomas, J. L., Krist, S. T., and Anderson, W. K., "Navier-Stokes computations of vortical flows over low-aspect-ratio wings," *AIAA journal*, Vol. 28, No. 2, Feb. 1990, pp. 205–212.

[73] Mohammad, A. H. and Wang, Z., "Large Eddy Simulation of Flow Over a Cylinder Using High-order Spectral Difference Method," *Advances in Applied Mathematics and Mechanics*, Vol. 2, No. 4, 2011, pp. 451–466.

[74] Rumsey, C., Slotnick, J., Long, M., Stuever, R., and Wayman, T., "Summary of the First AIAA CFD High-Lift Prediction Workshop," *Journal of Aircraft*, Vol. 48, No. 6, 2011, pp. 2068–2079.

VITA

Jon Taylor Erwin was born in July of 1983 in Little Rock, Arkansas, to Dennis and Evonna Erwin. He attended elementary school in Malvern, Arkansas, before moving to Conway, Arkansas, in 1995. There Taylor developed interest in and aptitude for mathematics and computer science. He graduated with honors from Conway High School in 2001. Taylor earned a Bachelor of Science degree in Mathematics, with a minor in Computer Science, from the University of Central Arkansas in 2007. He continued his education at the University of Central Arkansas, serving as a graduate teaching assistant while pursuing a Master of Science degree in Applied Mathematics. Upon graduation in 2009, Taylor moved to Chattanooga, Tennessee, accepting a graduate research assistantship at the University of Tennessee at Chattanooga SimCenter. Taylor graduated with a Ph.D. in Computational Engineering in December of 2013.