A Parallel Newton-Krylov-Schur Algorithm for the Reynolds-Averaged Navier-Stokes Equations

by

Michal Osusky

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Aerospace Science and Engineering
University of Toronto

Abstract

A Parallel Newton-Krylov-Schur Algorithm for the Reynolds-Averaged Navier-Stokes Equations

Michal Osusky
<michal.osusky@mail.utoronto.ca>

Doctor of Philosophy
Graduate Department of Aerospace Science and Engineering
University of Toronto
2013

Aerodynamic shape optimization and multidisciplinary optimization algorithms have the potential not only to improve conventional aircraft, but also to enable the design of novel configurations. By their very nature, these algorithms generate and analyze a large number of unique shapes, resulting in high computational costs. In order to improve their efficiency and enable their use in the early stages of the design process, a fast and robust flow solution algorithm is necessary.

This thesis presents an efficient parallel Newton-Krylov-Schur flow solution algorithm for the three-dimensional Navier-Stokes equations coupled with the Spalart-Allmaras one-equation turbulence model. The algorithm employs second-order summation-by-parts (SBP) operators on multi-block structured grids with simultaneous approximation terms (SATs) to enforce block interface coupling and boundary conditions. The discrete equations are solved iteratively with an inexact-Newton method, while the linear system at each Newton iteration is solved using the flexible Krylov subspace iterative method GMRES with an approximate-Schur parallel preconditioner.

The algorithm is thoroughly verified and validated, highlighting the correspondence of the current algorithm with several established flow solvers. The solution for a transonic flow over a wing on a mesh of medium density (15 million nodes) shows good agreement with experimental results. Using 128 processors, deep convergence is obtained in under 90 minutes. The solution of transonic flow over the Common Research Model wing-body geometry with grids with up to 150 million nodes exhibits the expected grid convergence

behavior. This case was completed as part of the Fifth AIAA Drag Prediction Workshop, with the algorithm producing solutions that compare favourably with several widely used flow solvers. The algorithm is shown to scale well on over 6000 processors. The results demonstrate the effectiveness of the SBP-SAT spatial discretization, which can be readily extended to high order, in combination with the Newton-Krylov-Schur iterative method to produce a powerful parallel algorithm for the numerical solution of the Reynolds-averaged Navier-Stokes equations. The algorithm can efficiently solve the flow over a range of clean geometries, making it suitable for use at the core of an optimization algorithm.

Acknowledgements

First and foremost, I would like to thank my supervisor, Dr. David W. Zingg, for his time and guidance during the past years. His enthusiasm and optimism have motivated my work and pushed it beyond what I had ever expected. Beyond that, he has been a part of many memorable moments on the basketball court, both as an opponent and a teammate.

I would like to thank the members of my doctoral committee, Drs. Clinton Groth and Philippe Lavoie, for their contributions to the direction in which this thesis has evolved, as well as Dr. Joaquim Martins for his input during the early stages of this work. For their involvement in the final oral examination, I thank Drs. Steve Allmaras and Markus Bussmann. In particular, Dr. Allmaras provided key insights through his appraisal of the thesis.

To all the students and staff at UTIAS, both past and present, I extend my deepest thanks for not only making UTIAS a great place to work, but also creating a supportive and fun community. In particular, I would like to thank Dr. Jason Hicken for the uncompromising foundation he set for the algorithm that served as the basis of this work; he established a level of excellence that I will always strive to approach. To Dr. Gaetan Kenway, a big thanks for many lively discussions and intense ultimate frisbee matches. The support received from Scott Northrup was critical in getting the code to run smoothly on Scinet, and it is much appreciated. And to all the students of the CFD and MDO groups, too many to name, I offer a huge thank you for their friendship, as well as all the time and discussions that helped shape this work.

None of this would have been possible without my family. Their unwavering support provided a constant source of motivation, especially during the weeks and months when it was needed most.

Most of all, to Lana, I am indebted and grateful for everything. Having you by my side during my time at UTIAS has made it all seem a little easier, within reach, and never lonely.

Financial support from the Natural Sciences and Engineering Research Council of Canada, the Government of Ontario, and the University of Toronto is gratefully acknowledged.

Michal Osusky

University of Toronto Institute for Aerospace Studies
June 15, 2013

Contents

List of Tables

List of Figures

List of Symbols and Abbreviations

Alphanumeric

a	sound speed
A	local (node) flow Jacobian
$\mathcal{A}$	global flow Jacobian
c	reference chord length
C_D, C_d	coefficient of drag
C_L, C_l	coefficient of lift
C_M	coefficient of moment
C_p, C_f	coefficients of pressure and friction
d	distance to closest surface boundary
D_1, D_2	SBP operators for first and second derivative
e	energy
$\hat{\mathbf{E}}, \hat{\mathbf{F}}, \hat{\mathbf{G}}$	inviscid flux vectors
$\hat{\mathbf{E}}_\mathbf{v}, \hat{\mathbf{F}}_\mathbf{v}, \hat{\mathbf{G}}_\mathbf{v}$	viscous flux vectors
$\mathcal{I}$	identity matrix
j, k, m	coordinate indices
J	metric Jacobian
M	Mach number
N	number of nodes in computational domain
p	pressure
Pr, Pr_t	laminar and turbulent Prandtl number
$\mathbf{Q}$	conservative flow variables at individual node
$\boldsymbol{\mathcal{Q}}$	conservative flow vector for computational domain
$\boldsymbol{\mathcal{R}}$	flow residual vector for computational domain
Re	Reynolds number
S^*	Sutherland's law constant (198.6°R)
$S_\mathrm{r}, S_\mathrm{c}, S_\mathrm{a}$	row and column scaling matrices
$S, \tilde{S}$	vorticity and vorticity-like term
S_ref	reference area
Δt	time step value

T	temperature
u, v, w	Cartesian velocities
U, V, W	contravariant velocities
x, y, z	Cartesian coordinate axes

Greek

ϵ	Frechet derivative perturbation parameter
γ	specific heat ratio
κ_2, κ_4	dissipation coefficients
κ_{2t}, κ_{4t}	dissipation coefficients for turbulence model
μ	dynamic laminar viscosity
μ_t	turbulent eddy viscosity
$\tilde{\nu}$	turbulence model working variable
ρ	density
σ_{dlf}	dissipation lumping factor
σ_{SA}	dissipation lumping factor for turbulence model
τ_{ij}	viscous stress tensor
ξ, η, ζ	curvilinear coordinates

Abbreviations

AIAA	American Institute for Aeronautics and Astronautics
CFD	Computational Fluid Dynamics
DBC	Dissipation Based Continuation
DPW	Drag Prediction Workshop
GMRES	Generalized Minimal RESidual
ILU	Incomplete Lower-Upper
MAC	Mean Aerodynamic Chord
PTC	Pseudo-Transient Continuation
RANS	Reynolds-Averaged Navier-Stokes
SAT	Simultaneous Approximation Term
SBP	Summation-by-Parts
TMR	Turbulence Modeling Resource

Chapter 1

Introduction

1.1 Background

The numerical solution of fluid flow has long been envisioned as a replacement for experimental studies, replacing purpose-built wind tunnels with large data centres, housing computers that can be applied not only to the solution of fluid flow problems, but also to unlocking the genetic structure of humans or the origins of the universe. Computational Fluid Dynamics (CFD) provides the capability of analyzing arbitrary aerodynamic shapes at arbitrary flow conditions, just by changing the input parameters supplied to the solution algorithm. This would not only be impractical in an experimental setting, but often impossible, as many wind tunnels are built to operate in a narrow range of flow conditions. Continually improved computer hardware and algorithms have advanced the field of CFD tremendously in recent decades. For example, simulations of two-dimensional airfoils can be routinely run on a laptop computer in a matter of seconds, a task that required cutting-edge computer hardware in the recent past. Many advancements have allowed CFD to become the primary aerodynamic design tool in the development of new aircraft. Nevertheless, much work remains to be done, in terms of algorithm and hardware advances, in order to enable high-fidelity multi-disciplinary optimization to be used early in the design of an aircraft, which is when many important design decisions are made. In particular, the results from the AIAA Drag Prediction Workshop [61] indicate that grids with over $O(10^8)$ grid nodes are required for accurate, grid-converged lift and drag values for flows over a wing-body configuration. Grids of this size still present a considerable challenge even for state of the art solution algorithms. Hence, significant advances can still be made in algorithm development, allowing high-fidelity tools to be introduced into earlier parts of the design process where faster turnaround times are a necessity.

As CFD is intimately tied to the capabilities of the computational hardware on which it is run, many of the recent developments in the field have been tied to a paradigm shift in the development of large computer systems. The 1990s saw a push for ever increasing speeds of computer processors, fueled by Moore's law, which led to the development of highly

efficient and capable serial solution algorithms. However, massive power requirements and a stall in the ability of computer chip manufacturers to continually decrease the size of chip manufacturing processes (allowing them to increase the number of transistors on a chip die, which was responsible for much of the enormous speed increases), led to a slow-down in the increase of processor speeds. To circumvent this, a shift to parallel computer systems in mainstream computing took place. Instead of containing a few high-speed processors, many modern computer systems boast arrays of multi-core processors [26, 47], naturally giving rise to parallel flow solution algorithms. In fact, it is not atypical for research institutes to have access to supercomputer clusters with in excess of tens of thousands of processors. At the extreme end, graphical processing units, or GPUs, which possess hundreds of individual processing cores on a single chip, are seeing use in the field of parallel computational algorithms, though with limited adoption due to a high entry barrier in terms of ease of use and the availability of efficient and user-friendly coding languages and compilers.

Even though supercomputers provide access to large numbers of processors, the high cost of memory has resulted in systems with distributed memory characteristics, leading to the need for parallel algorithms that efficiently split up the computational problem into smaller chunks (to be solved concurrently across hundreds, or even thousands, of processors). This is an important requirement that can shape the choice of methods used in an algorithm.

By making use of the ever-increasing access to large supercomputer clusters, large-scale CFD simulations are becoming more suitable for dealing with problems of practical interest. Hence, algorithms that scale well with thousands of processors are required to make efficient use of computational resources and advance the state of the art in CFD. Additionally, aerodynamic shape and multidisciplinary optimization algorithms, which have the potential to be used in early stages of the aircraft design process, perform many flow solutions on a potentially large number of unique shapes during the course of an optimization. With a requirement for fast turnaround times, such algorithms require fast and robust flow solvers.

1.2 Solving the Reynolds-Averaged Navier-Stokes Equations

The methods used for the simulation of fluid flow can be broadly divided by the level of fidelity they provide. The panel method [32] provides an example of a low-fidelity approach to obtaining fluid flow properties around an aerodynamic shape. While inexpensive to compute, this method is limited to fully-attached subsonic flows and provides only pressure drag information, inherently placing a barrier on the range of its applicability in the analysis of

complex flows. High-fidelity methods provide a more complete representation of the fluid flow at the expense of increased computational cost. Several approaches fall under this category, divided by the assumptions they make in the simulation of turbulence. At one end, the solution of the Reynolds-averaged Navier-Stokes (RANS) equations requires the modeling of turbulence within the fluid flow. For this purpose, several turbulence modeling approaches have been developed, ranging from algebraic methods (for examples, see references [8, 90]) to one- and two-equation turbulence models. The Spalart-Allmaras one-equation turbulence model [91] has been shown to be especially effective for external aerodynamic flows and is in widespread use today. Two-equation models are also commonly used, such as the κ-ω two-equation model, with several variants in existence [31, 107], and the Menter SST model [63]. The RANS equations provide an accurate and detailed representation of the fluid flow, but are often limited by the range of applicability of the turbulence model being used. At the other extreme, direct numerical simulation (DNS) provides the most rigorous representation of the fluid flow properties by directly simulating all aspects of the flow, including turbulence. However, the enormous computational requirements for these simulations, stemming from the need for extremely fine grids, limits the applicability of this approach to small, model problems that are aimed at understanding the fundamentals of fluid flow, rather than the simulation of flows of practical interest. Several methods that lie between RANS and DNS enjoy widespread use, such as large-eddy simulation (LES) [87], which directly simulates large turbulent structures, but applies modeling to the smallest scales of turbulent eddies. For the purposes of aircraft design, the RANS equations provide a sufficient representation of fluid flow at a reasonable computational expense. This is especially important from the standpoint of a high-fidelity aerodynamic optimization algorithm, which requires the computation of a flow solution during each design cycle.

The solution of the RANS equations requires the algorithm developer to make a number of choices. At the core of this decision is the choice between explicit and implicit solution strategies. Explicit solution algorithms require less development time and memory during the computational process, but possess small stability limits. This results in a necessary limit on the time step employed in evolving a solution, leading to long computational times for deep convergence of the governing equations. In addition, the convergence rate of these algorithms is adversely affected by the presence of high-aspect-ratio cells in the computational grid, which are often found in the boundary layers of the grids used in turbulent flow solutions. The use of multigrid for accelerating convergence is critical in obtaining a computationally efficient explicit method. An overview of multigrid is presented by Brandt [14]. An example of using an explicit method for solving the Navier-Stokes equations can be found in the work

of Martinelli *et al.* [56].

The limit on the stability of explicit solution techniques led to the development of implicit algorithms. A notable early entrant into this class of methods is the approximate factorization method developed by Beam and Warming [10] and further developed by Pulliam and Steger [82] in ARC3D, which evolved into a very efficient and widely used flow solver.

The use of fully implicit approaches in the solution algorithm is focused around the Newton method. Unlike explicit methods, most implicit methods are unconditionally stable, but due to the linearization of the governing equations, the calculation and storage of the flow Jacobian matrix is required, placing heavy requirements on memory. Several methods for solving the linear system have been developed. Direct solvers enable quadratic convergence, with examples of this approach presented by Venkatakrishnan [104], Hafez *et al.* [30], and Venkatakrishnan and Barth [105]. While the use of direct solvers in the Newton approach results in impressive convergence characteristics, the high computational cost, in terms of both processor time and memory, make the application of these methods to large problems prohibitively expensive.

To reduce the memory requirements of the direct solver, the linear system can instead be solved by an iterative approach. Iterative solvers can be allowed to terminate once a desired level of convergence is achieved in the solution of the linear system, substantially reducing the computational expense. Saad and van der Vorst [86] provide a comprehensive summary of iterative methods. In particular, the subclass of Krylov methods, resulting in Newton-Krylov algorithms, has gained popularity in the solution of fluid dynamics problems. One key aspect of Krylov methods is that they allow for the reduction of a substantial memory storage burden. By requiring only matrix-vector products, which can be approximated using Frechet derivatives, explicit formation of the flow Jacobian matrix is not required, resulting in what are known as Jacobian-free methods. A comprehensive survey of methods of this type is provided by Knoll and Keyes [48]. Of particular interest among Krylov methods, the Generalized Minimal Residual (GMRES) linear solver of Saad and Schultz [84] has been used extensively, including in the present work. Examples of the use of Newton-Krylov methods can be found in the works of Ajmani *et al.* [2], Nielsen *et al.* [69], and Anderson *et al.* [5]. Pueyo and Zingg [80] have compared the Newton-Krylov and approximate factorization methods, concluding that the former results in faster convergence. Newton-Krylov methods have been shown to solve turbulent flow problems efficiently in serial implementations [19,80].

The effectiveness of the Newton-Krylov algorithm in solving the equations of flow hinges on the use of efficient preconditioning. By improving the conditioning of the linear system, a good preconditioner can have a significant impact on the rate of convergence of the algorithm.

Incomplete lower upper (ILU) factorization has been widely used as part of Newton-Krylov methods. The storage requirements of this preconditioner can be controlled by limiting the level of fill, p, in the factorization, with notation of ILU(p). Examples of the use of ILU can be found in the work of Nielsen *et al.* [69], Anderson *et al.* [5], Forsyth and Jiang [25], Pueyo and Zingg [80], Blanco and Zingg [12], and Geuzaine [27]. While many efficient approaches have been developed for serial algorithms, most of these do not readily apply to parallel computations. Hicken *et al.* [36] have presented a study of both the additive-Schwarz [45,83] and the approximate-Schur [85] parallel preconditioners, concluding that the approximate-Schur preconditioner scales well to at least 1000 processors when inviscid and laminar flows are considered. This is an important result that will serve as the basis of the current work.

In addition to the solution method, the algorithm developer has to decide which type of spatial discretization strategy is to be used. The choice of spatial discretization method and grid type is heavily influenced by the class of flows and the complexity of geometries the algorithm is to be applied to. For smooth geometries, Shu [89] states that finite-difference methods are more efficient in terms of computational cost and programming complexity than finite-volume or discontinuous-Galerkin methods. Furthermore, finite-difference methods can be extended to higher order in a relatively straightforward manner.

While the use of unstructured grids can result in automated meshing for arbitrary geometries, structured multi-block grids offer a sufficiently effective meshing approach for clean geometries. Additionally, they allow for a straightforward application of the finite-difference discretization approach. By breaking the computational domain into several subdomains, or blocks, the approach lends itself readily for use with parallel solution algorithms. Grids created using this method, however, introduce block interfaces into the computational domain. The treatment of the spatial discretization at interfaces can impact the efficiency of an algorithm, especially in large parallel applications.

In the current algorithm, simultaneous approximation terms (SATs) are used to impose boundary conditions, as well as inter-block solution coupling, through a penalty method approach. SATs were originally introduced to treat boundary conditions in an accurate and time-stable manner [15], and later extended to apply to block interfaces [16, 33, 71]. Svärd *et al.* [95, 96] and Nordström *et al.* [72] have shown the application of SATs for the Navier-Stokes equations to unsteady problems, as well as some steady model problems. This approach has several advantages over more traditional approaches: it eliminates the need for mesh continuity across block interfaces, further simplifying mesh generation; it reduces the communication for parallel algorithms, especially when extended to higher-order discretizations; and it ensures linear time stability when coupled with summation-by-parts (SBP) op-

erators. Gong and Nordström [28] have recently extended the SBP discretization approach to hybrid meshes, using both structured and unstructured subdomains. SBP operators provide finite-difference approximations to derivatives, while also presenting a systematic means of deriving higher-order operators with a stable and suitably high-order boundary treatment. Consequently, a high-order SBP-SAT finite-difference discretization is a promising alternative to other high-order approaches, such as finite-volume and discontinuous-Galerkin methods. However, the SBP-SAT approach has received limited use in computational aerodynamics applications, in part because SATs present a difficulty in that they can necessitate the use of small time steps with explicit solvers [70]. Hence, the combination of SATs with a parallel Newton-Krylov solver has the potential to be an efficient approach, as demonstrated by Hicken and Zingg [37]. Although the algorithm presented is second-order in space, with the exception of the convective terms in the turbulence model, it can be efficiently extended to higher-order [21, 75].

State of the art flow solution algorithms today employ a combination of the methods described above. Solvers such as OVERFLOW [42], FUN3D [69], CFL3D [50], and NSU3D [60] implement a variety of numerical approaches, including the use of structured or unstructured grids, finite-volume, finite-element, or finite-difference approximations, explicit or implicit solution strategies, and a wide range of linear solvers and preconditioners. The current work presents an efficient parallel three-dimensional multi-block structured solver for turbulent flows over aerodynamic geometries, extending previous work [37] on an efficient parallel Newton-Krylov flow solver for the Euler equations. The combination of techniques employed in the current solver also has the benefit of lending itself to a unified approach for performing both steady and implicit unsteady computations [75]. An efficient and robust Newton-Krylov flow solver can readily serve as the core of an aerodynamic optimization algorithm, as was demonstrated by Nemec and Zingg [66] for two-dimensional turbulent flow and by Hicken and Zingg [39] for three-dimensional inviscid flow. More recent work by Osusky and Zingg [74] has made use of the current flow solution algorithm for three-dimensional aerodynamic shape optimization in the turbulent flow regime.

Due to the large number of approaches available in the solution of the RANS equations, the verification and validation of the approaches on a common set of test cases is required. For this purpose, the Turbulence Modeling Resources website [1] provides a large selection of two- and three-dimensional cases that can be used to validate the implementation of the governing equations, including the turbulence model. In particular, the resources provided on the website allow for detailed solution comparisons against experimental results, and, more importantly, against well-established flow solution algorithms with a variety of turbu-

lence models. This resource is an invaluable tool in the verification and validation of newly developed algorithms.

While the simple test cases are an important part of the development process, a means of evaluating an algorithm for cases of practical interest is required. This opportunity is provided by the AIAA Drag Prediction Workshop (DPW) series, which invites participants from government, industry, and academia to apply their algorithms to the solution of complex CFD problems. The first workshop, summarized by Levy *et al.* [53], involved the solution of the DLR-F4 wing-body geometry. Large discrepancies were observed between the solutions produced by the 18 participants. The second and third workshops, using the DLR-F6 geometry, resulted in similar observations. The large number of grids used by the various solution algorithms was thought to contribute to the scatter in results. The third workshop also introduced wing-alone configurations in an attempt to provide a simpler geometry with simpler flow features. The fourth workshop introduced the Common Research Model wing-body geometry [102], designed to reduce the amount of side-of-body separation that was thought to be the cause of much of the scatter in the results for the DLR geometries. The results of this workshop are summarized by Vassberg *et al.* [103]. Less scatter was observed in the results, but significant variance was still seen, especially between algorithms that make use of different turbulence models. The fifth workshop introduced a common grid family to be used by all participants [101], in an effort to further reduce the variance in the solutions. Participating in the fifth workshop can serve as a demonstration of the efficiency, robustness, and accuracy of the current algorithm.

1.3 Thesis Outline and Summary of Objectives

The primary goal of this thesis is to determine whether the use of a parallel Newton-Krylov-Schur algorithm with the SBP-SAT discretization can result in an accurate, efficient, and robust flow solver for the RANS equations. To facilitate this, the primary objective is the development and characterization of the NKS-SBP-SAT approach for the viscous terms of the Navier-Stokes equations and the Spalart-Allmaras turbulence model. The use of the SAT penalties presented in [72,95,96] will be novel in that they will be applied in a steady Newton-Krylov-Schur flow solver for the RANS equations. The algorithm will be used to solve compressible turbulent flows around clean geometries, either for analysis or optimization, so it is important that its behaviour for a wide range of flow conditions and geometries be understood; in the case of aerodynamic shape optimization, algorithm efficiency and robustness is critical, as a wide range of geometries could potentially be encountered in the

iterative optimization process.

The thesis is organized as follows. Chapter 2 presents a summary of the governing equations, while Chapter 3 provides an overview of the spatial discretization technique at the core of the current flow solution strategy. The solution strategy by which the discretized governing equations are solved is detailed in Chapter 4. Chapter 5 summarizes the efforts behind optimizing the current algorithm, both in terms of accuracy and efficiency. Results are presented in Chapter 6, which contains a wide array of flow solution examples demonstrating the robustness of the flow solver. Conclusions, contributions, and recommendations are given in Chapter 7.

Chapter 2

Governing Equations

This chapter describes the governing equations that are solved by the algorithm developed for this thesis. These include the three-dimensional Navier-Stokes equations and the Spalart-Allmaras one-equation turbulence model, as well as the boundary conditions and turbulence model modifications investigated.

2.1 Navier-Stokes Equations

The compressible Navier-Stokes equations govern the flow of a compressible fluid. They consist of a set of partial differential equations that represent the conservation of mass, momentum, and energy. The three-dimensional Navier-Stokes equations in Cartesian coordinates in conservative, non-dimensional form are given by

$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{\partial \mathbf{E}}{\partial x} + \frac{\partial \mathbf{F}}{\partial y} + \frac{\partial \mathbf{G}}{\partial z} = \frac{1}{Re}\left(\frac{\partial \mathbf{E_v}}{\partial x} + \frac{\partial \mathbf{F_v}}{\partial y} + \frac{\partial \mathbf{G_v}}{\partial z}\right), \tag{2.1}$$

for which the non-dimensional quantities are defined as

$$t = \frac{\tilde{t}\tilde{a}_\infty}{c}, \quad x = \frac{\tilde{x}}{c}, \quad y = \frac{\tilde{y}}{c}, \quad z = \frac{\tilde{z}}{c}, \quad \rho = \frac{\tilde{\rho}}{\tilde{\rho}_\infty}, \quad u_i = \frac{\tilde{u}_i}{\tilde{a}_\infty}, \quad e = \frac{\tilde{e}}{\tilde{\rho}_\infty \tilde{a}_\infty^2}, \quad \mu = \frac{\tilde{\mu}}{\tilde{\mu}_\infty},$$

where (x, y, z) are the Cartesian coordinates, ρ is the density, a is the sound speed, taken as $\sqrt{\gamma p/\rho}$ for an ideal gas, e is the energy, u_i are the Cartesian velocity components (u, v, w), μ is the molecular viscosity, and c is the characteristic (reference) length. The specific heat ratio, γ, for air is 1.4. The '∞' subscript denotes a free-stream value for the given quantity, while the '˜' symbol denotes a dimensional quantity. The equation of state for an ideal gas is used to define the pressure, p, as

$$p = (\gamma - 1)\left(e - \frac{1}{2}\rho\left(u^2 + v^2 + w^2\right)\right). \tag{2.2}$$

Furthermore, the Reynolds number is given by

$$Re = \frac{\rho_\infty a_\infty c}{\mu_\infty}. \tag{2.3}$$

Since the Reynolds number is defined in terms of a characteristic length, it is critical that the geometry (and grid) used in computations is nondimensionalized by the same value. This is especially important when comparing to experimental results. For example, if a given Reynolds number is specified in terms of the mean aerodynamic chord (MAC), the geometry in the computational mesh should have an MAC of unity. Alternately, the Reynolds number can be adjusted by an appropriate scaling factor, taking into account the given Reynolds number reference length and whichever length is equal to unity in the computational grid. A typical conversion that is often encountered is between the Reynolds number specified for the MAC, Re_{MAC}, often used for experimental results, and that for the root chord, Re_{RC},

$$Re_{\mathrm{RC}} = Re_{\mathrm{MAC}} \frac{c_{\mathrm{RC}}}{c_{\mathrm{MAC}}}, \tag{2.4}$$

where c_{MAC} and c_{RC} are the respective chord lengths. Most of the studies presented in this thesis make use of the root chord definition of the Reynolds number and care is taken to highlight any solutions where this is not the case.

The Navier-Stokes equations operate on the non-dimensional conservative variables

$$\mathbf{Q} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ e \end{bmatrix}, \tag{2.5}$$

which represent mass, momentum, and energy. The inviscid flux vectors are defined as

$$\mathbf{E} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ u(e+p) \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vw \\ v(e+p) \end{bmatrix}, \quad \mathbf{G} = \begin{bmatrix} \rho v \\ \rho uw \\ \rho vw \\ \rho w^2 + p \\ w(e+p) \end{bmatrix}, \tag{2.6}$$

and the viscous flux vectors are defined as

$$\mathbf{E_v} = \begin{bmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ \tau_{xz} \\ E_{v,5} \end{bmatrix}, \quad \mathbf{F_v} = \begin{bmatrix} 0 \\ \tau_{yx} \\ \tau_{yy} \\ \tau_{yz} \\ F_{v,5} \end{bmatrix}, \quad \mathbf{G_v} = \begin{bmatrix} 0 \\ \tau_{zx} \\ \tau_{zy} \\ \tau_{zz} \\ G_{v,5} \end{bmatrix}. \tag{2.7}$$

The viscous stress and heat conduction terms are given by

$$\begin{aligned}
\tau_{xx} &= \tfrac{4}{3}(\mu+\mu_t)(u_x) - \tfrac{2}{3}(\mu+\mu_t)\left(v_y+w_z\right), \\
\tau_{xy} &= (\mu+\mu_t)\left(u_y+v_x\right), \\
\tau_{xz} &= (\mu+\mu_t)\left(u_z+w_x\right), \\
\tau_{yy} &= \tfrac{4}{3}(\mu+\mu_t)(v_y) - \tfrac{2}{3}(\mu+\mu_t)\left(u_x+w_z\right), \\
\tau_{yz} &= (\mu+\mu_t)\left(v_z+w_y\right), \\
\tau_{zz} &= \tfrac{4}{3}(\mu+\mu_t)(w_z) - \tfrac{2}{3}(\mu+\mu_t)\left(u_x+v_y\right), \\
\tau_{yx} &= \tau_{xy}, \quad \tau_{zx} = \tau_{xz}, \quad \tau_{zy} = \tau_{yz}, \\
E_{v,5} &= u\tau_{xx} + v\tau_{xy} + w\tau_{xz} + (\mu Pr^{-1} + \mu_t Pr_t^{-1})(\gamma-1)^{-1}\partial_x(a^2), \\
F_{v,5} &= u\tau_{yx} + v\tau_{yy} + w\tau_{yz} + (\mu Pr^{-1} + \mu_t Pr_t^{-1})(\gamma-1)^{-1}\partial_y(a^2), \\
G_{v,5} &= u\tau_{zx} + v\tau_{zy} + w\tau_{zz} + (\mu Pr^{-1} + \mu_t Pr_t^{-1})(\gamma-1)^{-1}\partial_z(a^2),
\end{aligned} \tag{2.8}$$

where μ_t is the turbulent eddy viscosity. The laminar and turbulent Prandtl numbers are defined as $Pr = 0.72$ and $Pr_t = 0.90$. The above terms make use of the simplified derivative notation, where both $\partial_x u$ and u_x are taken to represent $\frac{\partial u}{\partial x}$.

Molecular viscosity is calculated using the non-dimensional form of Sutherland's law [106],

$$\mu = \frac{a^3(1+S^*/T_\infty)}{a^2+S^*/T_\infty}, \tag{2.9}$$

where S^* is Sutherland's constant (198.6°R), and T_∞ is the free-stream temperature, typically set to 460°R for air.

2.1.1 Curvilinear coordinate transformation

In order to solve the governing equations in the domain surrounding an aerodynamic shape, such as a wing, a structured grid is created to represent discrete points in the physical space. A mathematical transformation is used to map the governing equations from the physical grid to an equally spaced, uniform computational grid, greatly simplifying the process of discretizing the governing equations. Instead of solving the equations in physical coordinates (x, y, z), a computational coordinate system (ξ, η, ζ) is defined such that

$$\begin{aligned}
\xi &= \xi(x, y, z), \\
\eta &= \eta(x, y, z), \\
\zeta &= \zeta(x, y, z).
\end{aligned} \tag{2.10}$$

The details of this transformation can be found in [81]. The grid spacings on the uniform computational grid are $\Delta\xi = \Delta\eta = \Delta\zeta = 1$.

Applying the coordinate transformation, the Navier-Stokes equations can be rewritten as

$$\partial_t \hat{\mathbf{Q}} + \partial_\xi \hat{\mathbf{E}} + \partial_\eta \hat{\mathbf{F}} + \partial_\zeta \hat{\mathbf{G}} = \frac{1}{Re}\Big(\partial_\xi \hat{\mathbf{E}}_\mathbf{v} + \partial_\eta \hat{\mathbf{F}}_\mathbf{v} + \partial_\zeta \hat{\mathbf{G}}_\mathbf{v}\Big), \tag{2.11}$$

where

$$\begin{aligned}
&\hat{\mathbf{Q}} = J^{-1}\mathbf{Q},\\
&\hat{\mathbf{E}} = J^{-1}\Big(\xi_x\mathbf{E} + \xi_y\mathbf{F} + \xi_z\mathbf{G}\Big), && \hat{\mathbf{E}}_\mathbf{v} = J^{-1}\Big(\xi_x\mathbf{E}_\mathbf{v} + \xi_y\mathbf{F}_\mathbf{v} + \xi_z\mathbf{G}_\mathbf{v}\Big),\\
&\hat{\mathbf{F}} = J^{-1}\Big(\eta_x\mathbf{E} + \eta_y\mathbf{F} + \eta_z\mathbf{G}\Big), && \hat{\mathbf{F}}_\mathbf{v} = J^{-1}\Big(\eta_x\mathbf{E}_\mathbf{v} + \eta_y\mathbf{F}_\mathbf{v} + \eta_z\mathbf{G}_\mathbf{v}\Big),\\
&\hat{\mathbf{G}} = J^{-1}\Big(\zeta_x\mathbf{E} + \zeta_y\mathbf{F} + \zeta_z\mathbf{G}\Big), && \hat{\mathbf{G}}_\mathbf{v} = J^{-1}\Big(\zeta_x\mathbf{E}_\mathbf{v} + \zeta_y\mathbf{F}_\mathbf{v} + \zeta_z\mathbf{G}_\mathbf{v}\Big),
\end{aligned}$$

and J is the metric Jacobian of the coordinate transformation, defined as

$$J = \left(x_\xi y_\eta z_\zeta + y_\xi z_\eta x_\zeta + z_\xi x_\eta y_\zeta - x_\xi z_\eta y_\zeta - y_\xi x_\eta z_\zeta - z_\xi y_\eta x_\zeta\right)^{-1}. \tag{2.12}$$

The transformed inviscid flux vectors can be expanded to

$$\hat{\mathbf{E}} = \begin{bmatrix} \rho U \\ \rho u U + \xi_x p \\ \rho v U + \xi_y p \\ \rho w U + \xi_z p \\ (e+p)U \end{bmatrix}, \quad \hat{\mathbf{F}} = \begin{bmatrix} \rho V \\ \rho u V + \eta_x p \\ \rho v V + \eta_y p \\ \rho w V + \eta_z p \\ W(e+p) \end{bmatrix}, \quad \hat{\mathbf{G}} = \begin{bmatrix} \rho W \\ \rho u W + \zeta_x p \\ \rho v W + \zeta_y p \\ \rho w W + \zeta_z p \\ W(e+p) \end{bmatrix}, \tag{2.13}$$

where the contravariant velocities (U, V, W) are defined by

$$U = \xi_x u + \xi_y v + \xi_z w, \quad V = \eta_x u + \eta_y v + \eta_z w, \quad W = \zeta_x u + \zeta_y v + \zeta_z w. \tag{2.14}$$

The viscous stresses also undergo the coordinate transformation and result in the following expressions:

$$\begin{aligned}
&\tau_{xx} = \tfrac{4}{3}(\mu+\mu_t)(\xi_x u_\xi + \eta_x u_\eta + \zeta_x u_\zeta) - \tfrac{2}{3}(\mu+\mu_t)\left(\xi_y v_\xi + \eta_y v_\eta + \zeta_y v_\zeta + \xi_z w_\xi + \eta_z w_\eta + \zeta_z w_\zeta\right),\\
&\tau_{xy} = (\mu+\mu_t)\left(\xi_y u_\xi + \eta_y u_\eta + \zeta_y u_\zeta + \xi_x v_\xi + \eta_x v_\eta + \zeta_x v_\zeta\right),\\
&\tau_{xz} = (\mu+\mu_t)\left(\xi_z u_\xi + \eta_z u_\eta + \zeta_z u_\zeta + \xi_x w_\xi + \eta_x w_\eta + \zeta_x w_\zeta\right),\\
&\tau_{yy} = \tfrac{4}{3}(\mu+\mu_t)(\xi_y v_\xi + \eta_y v_\eta + \zeta_y v_\zeta) - \tfrac{2}{3}(\mu+\mu_t)\left(\xi_x u_\xi + \eta_x u_\eta + \zeta_x u_\zeta + \xi_z w_\xi + \eta_z w_\eta + \zeta_z w_\zeta\right),\\
&\tau_{yz} = (\mu+\mu_t)\left(\xi_z v_\xi + \eta_z v_\eta + \zeta_z v_\zeta + \xi_y w_\xi + \eta_y w_\eta + \zeta_y w_\zeta\right),\\
&\tau_{zz} = \tfrac{4}{3}(\mu+\mu_t)(\xi_z w_\xi + \eta_z w_\eta + \zeta_z w_\zeta) - \tfrac{2}{3}(\mu+\mu_t)\left(\xi_x u_\xi + \eta_x u_\eta + \zeta_x u_\zeta + \xi_y v_\xi + \eta_y v_\eta + \zeta_y v_\zeta\right),\\
&\tau_{yx} = \tau_{xy}, \quad \tau_{zx} = \tau_{xz}, \quad \tau_{zy} = \tau_{yz},\\
&E_{v,5} = u\tau_{xx} + v\tau_{xy} + w\tau_{xz} + (\mu Pr^{-1} + \mu_t Pr_t^{-1})(\gamma-1)^{-1}[\xi_x\partial_\xi(a^2) + \eta_x\partial_\eta(a^2) + \zeta_x\partial_\zeta(a^2)],\\
&F_{v,5} = u\tau_{yx} + v\tau_{yy} + w\tau_{yz} + (\mu Pr^{-1} + \mu_t Pr_t^{-1})(\gamma-1)^{-1}[\xi_y\partial_\xi(a^2) + \eta_y\partial_\eta(a^2) + \zeta_y\partial_\zeta(a^2)],\\
&G_{v,5} = u\tau_{zx} + v\tau_{zy} + w\tau_{zz} + (\mu Pr^{-1} + \mu_t Pr_t^{-1})(\gamma-1)^{-1}[\xi_z\partial_\xi(a^2) + \eta_z\partial_\eta(a^2) + \zeta_z\partial_\zeta(a^2)].
\end{aligned}$$

2.1.2 Boundary conditions

Farfield boundary

The outer boundary of the computational domain, referred to as the farfield boundary, dictates the free-stream flow conditions which the aerodynamic shape is being subjected to. It has been shown that specifying farfield boundary conditions for all flow variables leads to an over-determined system, necessitating the use of characteristic boundary conditions [40].

Within this work, characteristic-type boundary conditions are applied based on the direction of flow and whether or not the flow is locally supersonic. This determines which of the flow quantities are taken from the interior of the computational domain, and which are obtained from the free-stream conditions. Based on the location and direction of flow, different farfield quantities may be required. The complete farfield solution vector, $\mathbf{Q}_{\text{ff}}$, is defined in dimensionless quantities as

$$\mathbf{Q}_{\text{ff}} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ e \end{bmatrix}_{\text{ff}} = \begin{bmatrix} 1 \\ M_\infty \cos(\alpha) \\ 0 \\ M_\infty \sin(\alpha) \\ \frac{1}{\gamma(\gamma-1)} + \frac{1}{2} M_\infty^2 \end{bmatrix}, \tag{2.15}$$

with the assumption that x is the streamwise and z the normal direction. Additionally, M_∞ is the free-stream Mach number and α is the angle of attack.

The use of characteristic boundary conditions breaks down in the area of the viscous wake. Pueyo and Zingg [80] have demonstrated that a zeroth-order extrapolation of the flow variables is sufficient for closure of the equations in this region. This treatment has been observed to prevent reflection of conservative flow quantities in the outflow region of the farfield boundary. Details of where the outflow boundary is specified can be found in §3.1.1.

Solid surface (wall) boundary

The enforcement of the boundary conditions for a solid surface makes use of both the inviscid slip-wall conditions and the viscous no-slip conditions. This approach allows the same basic algorithm to deal with both inviscid and viscous flows without extensive modification or tuning required for each class of flow.

With the inviscid slip-wall conditions, the velocity at the surface is required to be tangential, resulting in a normal velocity, V_n, of zero. No other conditions are imposed on the flow in this case.

The viscous no-slip condition imposes a further restriction on the velocity. Here, the surface velocity is required to be zero, resulting in $u = 0$, $v = 0$, and $w = 0$ along the entire surface boundary. It should be noted that the use of both the slip-wall and no-slip conditions simultaneously is possible, since the viscous no-slip condition is effectively a subset of the inviscid slip-wall condition. The details of the implementation, and the reason for the use of both boundary conditions, will be discussed in Chapter 3.

For laminar and turbulent flows, an additional solid surface boundary condition is required. To this end, the adiabatic condition is also assumed on the surface boundary. This requires the gradient of the temperature normal to a surface boundary to be zero,

$$\frac{\partial T}{\partial n} = 0. \tag{2.16}$$

While for the purpose of this thesis, the above boundary condition is used, the framework implemented is also readily applicable to the case of an isothermal boundary condition, where a wall temperature, T_w, is specified.

Symmetry boundary

In order to reduce the size of the computational domain needed to obtain a realistic representation of the flow, certain physical features of the domain can be exploited. In particular, the midline of an airplane can serve as a natural split where an assumption of symmetric flow can be made as long as there is no side-slip velocity component. At any such boundary, all flow variables possess a zero gradient normal to the boundary. In addition, velocity is assumed to be fully tangential, as with the slip-wall condition used for the inviscid solid surface.

2.1.3 Initial conditions

The mean-flow quantities in Q are initialized to the farfield boundary values, as defined in (2.15).

2.2 Turbulence Model

In order to solve turbulent flows, a turbulent, or eddy, viscosity, μ_t, can be added to the molecular viscosity, μ. The eddy viscosity is normalized as $\tilde{\mu}_t/\tilde{\mu}_\infty$. The Spalart-Allmaras one-equation turbulence model [91] is used to compute the turbulent viscosity. This model has been applied extensively in various CFD codes, and has been shown to provide good

approximations of the behaviour of turbulence in external aerodynamic flows [103], for which it has been calibrated.

The model solves a transport equation for a turbulence-like variable, $\tilde{\nu}$, that is related to the turbulent viscosity term required for the Navier-Stokes equations. The model itself is a sixth equation that is solved concurrently with the five Navier-Stokes equations.

The non-dimensional version of the model used in this work is given by

$$\begin{aligned}\frac{\partial\tilde{\nu}}{\partial t}+u_i\frac{\partial\tilde{\nu}}{\partial x_i}=&\frac{c_{b1}}{Re}[1-f_{t2}]\tilde{S}\tilde{\nu}+\frac{1+c_{b2}}{\sigma Re}\nabla\cdot[(\nu+\tilde{\nu})\nabla\tilde{\nu}]-\frac{c_{b2}}{\sigma Re}(\nu+\tilde{\nu})\nabla^2\tilde{\nu}\\&-\frac{1}{Re}\left[c_{w1}f_w-\frac{c_{b1}}{\kappa^2}f_{t2}\right]\left(\frac{\tilde{\nu}}{d}\right)^2+Ref_{t1}\Delta U^2,\end{aligned}\tag{2.17}$$

where $\nu=\frac{\mu}{\rho}$. Non-dimensional values of $\tilde{\nu}$ and $\tilde{S}$ were obtained by normalizing with ${}^{\tilde{\mu}_\infty}/_{\tilde{\rho}_\infty}$ and ${}^{\tilde{\mu}_\infty}/_{\tilde{\rho}_\infty c^2}$, respectively. The spatial derivatives on the left side of the equation represent advection. The first term on the right side represents production, while the second and third terms account for diffusion. The fourth term represents destruction. The final term allows for the specification of an explicit laminar-to-turbulent transition location, but it should be stressed that the model cannot predict this location; it has to be specified either by the user or some other means, such as the e^N method [100].

The eddy viscosity, $\nu_t=\frac{\mu_t}{\rho}$, can be obtained from $\tilde{\nu}$ by using

$$\nu_t=f_{v1}\tilde{\nu},\tag{2.18}$$

where

$$f_{v1}=\frac{\chi^3}{\chi^3+c_{v1}^3},\tag{2.19}$$

and

$$\chi=\frac{\tilde{\nu}}{\nu}.\tag{2.20}$$

The modified vorticity value, $\tilde{S}$, is given by

$$\tilde{S}=ReS+\frac{\tilde{\nu}}{\kappa^2d^2}f_{v2},\tag{2.21}$$

where S is the magnitude of the vorticity, nondimensionalized with ${}^{\tilde{a}_\infty}/_{c}$:

$$S=\left[\left(\frac{\partial w}{\partial y}-\frac{\partial v}{\partial z}\right)^2+\left(\frac{\partial u}{\partial z}-\frac{\partial w}{\partial x}\right)^2+\left(\frac{\partial v}{\partial x}-\frac{\partial u}{\partial y}\right)^2\right]^{-\frac{1}{2}},\tag{2.22}$$

d is the distance to the closest solid surface, and

$$f_{v2}=1-\frac{\chi}{1+\chi f_{v1}}.\tag{2.23}$$

Within the destruction term, f_w is given by

$$f_w = g\left[\frac{1+c_{w3}^6}{g^6+c_{w3}^6}\right]^{\frac{1}{6}}, \tag{2.24}$$

where

$$g = r + c_{w2}\left(r^6 - r\right), \tag{2.25}$$

and

$$r = \min\left(\frac{\tilde{\nu}}{\tilde{S}\kappa^2 d^2}, 10\right). \tag{2.26}$$

To enforce laminar-to-turbulent transition in a specific region of the flow, the model also contains two trip terms, the first ensuring that a higher level of turbulence is found in the vicinity of the trip location, and the second bringing small values of $\tilde{\nu}$ towards zero. The terms are defined as

$$f_{t1} = c_{t1} g_t \exp\left[-c_{t2}\frac{\omega_t^2}{\Delta U^2}\left(d^2 + g_t^2 d_t^2\right)\right], \tag{2.27}$$

$$f_{t2} = c_{t3}\exp\left(-c_{t4}\chi^2\right). \tag{2.28}$$

To ensure that the trip terms are active over at least a few nodes, the factor g_t is defined as

$$g_t = \min\left(0.1, \frac{\Delta U}{\omega_t \Delta x_t}\right), \tag{2.29}$$

where ω_t is the vorticity at the trip point, Δx_t is the grid spacing at the trip point, and ΔU is the velocity difference between the point at which the trip function is being calculated and the trip point. Additionally, d_t is the distance to the trip point.

The remaining constants are given by

$$\begin{aligned}
c_{b1} &= 0.1355, & c_{b2} &= 0.622,\\
\sigma &= 2/3, & c_{v1} &= 7.1,\\
c_{w2} &= 0.3, & \kappa &= 0.41,\\
c_{w3} &= 2.0, & c_{w1} &= \frac{c_{b1}}{\kappa^2} + \frac{1}{\sigma}(1+c_{b2}),\\
c_{t1} &= 5.0, & c_{t2} &= 2.0,\\
c_{t3} &= 1.2, & c_{t4} &= 0.5.
\end{aligned}$$

The default configuration of the turbulence model does not enforce a specific transition point, but rather assumes a fully turbulent flow. This is achieved by setting f_{t1} to zero.

To prevent numerical instabilities, precautions are taken to ensure that neither the vorticity, S, nor the modified vorticity, $\tilde{S}$, approaches zero or becomes negative. In particular,

$\tilde{S}$ is modified according to

$$\tilde{S} = \begin{cases} \tilde{S} & : \quad \tilde{S} > 0.3S \\ 0.3S & : \quad \text{otherwise} \end{cases} . \tag{2.30}$$

Appendix F presents a discussion of an alternate approach to enforcing transition locations through the use of an intermittency function. Unlike the f_{t1} source term approach, the intermittency function has the capability of enforcing a specific transition region length, an important feature if transition prediction is to be considered.

2.2.1 "Negative" turbulence model formulation

Even though the standard turbulence model formulation should not possess any negative values of $\tilde{\nu}$ once a converged flow solution is obtained, the presence of transient states during the solution process may introduce these nonphysical values into the flow. Negative turbulence quantities can lead to unwanted behaviour in the source terms for production and destruction, often resulting in numerical instabilities and algorithm divergence. In order to stabilize the solution, the default behaviour is to trim any negative values of $\tilde{\nu}$ during a solution update, as will be presented in (4.16).

Allmaras *et al.* [3] have proposed a modification to the original turbulence model that accepts negative values of $\tilde{\nu}$ as an alternate means of dealing with the problematic transient states. Their work was also motivated by instances where under-resolved meshes and higher-order spatial discretization techniques could lead to negative values of $\tilde{\nu}$ in the converged steady-state flow solutions [9, 73]. In particular, this approach supersedes the use of an additional PDE for artificial dissipation applied to the turbulence model [67], which has been used to deal with the same turbulence quantity behaviour.

In their "negative" Spalart-Allmaras model*, the modified vorticity, $\tilde{S}$, is defined as

$$\tilde{S} = \begin{cases} Re\left(S + \bar{S}\right) & : \quad \bar{S} \geq -c_{v2}S \\ Re\left(S + \frac{S(c_{v2}^2 S + c_{v3}\bar{S})}{(c_{v3} - 2c_{v2})S - \bar{S}}\right) & : \quad \bar{S} < -c_{v2}S \end{cases} , \tag{2.31}$$

where

$$\bar{S} = \frac{\tilde{\nu}}{\kappa^2 d^2} f_{v2}, \tag{2.32}$$

$c_{v2} = 0.7$, and $c_{v3} = 0.9$. Additionally,

$$\text{production} = \begin{cases} \frac{c_{b1}}{Re}\left(1 - f_{t2}\right)\tilde{S}\tilde{\nu} & : \quad \tilde{\nu} \geq 0 \\ c_{b1}\left(1 - c_{t3}\right)S\tilde{\nu} & : \quad \tilde{\nu} < 0 \end{cases} , \tag{2.33}$$

*All expressions are shown here in dimensionless form.

$$\text{destruction} = \begin{cases} -\frac{1}{Re}\left[c_{w1}f_w - \frac{c_{b1}}{\kappa^2}f_{t2}\right]\left(\frac{\tilde{\nu}}{d}\right)^2 & : \quad \tilde{\nu} \geq 0 \\ \frac{1}{Re}c_{w1}\left(\frac{\tilde{\nu}}{d}\right)^2 & : \quad \tilde{\nu} < 0 \end{cases}, \tag{2.34}$$

$$\text{diffusion} = \begin{cases} \frac{1+c_{b2}}{\sigma Re}\nabla\cdot[(\nu+\tilde{\nu})\nabla\tilde{\nu}] - \frac{c_{b2}}{\sigma Re}(\nu+\tilde{\nu})\nabla^2\tilde{\nu} & : \quad \tilde{\nu} \geq 0 \\ \frac{1}{\sigma Re}\nabla\cdot[(\nu+f_n\tilde{\nu})\nabla\tilde{\nu}] + \frac{c_{b2}}{\sigma Re}\left\{\nabla\cdot(\nu+\tilde{\nu})\nabla\tilde{\nu} - (\nu+\tilde{\nu})\nabla^2\tilde{\nu}\right\} & : \quad \tilde{\nu} < 0 \end{cases}, \tag{2.35}$$

where

$$f_n = \frac{c_{n1}+\chi^3}{c_{n1}-\chi^3},$$

and $c_{n1} = 16$.

The expressions in (2.33), (2.34), and (2.35) are presented in the form that would be added to the right hand side of (2.17). To reiterate the purpose of the modifications, the standard turbulence model of (2.17) remains valid for regions of the flow where $\tilde{\nu} \geq 0$, while the modifications to the diffusion, production, and destruction terms take effect in areas of negative $\tilde{\nu}$.

Finally, to account for the presence of negative values of $\tilde{\nu}$, which would result in a nonphysical eddy viscosity if calculated based on (2.18), any negative $\tilde{\nu}$ will result in zero eddy viscosity, μ_t.

The effectiveness of the "negative" model variant will be compared to that of the standard turbulence model for a variety of flow conditions. In particular, it is important to determine whether the model variant significantly impacts the convergence of the overall solution algorithm for cases where excessive negative turbulence quantities occur during iteration to a steady-state solution.

2.2.2 Curvilinear coordinate transformation

The coordinate transformation has to be performed on all terms in the turbulence model that possess spatial derivatives. This includes the advective and diffusive terms, as well as the vorticity, S.

The advective terms, in Cartesian coordinates, are

$$u_i\frac{\partial\tilde{\nu}}{\partial x_i} = u\frac{\partial\tilde{\nu}}{\partial x} + v\frac{\partial\tilde{\nu}}{\partial y} + w\frac{\partial\tilde{\nu}}{\partial z}, \tag{2.36}$$

where (u, v, w) are the three-dimensional velocity components. Performing the coordinate transformation, the advective terms become

$$\begin{aligned} U_i\frac{\partial\tilde{\nu}}{\partial\xi_i} &= (u\xi_x + v\xi_y + w\xi_z)\frac{\partial\tilde{\nu}}{\partial\xi} + (u\eta_x + v\eta_y + w\eta_z)\frac{\partial\tilde{\nu}}{\partial\eta} + (u\zeta_x + v\zeta_y + w\zeta_z)\frac{\partial\tilde{\nu}}{\partial\zeta} \\ &= U\frac{\partial\tilde{\nu}}{\partial\xi} + V\frac{\partial\tilde{\nu}}{\partial\eta} + W\frac{\partial\tilde{\nu}}{\partial\zeta}, \end{aligned} \tag{2.37}$$

where (U, V, W) are the contravariant velocities.

Due to the relative complexity of the diffusive terms, only the final results of the coordinate transformation will be presented. It should be noted, however, that the cross-derivative terms are dropped during the coordinate transformation.

The model in (2.17) contains two separate diffusive terms, each of which possesses a slightly different form. The first term is transformed into curvilinear coordinates as

$$\begin{aligned}\nabla \cdot [(\nu + \tilde{\nu})\nabla\tilde{\nu}] &= \partial_x \left[(\nu + \tilde{\nu})\partial_x\tilde{\nu}\right] + \partial_y \left[(\nu + \tilde{\nu})\partial_y\tilde{\nu}\right] + \partial_z \left[(\nu + \tilde{\nu})\partial_z\tilde{\nu}\right] \\ &\approx \xi_x\partial_\xi[(\nu + \tilde{\nu})\xi_x\partial_\xi\tilde{\nu}] + \xi_y\partial_\xi[(\nu + \tilde{\nu})\xi_y\partial_\xi\tilde{\nu}] + \xi_z\partial_\xi[(\nu + \tilde{\nu})\xi_z\partial_\xi\tilde{\nu}] + \\ &\quad \eta_x\partial_\eta[(\nu + \tilde{\nu})\eta_x\partial_\eta\tilde{\nu}] + \eta_y\partial_\eta[(\nu + \tilde{\nu})\eta_y\partial_\eta\tilde{\nu}] + \eta_z\partial_\eta[(\nu + \tilde{\nu})\eta_z\partial_\eta\tilde{\nu}] + \\ &\quad \zeta_x\partial_\zeta[(\nu + \tilde{\nu})\zeta_x\partial_\zeta\tilde{\nu}] + \zeta_y\partial_\zeta[(\nu + \tilde{\nu})\zeta_y\partial_\zeta\tilde{\nu}] + \zeta_z\partial_\zeta[(\nu + \tilde{\nu})\zeta_z\partial_\zeta\tilde{\nu}].\end{aligned} \tag{2.38}$$

Similarly, the second term can be transformed as

$$\begin{aligned}\nabla^2\tilde{\nu} &= \frac{\partial^2\tilde{\nu}}{\partial x^2} + \frac{\partial^2\tilde{\nu}}{\partial y^2} + \frac{\partial^2\tilde{\nu}}{\partial z^2} \\ &\approx \xi_x\partial_\xi[\xi_x\partial_\xi\tilde{\nu}] + \xi_y\partial_\xi[\xi_y\partial_\xi\tilde{\nu}] + \xi_z\partial_\xi[\xi_z\partial_\xi\tilde{\nu}] + \\ &\quad \eta_x\partial_\eta[\eta_x\partial_\eta\tilde{\nu}] + \eta_y\partial_\eta[\eta_y\partial_\eta\tilde{\nu}] + \eta_z\partial_\eta[\eta_z\partial_\eta\tilde{\nu}] + \\ &\quad \zeta_x\partial_\zeta[\zeta_x\partial_\zeta\tilde{\nu}] + \zeta_y\partial_\zeta[\zeta_y\partial_\zeta\tilde{\nu}] + \zeta_z\partial_\zeta[\zeta_z\partial_\zeta\tilde{\nu}].\end{aligned} \tag{2.39}$$

The vorticity term, S, which contains spatial derivatives of the three velocity components, was introduced in (2.22). After performing the coordinate transformation, the vorticity becomes

$$S = \left(S_1^2 + S_2^2 + S_3^2\right)^{-\frac{1}{2}}, \tag{2.40}$$

where

$$\begin{aligned}S_1 &= (\xi_y w_\xi + \eta_y w_\eta + \zeta_y w_\zeta) - (\xi_z v_\xi + \eta_z v_\eta + \zeta_z v_\zeta), \\ S_2 &= (\xi_z u_\xi + \eta_z u_\eta + \zeta_z u_\zeta) - (\xi_x w_\xi + \eta_x w_\eta + \zeta_x w_\zeta), \\ S_3 &= (\xi_x v_\xi + \eta_x v_\eta + \zeta_x v_\zeta) - (\xi_y u_\xi + \eta_y u_\eta + \zeta_y u_\zeta).\end{aligned} \tag{2.41}$$

2.2.3 Boundary conditions

Farfield boundary

The farfield, or free-stream, value of $\tilde{\nu}$ is dependent on the type of flow being solved. If fully turbulent flow is assumed, the farfield boundary value is set to 3.0, following the work of Spalart and Rumsey [92]. For flows with explicit trip location definitions, the farfield boundary value is set to 0.1 in order to prevent laminar-to-turbulent transition in areas upstream of the specified trip locations. The value for tripped flows is at the high end of the recommended range, but has resulted in more robust algorithm performance.

Solid surface (wall) boundary

The value of $\tilde{\nu}$ on any solid surface boundary is required to be zero, since no turbulence can be sustained directly on a solid surface.

Symmetry boundary

As with the Navier-Stokes equations, the symmetry boundary requires the gradient of $\tilde{\nu}$ to be zero normal to the boundary, specified as

$$\frac{\partial \tilde{\nu}}{\partial n} = 0. \tag{2.42}$$

2.2.4 Initial conditions

Fully turbulent flow

For fully turbulent flow, the initial value of $\tilde{\nu}$ is set to 3.0, which corresponds to the farfield boundary condition. This ensures that the computational domain is seeded with a high enough turbulence value to ensure the development of a turbulent boundary layer at the leading edge of the aerodynamic shape, without introducing large turbulence values to regions of flow where little to no turbulence can be sustained.

Explicitly tripped flow

For flow where an explicit laminar-to-turbulent transition region is specified, the initial conditions are set according to the location of the individual nodes with respect to the transition region. Any nodes upstream of the transition region take on the farfield value of $\tilde{\nu}$ used for tripped flows, 0.1, while any nodes downstream of the transition region are initialized with $\tilde{\nu} = 10$. This procedure ensures that a turbulent boundary layer forms downstream of the transition region and provides a better estimate of the initial flowfield for transitional flow than initialization to a single value would, greatly improving the convergence characteristics of the algorithm in the early stages of the flow solution.

It is important to make sure that nodes upstream of the transition region possess very low turbulence. Initializing these nodes with higher $\tilde{\nu}$ values could lead to the turbulence model self-tripping upstream of the required trip location. If this occurs, it is not uncommon for the converged flow solution to completely ignore the specified trip location and instead transition from laminar to turbulent flow upstream of the trip region definition.

This technique is used with both transition enforcement methods, using either the f_{t1} term or the S-curve intermittency function.

Chapter 3

Spatial Discretization

The spatial discretization of the Navier-Stokes equations and the turbulence model is obtained by the use of Summation-By-Parts (SBP) operators, while inter-block coupling and boundary conditions are enforced by the use of Simultaneous Approximation Terms (SATs). This type of discretization gives rise to multi-block schemes that possess less communication overhead than typical schemes using halo nodes, as no derivatives are formed across block interfaces. Furthermore, since only C^0 continuity between blocks at interfaces is required, the relaxed geometric requirements make grid generation and domain decomposition substantially easier.

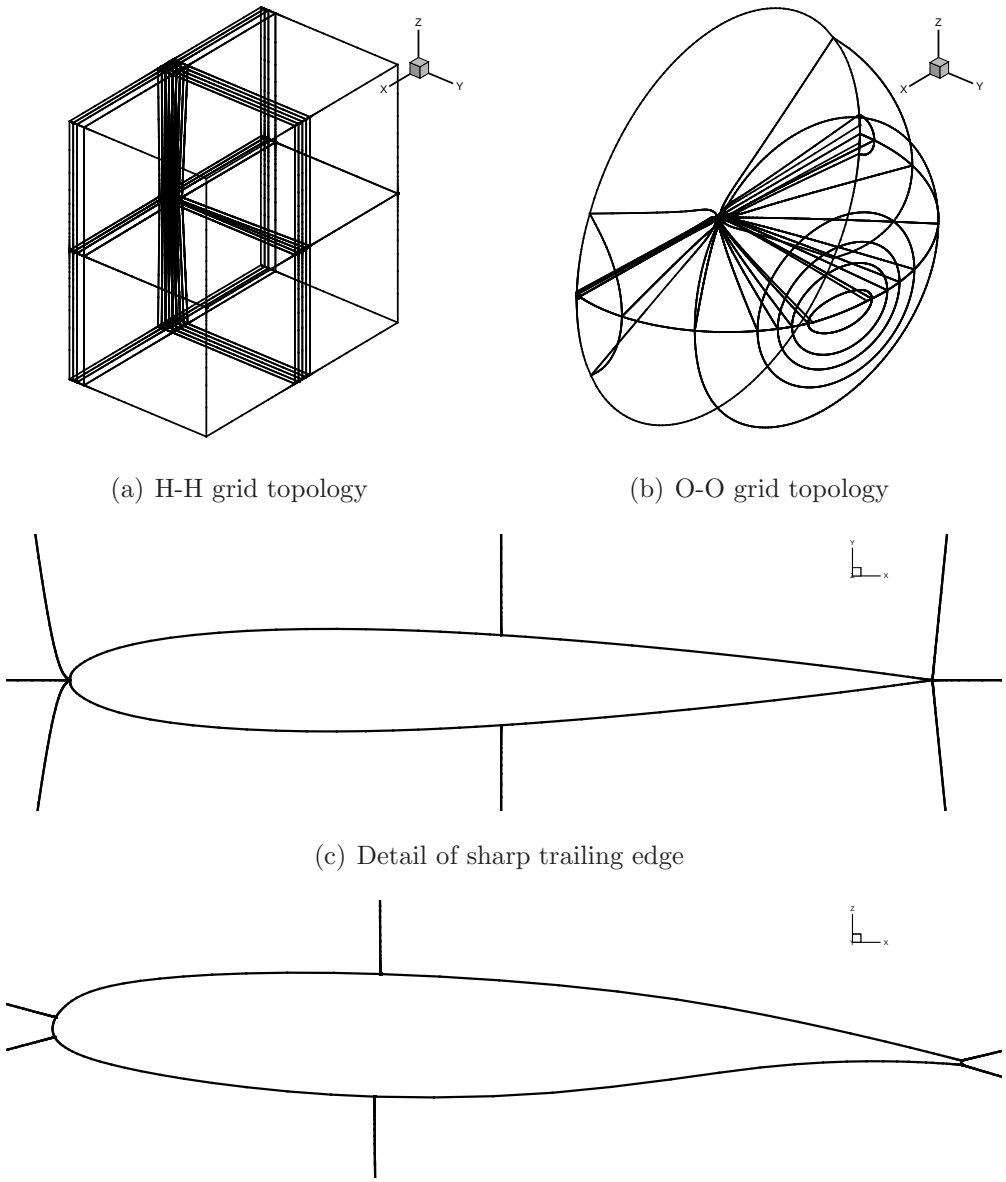
This chapter presents the SBP operators for first and second derivatives and their application to the governing equations, as well as the various SATs required. The focus of this work is to present the implementation of the discretization for the viscous terms and turbulence model, with the details of the Euler equation implementation presented previously by Hicken and Zingg [37]. Additional details, including the theory behind the development of the SBP-SAT approach, can be found in Refs. 37, 76, 77, 95, 96, 72, 49, and 94.

Numerical dissipation is added using either the scalar dissipation model developed by Jameson *et al.* [41] and later refined by Pulliam [81], or the matrix dissipation model of Swanson and Turkel [97]. With the current second-order spatial discretization (with the exception of the convective terms in the turbulence model), the numerical dissipation consists of second- and fourth-difference dissipation operators, whose magnitudes are controlled by the κ_2 and κ_4 coefficients, respectively.

Grid metrics, which result from the coordinate transformation, are computed using the second-order SBP operator for a first derivative.

3.1 Domain Decomposition

The current strategy decomposes the computational domains into multiple blocks, resulting in multi-block structured grids. Not only does this type of blocking strategy work well with the SBP-SAT discretization, it also allows for relatively straightforward creation of meshes



(a) H-H grid topology

(b) O-O grid topology

(c) Detail of sharp trailing edge

(d) Detail of blunt trailing edge

Figure 3.1: Multi-block grid topology examples

around complex geometries. This blocking is conducive to localization of high grid density in certain areas without affecting the grid resolution elsewhere, which can be especially effective when resolving turbulent boundary layers or shocks. Finally, multi-block meshes can effectively make use of a number of different grid topologies, making them suitable for dealing with geometries with either sharp or blunt trailing edges. Figure 3.1 presents examples of multi-block grids for various geometries, highlighting both overall topology choices as well as details of block orientations in select areas near wing and wing-body surfaces.

The algorithm has been created to accept arbitrary orientations and connectivities of

blocks within the domain, such that there is no extra burden on the grid generation process. As long as the algorithm is provided with a connectivity file that describes how the blocks are oriented and attached to each other, the governing equations will be discretized for the computational domain without additional input from the user. An example of the flexibility of the current algorithm would be a situation where a single face of a block is attached to two adjoining blocks. This becomes useful for load balancing of the parallel algorithm, as it allows for select blocks within the computational domain to be split without the need to propagate the split through the remainder of the domain. The method of enforcement of boundary conditions, discussed in §3.3, also allows for subsections of faces to enforce different boundary conditions. For example, if simulating a flat plate with a single rectangular block. a section of the bottom block face can be specified as a no-slip wall boundary, while a small section upstream of this can be specified as a symmetry plane, as is often done in flat plate simulations. At the present time, the only requirement is that the block faces are node-matched, but future work may remove even this requirement, opening the possibility for adaptive mesh refinement in a block-structured grid setting.

3.1.1 Farfield outflow boundary

The wake region of viscous flows is not suited for use with the typical farfield boundary conditions, such as the characteristic approach described in §2.1.2, and a zeroth-order extrapolation of the flow variables can be performed instead. However, for this approach to work, it is necessary to specify the region of effect for the outflow boundary. This becomes especially acute for complex three-dimensional grid topologies, for which it is not possible to simply specify an entire block boundary as an outflow region. In addition, specifying individual boundaries ignores the possibility of using substantially different flow directions, as specified through the angle of attack, with the same grids. The approach presented here is well suited for external flows.

The approach taken here is to create a "pyramid" emanating from the aerodynamic body in the direction of free-stream flow, with a spread of approximately 14°. The shape is extended to the downstream farfield boundary, and any nodes that lie within it are classified as outflow boundary nodes, for which the zeroth-order extrapolation will be performed during a flow solution. Figure 3.2 shows an example of the location of the outflow boundary for an H-H topology grid running a flow solution at an angle of attack, α, of 3°. Any nodes located within the blue square are classified as outflow boundary nodes.

One of the advantages of this approach is that it requires no input from the user. It can also accommodate any grid topologies by relying on the free-stream flow characteristics,

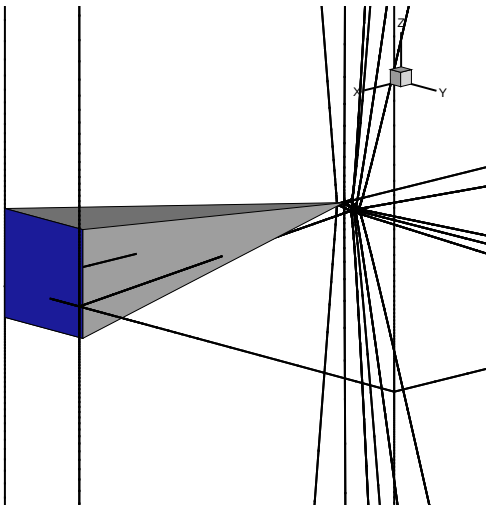


Figure 3.2: Outflow boundary specification (in blue) on H-H topology grid at $\alpha = 3°$

namely α, to determine the outflow region. Finally, the use of the "pyramid" geometry allows the algorithm to automatically account for varying distances to the farfield boundary; farfield boundaries located further downstream of the geometry will be assigned a larger outflow region to account for a spreading wake. Since the determination of the outflow region only needs to be performed once for each flow solution, the calculation adds minimal computational overhead. Two-dimensional grids are accommodated by collapsing the outflow region onto the two-dimensional plane of the grid.

All flow solutions presented in this thesis incorporate the above outflow region specification approach, demonstrating the flexibility of the method in dealing with somewhat arbitrary grid topologies.

3.2 Summation-by-Parts Operators

SBP operators allow for the construction of finite-difference approximations to derivatives. When dealing with the governing equations considered in this work, approximations to both the first and second derivatives are required.

The SBP operators for the first derivative were originally derived by Kreiss and Scherer [49], subsequently extended by Strand [94], and applied by various authors (see [21, 22, 37, 59, 72]). SBP operators are centered difference schemes that do not include boundary conditions; in our case these are enforced using SATs. They are constructed so that the discrete energy-method can be used to make time stability statements about a discretization and have been shown to be time-stable for the linearized Navier-Stokes equations [59]. For curvilinear coordinates, however, time-stability can only be guaranteed for SBP operators

constructed with a diagonal norm matrix. This type of operator is considered for this work.

3.2.1 SBP operator for first derivative

This section briefly presents the second-order SBP operator for the first derivative. A globally second-order accurate operator for a first derivative is given by

$$D_1 = H^{-1}\Theta, \tag{3.1}$$

where

$$H = h\begin{bmatrix} \frac{1}{2} & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ & & & & \frac{1}{2} \end{bmatrix}, \quad \Theta = \frac{1}{2}\begin{bmatrix} -1 & 1 & & & \\ -1 & 0 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 0 & 1 \\ & & & -1 & 1 \end{bmatrix},$$

and h takes on the value of the spatial difference in the pertinent coordinate direction, either $\Delta\xi$, $\Delta\eta$, or $\Delta\zeta$. In the context of the uniform computational grid, h has a value of 1 for all three coordinate directions. H is the diagonal norm matrix mentioned previously. The second-order operator takes on the form of a centered difference approximation on the interior of a block, with one-sided first-order treatment at block boundaries.

Application to Navier-Stokes equations

The D_1 operator is used to obtain a finite difference approximation of the inviscid fluxes for the entire computational domain. For example, the inviscid flux in the ξ-direction can be taken as

$$\partial_\xi \hat{\mathbf{E}} \approx D_{1\xi}\hat{\mathbf{E}}. \tag{3.2}$$

With the viscous component of the Navier-Stokes equations, the D_1 operator is used in the discretization of the cross-derivative terms, which have the form*

$$\partial_\xi(\beta\partial_\eta\alpha), \tag{3.3}$$

where β is a spatially varying coefficient, and α can be a flow quantity such as the x-component of velocity, u. Using (3.1), the cross-derivative can be approximated as

$$D_{1\xi}\boldsymbol{\beta}D_{1\eta}\boldsymbol{\alpha}, \tag{3.4}$$

*Within this section, the term α does not refer to the angle of attack, but rather an arbitrary flow quantity within a spatial derivative.

resulting in the following interior discretization (at node (j,k,m)):

$$\frac{1}{2}\beta_{j+1,k,m}\left(\frac{\alpha_{j+1,k+1,m}-\alpha_{j+1,k-1,m}}{2}\right)-\frac{1}{2}\beta_{j-1,k,m}\left(\frac{\alpha_{j-1,k+1,m}-\alpha_{j-1,k-1,m}}{2}\right). \tag{3.5}$$

An analogous approach is used for terms where cross-derivatives in any two coordinate directions are required.

Application to Spalart-Allmaras turbulence model

The advective terms that appear in the turbulence model consist of first derivatives of the turbulence variable, $\tilde{\nu}$, multiplied by velocities. An example of this is the term associated with the spatial derivative in the ξ-direction, given by

$$U\partial_\xi\tilde{\nu}, \tag{3.6}$$

where U is a contravariant velocity.

The authors of the model suggest the use of an upwinding strategy when discretizing this term, which is one of the approaches taken here. In the context of SBP operators, the connection between upwinding and artificial dissipation can be leveraged, namely that an upwinded operator can be expressed as a centered difference operator added to a dissipation operator. The derivative can be taken as

$$U\partial_\xi\tilde{\nu}\approx\mathbf{U}D_1\tilde{\boldsymbol{\nu}}+\frac{1}{2}|\mathbf{U}|H^{-1}D_d^TC_1D_d\tilde{\boldsymbol{\nu}} \tag{3.7}$$

where $\tilde{\boldsymbol{\nu}}$ represents a vector containing the turbulence quantity in the domain, and

$$\mathbf{U}=\operatorname{diag}\left(U_1,U_2,...,U_N\right),\quad |\mathbf{U}|=\operatorname{diag}\left(|U_1|,|U_2|,...,|U_N|\right),$$

$$D_d=\begin{bmatrix}-1 & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & -1 & 1\\ & & & & 1\end{bmatrix},\quad C_1=\begin{bmatrix}1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ & & & & 0\end{bmatrix}.$$

The above SBP discretization provides a clear approach to dealing with block boundaries. For completeness, the following shows the resulting discretization in different parts of the domain:

$$\begin{aligned}&\text{low side:} && \left(U_1-|U_1|\right)\left(\tilde{\nu}_2-\tilde{\nu}_1\right),\\ &\text{interior:} && \tfrac{1}{2}U_j\left(\tilde{\nu}_{j+1}-\tilde{\nu}_{j-1}\right)-\tfrac{1}{2}|U_j|\left(\tilde{\nu}_{j+1}-2\tilde{\nu}_j+\tilde{\nu}_{j-1}\right),\\ &\text{high side:} && \left(U_N+|U_N|\right)\left(\tilde{\nu}_N-\tilde{\nu}_{N-1}\right).\end{aligned}$$

The first derivative operator of (3.1) is also used in the discretization of the vorticity term, S.

A drawback of the current first-order upwinding approach is the limit it imposes on the order of accuracy of the discretization. While the effect is minimal for the second-order discretization used in the present work, future use of the current algorithm with higher-order discretizations requires an approach that is easily extensible to higher-order schemes. Hence, the use of a fourth-difference dissipation model has also been investigated, based on the scalar dissipation model used for the mean-flow equations. The discretization of the advective terms retains the second-order centered difference approximation of (3.7), removing the upwinding approach, instead applying the fourth-difference dissipation model. Using the formulation presented by Hicken [34], the discretization then becomes

$$U\partial_\xi \tilde{\nu} \approx \mathbf{U} D_1 \tilde{\boldsymbol{\nu}} + H^{-1} D_d^T \left(C_{\xi,U}^{(2)} D_d + C_{\xi,U}^{(4)} D_d C_2 D_d^T D_d \right) \tilde{\boldsymbol{\nu}}, \tag{3.8}$$

where

$$C_{\xi,U}^{(2)} = \begin{bmatrix} \kappa_{2t}|U|_{\frac{3}{2}}\varepsilon_{\frac{3}{2}}^{(\xi)} & & & & \\ & \kappa_{2t}|U|_{\frac{5}{2}}\varepsilon_{\frac{5}{2}}^{(\xi)} & & & \\ & & \ddots & & \\ & & & \kappa_{2t}|U|_{N-\frac{1}{2}}\varepsilon_{N-\frac{1}{2}}^{(\xi)} & \\ & & & & 0 \end{bmatrix},$$

$$C_{\xi,U}^{(4)} = \begin{bmatrix} \hat{\kappa}_{4t}|U|_{\frac{3}{2}} & & & & \\ & \hat{\kappa}_{4t}|U|_{\frac{5}{2}} & & & \\ & & \ddots & & \\ & & & \hat{\kappa}_{4t}|U|_{N-\frac{1}{2}} & \\ & & & & 0 \end{bmatrix}, \quad C_2 = \begin{bmatrix} 0 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ & & & & 0 \end{bmatrix},$$

and $|U|_{i+\frac{1}{2}} = \frac{1}{2}\left(|U|_i + |U|_{i+1}\right)$. $\varepsilon_{i+\frac{1}{2}}^{(\xi)}$ represents a pressure switch-like term that governs the inclusion of second-difference dissipation in areas of rapid change in the turbulence quantity. It is modeled after the pressure switch term used for shock capturing in the Navier-Stokes equations, and is defined by the following expressions:

$$\begin{aligned} \varepsilon_{i+\frac{1}{2}}^{(\xi)} &= \varepsilon_i^{(\xi)} + \varepsilon_{i+1}^{(\xi)}, \\ \varepsilon_i^{(\xi)} &= \tfrac{1}{4}\left(\Upsilon_{i-1}^* + 2\Upsilon_i^* + \Upsilon_{i+1}^*\right), \\ \Upsilon_i^* &= \max\left[\Upsilon_{i-1}, \Upsilon_i, \Upsilon_{i+1}\right], \\ \Upsilon_i &= \begin{cases} \frac{|\tilde{\nu}_{i-1} - 2\tilde{\nu}_i + \tilde{\nu}_{i+1}|}{|\tilde{\nu}_{i-1} + 2\tilde{\nu}_i + \tilde{\nu}_{i+1}|} & : \ |\tilde{\nu}_{i-1} + 2\tilde{\nu}_i + \tilde{\nu}_{i+1}| > 0 \\ 0 & : \ \text{otherwise} \end{cases}. \end{aligned}$$

Finally, the $\hat{\kappa}_{4t}$ coefficient is obtained from

$$\hat{\kappa}_{4t} = \max\left[0, \kappa_{4t} - \kappa_{2t}\varepsilon^{(\xi)}_{i+\frac{1}{2}}\right].$$

Analogous formulations exist for the η and ζ computational directions. The values for the dissipation coefficients, κ_{2t} and κ_{4t}, will be discussed in §5.2.2.

For completeness, and to help avoid possible implementation errors for the higher-order discretizations, the following is an alternate, but equivalent, formulation of the fourth-difference dissipation discretization, as presented by Dias [20]:

$$U\partial_\xi \tilde{\nu} \approx \mathbf{U}D_1\tilde{\boldsymbol{\nu}} + H^{-1}D_m\left(-C^{(2)}_{\xi,U}D_d - C^{(4)}_{\xi,U}D_m^{-1}\tilde{D}_2^T C_2\tilde{D}_2\right)\tilde{\boldsymbol{\nu}}, \tag{3.9}$$

where $D_m = -D_d^T$ and $\tilde{D}_2$ is defined as part of the second derivative operator in (3.10), below. Dias outlines the method by which the dissipation operator can be extended to higher orders of accuracy.

3.2.2 SBP operator for second derivative

The compressible Navier-Stokes equations require a discrete approximation to derivatives of the form $\partial_\xi(\beta\partial_\xi\alpha)$. The simplest means of discretizing these terms is to apply the first derivative twice. Alternatively, one can construct a discrete approximation that has the same stencil width as the first derivative, called a compact-stencil operator. The application of the first derivative twice has several disadvantages compared to compact-stencil operators: larger bandwidth, loss of one order of accuracy, higher global error, and less dissipation of high wavenumber modes. Given these shortcomings, our approach is to use compact SBP operators for the second derivative with variable coefficients.

The second-order operator, originally developed by Mattsson *et al.* [59], can be expressed as

$$D_2(\beta) = H^{-1}\left\{-\left(D_1\right)^T HBD_1 - \frac{1}{4h}\left(\tilde{D}_2\right)^T C_2 B\tilde{D}_2 + EBD_1^{(2)}\right\}, \tag{3.10}$$

where

$$\tilde{D}_2 = \begin{bmatrix} 1 & -2 & 1 & & & & \\ 1 & -2 & 1 & & & & \\ & 1 & -2 & 1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & 1 & -2 & 1 & \\ & & & & 1 & -2 & 1 \\ & & & & 1 & -2 & 1 \end{bmatrix}, \quad EBD_1^{(2)} = \tfrac{1}{h}\begin{bmatrix} \frac{3\beta_1}{2} & -2\beta_1 & \frac{\beta_1}{2} & & & \\ 0 & 0 & 0 & & & \\ & \ddots & \ddots & \ddots & & \\ & & 0 & 0 & 0 & \\ & & \frac{\beta_N}{2} & -2\beta_N & \frac{3\beta_N}{2} \end{bmatrix},$$

B is a diagonal matrix containing the spatially varying coefficients, and D_1 is defined in (3.1). Where necessary, the notation $D^{(b)}$ provides the order of the operator at block boundaries (b). The '˜' symbol signifies an undivided difference operator.

Application to governing equations

Both the viscous terms of the Navier-Stokes equations and the diffusive terms of the turbulence model contain double-derivatives which can be approximated as

$$\partial_\xi \left(\beta \partial_\xi \alpha\right) \approx D_2(\beta)\boldsymbol{\alpha}. \tag{3.11}$$

For an internal node, this will result in the narrow stencil used by Pulliam [81] (with k and m subscripts suppressed):

$$\frac{1}{2}(\beta_{j+1}+\beta_j)(\alpha_{j+1}-\alpha_j) - \frac{1}{2}(\beta_j+\beta_{j-1})(\alpha_j-\alpha_{j-1}). \tag{3.12}$$

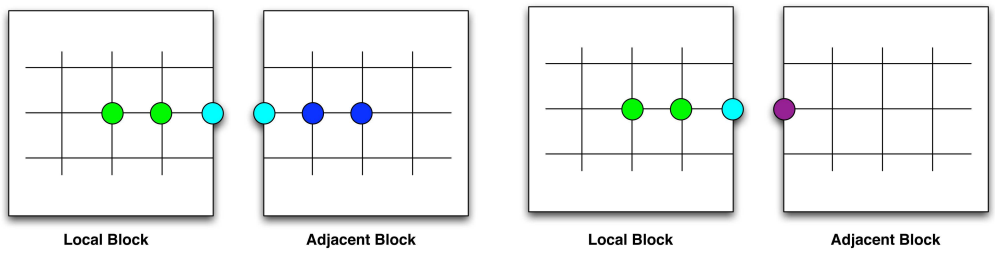
At block boundaries, where one-sided differences are employed, the discretization takes on the form

$$\begin{aligned} &\text{low side:} && -\beta_1\left[2(\alpha_2-\alpha_1)-(\alpha_3-\alpha_2)\right]+\beta_2(\alpha_2-\alpha_1), \\ &\text{high side:} && \beta_N\left[2(\alpha_N-\alpha_{N-1})-(\alpha_{N-1}-\alpha_{N-2})\right]-\beta_{N-1}(\alpha_N-\alpha_{N-1}). \end{aligned} \tag{3.13}$$

3.3 Simultaneous Approximation Terms

The use of SBP operators ties in closely to the application of SAT penalties at block boundaries, either interfaces or domain boundaries. SATs are used to preserve inter-block continuity, or enforce specific boundary conditions. The purpose of this section is not to derive the forms of the various SATs used, but rather to present the implementation used in the present algorithm. See Refs. 37, 95, 96, and 72 for analysis and derivation of the SATs applied to the Navier-Stokes equations. All SATs that follow are shown in the form in which they would be added to the right-hand side of the governing equations. The SATs presented here for the Spalart-Allmaras one-equation turbulence model were developed following the methodology used for the Navier-Stokes equations.

One of the important advantages of SATs lies in the manner in which they allow the algorithm to deal with both domain boundaries and block interfaces. This is illustrated in Figure 3.3, which contrasts how the governing equations would be handled at an interface with a halo-node approach and the current SBP-SAT discretization. The halo-node approach in Figure 3.3(a) requires spatial derivatives to be formed across the interface when dealing with the governing equations of nodes at a block boundary. This is achieved by the use of



(a) Halo-node approach (b) SBP-SAT discretization

Figure 3.3: Node information required for spatial discretization at block interface

halo nodes, effectively completing a full internal discretization stencil with information from the adjoining block. The SBP-SAT approach, illustrated in Figure 3.3(b), instead forms a local one-sided approximation to the required derivatives, as defined by the SBP operators. The SAT is added to the residual of the nodes on the interface. For inviscid flows, the penalty term is based on the direction of flow and the difference between the flow variables on the local node and the corresponding node on the adjoining block. When viscous terms are also considered, the difference in the viscous fluxes on the two sides of the interface also has to be accounted for. To reduce communication costs, the fluxes are calculated on the processor corresponding to the adjoining block before being exchanged, removing the requirement to share several layers of flow solution information. Analogous treatment is applied for the advective and diffusive terms of the Spalart-Allmaras turbulence model. The same approach can be applied at domain boundaries, where the information formerly supplied by an adjoining block is replaced by a boundary condition.

The SBP-SAT approach requires less information from adjoining blocks in order to obtain a discretization of the governing equations at block interfaces. This results in a reduced requirement for information sharing between blocks, which is especially advantageous for parallel algorithms. Inter-processor communication time can become a bottleneck for a parallel algorithm, and by reducing the amount of information that is shared, the overall efficiency of the algorithm improves. Higher-order discretizations will benefit from this feature to a larger extent due to the inherent size of the computational stencils. Additionally, the fact that this discretization does not need to form any derivatives across interfaces reduces the continuity requirements for mesh generation at interfaces. In fact, only C^0 continuity is necessary, allowing the algorithm to provide accurate solutions, even on grids with relatively high incidence angles for grid lines at interfaces, such as those in Figure 3.4. Using the halo-node approach on this grid would result in errors in any spatial derivatives taken across the interfaces.

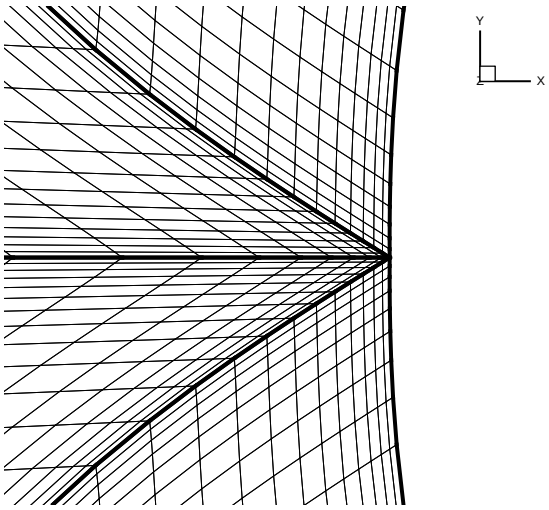


Figure 3.4: Multi-block grid at leading edge of airfoil with C^0 continuity

Sections 3.3.1 and 3.3.2 describe the various types of SATs that are applied in different parts of the domain for both the Navier-Stokes equations and the turbulence model, respectively. The description of each SAT is accompanied by information about which type of boundary or block interface the penalty term should be applied to, along with the appropriate target states required for the SAT.

3.3.1 SATs for the Navier-Stokes equations

The form of the inviscid, or Euler, portion of the SATs on the low side of a block is

$$\mathrm{SAT}_{\mathrm{inv}} = -H_b^{-1} J^{-1} A_\xi^+ \left(\mathbf{Q} - \mathbf{Q_{target}}\right), \tag{3.14}$$

where H_b is the boundary node element of the diagonal norm matrix H,

$$A_\xi^+ = \frac{A_\xi + |A_\xi|}{2}, \quad A_\xi = \frac{\partial \hat{\mathbf{E}}}{\partial \hat{\mathbf{Q}}},$$

and $\mathbf{Q}$ are the flow variables on the boundary node in the current block. $|A_\xi|$ denotes $X^{-1}|\Lambda|X$, where X is the right eigenmatrix of A_ξ, and Λ contains the eigenvalues along its diagonal. At a high-side boundary, A_ξ^- is used to capture the incoming characteristics and the sign of the penalty is reversed. When dealing with boundaries normal to the other two coordinate directions, A_ξ is replaced by either A_η or A_ζ. The variable $\mathbf{Q_{target}}$ takes on the target values to which the local values of $\mathbf{Q}$ are being forced. When dealing with a block interface, these are the flow variable values on a coincident node in a neighbouring block, or, when dealing with a far-field boundary, they can be the free-stream flow variable values. A number of different boundary conditions, such as a slip-wall or symmetry plane, can be

enforced using this approach for the Euler equations. In each case, the SAT works on a principle very similar to characteristic boundary conditions.

The basis of the viscous SATs is presented by Nordström *et al.* [72] and is summarized below, with special attention being paid to each type of block boundary.

The first type of viscous SAT deals with differences in viscous fluxes. In the ξ-direction, this term has the form

$$\mathrm{SAT}_{\text{visc\_flux}} = \frac{H_b^{-1}\sigma^V}{Re}\left(\hat{\mathbf{E}}_{\mathbf{v}} - \hat{\mathbf{E}}_{\mathbf{v,target}}\right), \tag{3.15}$$

where $\hat{\mathbf{E}}_{\mathbf{v}}$ is the local viscous flux, and $\hat{\mathbf{E}}_{\mathbf{v,target}}$ is the target value of the viscous flux. Additionally, $\sigma^V = 1$ at a low-side boundary, and $\sigma^V = -1$ at a high-side boundary. At a far-field boundary, $\hat{\mathbf{E}}_{\mathbf{v,target}} = 0$. Interface SATs also make use of (3.15), where $\hat{\mathbf{E}}_{\mathbf{v,target}}$ is equal to $\hat{\mathbf{E}}_{\mathbf{v2}}$, the viscous flux on the coincident node in the adjoining block.

A no-slip adiabatic wall boundary condition is enforced with the use of a different type of term, which is again added on top of the Euler SAT. The form of the viscous portion of the no-slip wall SAT for a boundary at the low or high side of a block in the ξ-direction, is

$$\mathrm{SAT}_{\text{visc\_wall,1}} = \frac{H_b^{-1}\sigma^W}{Re} I\left(\mathbf{Q} - \mathbf{Q}_{\mathbf{target}}\right), \tag{3.16}$$

where I is the identity matrix,

$$\sigma^W \leq -\frac{\xi_x^2 + \xi_y^2 + \xi_z^2}{J}\frac{\mu}{2\rho}\max\left(\frac{\gamma}{Pr}, \frac{5}{3}\right), \text{ and } \mathbf{Q}_{\mathbf{target}} = \left[\rho, 0, 0, 0, \frac{\rho T_2}{\gamma(\gamma-1)}\right]^T.$$

σ^W is calculated based on local values, while $\mathbf{Q}_{\mathbf{target}}$ is constructed in order to enforce an adiabatic no-slip wall boundary condition. The three momentum components are forced toward zero, thus satisfying the no-slip condition, while no condition is enforced on density, since the local value of density, ρ, will cancel out in the penalty term. The energy equation has a penalty term applied to it based on the value of the temperature of one node above the boundary, $T_2 = T_{j+1,k,m}$. This approach will result in a zero temperature gradient at the solid boundary, along with a no-slip velocity condition. The use of T_2 to enforce the adiabatic condition relies on the assumption that the grid is perpendicular to the surface of the wing, which may not always be true. The form of the SAT presented in (3.16) can also be readily used to enforce an isothermal boundary condition. This can be achieved by replacing the T_2 term in $\mathbf{Q}_{\mathbf{target}}$ with the desired wall temperature, T_w, as described in [96]. Unlike a more traditional method of applying the adiabatic no-slip surface condition, the penalty approach presented here does not apply a boundary condition on the equation for the conservation of mass. This is due to the fact that the Navier-Stokes equations are solved

on all nodes, including the boundaries, so the discretization does not need to provide an explicit value for all variables at the surface (as required for some traditional methods).

An alternate approach to dealing with the adiabatic condition involves a combination of previously discussed penalty terms. The surface penalty in (3.16) can be modified to enforce the no-slip condition only, while the viscous flux penalty in (3.15) can be modified to enforce the zero-temperature gradient necessary for the adiabatic condition. The overall form of this SAT is

$$\mathrm{SAT}_{\mathrm{visc\_wall},2} = \frac{H_b^{-1}}{Re}\left[\sigma^W I\left(\mathbf{Q}-\mathbf{Q_{target}}\right)+\sigma^V\left(\hat{\mathbf{E}}_{\mathbf{v}}-\hat{\mathbf{E}}_{\mathbf{v},\mathbf{target}}\right)\right], \tag{3.17}$$

where

$$\mathbf{Q_{target}} = \left[\rho, 0, 0, 0, e\right]^T,$$

and $\hat{\mathbf{E}}_{\mathbf{v},\mathbf{target}}$ is identical to $\hat{\mathbf{E}}_{\mathbf{v}}$, except that the temperature derivative terms normal to the wall are set to zero. The local values of density and energy are given by ρ and e, respectively, and the coefficients σ^W and σ^V retain their previously defined values. In this way, the first part of the SAT enforces only the no-slip condition, while the second part enforces the adiabatic condition. Since this approach uses the gradients as they appear in the viscous stresses, it makes no assumptions about the grid (whether or not it is perpendicular to the surface), and enforces a more general condition of $\frac{\partial T}{\partial n} = 0$.

In addition to the use of (3.15), block interfaces require viscous SATs for penalizing differences in conservative variable values. The form used is:

$$\mathrm{SAT}_{\mathrm{visc\_vars}} = -\frac{H_b^{-1}\sigma^{V_2}}{JRe} B_{\mathrm{int},\xi}\left(\mathbf{Q}-\mathbf{Q_{target}}\right), \tag{3.18}$$

where

$$\sigma^{V_2} \leq 0.5$$

for stability, and $\mathbf{Q_{target}}$ is the vector of conservative flow variables on the coincident node in the adjoining block. The B_{int} matrix is related to the viscous Jacobian, and is derived based on Nordström *et al.* [72]. The following is the complete form of the matrix, where $\varsigma = \xi,\ \eta,$ or ζ:

$$B_{\mathrm{int},\varsigma} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ -a_1u-a_2v-a_3w & a_1 & a_2 & a_3 & 0 \\ -a_2u-a_4v-a_5w & a_2 & a_4 & a_5 & 0 \\ -a_3u-a_5v-a_6w & a_3 & a_5 & a_6 & 0 \\ b_{51} & b_{52} & b_{53} & b_{54} & a_7 \end{bmatrix},$$

with

$$
\begin{aligned}
a_1 &= t_1\left(4/3\varsigma_x^2 + \varsigma_y^2 + \varsigma_z^2\right), & a_2 &= t_1\left(1/3\varsigma_x\varsigma_y\right),\\
a_3 &= t_1\left(1/3\varsigma_x\varsigma_z\right), & a_4 &= t_1\left(\varsigma_x^2 + 4/3\varsigma_y^2 + \varsigma_z^2\right),\\
a_5 &= t_1\left(1/3\varsigma_y\varsigma_z\right), & a_6 &= t_1\left(\varsigma_x^2 + \varsigma_y^2 + 4/3\varsigma_z^2\right),\\
a_7 &= t_2\gamma\left(\varsigma_x^2 + \varsigma_y^2 + \varsigma_z^2\right), & &\\
b_{52} &= -a_7u + a_1u + a_2v + a_3w, & b_{53} &= -a_7u + a_2u + a_4v + a_5w,\\
b_{54} &= -a_7u + a_3u + a_5v + a_6w, & &\\
t_1 &= \rho^{-1}\left(\mu + \mu_t\right), & t_2 &= \rho^{-1}\left(\mu/Pr + \mu_t/Pr_t\right),
\end{aligned}
$$

$$b_{51} = a_7\left(-e/\rho + (u^2+v^2+w^2)\right) - a_1u^2 - a_4v^2 - a_6w^2 - 2(a_2uv + a_3uw + a_5vw).$$

In order to reduce the size of the computational domain, symmetry boundaries can be imposed. SATs are again used to impose this boundary condition by using (3.14) to impose a purely tangential flow ($\mathbf{Q_{target}}$ is constructed in such a way as to force the normal velocity component to zero). In addition, (3.16) is used to enforce a zero normal gradient in all conservative variables.

The following is used to enforce the inviscid SAT on an outflow boundary in a viscous flow:

$$\mathrm{SAT}_{\mathrm{inv\_outflow}} = \frac{H_b^{-1}\sigma^I}{J}A_\xi^-\left(\mathbf{Q} - \mathbf{Q}_{N-1}\right), \tag{3.19}$$

in which the boundary is assumed to be on the high side of a block in the ξ-direction. The modification is appropriate in dealing with the viscous wake region. The advantage of this approach is that it requires minimal modification to the existing Euler SAT term, which typically uses the free-stream flow conditions instead of $\mathbf{Q}_{N-1}$. An alternate approach to dealing with the outflow condition is presented by Svärd *et al.* [95].

3.3.2 SATs for the turbulence model

The SAT for the advection portion of the turbulence model needs to account for the flow direction in much the same way as the Euler equation SATs. This can be achieved using the following form of the SAT:

$$\mathrm{SAT}_{\mathrm{adv}} = H_b^{-1}\sigma_{\mathrm{a}}\left(\tilde{\nu} - \tilde{\nu}_{\mathrm{target}}\right), \tag{3.20}$$

where $\tilde{\nu}$ is the local value of the turbulence variable, and $\tilde{\nu}_{\mathrm{target}}$ is the target value of the turbulence variable, which can either be specified by a boundary condition or, in the case of

a block interface, the corresponding value on an adjoining block. The SAT parameter σ_{a} is constructed so that it accounts for the direction of information propagation in the flow:

$$\sigma_{\mathrm{a}} = -\frac{1}{2}\left[\max\left(|U|, \phi\right) + \delta_{\mathrm{a}} U\right], \tag{3.21}$$

where δ_{a} is +1 on the low side of a block, and -1 on the high side of a block. On an interface, all flow related information in σ_{a}, such as the contravariant velocity U, is based on an average velocity between the coincident interface nodes, while at a domain boundary, it is constructed based on local information only. Finally, ϕ is a limiting factor introduced to prevent the SAT from completely disappearing in regions where the value of U goes to zero, such as near a solid surface. Following the work done on the Euler equation SATs, the value of ϕ was chosen to be

$$\phi = V_l\left(|U| + a\sqrt{\xi_x^2 + \xi_y^2 + \xi_z^2}\right), \tag{3.22}$$

where $V_l = 0.025$, and a is the speed of sound. The quantity appearing in the brackets above is the spectral radius of the inviscid flux Jacobian.

As with the SATs used for the viscous portion of the Navier-Stokes equations, the SATs for the diffusive portion of the turbulence model consist of two parts, one dealing with the difference in the turbulent quantity, the other dealing with the difference in the turbulent quantity gradient.

The diffusive SAT dealing with the difference in gradients of the turbulence variable has the general form

$$\mathrm{SAT}_{\mathrm{diff\_flux}} = H_b^{-1}\sigma_{\mathrm{df}}\left(g - g_{\mathrm{target}}\right), \tag{3.23}$$

where σ_{df} is +1 on the low side of a block and -1 on the high side of a block. Additionally, the turbulence quantity gradients, denoted by g, have the form

$$g = \frac{1}{\sigma Re}\left(\nu + \tilde{\nu}\right)\left(\xi_x^2 + \xi_y^2 + \xi_z^2\right)\delta_\xi \tilde{\nu}, \tag{3.24}$$

where $\delta_\xi \tilde{\nu}$ is a one-sided first derivative consistent with the definition of the second derivative SBP operator at block boundaries, specified in the $D_1^{(2)}$ matrix of (3.10). The parameter σ is specified as part of the turbulence model definition with a value of 2/3. This SAT is applied at the farfield boundary, with the target gradient set to 0, or at block interfaces, with the target gradient calculated based on values at the interface of the adjoining block.

The diffusive SAT that deals with the difference in flow variables has a form analogous to the viscous SAT presented by Nordström *et al.* [72] for the Navier-Stokes equations,

$$\mathrm{SAT}_{\mathrm{diff\_vars}} = -H_b^{-1}\frac{1}{4\sigma Re}\sigma_{\mathrm{dv}}\left(\tilde{\nu} - \tilde{\nu}_{\mathrm{target}}\right), \tag{3.25}$$

where

$$\sigma_{\text{dv}} = (\nu + \tilde{\nu}) \left(\xi_x^2 + \xi_y^2 + \xi_z^2 \right). \tag{3.26}$$

As with the advective SAT, the value of σ_{dv} is based on a state average when dealing with an interface, or simply the local state when at a domain boundary. Grid metrics are always taken from the local block information. This SAT is applied at block interfaces, wall boundaries (where the target value is 0), and symmetry planes (where the target value is taken from one node inside the boundary).

When dealing with the negative turbulence model described in §2.2.1, it is important to modify the coefficient values for the diffusive SATs, accounting for the presence of the f_n term. If the local value of $\tilde{\nu}$ is less than zero, the SATs make use of

$$g_{\text{neg}} = \frac{1}{\sigma Re} (\nu + f_n \tilde{\nu}) \left(\xi_x^2 + \xi_y^2 + \xi_z^2 \right) \delta_\xi \tilde{\nu}, \tag{3.27}$$

and

$$\sigma_{\text{dv,neg}} = (\nu + f_n \tilde{\nu}) \left(\xi_x^2 + \xi_y^2 + \xi_z^2 \right). \tag{3.28}$$

The farfield condition used with the turbulence model for fully turbulent flows sets the target farfield value of $\tilde{\nu}$ to 3.0, as suggested by Spalart and Rumsey [92], while flows with an explicit trip location specified use a target farfield value of 0.1. The target surface value of $\tilde{\nu}$ is set to 0.

3.3.3 Production and destruction terms

While the production and destruction terms act as source terms, therefore not necessitating the application of the SBP-SAT approach due to the absence of spatial derivatives, it is necessary to add a source term for nodes located directly on the surface of the aerodynamic body. Even though the distance value is zero at a solid boundary, the original reference for the turbulence model presents limiting values for both the production and destruction terms. The values are based on a local surface shear stress, τ_w. The initialization of the flow conditions to a uniform state results in the shear stress being negligible in the early iterations of the solution process, resulting in insignificant limiting values for the production and destruction terms. The lack of source terms for the surface nodes leads to a significant difference in the residual value between the surface nodes and the nodes directly above the surface. This difference can result in large, destabilizing updates to the turbulence variable, often causing the flow solution to diverge.

To mitigate this, a destruction source term is added to all nodes with a zero off-wall distance in order to stabilize the solution in the early stages of convergence. It is calculated

using a value of $d = d_{\min}/2$, where $d_{\min}$ is the smallest non-zero off-wall distance in the entire computational domain. This term acts as an estimate of the limiting value of the source term as $d \to 0$. The use of the term for the surface nodes does not have a significant impact on the converged solution, as it forces the surface $\tilde{\nu}$ towards zero. The destruction source term is compatible with both the standard and negative turbulence model definitions.

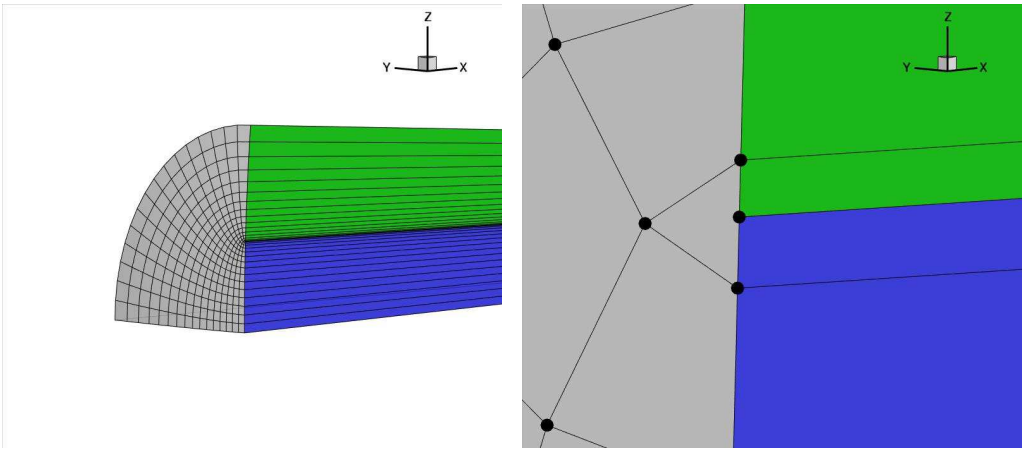
3.3.4 Application of SATs to block boundaries

In order to provide a clear outline of the application of SATs to various block boundaries, Tables 3.1 and 3.2 contain a summary of the SATs discussed in §3.3.1 and §3.3.2. The tables describe the SATs required for each type of block boundary for both the Navier-Stokes equations and the Spalart-Allmaras turbulence model, respectively, as well as the necessary target states. For the sake of simplicity and consistency, all SATs are presented in the form they would possess on a block boundary with a constant $\xi = 0$ (or index $j = 1$). Analogous versions are constructed for all other block faces. For completeness, the target states and values are described in Table 3.3. The coefficient values appearing in the SATs have already been described in the preceding sections, and references to particular equations are provided with each SAT.

Even though the inviscid and viscous SATs for the Navier-Stokes equations are described separately, they are in fact concurrently applied to the governing equations when solving laminar or turbulent flows. The distinction is made to highlight the important fact that the SATs are designed to enforce a penalty on either the inviscid or viscous fluxes. The same can be said of the advection and diffusion SATs for the Spalart-Allmaras turbulence model.

3.4 Spatial Discretization on Degenerate Grids

Grid generation around complicated three-dimensional geometries may, even for multi-block grids, necessitate the use of topological features that are not conducive to a finite-difference discretization. In striving to generalize the algorithm to be able to handle grids from third parties, even if they do not conform to topologies currently used with the algorithm, modifications to the code were necessary to be able to complete flow solutions on what are termed "degenerate" grids. One such example can be found in the grids used for the Fifth AIAA Drag Prediction Workshop (DPW5), for which the overall topology can be seen in Figure 3.1(b). The O-O topology that the workshop grids make use of contains several instances of degenerate edges, as illustrated in Figure 3.5. The degenerate edge occurs at the joining of the blue and green block faces, representing block faces with constant ξ- and η-coordinates



(a) Location of degenerate edge between blue and green block faces

(b) Detail of triangular cell

Figure 3.5: Degenerate edge in O-O multi-block grid

that would typically be perpendicular to each other. When these faces are forced to be parallel, as is the case for this topology, triangular cells are formed along the edge where the two faces meet, seen in Figure 3.5(b).

For a node-centered finite-difference discretization, which relies on values of flow variables and grid metrics at individual nodes, a triangular cell such as this causes severe problems. In particular, when performing the coordinate transformation, it is necessary to calculate grid metrics and the metric Jacobian at every node. However, the value of the metric Jacobian at the node midway along the straight line of the triangular cell is undefined, resulting in an `NaN` or `infinity` value. The residual calculated for this node will also be undefined, and the algorithm will terminate.

In order to be able to perform flow solutions on grids with degenerate edges, a modification to the grid metric and residual calculation has been implemented. This modification is applied at any nodes located along the edges where the original calculation would result in undefined values. First, the metric Jacobian is calculated as a weighted average of the metric Jacobian values of nodes adjacent to the degenerate edge. For an edge on the low side of a block in the ξ and η directions (with j and k index values equal to unity), one obtains

$$J_{\text{degen}} = \frac{d_j J_{j+1,k,m} + d_k J_{j,k+1,m}}{d_j + d_k}, \tag{3.29}$$

where d_j and d_k are the distances from the degenerate edge node to the nodes at the $(j+1, k, m)$ and $(j, k+1, m)$ locations (the end points along the straight edge with 3 nodes in Figure 3.5(b)), respectively, with $J_{j+1,k,m}$ and $J_{j,k+1,m}$ the metric Jacobian values at those nodes. Second, for the mean-flow equations, the residual value at the degenerate nodes is

replaced with

$$\mathbf{R}_{\text{degen}} = J_{\text{degen}}^{-1} \left(\mathbf{Q}_{j,k,m} - \frac{d_j \mathbf{Q}_{j+1,k,m} + d_k \mathbf{Q}_{j,k+1,m}}{d_j + d_k} \right), \tag{3.30}$$

where $\mathbf{Q}$ is the vector of mean-flow variables at a node, defined in (2.5). Equation (3.30) is used for the turbulence model by removing the inverse metric Jacobian term. The above formulation forces the flow variables along a degenerate edge to take on the weighted average of the two adjacent nodes, which has been shown to provide a good approximation of the solution along degenerate edges.

The current implementation of the above modification relies on a user-supplied input file that defines the locations of any degenerate edges in the grid. Appendix A.3.2 contains an example of such an input file.

Table 3.1: SAT application at block boundaries for Navier-Stokes equations (on $j = 1$ block face)

Inviscid SATs

Boundary type	**Form**	**Target vector**	**Eqn.**
wall (adiabatic)	$-H_b^{-1} J^{-1} A_\xi^+ \left(\mathbf{Q} - \mathbf{Q}_{\mathbf{target}}\right)$	$\mathbf{Q}_{V_t}$	(3.14)
wall (isothermal)			
symmetry			
farfield		$\mathbf{Q}_{\mathrm{ff}}$	
farfield (outflow)		$\mathbf{Q}_{j+1,k,m}$	
interface		$\mathbf{Q}_2$	

Viscous SATs

Boundary type	**Form**	**Target vector**	**Eqn.**
wall (adiabatic)	$\frac{H_b^{-1}\sigma^W}{Re} I \left(\mathbf{Q} - \mathbf{Q}_{\mathbf{target}}\right)$	$\mathbf{Q}_{\mathrm{w}}(\mathrm{T}_2)$	(3.16)
wall (isothermal)		$\mathbf{Q}_{\mathrm{w}}(\mathrm{T}_w)$	
symmetry		$\mathbf{Q}_{j+1,k,m}$	
farfield	$\frac{H_b^{-1}\sigma^V}{Re}\left(\hat{\mathbf{E}}_{\mathbf{v}} - \hat{\mathbf{E}}_{\mathbf{v},\mathbf{target}}\right)$	$\hat{\mathbf{E}}_{\mathbf{v},\mathrm{ff}} = 0$	(3.15)
farfield (outflow)			
interface	$-\frac{H_b^{-1}\sigma^{V_2}}{JRe} B_{\mathrm{int},\xi} \left(\mathbf{Q} - \mathbf{Q}_{\mathbf{target}}\right)$	$\mathbf{Q}_2$	(3.18)
	$\frac{H_b^{-1}\sigma^V}{Re}\left(\hat{\mathbf{E}}_{\mathbf{v}} - \hat{\mathbf{E}}_{\mathbf{v},\mathbf{target}}\right)$	$\hat{\mathbf{E}}_{\mathbf{v},2}$	(3.15)
wall (adiabatic #2)	$\frac{H_b^{-1}\sigma^W}{Re} I \left(\mathbf{Q} - \mathbf{Q}_{\mathbf{target}}\right)$	$\mathbf{Q}_{\mathrm{w}}(\mathrm{T}_1)$	(3.17)
	$\frac{H_b^{-1}\sigma^V}{Re}\left(\hat{\mathbf{E}}_{\mathbf{v}} - \hat{\mathbf{E}}_{\mathbf{v},\mathbf{target}}\right)$	$\hat{\mathbf{E}}_{\mathbf{v},\partial\mathrm{T}=0}$	

Table 3.2: SAT application at block boundaries for Spalart-Allmaras turbulence model (on $j = 1$ block face)

Advection term SATs

Boundary type	**Form**	**Target value**	**Eqn.**
wall (adiabatic)	$H_b^{-1}\sigma_{\mathrm{a}}\left(\tilde{\nu} - \tilde{\nu}_{\mathrm{target}}\right)$	$\tilde{\nu}_{\mathrm{w}} = 0$	(3.20)
wall (isothermal)			
symmetry		$\tilde{\nu}_{j+1,k,m}$	
farfield		$\tilde{\nu}_{\mathrm{ff}}$	
farfield (outflow)			
interface		$\tilde{\nu}_2$	

Diffusion term SATs

Boundary type	**Form**	**Target value**	**Eqn.**
wall (adiabatic)	$-H_b^{-1}\frac{1}{4\sigma Re}\sigma_{\mathrm{dv}}\left(\tilde{\nu} - \tilde{\nu}_{\mathrm{target}}\right)$	$\tilde{\nu}_{\mathrm{w}} = 0$	(3.25)
wall (isothermal)			
symmetry		$\tilde{\nu}_{j+1,k,m}$	
farfield	$H_b^{-1}\sigma_{\mathrm{df}}\left(g - g_{\mathrm{target}}\right)$	$g_{\mathrm{ff}} = 0$	(3.23)
farfield (outflow)			
interface	$-H_b^{-1}\frac{1}{4\sigma Re}\sigma_{\mathrm{dv}}\left(\tilde{\nu} - \tilde{\nu}_{\mathrm{target}}\right)$	$\tilde{\nu}_2$	(3.25)
	$H_b^{-1}\sigma_{\mathrm{df}}\left(g - g_{\mathrm{target}}\right)$	g_2	(3.23)

Table 3.3: SAT target state descriptions

Target	Description
$\mathbf{Q}_{V_t}$	Local flow variables with normal velocity component removed
$\mathbf{Q}_{\mathrm{ff}}$	Farfield flow variables, defined in (2.15)
$\mathbf{Q}_{j+1.k.m}$	Flow variables from boundary-adjacent node
$\mathbf{Q}_2$	Flow variables from coincident node on adjacent block
$\mathbf{Q}_{\mathrm{w}}(T_2)$	Flow variables with zero velocity and boundary-adjacent temperature, defined in (3.16)
$\mathbf{Q}_{\mathrm{w}}(T_w)$	Flow variables with zero velocity and specified wall temperature, defined in (3.16)
$\mathbf{Q}_{\mathrm{w}}(T_1)$	Flow variables with zero velocity, defined in (3.17)
$\hat{\mathbf{E}}_{\mathbf{v},\mathrm{ff}}$	Viscous flux vector in farfield
$\hat{\mathbf{E}}_{\mathbf{v},2}$	Viscous flux vector from coincident node on adjacent block
$\hat{\mathbf{E}}_{\mathbf{v},\partial \mathrm{T}=0}$	Viscous flux vector with temperature derivatives removed
$\tilde{\nu}_{\mathrm{w}}$	Surface value of turbulence quantity
$\tilde{\nu}_{\mathrm{ff}}$	Farfield value of turbulence quantity
$\tilde{\nu}_{\mathrm{j+1,k,m}}$	Boundary-adjacent value of turbulence quantity
$\tilde{\nu}_2$	Turbulence quantity value from coincident node on adjacent block
g_{ff}	Farfield value of turbulence quantity gradient
g_2	Turbulence quantity gradient from coincident node on adjacent block

Chapter 4

Iteration to Steady-State

Several strategies exist by which one can obtain converged flow solutions from an initial guess. These range from explicit to implicit solution techniques, with the possibility of simulating the flow in a time-accurate manner, and can be independent of the spatial discretization used. The objective of this thesis is the creation of an efficient and robust solution algorithm for the steady RANS equations. Hence, the algorithm can make use of techniques that accelerate convergence to steady-state while sacrificing time-accuracy. Grids required for the accurate solution of the RANS equations often possess high aspect-ratio cells in the boundary layer, which have an adverse effect on the convergence characteristics of explicit methods. Additionally, the SBP-SAT discretization (Chapter 3) can necessitate very small time steps when used as part of an explicit solution algorithm; hence, the current work focuses on implicitly solving the RANS equations.

This chapter describes the solution strategy implemented as part of the current algorithm, in order to develop an efficient and robust method for computing steady-state flow solutions around aerodynamic configurations. Optimization of the solution algorithm will be performed in Chapter 5.

4.1 Nonlinear System of Equations

Applying the SBP-SAT discretization described in Chapter 3 to the Navier-Stokes equations and the Spalart-Allmaras one-equation turbulence model results in a large system of nonlinear ordinary differential equations, represented as

$$\frac{d\boldsymbol{\mathcal{Q}}}{dt} + \boldsymbol{\mathcal{R}}(\boldsymbol{\mathcal{Q}}) = 0, \tag{4.1}$$

where $\boldsymbol{\mathcal{R}}$ is the residual, and $\boldsymbol{\mathcal{Q}}$ represents the complete solution vector. The current work is focused on obtaining steady-state solutions for the governing equations, simplifying (4.1) to the following nonlinear algebraic system of equations:

$$\boldsymbol{\mathcal{R}}(\boldsymbol{\mathcal{Q}}) = 0. \tag{4.2}$$

4.2 Solution of the Nonlinear Algebraic System

In order to obtain a solution to the nonlinear system defined in (4.2), an iterative strategy is used to evolve the solution vector from an initial state to a converged steady-state solution. Newton's method will converge quadratically if a suitable initial iterate is known. This initial iterate must be sufficiently close to the solution of (4.2). Since it is unlikely that any initial guess made for a steady-state solution will satisfy this requirement, the present algorithm makes use of a globalization (or continuation) strategy whose purpose it is to find a suitable initial iterate. A common choice for many CFD algorithms, pseudo-transient continuation (PTC) [44, 48] makes use of a time-marching analogy, providing the algorithm with increasing time step values during the solution process. More recent work has focused on homotopy-based continuation methods [38], such as dissipation-based continuation (DBC) [35]. DBC adds substantial dissipation to the discretization, resulting in a less stiff problem. Once the solution is obtained, the dissipation is reduced and the process is repeated. This continues until the additional dissipation vanishes and a steady solution is obtained. Line-search and trust-region methods have also been used in the globalization of the Navier-Stokes equations [78]. In the context of this algorithm, several continuation strategies have been investigated for inviscid and laminar viscous flows [35], with dissipation-based continuation (DBC) showing promise as a robust alternative to commonly used PTC. For turbulent flows, the current implementation of DBC lacks stability, often resulting in divergence of the nonlinear residual due to the presence of large updates to the turbulent quantity. Hence, PTC is used here. It should be noted that the work in this thesis was not focused on studying the applicability of the DBC approach for the RANS equations, and further research is warranted in this area.

The selection of the method is based on two factors. First, it is not necessary for the solution to remain time-accurate during the solution process, removing accuracy requirements for the intermediate states generated during the solution process. Second, an implicit solution technique will allow for large time steps. Based on previous work in our group [12, 19, 68, 80], the implicit Euler time-marching method has been chosen as the backbone of the solution algorithm, with the ability to further speed up the solution process with the use of spatially-varying time steps, as will be discussed below.

When time-marched with the implicit Euler time-marching method and a local time linearization, (4.1) results in a large system of linear equations of the form [55]:

$$\left(\mathcal{T}^{(n)} + \mathcal{A}^{(n)}\right) \Delta \boldsymbol{\mathcal{Q}}^{(n)} = -\boldsymbol{\mathcal{R}}^{(n)}, \tag{4.3}$$

where n is the outer (nonlinear) iteration index, $\mathcal{T}^{(n)}$ is a diagonal matrix that contains the

inverse local time step values, $\boldsymbol{\mathcal{R}}^{(n)} = \boldsymbol{\mathcal{R}}(\boldsymbol{\mathcal{Q}}^{(n)})$, $\Delta\boldsymbol{\mathcal{Q}}^{(n)} = \boldsymbol{\mathcal{Q}}^{(n+1)} - \boldsymbol{\mathcal{Q}}^{(n)}$, and

$$\mathcal{A}^{(n)} = \frac{\partial \boldsymbol{\mathcal{R}}^{(n)}}{\partial \boldsymbol{\mathcal{Q}}^{(n)}}$$

is the flow Jacobian. In the infinite time step limit, the above describes Newton's method.

In the current algorithm, the PTC implementation consists of two phases. The first phase, the approximate-Newton phase, starts from the initial guess of a uniform free-stream flow and converges the residual of (4.2) by several orders of magnitude. At this point, the algorithm switches to the inexact-Newton phase, which converges the system fully, obtaining a steady-state flow solution. Both phases result in a large set of linear equations at each outer iteration, which are solved to a specified tolerance using the preconditioned Krylov iterative solver GMRES [84].

The following sections describe the two phases of the solution process in detail, along with aspects of the algorithm critical in obtaining an efficient solution process, while Chapter 5 presents parameter studies that were performed to obtain efficient and robust performance.

4.2.1 Approximate-Newton phase

The approximate-Newton method serves as a start-up phase to find a suitable initial iterate for Newton's method. Since a time-accurate solution is not of interest, some useful modifications can be made. These include a first-order Jacobian matrix and a spatially varying time step.

A first-order Jacobian matrix, $\mathcal{A}_1$, has been shown to be an effective replacement for the true Jacobian, $\mathcal{A}$, during the start-up phase [12, 68, 80]. The implicit Euler method requires a time step whose inverse is added to the diagonal elements of $\mathcal{A}_1$. A spatially varying time step has been shown to improve the convergence rates of Newton-Krylov algorithms [81], leading to the use of

$$\Delta t^{(n)}_{j,k,m} = \frac{J_{j,k,m}\Delta t^{(n)}_{\text{ref}}}{1 + \sqrt[3]{J_{j,k,m}}} \tag{4.4}$$

for three-dimensional flows, while two-dimensional simulations use [93]

$$\Delta t^{(n)}_{j,k} = \frac{J_{j,k}\Delta t^{(n)}_{\text{ref}}}{1 + \sqrt{J_{j,k}}}, \tag{4.5}$$

where (j, k, m) denote the indices of the node to which this time step is being applied. Since the solver uses the unscaled flow variables $\mathbf{Q}$, instead of the transformed variables $\hat{\mathbf{Q}}$, the J term that results from the coordinate transformation is lumped into the numerators of (4.4) and (4.5). The reference time step is

$$\Delta t^{(n)}_{\text{ref}} = a(b)^n, \tag{4.6}$$

where typical values used for turbulent flow solutions are $a = 0.001$ and $b \in [1.1, 1.3]$. Further details are presented in §5.3.3.

An important part of using a start-up phase is knowing when a suitable iterate has been found to initiate the inexact-Newton phase. For this purpose, the relative drop in the residual is used:

$$R_d^{(n)} \equiv \frac{||\boldsymbol{\mathcal{R}}^{(n)}||_2}{||\boldsymbol{\mathcal{R}}^{(0)}||_2}. \tag{4.7}$$

For turbulent flows, once this value reaches 1×10^{-4}, i.e. the residual has dropped by 4 orders of magnitude in the approximate-Newton phase, the algorithm switches to the inexact-Newton method. This initial drop is larger than is required for inviscid or laminar solutions for two reasons. First, the turbulence quantity fluctuates substantially more than the mean-flow quantities during the start-up phase, necessitating a longer start-up than flow solutions dealing with inviscid or laminar flows. Second, due to the use of grids with much finer spacing near the surface of the aerodynamic shape, the initial residual, $\boldsymbol{\mathcal{R}}^{(0)}$, starts with a larger value, but drops by one to two orders of magnitude very quickly before settling into a convergence pattern similar to that observed with inviscid or laminar solves. Hence, the relative residual drop threshold, R_d, is adjusted to compensate for these differences. This parameter may need to be adjusted slightly depending on the complexity of the flow being solved, in particular for explicitly tripped turbulent flow solutions.

4.2.2 Approximate flow Jacobian

The implicit solution algorithm used in the current work requires the formation of an approximate flow Jacobian matrix, $\mathcal{A}_1$, which represents an approximation to the partial derivative of the global residual with respect to the global solution vector. The formation and factorization of the Jacobian represents a substantial portion of the overall solution time and memory requirements; hence, an efficient implementation is necessary. Several methods exist by which the flow Jacobian matrix can be obtained, such as using the finite-difference method, complex-step method [4,57], automatic differentiation, or analytical differentiation.

While the analytical differentiation of the residual requires the highest algorithm development investment, it also provides the fastest execution time. For this reason, the majority of the flow Jacobian terms were derived analytically. In select cases, such as several of the SATs, the complex-step method was used to obtain their linearization. While slower than an analytical derivative, the complex-step method eases the development process, and its use for a small subset of the Jacobian formulation does not add significant computational time.

A number of approximations are made in creating the first-order approximation, $\mathcal{A}_1$.

When dealing with the inviscid terms, the fourth-difference dissipation coefficient, κ_4, is combined with the second-difference dissipation coefficient, κ_2, to form a modified second-difference dissipation coefficient, $\tilde{\kappa}_2$, such that

$$\tilde{\kappa}_2 = \kappa_2 + \sigma_{\text{dlf}}\kappa_4,$$

where σ_{dlf} is a lumping factor. A value of $\sigma = 8$ has been shown to work well for the Navier-Stokes solutions with scalar dissipation [43]. Current work with matrix dissipation uses $\sigma = 12$. The modified fourth-difference dissipation coefficient, $\tilde{\kappa}_4$, is set to zero. Applying this lumping approach reduces the number of matrix entries for the inviscid terms, reducing the memory requirements for the code. Only information from nearest neighbours nodes is used. The lumping approach is also used for the fourth-difference dissipation discretization option of the turbulence model, but with a unique lumping factor, σ_{SA}, which will be discussed in §5.2.2.

The viscous terms, however, still possess a relatively large stencil. To mitigate this, the cross-derivative terms that appear in the viscous stresses are dropped when constructing the first-order Jacobian. This approach reduces the stencil of all interior nodes to nearest neighbors only, matching the stencil size of the inviscid terms, which is substantially smaller than that of the full flow Jacobian. The linearization of the viscous flux SATs for the Navier-Stokes equations is also modified to ignore the tangential derivatives, which are analogous to the cross-derivatives. Additionally, the viscosity value appearing in the viscous fluxes is treated as a constant when forming the approximate Jacobian.

If first-order upwinding is used, no approximations are made to the discretization of the turbulence model when constructing the Jacobian entries that arise due to the solution of this extra equation, since all cross-derivatives were dropped during the coordinate transformation and the turbulence model already possesses the minimum stencil size.

One of the important features of the approximate flow Jacobian matrix is its sparse nature, due to the localized effect of the flow solution variables at individual nodes on the residual vector. This can be exploited in order to significantly reduce the memory requirements for the storage of the Jacobian. In the current algorithm, the block-compressed-row (BCR) storage scheme is used to store the nonzero entries of the Jacobian.

4.2.3 Inexact-Newton phase

The inexact-Newton phase uses a different scheme for the reference time step, designed to ramp the value toward infinity more rapidly than in the approximate-Newton phase. This

eventually eliminates the inverse time term from the left-hand-side of the discretized Navier-Stokes equations. The present work involves the use of a scheme based on the work of Mulder and van Leer [65], by which a new reference time step is calculated and used in (4.4):

$$\Delta t_{\text{ref}}^{(n)} = \max\left[\alpha\left(R_d^{(n)}\right)^{-\beta}, \Delta t_{\text{ref}}^{(n-1)}\right], \tag{4.8}$$

where $\beta \in [1.5, 2.0]$, α is calculated as

$$\alpha = a(b)^{n_{\text{Newt}}}\left(R_d^{(n_{\text{Newt}})}\right)^{\beta},$$

and n_{Newt} is the first iteration of the inexact-Newton phase. The above scheme ensures a smooth transition of time step values between the approximate- and inexact-Newton phases, preventing any possible abrupt increases, while also providing a non-decreasing progression of reference time step values.

In contrast with the approximate-Newton phase, the inexact-Newton phase uses the full second-order accurate Jacobian. However, since the algorithm uses a Krylov subspace method, there is no need to form the full Jacobian matrix, $\mathcal{A}$, explicitly. Instead, only Jacobian-vector products are required, which can be approximated using a first-order forward difference, known as a Frechet derivative:

$$\mathcal{A}^{(n)}\mathbf{v} \approx \frac{\boldsymbol{\mathcal{R}}(\boldsymbol{\mathcal{Q}}^{(n)} + \epsilon\mathbf{v}) - \boldsymbol{\mathcal{R}}(\boldsymbol{\mathcal{Q}}^{(n)})}{\epsilon}. \tag{4.9}$$

The parameter ϵ is determined from

$$\epsilon = \sqrt{\frac{N_u\delta}{\mathbf{v}^T\mathbf{v}}},$$

where N_u is the number of unknowns, and δ is a perturbation parameter. The approximate Jacobian, $\mathcal{A}_1$, is still used for preconditioning the system. By using the Frechet derivative and not requiring the full flow Jacobian, the algorithm provides a significant memory savings.

Finally, neither the approximate-Newton nor the inexact-Newton phase solves its respective linear system exactly. Instead, the following inequality is used to govern how far the system is solved:

$$||\boldsymbol{\mathcal{R}}^{(n)} + \mathcal{A}^{(n)}\Delta\boldsymbol{\mathcal{Q}}^{(n)}||_2 \leq \eta_n||\boldsymbol{\mathcal{R}}^{(n)}||_2,$$

where the forcing parameter η_n is specified. If it is too small, the linear system will be over-solved and will take too much time without benefiting the convergence of the outer iterations, but if it is too large, nonlinear convergence will suffer. For the present work, a value of 0.05 is used for the approximate-Newton phase, while 0.01 is used for the inexact-Newton phase.

An important aspect to consider when using GMRES is the amount of memory available on the computational hardware. Since GMRES stores the vectors from the individual

search directions, it is important to limit the maximum number that are stored (to prevent exceeding the available system memory). This is especially critical in the inexact-Newton phase, where the linear solver may have difficulty attaining the above-mentioned relative tolerance before the maximum number of search directions is reached, especially for complex flows. For the flow solutions presented in this work, limiting the number of search directions to 70 provides sufficient linear convergence without reaching memory limitations of the computational systems. It should be stressed that this limit is typically triggered in the inexact-Newton phase, which deals with more complex linear systems (no approximations in flow Jacobian). A possible work-around for the memory limitation is the use of restarted GMRES, but previous work by Chisholm [18] states that this approach leads to a significant increase in the number of linear iterations, and is therefore not considered here.

4.2.4 Preconditioning

Effective preconditioning is critical to an efficient parallel linear solver. The first-order Jacobian, $\mathcal{A}_1$, is factored using block incomplete lower-upper factorization (BILU) with fill level p in order to construct the preconditioner. This is a computationally expensive task, especially in the approximate-Newton phase, which requires many outer iterations. Previous work [37,46] has shown that lagging the update of the preconditioner (freezing it for a number of iterations) during the start-up phase can positively impact the efficiency of the flow solver. Experience with turbulent flow solutions is mixed, with lagging often resulting in divergence of the nonlinear problem. This is most likely due to the relatively large updates in the turbulent flow quantities during the early stages of convergence, rendering preconditioners from previous nonlinear iterations ineffective. A typical fill level for the factorization is 2, but levels as high as 4 are sometimes required for more problematic cases. Further details about fill level are provided in §5.3.3.

Two approaches to parallel preconditioning, additive-Schwarz [45] and approximate-Schur [85], have been previously investigated in the context of a parallel Newton-Krylov flow solver for the Euler [36, 37] and Navier-Stokes [36] equations, with a thorough description of their application to the current linear system provided in the references. The approximate-Schur parallel preconditioner is used in the current work, making use of the interface nodes in constructing the global Schur complement.

Approximate-Schur parallel preconditioner

This section provides a brief description of the approximate-Schur preconditioner used in the current algorithm, originally implemented by Hicken [37]. The preconditioning step requires the solution of the generic global system

$$A\mathbf{x} = \mathbf{b}, \tag{4.10}$$

where $A \in \mathbb{R}^{n\times n}$ and $\mathbf{x}, \mathbf{b} \in \mathbb{R}^n$. In the current implementation, $A = \mathcal{A}_1$.

In the context of a distributed parallel implementation, it is necessary to define a representation of the individual subsystems located on various processes. To achieve this, the rectangular matrix P_i is used to project the global vector onto a local vector corresponding to the unknowns stored on process i. Applying P_i to the linear system (4.10), and considering process-local solutions of the form $\mathbf{x} = P_i^T\mathbf{x}_i$, we obtain the block diagonal system

$$A_i\mathbf{x}_i = \mathbf{b}_i, \tag{4.11}$$

where $\mathbf{b}_i = P_i\mathbf{b}$ and $A_i = P_iAP_i^T$. The local matrix A_i is factored into L_iU_i using BILU(p) [62].

The local unknowns, $\mathbf{x}_i$, are separated into two groupings, consisting of unknowns dependent on local information only, $\mathbf{u}_i$, and unknowns that are coupled to information on process $j \neq i$, defined as $\mathbf{y}_i$. Additionally, $\mathbf{y}_j$ represent interface unknowns on non-local processes that are part of local equations. The coupling is a result of the application of SATs at block interfaces. By placing the unknowns $\mathbf{y}_i$ last, this ordering partitions $P_iA\mathbf{x} = P_i\mathbf{b}$ into the following block structure:

$$\begin{pmatrix} B_i & F_i \\ H_i & C_i \end{pmatrix} \begin{pmatrix} \mathbf{u}_i \\ \mathbf{y}_i \end{pmatrix} + \begin{pmatrix} 0 \\ \sum_{j\in N_i} E_{ij}\mathbf{y}_j \end{pmatrix} = \begin{pmatrix} \mathbf{f}_i \\ \mathbf{g}_i \end{pmatrix}, \tag{4.12}$$

where N_i is the set of neighbour processes for the local process. The variables $\mathbf{u}_i$ are not coupled across processes and can be eliminated by reordering the first equation as

$$\mathbf{u}_i = B_i^{-1}(\mathbf{f}_i - F_i\mathbf{y}_i). \tag{4.13}$$

Substituting $\mathbf{u}_i$ into (4.12), a system of equations for the coupling variables in the global computational domain is obtained:

$$\underbrace{\begin{pmatrix} S_1 & E_{12} & \dots & E_{1P} \\ E_{21} & S_2 & \dots & E_{2P} \\ \vdots & & \ddots & \vdots \\ E_{P1} & E_{P2} & \dots & S_P \end{pmatrix}}_{S} \begin{pmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \\ \mathbf{y}_P \end{pmatrix} = \begin{pmatrix} \mathbf{g}_1' \\ \mathbf{g}_2' \\ \vdots \\ \mathbf{g}_P' \end{pmatrix}, \tag{4.14}$$

where $S_i = C_i - H_i B_i^{-1} F_i$, and $\mathbf{g}_i' = \mathbf{g}_i - H_i B_i^{-1} \mathbf{f}_i$. The coefficient matrix S is the Schur complement corresponding to the variables coupled between processes. Solving (4.14) requires the inverse of S_i, which can be expensive to form explicitly. Instead, Saad and Sosonkina [85] recognized that an ILU factorization of S_i can be extracted from an ILU(p) factorization of A_i. Their Schur-based preconditioner consists of a GMRES-accelerated approximate solution of (4.14), with S_i replaced by its ILU factorization. Once approximate solutions to the $\mathbf{y}_i$ are obtained, they are substituted into (4.13), with B_i replaced with its ILU(p) factorization, to obtain $\mathbf{u}_i$. A parallel implementation of this approach is used in the current work.

4.2.5 Node ordering

The effectiveness and efficiency of the linear solution algorithm, in particular the preconditioner, is greatly affected by the location of the entries in the large sparse matrix of the flow Jacobian [11]. In particular, the offset of the entries from the diagonal, characterized as the bandwidth of the matrix and affected by the ordering of the nodes in the system, is critical. Higher clustering near the diagonal, known as lower bandwidth, has been shown to improve the accuracy of the ILU(p) factorization, improving the convergence of the linear solver. In particular, previous work within our group [19, 80] has shown the reverse Cuthill-McKee (RCM) ordering [54] to be most effective for large sparse systems that result from the discretization of the governing equations.

RCM reordering is used in the current algorithm, with two modifications that are pertinent to the parallel multi-block system decomposition described in (4.11):

1. RCM reordering is performed on each process i separately. A global ordering for multi-block systems in a parallel algorithm would introduce substantial computational and memory overhead, as well as unnecessary algorithm complexity.

2. On each process, the local unknowns, $\mathbf{u}_i$, and the interface unknowns, $\mathbf{y}_i$, are reordered separately. This is done in order to preserve the overall separation of the unknowns required by the approximate-Schur parallel preconditioner, as outlined in §4.2.4.

A "downstream bias" modification to the RCM algorithm is used, as outlined by Chisholm [18]: when two nodes of the same order are encountered, the node located further downstream is given preference in the ordering process during the forward sweep of the RCM algorithm. The selection of the root node for the RCM reordering strategy has a significant impact on the performance of the reordering. Chisholm demonstrated that a downstream root node is an effective choice for Newton-Krylov algorithms used for the solution of the RANS equations. The importance of effective node ordering is demonstrated in §5.3.2.

4.2.6 Equation and variable scaling

The introduction of the inverse metric Jacobian term, J, into the Navier-Stokes equations as a result of the coordinate transformation discussed in §2.1.1 can have a negative impact on the approximate solution of the linear system. This is due to the large difference in the magnitudes of J near the solid body and in the farfield. If left unaddressed, large differences in the scales of the linear system entries can result in unpredictable behaviour of the approximate linear solver. For example, the algorithm may not reduce the residuals of nodes close to the surface (where large values of J occur), in extreme cases leading to divergence. An effective means of dealing with the scaling discrepancy in the Navier-Stokes equations is the use of equation scaling, during which each equation is scaled by the local value of J.

The addition of the turbulence model to the linear system of (4.3) presents some unique challenges, as the scaling of the linear system can be adversely affected, resulting in unpredictable behavior of the linear solver, as well as the nonlinear iterations. The improper scaling arises from several factors. First, the turbulence model does not contain the inherent geometric scaling present in the mean-flow equations (division by J). Second, the turbulence quantity can be as large as 1000 or higher in the converged solution, while the nondimensionalized mean-flow quantities rarely exceed 2. Finally, the terms that result from the linearization of the turbulence model with respect to the mean-flow variables add large off-diagonal values to the Jacobian. Hence, a more sophisticated scaling approach has been implemented to account for these discrepancies, based on the work done by Chisholm and Zingg [19], in order to obtain an efficient and accurate solution of the linear system.

The row, or equation, scaling of the mean-flow equations is achieved by multiplying the equations by a factor that includes the metric Jacobian, removing the inherent geometric scaling, while the turbulence model is scaled by $\tilde{\nu}_{\max}^{-1}$. This value is selected in order to balance the large turbulence variable quantities that can occur during a flow solution, often several orders of magnitude larger than the mean-flow solution quantities. This approach effectively normalizes the turbulence equation by that quantity. The value of this parameter impacts the numerical properties of the linear system provided to the linear solution algorithm, GMRES, and hence has an impact on the accuracy with which the algorithm solves the linear system during each nonlinear iteration. Section 5.3.1 presents a summary of the various scaling approaches used in the current algorithm. To complete the scaling, and in order to normalize the flow variable values, the turbulence variable quantity is also multiplied by $\tilde{\nu}_{\max}^{-1}$. Hence, instead of solving the system presented in (4.3), the linear solution algorithm solves a scaled

system of the form

$$S_a S_r \left(\frac{\mathcal{I}}{\Delta t} + \mathcal{A}^{(n)} \right) S_c S_c^{-1} \Delta \boldsymbol{\mathcal{Q}}^{(n)} = -S_a S_r \boldsymbol{\mathcal{R}}^{(n)}, \tag{4.15}$$

where S_r and S_c are the row and column scaling matrices, respectively. S_a is an auto-scaling matrix used to bring the values of the individual equation components within an order of magnitude, further improving the scaling of the linear system. In the current implementation, these matrices are defined as

$$S_r = \mathrm{diag}\left(S_{r1}, ..., S_{rN}\right), \quad S_c = \mathrm{diag}\left(S_{c1}, ..., S_{cN}\right),$$

where

$$S_{ri} = \begin{bmatrix} J_i^{2/3} & & & & & & \\ & J_i^{2/3} & & & & & \\ & & J_i^{2/3} & & & & \\ & & & J_i^{2/3} & & & \\ & & & & J_i^{2/3} & & \\ & & & & & \tilde{\nu}_{\max}^{-1} J_i^{-1/3} \end{bmatrix}, \quad S_{ci} = \begin{bmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & 1 & & & & \\ & & & 1 & & & \\ & & & & 1 & & \\ & & & & & \tilde{\nu}_{\max} \end{bmatrix},$$

and J_i is the value of the metric Jacobian at the i^{th} node in the computational domain. The values in the auto-scaling matrix are calculated based on the equation-wise residual L_2-norms of the partially scaled system $S_r \boldsymbol{\mathcal{R}}^{(n)}$. Instead of scaling each node by a unique value, they scale the component equations by different amounts. Any residual values required for the time step calculation make use of the partially scaled residual $S_r \boldsymbol{\mathcal{R}}^{(n)}$.

4.2.7 Special considerations for turbulence model

Negative turbulence quantities

As discussed in §2.2.1, it is not atypical to encounter negative values of $\tilde{\nu}$ in the flowfield when using the standard version of the turbulence model. These values are nonphysical, and merely a result of the occurrence of large transients in the solution, especially during the early stages of convergence or after the switch from the approximate-Newton phase to the inexact-Newton phase. However, it is important to address these negative values, since they could destabilize the solution process. The approach taken is to trim $\tilde{\nu}$ during the solution update step according to the following:

$$\tilde{\nu} = \begin{cases} \tilde{\nu} & : \ \tilde{\nu} \geq 0 \\ 10^{-14}\nu & : \ \tilde{\nu} < 0 \text{ (on solid surface)} \\ 10^{-3}\nu & : \ \tilde{\nu} < 0 \text{ (elsewhere)} \end{cases} \tag{4.16}$$

The local ν value is introduced such that advective and diffusive fluxes do not vanish completely from regions where several adjacent nodes are trimmed during the same iteration. The above trimming approach is only used for the standard version of the turbulence model.

Additional trimming is used when dealing with the value of vorticity, S. In order to avoid numerical problems, this value is not allowed to fall below 8.5×10^{-10}. Finally, the vorticity-like term, $\tilde{S}$, cannot be allowed to become nonpositive, which would have a destabilizing effect on the values of the production and destruction terms. The approach presented in (2.30) works well in this context.

An alternative to the above is the use of the "negative" turbulence model outlined in §2.2.1. Rather than trim negative values of $\tilde{\nu}$, the model makes use of modified source term expressions to force the solution towards a physical state. Studies presented in Chapter 5 will highlight instances where the use of the "negative" turbulence model is warranted.

Off-wall distance calculation

An important consideration in the use of the Spalart-Allmaras turbulence model is the method by which the off-wall distance, d, is calculated. Many implementations, in an effort to save computational time and reduce code complexity, make use of approximations in calculating the distance from each volume node to the closest solid boundary. These range from following grid lines towards a surface, a method that itself becomes very complex when dealing with multi-block grids, to calculating distances to individual grid nodes on the surface. While these methods provide reasonably accurate representations of d, certain grid topologies can result in values that have a substantial effect on the force coefficients. This effect is due to the sensitivity of the Spalart-Allmaras turbulence model to the values used for d. It is therefore highly recommended that a method which identifies the entire surface, including edges and surface segments between the nodes on a solid boundary, be used in calculating d, as it will provide more accurate off-wall distance values on a wide variety of grids, regardless of blocking or gridding anomalies.

Several methods for calculating d exist, ranging from brute-force searches to the solution of differential equations [98, 99]. Depending on the complexity of the geometry and the blocking strategy for the mesh, the calculation of d can present a substantial computational expense. In order to speed up the calculation of the off-wall distances, methods that rely on a priori knowledge of the grid topology have also been used in the past; for example, some algorithms follow grid lines emanating from the surface and use the chord lengths along those lines in computing the off-wall distance of any nodes located on the grid lines. This approach has several disadvantages, foremost of which is the inability to easily generalize

to multi-block grids. While not hindered by assumptions about grid topology or geometry complexity, many differential approaches provide increasingly inaccurate values of d as one moves away from the surface. The extent to which this may affect the accuracy of the turbulence model depends on the specifics of the implementation. The current algorithm makes use of two search approaches. The first approach finds the distance from each internal node to the closest surface node, while the second takes into account distances to grid edges and surfaces on the solid boundary, in addition to the surface nodes. As will be demonstrated, ignoring the grid edges and surfaces of a solid boundary may have a significant impact on the computed flow solution. This is particularly noticeable on grids with non-perpendicular grid lines.

The calculation of d can become especially problematic for parallel algorithms, especially when the solid surface boundaries are located on blocks being solved by other processes. In the current implementation, all solid surface information is first collected into an array that is distributed to all processes., ensuring that all processes, regardless of whether or not they contain solid boundaries, are aware of the entire solid surface geometry. This framework also lends itself easily to efficient parallel computations, as each process is responsible for calculating the value of d only for local nodes. By ensuring that the number of nodes per processor remains approximately the same regardless of overall grid size, which is often a necessary requirement for implicit algorithms due to memory requirements, the calculation of d scales with the total number of surface nodes, a value much smaller than the total number of nodes.

In some cases, use of one calculation method versus the other results in the residual diverging, but this is very rare. Test cases highlighting the differences in the two approaches are presented in §5.2.3, in addition to descriptions of the algorithms implemented for the distance calculation.

Time step for turbulence model

In their work on a two-dimensional Newton-Krylov flow solution algorithm, Chisholm and Zingg [19] discuss the need to introduce limits on the time step used for the turbulence model. In their experience, the limits were required to achieve robust performance of the algorithm, which could become destabilized if large updates to the turbulence variable occurred in the early stages of convergence.

For a strong boundary condition treatment, the large fluctuations in the turbulence quantities are usually observed near solid boundaries and are the result of a large difference between the initial values of the turbulence quantities of the surface and the surface-adjacent

nodes. Through the use of a strong boundary condition, the surface values are forced to remain at a constant value throughout the solution process, regardless of what is happening elsewhere in the flowfield. This can result in large flux values, eventually leading to large, destabilizing solution updates. By limiting the time step used for the turbulence model, the large updates to the turbulence variable are significantly reduced, resulting in more robust performance over a range of flow conditions. The SBP-SAT discretization, on the other hand, does not initially differentiate between surface and surface-adjacent nodes; all nodes are initialized to the same flow variable values. Instead, the SATs act to weakly enforce the boundary conditions, slowly changing the values of the surface variables as the solution evolves. In this way, the SBP-SAT discretization rarely sees large updates to the turbulence quantity, and therefore does not require the use of special time step values for the turbulence model when solving fully turbulent flows. Similar convergence improvements have been demonstrated by Eliasson *et al.* [24], who have shown that weakly enforced boundary conditions provide faster convergence to steady state for the Navier-Stokes equations.

The only special time step limitations required for the turbulence model used in this algorithm are related to explicitly tripped turbulent flow solutions. As expected, the substantial nonlinearities present in such a flow pose a difficult problem for the solution algorithm, necessitating the use of a reduced time step. This was also true of the work done in [19], and the current approach is based on the results presented therein.

The use of the explicit trip terms in the turbulence model when simulating flows with laminar to turbulent transition can introduce substantial instabilities into the solution process, potentially resulting in divergence if not addressed. An effective strategy is to limit the time step used for the turbulence model, while the mean-flow equation time step remains unchanged. In particular, the following has been found to work well for cases when explicit trip terms are active:

$$\Delta t_{\mathrm{SA\_trip}} = \frac{1}{100}\Delta t, \tag{4.17}$$

where Δt is is defined in Eq. (4.4). This modification is not necessary for fully turbulent flow simulations.

Chapter 5

Algorithm Optimization

In developing a robust flow solution algorithm, many of the components have to be tuned for optimal performance. These include, but are not limited to, preconditioner level of fill, residual reduction for switching between solver phases, time marching parameters, and SAT implementation details. While some of the previous sections presented details of many of the components of the algorithm, the intent here is to provide justification behind those choices. The goal of this section is to provide parameters that will result in an efficient, robust, and accurate flow solution algorithm.

The following sections present a summary of the steps undertaken in the optimization of the current algorithm. They are broken down into two groups, one covering modifications to the spatial discretization, the other covering tuning of the nonlinear solution algorithm. Additionally, §5.1 provides a summary of the grid characteristics and flow conditions of the test cases used in the optimization of the current algorithm.

5.1 Test Case Specification

In order to optimize the algorithm for both efficiency and robustness, it is necessary to select a set of test cases that provides a sample sufficiently representative of the types of geometries and flows that would typically be encountered. To this end, Table 5.1 presents a summary of the geometries and grids used for the algorithm optimization in this section, while Table 5.2 summarizes the flow conditions used. Geometries 1 and 2 represent two-dimensional shapes, while the remainder of the cases contain three-dimensional configurations. Together, the information from the tables is used to fully specify the test case; for example, test case 2A consists of a subsonic flow solution around the RAE2822 airfoil geometry. Descriptions of the grid topologies can be found in Appendix C. Note that the O-O topology contains degenerate edges.

The grid sizes used in this chapter are smaller than what would typically be used for production runs. This was a conscious choice made in order to facilitate a higher throughput of runs for the parameter studies. While this may result in errors in force coefficients due to

Table 5.1: Geometries and grids used in test cases

	Geometry	Topology	Blocks	Nodes	Off-wall spacing
1	NACA0012 airfoil	H	8	1.9×10^4	9.2×10^{-7}
2	RAE2822 airfoil	H	8	2.4×10^4	1.5×10^{-6}
3	ONERA M6 wing	Y-H	128	15.1×10^6	9.0×10^{-7}
4	CRM wing-body	O-O	704	6.0×10^6	5.1×10^{-6}

Table 5.2: Flow conditions used in test cases

	M	Re	α	Note
A	0.50	6.00×10^6	3.00°	Fully turbulent flow
B	0.84	11.72×10^6	3.06°	Fully turbulent flow
C	0.85	5.00×10^6	2.30°	Fully turbulent flow
D	0.30	1.00×10^6	1.00°	Explicitly tripped flow
E	0.40	40.00×10^6	1.00°	Fully turbulent flow

grid resolution, most studies are aimed at improving the efficiency of the algorithm, which is not affected by the accuracy of the converged force coefficients. Importantly, the parameters obtained in the following sections were used for a wide range of results presented in Chapter 6, on grids with up to 150 million nodes.

5.1.1 Force coefficients

The following convention is adopted for test cases where force coefficients are provided. When two-dimensional cases are presented alone, the coefficients for lift and drag per unit span are labeled as C_l and C_d, respectively. For three-dimensional cases, the coefficients of lift and drag are labeled as C_L and C_D to differentiate them from their two-dimensional counterparts. However, if both two- and three-dimensional results are presented together, for example in a table, the labels of C_L and C_D will be used, with the explicit assumption that the two-dimensional values still represent the coefficients per unit span.

The lift, $\tilde{L}$, and drag, $\tilde{D}$, experienced by an aerodynamic body are defined as

$$\tilde{L} = \frac{1}{2}\tilde{\rho}_\infty \tilde{u}_\infty^2 S C_L, \text{ and } \tilde{D} = \frac{1}{2}\tilde{\rho}_\infty \tilde{u}_\infty^2 S C_D, \tag{5.1}$$

where $\tilde{\rho}_\infty$ and $\tilde{u}_\infty$ are the (dimensional) free-stream values of density and velocity, respectively, and S is a reference area. C_L and C_D represent the coefficients of lift and drag, respectively. Defining non-dimensional values of lift and drag using the approach in §2.1, we

get:

$$L = \frac{\tilde{L}}{\tilde{\rho}_\infty \tilde{a}_\infty^2 c^2}, \text{ and } D = \frac{\tilde{D}}{\tilde{\rho}_\infty \tilde{a}_\infty^2 c^2}. \tag{5.2}$$

The force coefficients can be written as

$$C_L = \frac{L}{\frac{1}{2}\left(\frac{\tilde{u}_\infty}{\tilde{a}_\infty}\right)^2 S_{\text{ref}}} = \frac{L}{\frac{1}{2} M_\infty^2 S_{\text{ref}}}, \text{ and } C_D = \frac{D}{\frac{1}{2}\left(\frac{\tilde{u}_\infty}{\tilde{a}_\infty}\right)^2 S_{\text{ref}}} = \frac{D}{\frac{1}{2} M_\infty^2 S_{\text{ref}}}, \tag{5.3}$$

where S_{ref} is the non-dimensional reference area, defined as

$$S_{\text{ref}} = \frac{S}{c^2}. \tag{5.4}$$

For the ONERA M6 wing, the reference area is $S = 1.506m^2$, with the root chord length of $0.8059m$ used as the reference length. The non-dimensional reference area S_{ref} is 2.3188.

For the NASA Common Research Model (CRM) wing-body geometry [102], the reference area is specified as $S = 383.69m^2$, with the mean aerodynamic chord (MAC) of $7.0053m$ used as the reference length. Hence, the non-dimensional reference area S_{ref} is 7.8185.

When calculating the lift and drag coefficients for half-wing or half-body geometries, as are commonly used in computational grids, the above reference areas have to be divided by two.

5.2 Spatial Discretization Modifications

Changes to the spatial discretization impact the robustness and accuracy of the algorithm. Several choices exist in the discretization of the governing equations, such as the type of numerical dissipation model or modifications available for the Spalart-Allmaras turbulence model. These choices have a direct effect on the residual; hence any modifications are also reflected in the preconditioner matrix, $\mathcal{A}_1$. All other algorithm inputs, such as the PTC time-stepping parameters, are held constant to ensure that the observed performance differences between two variations in the spatial discretization are purely due to the inherent discretization properties.

As discussed in Chapter 3, the current algorithm uses either a scalar or a matrix artificial dissipation model. The scalar dissipation model is used by default, as it results in more robust convergence for a wide range of flow solutions. However, the matrix dissipation model provides increased accuracy for a given grid resolution. Additional discussion on the dissipation models is presented in Appendix D.

During the course of this thesis work, a number of modifications aimed at improving the accuracy and robustness of the Spalart-Allmaras turbulence model have been investigated.

Presented here is a discussion of some of the approaches that have resulted in improved behaviour of the model for a substantial range of flow conditions.

5.2.1 Turbulence model variant

One of the choices available within the current algorithm allows the user to select between the standard and "negative" Spalart-Allmaras turbulence models. While the converged solutions for both model variants will be identical for most flow solutions, it is important to understand the differences between the models and how they impact the convergence process.

As mentioned in §2.2.1, the "negative" turbulence model was recently developed to handle instances when negative turbulence quantity values occur in the flow field. The most common instance of this arises during the convergence process, when large updates to the solution vector push the turbulence quantity into nonphysical values. With the standard model, this is addressed by clipping the solution updates at a small positive value whenever a negative turbulence quantity occurs during the solution update process. However, even though a steady-state solution to the RANS equations should not possess any of these nonphysical quantities, there exist cases in which under-resolved grids can lead to negative quantities in the converged solutions. In these cases, the clipping approach will result in the nonlinear residual stalling out, since the solution updates repeatedly force the turbulence quantity to become negative. It is for these cases that the "negative" turbulence model can provide better convergence characteristics. Instead of repeatedly clipping the solution updates, the "negative" model variant modifies the diffusion, production, and destruction terms in areas of negative turbulence, stabilizing the solution and forcing the turbulence variable towards a physical state. In this way, negative turbulence quantities remain a valid solution to the turbulence model even at steady-state. It should be stressed, though, that while a negative turbulence quantity, $\tilde{\nu}$, is acceptable for the model variant solution, the RANS equations still require non-negative values of turbulent viscosity, μ_t. Hence, a form of clipping is still used with the "negative" model when calculating μ_t, such that any negative value of $\tilde{\nu}$ results in zero turbulent viscosity.

While the "negative" turbulence model does provide improved convergence characteristics for a small subset of cases, the vast majority of flow solutions, and the grids used for them, do not contain negative values of $\tilde{\nu}$ in the steady-state converged solutions, and the standard clipping approach can easily deal with transient negative values that occur during the convergence process. For this reason, it is difficult to justify the use of the more complex "negative" model variant for the current algorithm, especially with the second-order spatial discretization. For higher-order spatial discretizations, the problem of negative $\tilde{\nu}$ in

the converged solution becomes more prevalent, as discussed in [9, 73]. In these instances, the switch to the "negative" model will be warranted, and most likely unavoidable.

5.2.2 Dissipation for turbulence model

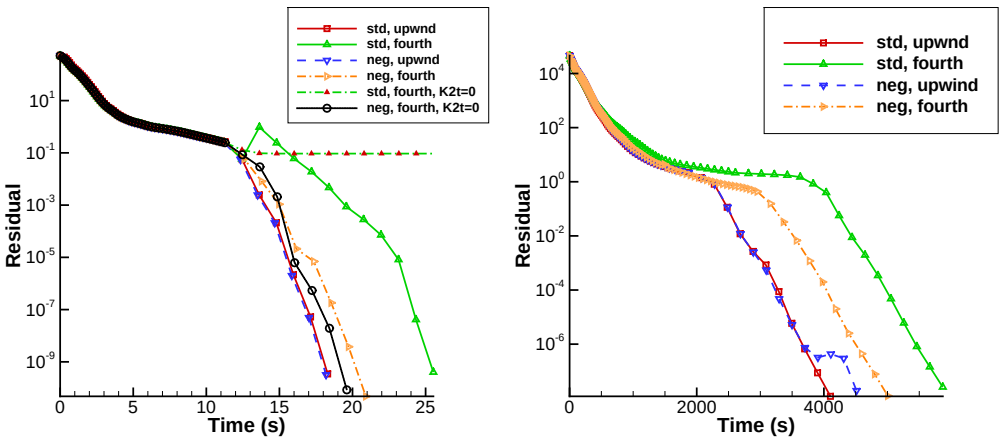
As this algorithm will serve as the basis of future expansions to higher-order spatial discretizations, it is important to address aspects of the discretization that may hinder such efforts. The foremost of these is the use of first-order upwinding in the discretization of the advection terms of the Spalart-Allmaras turbulence model. The upwinding strategy, suggested for use by the original authors, stabilizes the turbulence model discretization, effectively adding additional dissipation. The formulation is identical to the second-difference dissipation added to the mean-flow equations, without the use of the pressure switch for shock capturing. For this reason, the upwinding approach is limited to first-order accuracy, which, while not detrimental to the current second-order discretization, would prevent a high-order algorithm from achieving its theoretical convergence order.

The use of a fourth-difference dissipation model for the turbulence model equation is investigated, replacing the first-order upwinding strategy. The details of the approach were presented in §3.2.1. The critical advantage of this discretization approach is that it is not limited in order, and can be selected to match the order of accuracy of the overall spatial discretization. The second-difference dissipation term, while limited to first-order accuracy, is only active in select areas of the flow, limiting its effect.

As with the dissipation model for the mean-flow equations, the fourth-difference dissipation approach requires values for the dissipation coefficients, κ_{2t} and κ_{4t}, and the dissipation lumping parameter, σ_{SA}. Through extensive testing, the following were found to work well for a wide sample of flow simulations:

$$\kappa_{2t} = 2.0, \quad \kappa_{4t} = 0.04, \quad \sigma_{SA} = 10.0.$$

Cases 1E and 3A are used for the comparison of the two dissipation approaches. The tests also serve as a means of comparing the two turbulence model variants discussed above. For each case, flow solutions are obtained with both model variants, each coupled with both dissipation discretizations. For the two-dimensional case, simulations are also performed with the fourth-difference dissipation approach with κ_{2t} set to zero. The convergence histories of all runs can be seen in Figure 5.1, providing a clear comparison of the model variant and dissipation discretization combinations. For the two-dimensional case, the only combination which results in a stalled residual is the standard turbulence model with the fourth-difference dissipation approach and $\kappa_{2t} = 0$. For converged cases, the dissipation model choice has a



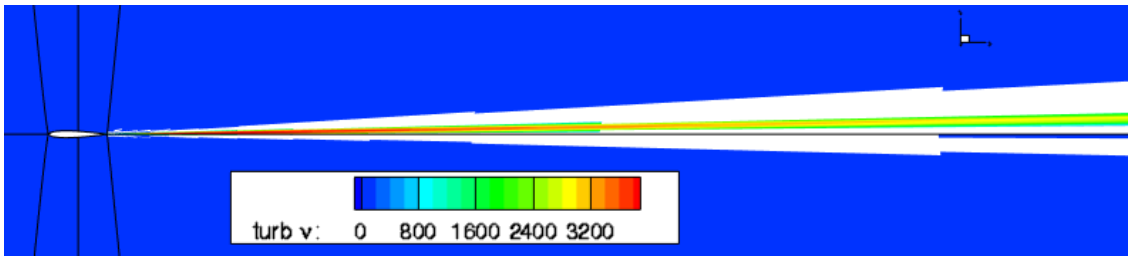
(a) Case 1E (b) Case 3A

Figure 5.1: Convergence histories for turbulence model dissipation study

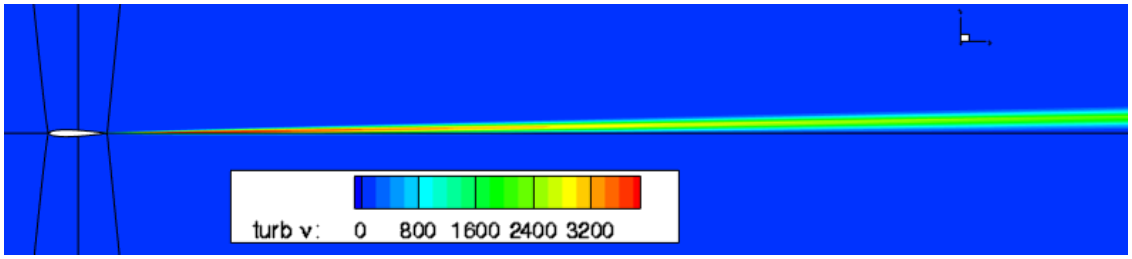
minimal impact on the steady-state solutions, with the force coefficients differing by less than 1%. In the three-dimensional case, the fourth-difference dissipation approach results in visibly slower convergence for both turbulence model variants. As with case 1E, minimal differences are observed in the steady-state values of the lift and drag coefficients.

The lack of convergence of the fourth-difference dissipation approach with $\kappa_{2t} = 0$ for the standard turbulence model provides an interesting insight into the effect the dissipation has on the steady-state solution for the turbulence variable, $\tilde{\nu}$. This model and dissipation combination results in large areas of negative $\tilde{\nu}$ appearing during the solution process, causing the stall in the nonlinear residual. In fact, examining the converged solution of this dissipation approach with the "negative" turbulence model variant shows large areas of negative $\tilde{\nu}$ in the wake region, as seen in Figure 5.2(a). When κ_{2t} is set to its default value, the negative $\tilde{\nu}$ values are no longer present, as seen in Figure 5.2(b). This suggests that the fourth-difference dissipation terms do not add enough dissipation to the discretization of the advection terms, especially in the wake area, where large gradients in the value of $\tilde{\nu}$ are present. When κ_{2t} is set to zero, an oscillation in the turbulence variable occurs, producing large negative values (on the order of -500). However, introducing the pressure-switch-like term, which activates additional dissipation in regions of high turbulence quantity gradients, eliminates the negative values completely.

Even though the fourth-difference dissipation approach provides adequate convergence properties, it results in longer convergence times than the upwinding approach. In addition, the current second-order spatial discretization strategy does not benefit from the potential higher-order accuracy of this dissipation model. However, future work on high-order



(a) Solution with $\kappa_{2t} = 0.0$ (negative regions in white)



(b) Solution with $\kappa_{2t} = 1.0$

Figure 5.2: Turbulence quantity for advection with fourth-difference dissipation

codes can most definitely benefit from discretizing the advection terms of the turbulence model with the fourth-difference dissipation approach. Additional second-difference dissipation, governed by a pressure-switch-like term, is critical in stabilizing this discretization by maintaining positivity in $\tilde{\nu}$, particularly in the converged solution.

5.2.3 Off-wall distance calculation

A key component contributing to the accuracy of the Spalart-Allmaras turbulence model is the value of the off-wall distance, d, which represents the location of the closest solid surface. This quantity has a direct impact on the source terms of the model, influencing the rate at which the turbulence quantity, $\tilde{\nu}$, is produced or destroyed, especially in the near-wall regions of the flow.

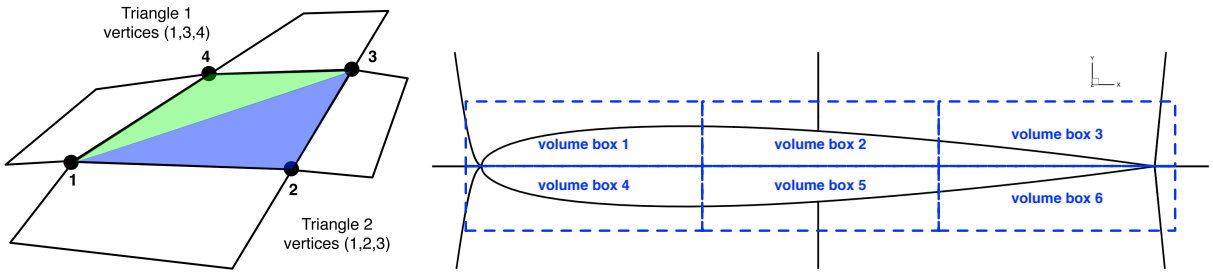
Two distinct algorithms were developed for the calculation of d. The first, based on the distance to the closest surface node, is outlined in Algorithm 1. The parallel implementation results in a fast calculation of d for the entire computational domain, but makes the assumption that the closest solid boundary is in fact a grid node, and not an edge or surface between two boundary nodes. An important feature of this algorithm is that the entire solid surface boundary information vector is distributed to all processes. This ensures that processes without solid surface boundaries are "aware" of all solid surfaces, a critical feature for a parallel algorithm designed to work with multi-block meshes.

Algorithm 1: Node-based distance calculation

```
1 on root process, gather node coordinates from complete solid surface into single vector;
2 distribute surface coordinate vector to all processes;
3 for all processes do
4     for all local volume nodes do
5         d ← 1e20 ;                                   /* temporary large value */
6         for surface node vector do
7             calculate distance, tmp, from surface node to volume node;
8             if tmp < d then
9                 d ← tmp;
```

Depending on the characteristics of the grid, surface nodes may not be the closest points on the solid surface boundaries. For this reason, a second approach, outlined in Algorithm 2, was developed. Instead of treating the surface as a collection of individual nodes, the solid boundary is represented as a collection of triangular elements, as highlighted in Figure 5.3(a). In order to account for all possible surface orientations for volume nodes, the algorithm first calculates the orthogonal distance to an intercept point within the area of the surface element, then the orthogonal distance to the three triangle edges (bounded by the triangle vertices), and finally the distance to each vertex. For two-dimensional grids, the surface triangles are collapsed into surface edges.

While the surface-based approach provides a more accurate representation of d, the added complexity of the calculation requires substantially more computational time. In order to alleviate the computational burden, an important additional step was added to the algorithm. The volume containing the complete surface boundary is divided into boxes, and all surface triangles are sorted into these subdivisions, as shown in Figure 5.3(b). The number of volume subdivisions is related to the number of surface nodes; grids with more surface nodes will result in a larger number of volume subdivisions for the off-wall distance calculation. The algorithm proceeds by checking the off-wall distances to the triangles in the closest subdivision boxes. However, it is important to also account for surface triangles that belong to two of the boxes at the same time, labeling them as separate "crossover" triangles. The final algorithm makes use of the positions of the closest box to substantially decrease computational time: when evaluating the box subdivisions, if the closest off-wall distance calculated thus far is smaller than the distance to any unchecked boxes, all remaining boxes are skipped.



(a) Surface triangle elements

(b) Example of volume subdivision boxes

Figure 5.3: Surface representation and volume subdivision for surface-based wall distance calculation

Cases 2A and 4C were used in the comparison of the two distance calculation algorithms, with a focus on timing and the effect of distance calculation on the converged steady-state solutions. Table 5.3 presents grid and surface information for each case, as well as a comparison of the computational times for the various algorithms. The effects on the converged solutions can be seen through the coefficients of lift and drag, C_L and C_D, respectively.

Since the off-wall distance value only affects a small portion of the turbulence model equation, the effects on convergence are negligible. Instead, the value of d can have a substantial impact on the steady-state force coefficients, especially for the three-dimensional solution. The cause of this can be deduced from examining the regions of the grid that result in the largest differences in the value of d produced by the two algorithms. For the wing-body geometry, the largest relative differences occur near the junction of the wing and fuselage, due to high incidence angles of grid lines and surface boundaries, and can be as high as 80%; the high incidence angles result in larger off-wall distance values for Algorithm 1, impacting the converged solution. In addition, the largest differences occur in the region of the flow that experiences recirculation, a flow feature that has a significant effect on turbulence, resulting in the observed change in force coefficients. This case highlights the sensitivity of the Spalart-Allmaras turbulence model to the value of d and reinforces the need for an accurate calculation of off-wall distance.

Based on the robustness and wide-ranging applicability of Algorithm 2, it is used as the default choice. It introduces a slightly longer computational time, but provides more accurate values of d, and the computational time is still insignificant compared to the overall time required for obtaining a steady-state solution. This can have a significant impact on the steady-state force coefficients, in particular for grids with non-orthogonal grid lines.

Algorithm 2: Surface-based distance calculation

```
1  on root process, gather surface triangle coordinates from complete solid surface;
2  distribute surface triangle coordinates to all processes;
3  for all processes do
4      sort surface elements into volume subdivision boxes;
5      for all local volume nodes do
6          d ← 1e20 ;                                   /* temporary large value */
7          for crossover surface triangles do
8              calculate orthogonal distance, tmp, from triangle surface to volume node;
9              if tmp < d then
10                 d ← tmp;
11             for surface triangle edges do
12                 calculate orthogonal distance, tmp, from triangle edges to volume node;
13                 if tmp < d then
14                     d ← tmp;
15             for surface triangle vertices do
16                 calculate distance, tmp, from triangle vertices to volume node;
17                 if tmp < d then
18                     d ← tmp;
19         for all unchecked subdivision boxes do
20             calculate orthogonal distance, tmp, from closest box to volume node;
21             if tmp < d then
22                 for box surface triangles do
23                     calculate orthogonal distance, tmp, from triangle surface to volume node;
24                     if tmp < d then
25                         d ← tmp;
26                     for surface triangle edges do
27                         calculate orthogonal distance, tmp, from triangle edges to volume node;
28                         if tmp < d then
29                             d ← tmp;
30                     for surface triangle vertices do
31                         calculate distance, tmp, from triangle vertices to volume node;
32                         if tmp < d then
33                             d ← tmp;
34             else
35                 cycle to next volume node;
```

Table 5.3: Results for wall-distance calculation algorithm comparison

Alg.	Case	Nodes	Surface nodes	Solution time (s)	Distance calc. (s)	C_L	C_D
1	2A	1.9×10^4	196	23.2	$3.4 \times 10{-}3$	0.6039	0.01058
2				23.2	$2.7 \times 10{-}2$	0.6040	0.01057
1	4C	6.0×10^6	59,696	1254	4.30	0.5000	0.02650
2				1421	8.96	0.5099	0.02590

5.3 Solution Algorithm Tuning

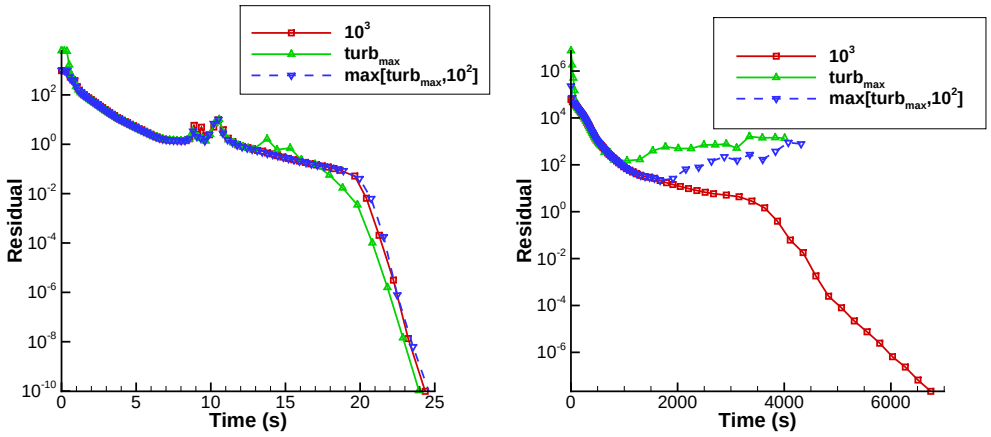
Modifications to the algorithm parameters discussed in this section do not affect the converged solution, as long as a converged solution is obtained, since the residual of the governing equations is not modified. Instead, these parameters affect how quickly a solution can be evolved from the initial guess to a steady state.

5.3.1 Equation scaling

The addition of the Spalart-Allmaras turbulence model into the linear system of equations can cause unpredictable behaviour of the linear solver. If not scaled properly, the large entries in the linearization of the turbulence model can result in the linear solver only reducing the residual of the turbulence equation; all other residuals are either converged very little, or even diverged. In addition, the turbulence quantity can take on values several orders of magnitude larger than the mean-flow variables, resulting in much larger updates during the nonlinear iterations. A scaling strategy for the turbulence model, as it relates to the linear system, was described in §4.2.6. However, the parameter that normalizes the turbulence model equation, denoted $\tilde{\nu}_{\max}$, has not yet been specified. Chisholm and Zingg [19] have shown a constant value of 10^3 works well for two dimensional flows. In the following study, alternative approaches will be investigated, accounting for iteration-specific values of the largest turbulence quantity magnitude. The following scaling values were implemented:

1. $\tilde{\nu}_{\max}$ is set to a constant 10^3,
2. $\tilde{\nu}_{\max}$ is set to $\tilde{\nu}_{\max}^{(n)}$,
3. $\tilde{\nu}_{\max}$ is set to $\max\left[\tilde{\nu}_{\max}^{(n)}, 10^2\right]$,

where $\tilde{\nu}_{\max}^{(n)}$ is the global maximum turbulence quantity at the nonlinear iteration n.



(a) Case 2B (b) Case 3B

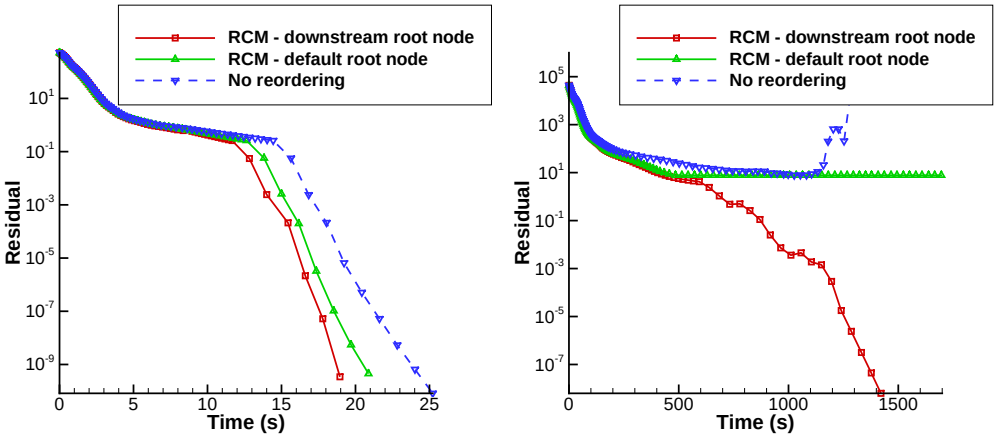
Figure 5.4: Convergence histories for equation scaling study

The effect of the scaling quantities was investigated using cases 2B and 3B. Figure 5.4 presents the convergence histories of all runs. The two-dimensional case shows very little sensitivity to the scaling approach used, with all scaling approaches resulting in nearly identical convergence. For the three-dimensional case, however, the convergence is greatly affected by the scaling approach; the use of the iteration-specific value of maximum $\tilde{\nu}$ introduces substantial variance into the scaling of the system, as this value can change by large amounts between two successive iterations. Additionally, using smaller values for scaling the turbulence model during the initial phases of convergence results in faster growth of the turbulence variable. This growth destabilizes the solution method, resulting in divergence, as can be seen with scaling approaches 2 and 3. Using the constant scaling value of 10^3 results in full convergence. Of note, modifying scaling approach 3 to use a minimum value of 10^3 also results in full convergence, further pointing to the need for larger scaling values in the initial stages of convergence.

The default value for $\tilde{\nu}_{\max}$ was selected to be 10^3. It provides good convergence characteristics, especially for three-dimensional flow, and requires substantially less computational time than the more complex approaches. In particular, it does not require the determination of a global maximum for $\tilde{\nu}$ at each nonlinear iteration.

5.3.2 Node ordering

The node ordering in the system can have a significant impact on the convergence of the linear and nonlinear systems. Pueyo [79] investigated several ordering strategies in the context of a two-dimensional solution algorithm, and found the ordering of the nodes, which affects the



(a) Case 1E (b) Case 4C

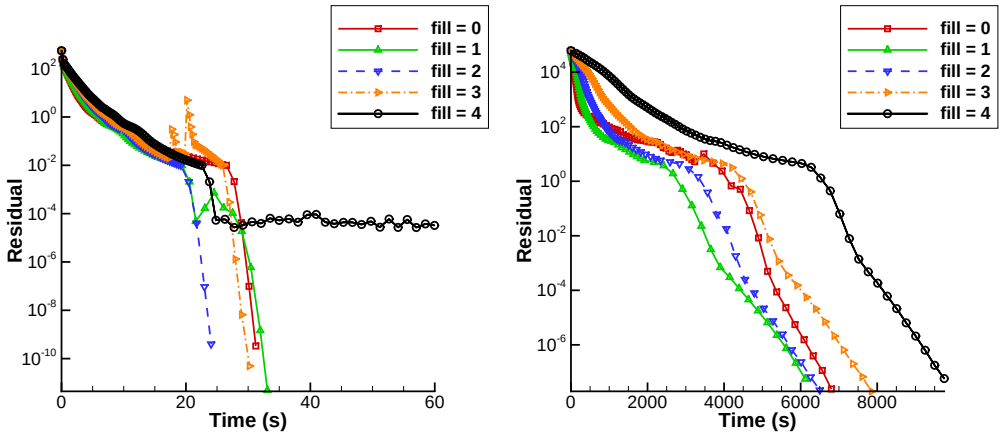
Figure 5.5: Convergence histories for node reordering study

form of the linear system, to be crucial. In particular, the reverse Cuthill-McKee was found to be very effective as long as a suitable root node is chosen [19].

Three node ordering approaches were investigated: no reordering, reverse Cuthill-McKee (RCM) reordering with no special root node selection*, and RCM reordering with downstream root node and downstream ordering bias. Of note, the natural ordering of the nodes in the algorithm strives to minimize the bandwidth of the system by indexing nodes in loops through coordinate directions with fewest nodes first. This approach does not account for direction of flow or grid orientation.

To illustrate the effects of node ordering, two cases of substantially different grid complexities are investigated. The first, representing a "well-behaved" two-dimensional grid, is case 1E, while case 4C provides a more complicated, three-dimensional grid. Figure 5.5 presents the convergence plots. Case 1E converged well for all ordering strategies, owing to the fact that the initial grid orientation and index values resulted in a default ordering sufficiently similar to the RCM strategy. Only marginal convergence improvement is seen with the RCM reordering strategy with downstream root node selection. The three-dimensional case, on the other hand, gives a stark illustration of why node ordering and the choice of root node is critical for efficient solution of the linear system, especially for complex three-dimensional grids with varying block orientations. While the two-dimensional case converged for all ordering strategies, only the RCM reordering with downstream root node and ordering bias obtained a steady-state solution in the three-dimensional case.

*The root node is set to the node with the lowest index prior to reordering



(a) Case 2A (b) Case 3B

Figure 5.6: Convergence histories for approximate-Newton phase fill level study

5.3.3 Approximate-Newton phase

The approximate-Newton phase, which comprises the start-up phase for the pseudo-transient continuation strategy, provides several user-tunable parameters. Changing their values can have a significant effect on the convergence of the system, not only affecting the time required to reach a steady-state solution, but also whether the start-up phase provides a suitable initial iterate for the inexact-Newton phase. This section presents the parametric studies for several values responsible for the effectiveness of the approximate-Newton phase.

Preconditioner level of fill

The level of fill used in the factorization of the preconditioner affects the robustness of the solution process. High fill levels provide a better representation of the preconditioner matrix, but require more memory and computational time, while lower fill levels require less memory at the expense of robustness. The goal is to select a fill level that enables efficient solution of the linear system in terms of time and memory.

The current algorithm allows for separate specifications of factorization fill levels for the approximate- and inexact-Newton phases. This section examines the effect of fill level on start-up, using cases 2A and 3B. Figure 5.6 shows the convergence histories of the cases considered, with fill levels between 0 and 4.

As with previous parameter studies, the two-dimensional case is easier to converge due to the smaller system size and reduced geometry and grid complexity. The stall in convergence observed for a fill level of 4 is related to the turbulence model; the mean-flow equations converge fully, but the overall residual stalls due to high turbulence model residual values

at a few nodes in the grid. This behaviour has been observed for other cases, but its cause is presently unknown. The three-dimensional case exhibits very good robustness at all fill levels. The results show that, for the startup phase, fill levels of 1 and 2 result in the fastest overall convergence. Lower and higher fill levels result in much slower convergence times, either due to improper initial iterates for the inexact-Newton phase (with a fill level of 0), or too much computational time spent in the factorization of the preconditioner (with fill levels above 2).

Preconditioner freezing

Previous work by Pueyo [79] and Hicken [34] has shown that freezing the preconditioner update during the start-up phase can provide a significant reduction in the solution time, with the work of Hicken focusing on a parallel three-dimensional solution algorithm for the Euler equations. Lagging the preconditioner update and factorization eliminates significant computational time during the start-up phase. Experience with the current solver has shown very little benefit to freezing the preconditioner, instead often resulting in residual divergence. This is due to the large solution changes, especially in the turbulence variable, during the early stages of convergence. Hence, the frozen preconditioner quickly becomes ineffective, or even detrimental to the solution process.

Time-marching parameters

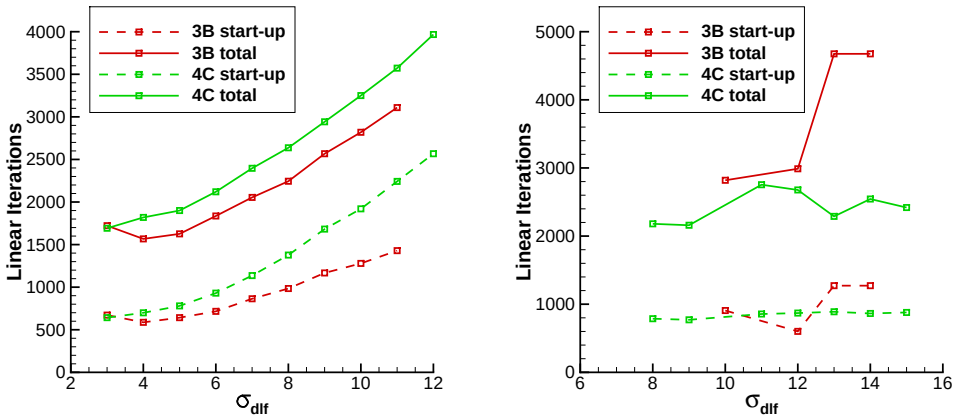
Perhaps the most important parameters affecting the efficiency and robustness of the approximate-Newton phase are related to the time step determination, as defined in (4.6). The initial time step value, governed by the parameter a, has to be set to a maximum of 0.001 due to small off-wall spacing. Any larger values will result in nonphysical updates to pressure and density during the first nonlinear iteration, causing the algorithm to abort. This value is appropriate for grids with an approximate off-wall spacing of 1×10^{-6} and must be decreased for grids with smaller off-wall spacings.

Cases 1A and 3B are used in this parameter study. The parameter of concern here is the value of b, which determines the rate of growth of the time step between nonlinear iterations. Table 5.4 presents a summary of the convergence characteristics of the cases run, showing the time spent in the start-up phase, overall solution time, and the total number of linear and nonlinear iterations required. Case 1A was run with 8 processors, while case 3B used 192 processors. Cases labeled with DNC either diverged or resulted in residual stall in the inexact-Newton phase, while cases labeled DNF ran out of time or reached the maximum number of nonlinear iterations before convergence could be achieved. The two-dimensional

Table 5.4: Convergence characteristics for time step parameter study

Case	b	Start-up time (s)	Total time (s)	Linear iterations	Nonlinear iterations
1A	1.05	DNF	-	-	-
	1.10	20.9	25.8	1189	126
	1.15	18.3	22.1	1147	101
	1.20	16.8	20.6	1150	88
	1.25	16.0	19.7	1155	80
	1.30	15.6	19.3	1145	75
	1.40	14.4	18.1	1128	69
	1.50	14.0	17.7	1115	66
	1.60	DNC	-	-	-
3B	1.05	DNF	-	-	-
	1.10	4025	7814	2062	148
	1.15	3011	6763	1912	110
	1.20	2549	6259	1873	92
	1.25	2629	6190	1907	81
	1.30	2541	6109	1918	74
	1.35	2329	6609	2054	71
	1.40	2371	6415	2102	68
	1.50	2395	6442	2096	63

case shows increased convergence speed with increasing b, until too high a value is reached at 1.6, resulting in divergence. The three-dimensional case provides a clearer optimum value. The tabulated results demonstrate an important fact about the selection of the time stepping parameters, especially b. Too low a parameter will result in unnecessarily slow convergence, while too high a parameter provides fast initial convergence, but slower overall solution time. This is due to the destabilizing effect a fast time step ramping has on the solution, especially the turbulence quantity, leading to slower initial convergence in the inexact-Newton phase. Hence, a parameter that provides a good compromise between start-up speed and overall convergence time should be selected. For the three-dimensional case, the value of b that best fits this requirement is 1.30. As discussed further in Appendix D, the time-marching parameters have to be decreased slightly when the matrix dissipation model is used.



(a) Scalar dissipation model (b) Matrix dissipation model

Figure 5.7: Convergence behaviour for dissipation lumping factor

Dissipation lumping

Dissipation lumping, or the amount of additional second-difference dissipation that is added to the preconditioner in lieu of fourth-difference dissipation, can have a significant impact on convergence. Dissipation lumping decreases the size of the computational stencil for the preconditioner matrix by eliminating next-to-nearest neighbouring node information introduced by fourth-difference dissipation, instead lumping some of the contribution into the compact second-difference dissipation operator. The lumping parameter, σ_{dlf}, controls the amount of lumping, but has to be selected carefully to ensure the preconditioner remains a good approximation of the flow Jacobian matrix. The details of the approach are presented in §4.2.2. While this value affects the entire solution process, its effect is more pronounced in the start-up phase, since the preconditioner matrix is also used for the matrix-vector product in the linear system solution.

Two transonic three-dimensional cases were considered when studying both the scalar and the matrix dissipation model, namely 3B and 4C. Figure 5.7 presents a comparison of the linear iterations and time required to obtain fully converged solutions, as well as the respective values for the start-up phase. The results clearly show an optimal value at which fastest convergence is achieved for the scalar dissipation model. A good choice for value of σ_{dlf} would lie in the range of 4 to 6, with 5 selected as the default. This value provides fast convergence while avoiding the extreme limits of convergence capability for the algorithm. Dissipation lumping values outside of the range presented in the plot resulted in divergence of the algorithm. The effect of the lumping factor is more pronounced in the start-up phase, demonstrated by the nearly constant number of linear iterations separating the total and

start-up linear iteration counts.

As expected, the matrix dissipation model presents less robust behaviour. In the range of dissipation lumping values considered, only a small band resulted in converged solutions. The performance of the algorithm is less well-behaved, with unexpected jumps in the total number of linear iterations. Additionally, solutions with some σ_{dlf} values within the presented ranges diverge, such as with $\sigma_{\text{dlf}} = 11$ for case 3B and $\sigma_{\text{dlf}} = 10$ for case 4C. The cause of this behaviour is not fully understood. However, two trends can be identified: first, low values of the lumping factor result in divergence during the approximate-Newton phase, likely due to the reduced dissipation and diagonal dominance of the matrix dissipation model; second, the behavior of the algorithm with larger lumping factors appears to be case-dependent, with case 3B experiencing a substantial jump in the number of linear iterations, especially for the inexact-Newton phase, while case 4C presents a flatter linear iteration curve. A value between 10 and 12 for σ_{dlf} is recommended for use with the matrix dissipation model.

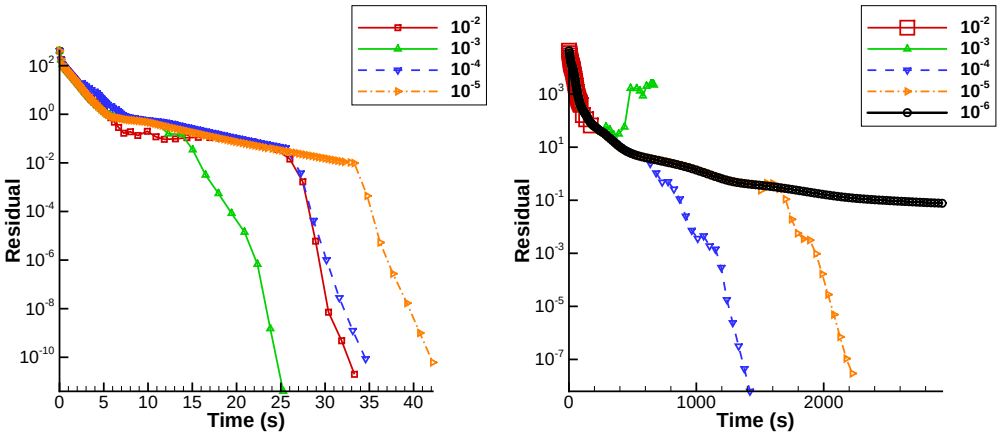
Linear residual tolerance, η_n

An important detail affecting the effectiveness of an approximate linear solution algorithm is the specification of the linear residual tolerance, η_n. If this value is set too high, the solution updates produced from the linear solutions will be inaccurate, resulting in slower nonlinear convergence or divergence. If this value is set too low, computational time will be wasted in obtaining unnecessarily accurate solution updates. Hence, a balance has to be achieved.

The uniform nature of the initialized flow and a simplified linear system allow for very rapid linear convergence during the approximate-Newton phase, often reducing the linear residual be several orders of magnitude in fewer than 10 linear iterations. Additionally, by using the approximate Jacobian matrix for the matrix-vector products, the cost of linear iterations is smaller than during the inexact-Newton phase. This, coupled with the fact that the total number of linear iterations performed during the start-up phase corresponds to roughly a third of the total required for a flow solution, minimizes the impact of changing the value of η_n. Values between 0.1 and 0.05 result in nearly identical convergence, in terms of both linear iterations and time. The conservative choice of 0.05 is used as the default for the approximate-Newton phase.

5.3.4 Residual reduction for switch to inexact-Newton phase

A critical component of the PTC strategy is determining when the switch from the approximate- to inexact-Newton phase should occur. The parameter responsible for this is the residual



(a) Case 2E (b) Case 4C

Figure 5.8: Convergence histories for approximate-Newton residual reduction study

reduction, defined as the relative decrease of the global nonlinear residual from its initial value.

Case 4C is used to investigate the solver performance for three-dimensional flows, while case 2E is used in two dimensions. The residual reduction is varied from 1×10^{-2}, a value commonly used for inviscid and laminar flows, to 1×10^{-6}. These values represent residual reductions of 2 and 6 orders of magnitude, respectively. Figure 5.8 presents the convergence histories for both cases. Case 2E is able to converge for all residual reduction values, but the lowest value results in a flat residual profile for the inexact-Newton phase before the solution fully converges. The three-dimensional case diverges for the highest residual reduction values (10^{-2}, 10^{-3}). (The symbol for the 10^{-2} residual convergence was enlarged to clearly show algorithm termination.)

Due to the presence of small off-wall spacing values, which result in large residual magnitudes for the turbulence model (contributed to mainly by the production and destruction source terms), the initial residual for the RANS equations is typically several orders of magnitude larger than that seen for the Euler or laminar Navier-Stokes equations. However, this residual decreases rapidly in the first few iterations before a more gradual convergence profile is seen. Hence, using residual reduction values that switch between the phases too quickly leads to divergence or poor inexact-Newton phase performance, as the solution has not evolved enough to serve as a good initial guess for the second solution phase. On the other hand, remaining in the start-up phase too long wastes computational time. A residual reduction of 1×10^{-4} provides a good balance of robustness and speed, and has been observed to work well for a wide range of two- and three-dimensional flow solutions.

5.3.5 Inexact-Newton phase

Once a suitable initial iterate for the inexact-Newton phase is provided from the approximate-Newton phase, the goal is to reach a steady-state solution as quickly as possible. This phase provides fewer tunable parameters than the start-up phase, almost exclusively related to the robustness and accuracy of the linear solution.

Time-marching parameters

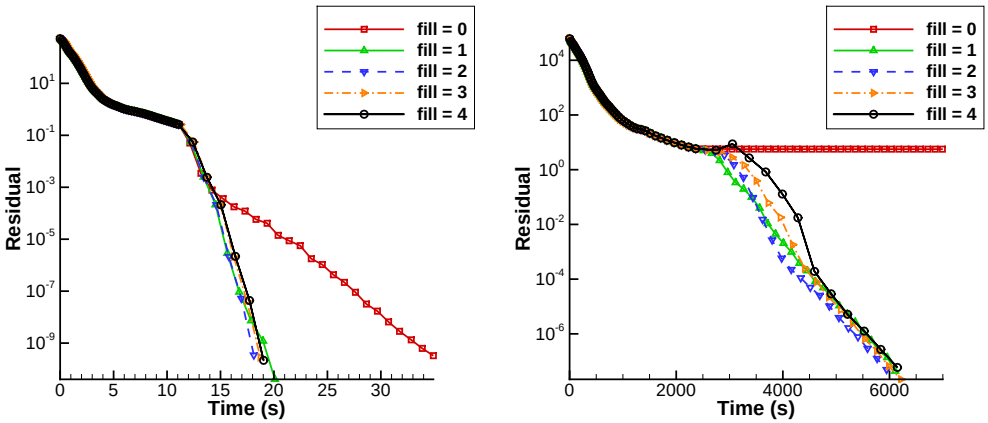
The purpose of the time ramping in the inexact-Newton phase of the solution algorithm is to increase the time step as quickly as possible to recover the full Newton method without destabilizing the solution. For this purpose, the geometric relationship outlined in (4.8) provides appropriate time step increases between nonlinear iterations. This approach is relatively insensitive to the tunable parameter β within a small range, with values between 1.5 and 2.0 providing nearly identical convergence characteristics.

Preconditioner level of fill

As with the approximate-Newton phase, the level of fill applied during the factorization of the preconditioner can have a substantial impact on the convergence characteristics of the flow solution algorithm during the inexact-Newton phase.

The results obtained for cases 1E and 3B are summarized in Figure 5.9 and Table 5.5. As expected, there exists a trade-off in the selection of the optimum value. While lower fill levels require less memory and computational time for the optimization, they also result in more linear iterations, sometimes leading to divergence. Higher fill levels result in better convergence characteristics, but require more memory and computational time for the factorization.

From these data, the fastest solution os obtained with a fill level of 2. However, while a noticeable difference exists in the nonlinear and linear iteration counts for the three-dimensional case, the total computational time varies only slightly. Hence, the choice of fill level can instead be motivated by the complexity of the problem being solved and the amount system memory available (higher fill levels possess a larger memory footprint). Importantly, larger computational grids often require higher fill levels during the inexact-Newton phase due to the increased size of the linear system. Using fill levels of 2 or less often results in poor linear solver convergence and nonlinear residual stall. The recommended fill level for the inexact-Newton phase is therefore 3, providing a balance between memory requirements and linear solver performance.



(a) Case 1E (b) Case 3B

Figure 5.9: Convergence histories for inexact-Newton phase fill level study

Table 5.5: Convergence characteristics for inexact-Newton phase fill level study

Case	level of fill	Total time (s)	Linear iterations	Nonlinear iterations
1E	0	34.9	2115	86
	1	20.1	1065	71
	2	18.1	925	69
	3	18.7	925	69
	4	19.0	925	69
3B	0	DNC	-	-
	1	6106	2614	79
	2	5945	2264	74
	3	6214	2054	71
	4	6147	1699	66

Frechet-derivative perturbation parameter

In order to avoid explicitly forming the flow Jacobian, which is required for matrix-vector products in the inexact-Newton phase, the forward finite-difference approximation shown in (4.9) can be used. For this approach to be effective, a perturbation parameter, δ, has to be selected. As with most finite-difference approximations, the value of this parameter will have a significant impact on the accuracy of the derivative. Selecting a value that is too high will result in inaccurate derivative approximations, while excessively low values will result in subtractive cancellation errors, also impacting the accuracy.

Table 5.6: Convergence characteristics for Frechet-derivative perturbation parameter study

Case	δ	Total time (s)	Linear iterations	Nonlinear iterations
2A	1×10^{-7}	24.2	1076	81
	1×10^{-8}	23.1	1027	80
	1×10^{-9}	23.1	1027	80
	1×10^{-10}	23.2	1027	80
	1×10^{-11}	23.1	1027	80
	1×10^{-12}	23.2	1027	80
	1×10^{-13}	23.1	1027	80
	1×10^{-14}	23.0	1021	80
	1×10^{-15}	24.1	1073	81
3B	1×10^{-8}	DNC	-	-
	1×10^{-9}	8268	2761	83
	1×10^{-10}	7187	2458	78
	1×10^{-11}	7668	2528	79
	1×10^{-12}	7966	2668	81
	1×10^{-13}	DNC	-	-

In order to determine the optimal value of this parameter for RANS computations, the convergence characteristics of cases 2A and 3B are examined for a range of δ values. The results are summarized in Table 5.6. Since the start-up phase for all runs remains unchanged, any differences in performance are purely due to the impact of δ on the inexact-Newton phase performance. The perturbation parameter was varied between 1×10^{-8} and 1×10^{-13}.

The results show that the two-dimensional flow solution is insensitive to the value of the perturbation parameter; any error in the matrix-vector approximation is not enough to cause convergence difficulties. For a significant range of perturbation parameters (10^{-8} to 10^{-13}), the linear solver is stopped by the maximum linear iteration limit. Hence, even though the linear solver is not achieving identical convergence for that range of perturbation parameter values, sufficient linear convergence is achieved to lead to similar nonlinear convergence. This behaviour is also affected by the small number of nonlinear iterations needed for the inexact-Newton phase; only 5 or 6 iterations are required to converge the system fully, which reduces the effect of the Frechet-derivative perturbation parameter on the overall solution time and iterations. It is only at the extremes of the range of perturbation values that any

change in convergence is observed.

The three-dimensional case is more sensitive to the value of δ. Unlike with the smaller two-dimensional case, the linear solver is not able to achieve a value close to the required level of convergence, again being cut off by the maximum number of linear iterations. With the increased final residual, the variations in the cut-off converged updates are more pronounced, and will therefore have a noticeable effect on the overall convergence of the algorithm. The most efficient solutions for the transonic flow are achieved at a δ value of approximately 1×10^{-10}, which lies within the region of flat convergence characteristics for the two-dimensional flow.

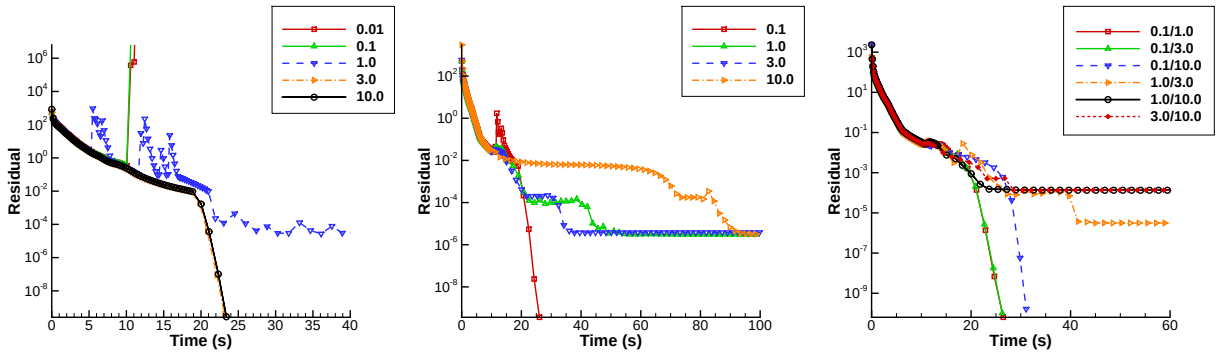
Linear residual tolerance, η_n

In order to approach quadratic convergence, Eisenstat and Walker [23] proposed a strategy for varying the value of the linear solution tolerance, η_n. Even though using a constant value of η_n will not result in quadratic convergence, Pueyo [79] demonstrated that this approach produces converged flow solutions in less overall computational time. For this reason, η_n is set to 0.01 for this phase. In practice, however, this tolerance is rarely achieved before the maximum linear iteration count is reached, terminating the linear solution. This is especially noticeable for three-dimensional solutions, with smaller than expected residual drops between outer iterations (an example of this can be seen in Figure 5.8). Based on extensive testing, the cause of this behaviour appears to be related to the complexity of the flow being simulated and the size of the system being solved, with larger grids exhibiting slower linear convergence in the inexact-Newton phase. Regardless of the actual rate of linear convergence for a specific case, the value of the linear residual tolerance is selected to achieve as fast a convergence as possible for each case.

5.3.6 Initialization of turbulence variable

An important consideration when performing turbulent flow simulation is the initialization of the turbulence variable. Two competing factors motivate the selection of this value. On one hand, initializing the turbulence variable to a value that is too low will result in unpredictable and often slow initial convergence due to the self-tripping of the turbulence model. Larger values, while providing a better starting point for fully turbulent flow solves, can cause problems for explicitly tripped flows, as the solution may remain fully turbulent upstream of the specified trip point.

For fully turbulent flows, case 2A is used to demonstrate the effect various initial values



(a) Case 2A - constant $\tilde{\nu}_{\text{init}}$ (b) Case 1D - constant $\tilde{\nu}_{\text{init}}$ (c) Case 1D - location-based $\tilde{\nu}_{\text{init}}$

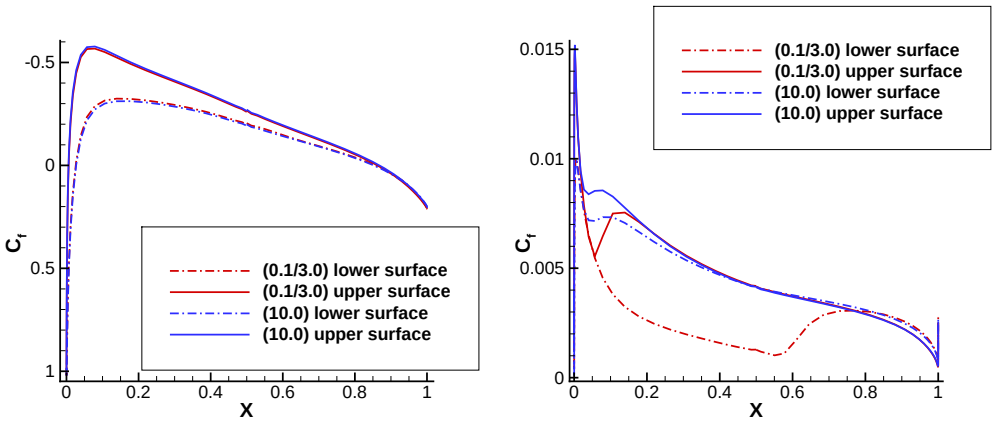
Figure 5.10: Convergence histories for initialization of $\tilde{\nu}$

of $\tilde{\nu}$ have on the convergence process. In all cases, the farfield value of $\tilde{\nu}$ is 3.0. Figure 5.10(a) presents the convergence histories, highlighting the importance of the initial value of $\tilde{\nu}$. Selecting a value that is too low results in divergence or convergence stall, while a value greater than or equal to 3.0 results in the expected convergence to steady-state. The cause of this behaviour can be traced to the lack of turbulence in the initial flow field if $\tilde{\nu}$ is initialized below 3.0. The use of the f_{t2} suppression term in the turbulence model, which possesses a boundary of attraction of $\chi = 0.6$, will force areas of the flow with small initial turbulence quantities to revert to laminar flow. This prevents the solver from obtaining a fully turbulent steady-state solution.

Case 1D represents an explicitly tripped flow solution, with the upper and lower surface trip locations set at 10 and 65% chord, respectively. Unlike the fully turbulent case, location-based initialization of $\tilde{\nu}$ is also investigated, with different initial values used upstream and downstream of the trip location. In all cases, the farfield value of $\tilde{\nu}$ is 0.1. As expected, the use of a constant initial value of $\tilde{\nu}$ results in convergence problems for the explicitly tripped flow solutions, as shown in Figure 5.10(b). In fact, the only case that converged for this set used the value of $\tilde{\nu}_{\text{init}} = 0.1$, with some instabilities still present in the convergence. All higher values resulted in a stall of the nonlinear residual. When location-based initialization is used, seen in Figure 5.10(c), better convergence characteristics are observed as long as the value of $\tilde{\nu}$ upstream of the trip location is set to 0.1, with fastest convergence provided by setting $\tilde{\nu}_{\text{init}}$ to (0.1/1.0) or (0.1/3.0)[†].

The constant value initialization for the explicitly tripped flow provides an important insight into the behaviour of the turbulence model. By comparing the surface distributions

[†]For location-based initialization, the notation (i/j) signifies values of $\tilde{\nu}$ (upstream/downstream) of the trip location



(a) Coefficient of pressure (b) Coefficient of skin friction

Figure 5.11: Coefficients of pressure and skin friction for case 2D

of C_p and C_f of the stalled $\tilde{\nu}_{\text{init}} = 10.0$ against the fully converged $\tilde{\nu}_{\text{init}} = (0.1/10.0)$ solution (Figure 5.11), a clear difference is visible: while the location-based initialization results in a flow that transitions from laminar to turbulent profiles near the specified trip locations, the case initialized with the higher value of $\tilde{\nu}$ appears to possess a C_f profile of a fully turbulent flow. This observation underlines the importance of properly specifying initial flow conditions based on flow type, either fully turbulent or explicitly tripped. Minimal differences are observed in the profiles for C_p.

While the results presented here are for two-dimensional cases, the same conclusions apply to three-dimensional flow simulations.

5.4 Summary of Default Parameters

The algorithm optimization tests presented in this chapter allow for the compilation of a set of default discretization choices and solution algorithm parameter values. While the derivation of the parameters is based on a small selection of cases, the resultant values have been applied successfully to a wide range of flow problems. The following are the default values for the parameters investigated as part of the algorithm optimization:

Turbulence model variant: standard variant

Turbulence model dissipation: first-order upwinding for advection terms[‡]

[‡]For higher-order codes, fourth-difference dissipation approach should be investigated due to order of accuracy limitations of first-order upwinding strategy

Off-wall distance calculation: surface-based calculation

Equation scaling for turbulence model: constant value of 10^3

Node ordering: RCM with downstream root node and ordering bias

Approximate-Newton phase preconditioner fill level: 2

Approximate-Newton phase time-marching parameters: $a = 0.001$, $b = 1.30$

Residual reduction for switch to inexact-Newton phase : 4 orders of magnitude (1×10^{-4})

Dissipation lumping factor: 5.0 for scalar dissipation, 10.0-12.0 for matrix dissipation

Inexact-Newton phase time-marching parameter: $\beta \in [1.5, 2.0]$

Inexact-Newton phase preconditioner fill level: 3

Frechet-derivative perturbation parameter: 1×10^{-10}

Linear solution tolerance: $\eta_n = 0.05$ for approximate-Newton phase, 0.01 for inexact-Newton phase

Initial $\tilde{\nu}$ value: for fully turbulent flow, use 3.0; for explicitly tripped flow, use 0.1 upstream of trip location and 3.0 downstream of trip location

It should be stressed that while the above represent a solution algorithm configuration that is both efficient and robust for a wide range of cases, instances will arise where modification to the settings will not only be beneficial, but also required. Many of these special cases were discussed in the preceding sections, and should serve as a starting point for improving the convergence characteristics of the solver for difficult cases. In particular, cases utilizing grids with off-wall distance much finer than 1×10^{-6}, the residual reduction for terminating the approximate-Newton phase will likely have to be decreased further, since the initial residual possesses a larger magnitude. The initial time step value will also have to be decreased for these grids.

Chapter 6

Results

The previous section presented the algorithmic optimization necessary to obtain an efficient and robust flow solution algorithm, addressing both the discretization of the governing equations as well as the solution of the discrete set of equations that results from applying the spatial discretization. The results presented in this section strive not only to address the accuracy of the current flow solution algorithm, but also to highlight its applicability to the solution of flows around geometries of engineering significance. Hence, a series of verification and validation cases will be presented, followed by a set of flow solutions around two- and three-dimensional geometries of practical interest, such as high-lift configurations, wing-body and non-planar geometries, as well as flows with explicitly specified laminar-to-turbulent transition regions. Finally, the parallel scaling characteristics of the algorithm are presented.

In particular, results from the participation of the current algorithm in the 5th Drag Prediction Workshop (DPW5) are included. The workshop provides a platform for the evaluation and comparison of leading flow solution algorithms and serves as an important verification and validation tool for flow problems of engineering interest.

When computing flow around three-dimensional geometries, a symmetry boundary condition is used at the root of the wing or wing-body configuration.

6.1 Verification and Validation

In order to verify the implementation and accuracy of the current algorithm, several verification and validation cases are shown. Making use of the extensive data available on the NASA Turbulence Modeling Resource (TMR) website [1], the current algorithm, named DIABLO, is verified against well-established and extensively validated finite-volume solvers FUN3D and CFL3D for the flat plate and bump-in-channel flows. As detailed on the TMR website, CFL3D is a cell-centered structured grid code, while FUN3D is a node-centered unstructured grid code. Both codes make use of Roe's flux difference splitting and a UMUSCL upwind approach, along with first-order upwinding for the advective terms of the turbulence model.

Finally, both codes were run with the full Navier-Stokes equations, bypassing the default thin-layer approximation in CFL3D. Further details can be found on the website.

Additionally, validation studies were performed with flow around a NACA0012 airfoil and the ONERA M6 wing, where comparison to experimental results could also be made.

6.1.1 2D zero-pressure-gradient flat plate

The first verification case considered is the two-dimensional turbulent flow over a flat plate. The flow conditions for this case are

$$M = 0.20,\ Re = 5 \times 10^6,\ T_\infty = 540^\circ \text{ R}.$$

The value of Re is based on a reference length, and the flat plat has a length of 2.0 reference units. Three grid levels are considered, with node numbers ranging from 137×97 to 545×385. Successively finer off-wall spacing values are used, the finest of which is 5.0×10^{-7} reference units. The coarser meshes were created by successively removing every second node in each coordinate direction from their respective finer counterpart. The grids provide average approximate y^+ values between 0.1 and 0.4, depending on the grid level. The data provided on the TMR website allows for a detailed comparison of the results obtained with the current algorithm with those obtained with CFL3D and FUN3D. Both scalar and matrix dissipation models were tested with this case.

Figure 6.1 shows the behavior of drag as the grid is refined, where N denotes the number of nodes in the grid. As a result of plotting versus N^{-1}, second-order behaviour should tend toward a straight line. Both dissipation models tend toward the same value of C_d as the grids are refined. However, the scalar dissipation model approaches this value in a non-monotonic manner from above, while the matrix dissipation model closely follows the trend of FUN3D, approaching from below. Both models are tending towards a grid-converged value of drag that lies between the trends of CFL3D and FUN3D.

Comparisons of the coefficient of skin friction, C_f, maximum turbulent viscosity in the boundary layer, and the turbulent viscosity profile at a vertical section of the finest grid are presented in Figure 6.2. Since the scalar and matrix dissipation results on this grid level are nearly indistinguishable, only the matrix dissipation result is shown. Some data points are omitted for clarity. Each of the comparisons shows excellent correspondence between DIABLO and the other algorithms, with nearly identical distributions of all pertinent quantities.

Table 6.1 summarizes the estimated orders of convergence for the drag experienced by the flat plate, using both the scalar and matrix dissipation models. The calculation is based

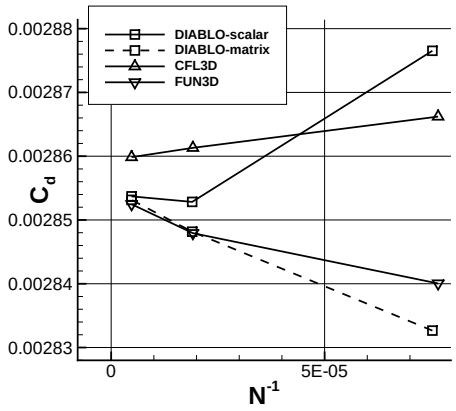
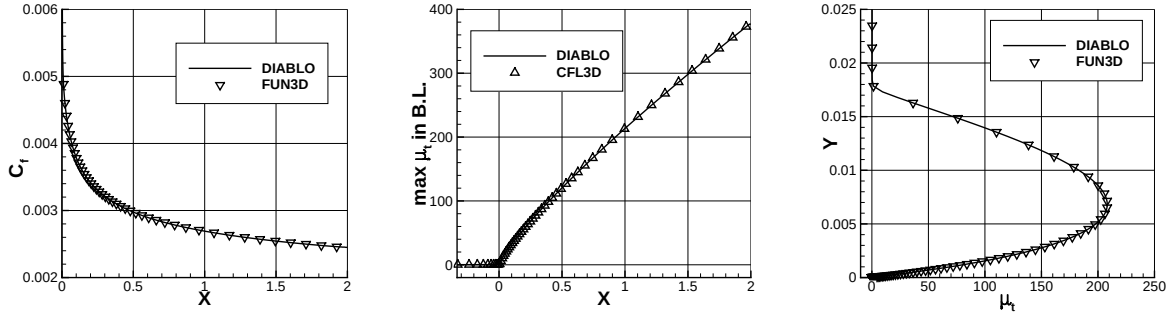


Figure 6.1: Grid convergence of drag for flow over a flat plate



(a) Skin friction

(b) Maximum μ_t in boundary layer

(c) μ_t at $x = 0.97$

Figure 6.2: Flow solution comparisons for flow over flat plate

on the procedure outlined by Baker [7] and Celik *et al.* [17]. CFL3D and FUN3D results are also available for comparison.

As can be seen from the data, the scalar dissipation solutions produce a non-monotonic set of drag values, preventing any calculations for the order of convergence. Matrix dissipation, however, shows better behaviour, with an estimated order of convergence of 1.64. This, along with the value of C_d^*, compares well to CFL3D and seems better than FUN3D.

It is not clear why the scalar dissipation model results in a non-monotonic sequence of drag values.

It is also worth mentioning that this case, along with the bump-in-channel case, required special pressure boundary conditions in order to match with the case description on the TMR website. These conditions were imposed with a standard injection method, bypassing the SBP-SAT discretization penalty terms. It was found that the use of SATs for the inflow and outflow boundaries resulted in a small discrepancy between the farfield values obtained

Table 6.1: Grid convergence characteristics for flat plate flow

Algorithm	C_d	
	order, p	C_d^*
DIABLO - scalar	NM	–
DIABLO - matrix	1.64	2.8555×10^{-3}
CFL3D	1.75	2.8593×10^{-3}
FUN3D	0.80	2.8586×10^{-3}

in the current algorithm and those reported by CFL3D and FUN3D. While this discrepancy was small, the inflow and outflow boundary treatment of the other numerical algorithms was adopted in order to demonstrate the effectiveness of the surface boundary SATs. Details of the implementation can be found in Appendix B.

6.1.2 2D bump-in-channel

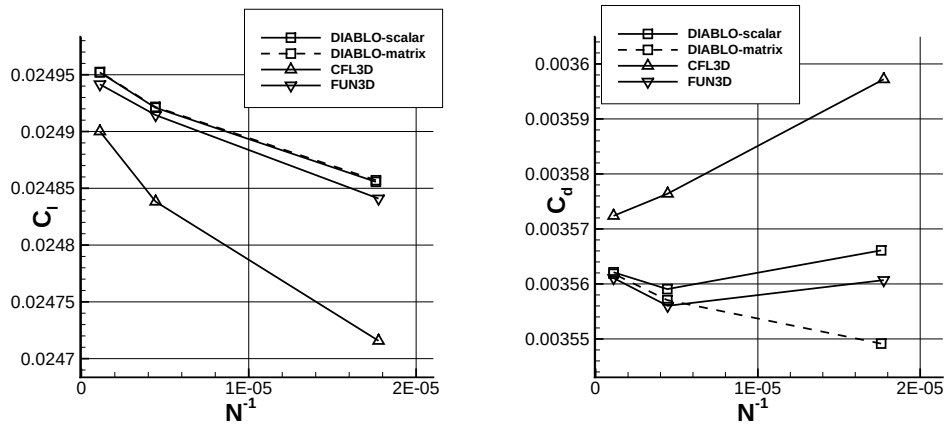
In order to verify the algorithm in a more complex flow regime where pressure gradients are present, the second case considered is the two-dimensional bump-in-channel flow. Three grid levels are considered, with node numbers ranging from 353×161 to 1409×641, with successively finer off-wall spacing values, the finest of which is 5.0×10^{-7} reference units (the bump has a length of 1.5 reference units). As with the previous case, the coarser grid levels were created by removing every second node in each coordinate direction from the finer grid level. The grids provide average approximate y^+ values between 0.06 and 0.23, depending on the grid level. The flow conditions for this case are

$$M = 0.20,\ Re = 3 \times 10^6,\ T_\infty = 540^\circ \text{ R}.$$

The Reynolds number is based on the reference unit length.

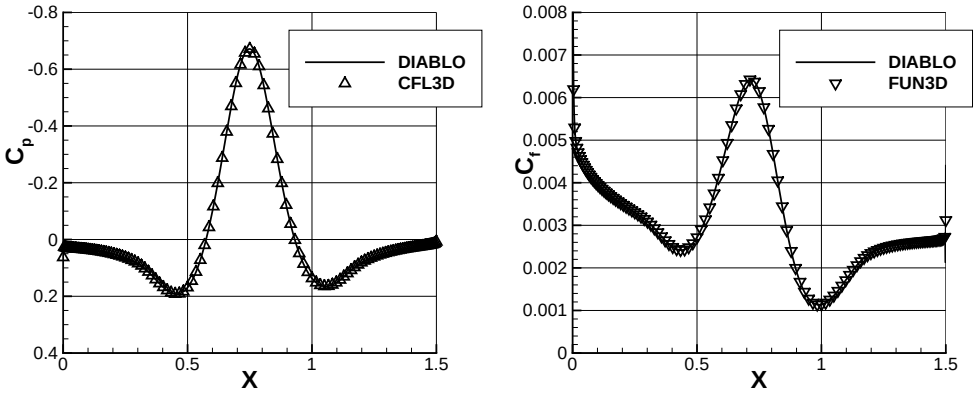
Figure 6.3 provides an overview of the grid convergence behaviour of lift and drag for the scalar and matrix dissipation models. For both quantities, the results obtained with DIABLO lie very close to those of the other solvers, with slightly better correspondence to the results of FUN3D. The grid convergence trends of DIABLO are in line with those of CFL3D and FUN3D.

Additionally, Figures 6.4 and 6.5 highlight the excellent correspondence between the current algorithm and the established solvers. This is evident not only for the coefficients of pressure, C_p, and friction, C_f, along the surface of the bump, but also for the value of μ_t in the boundary layer. This case verifies the implementation of DIABLO in a more complicated flow regime, with pressure gradients present in the flow due to the bump.



(a) Lift (b) Drag

Figure 6.3: Force coefficients for bump-in-channel flow

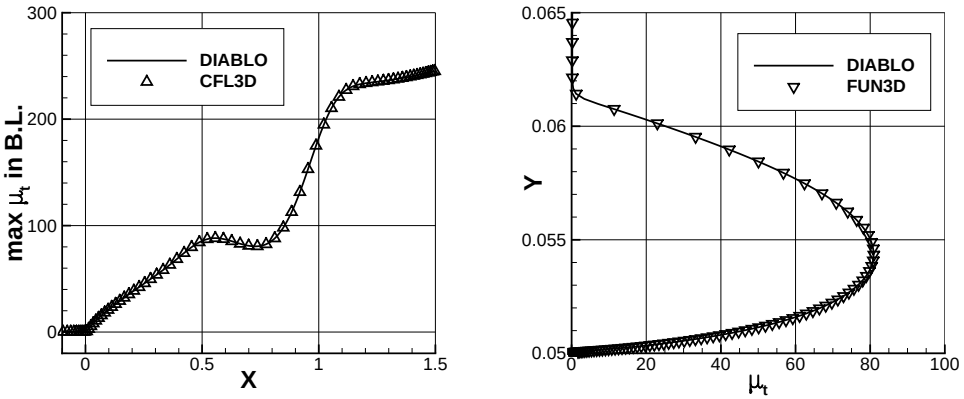


(a) Coefficient of pressure (b) Coefficient of friction

Figure 6.4: Surface coefficients for bump-in-channel flow

Table 6.2 summarizes the Richardson extrapolation results for for both lift and drag (and drag components) experienced by the geometry, using both the scalar and matrix dissipation models. CFL3D and FUN3D results are also available for comparison.

This case results in inconsistent behaviour for all three solution algorithms, with some quantities showing better than expected order of convergence, while others converge at a lower order. The results obtained with matrix dissipation compare favourably to the established flow solvers.



(a) Maximum μ_t in boundary layer (b) μ_t at $x = 0.75$

Figure 6.5: μ_t distribution comparisons for bump-in-channel flow

Table 6.2: Grid convergence characteristics for bump-in-channel flow

Algorithm	C_l		C_d	
	order, p	C_l^*	order, p	C_d^*
DIABLO - scalar	1.07	2.4980×10^{-2}	1.20	3.5086×10^{-3}
DIABLO - matrix	1.07	2.4980×10^{-2}	0.72	3.5693×10^{-3}
CFL3D	0.99	2.4963×10^{-2}	2.37	3.5714×10^{-3}
FUN3D	1.44	2.4957×10^{-2}	NM	–
	$C_{d,p}$		$C_{d,f}$	
	order, p	$C_{d,p}^*$	order, p	$C_{d,f}^*$
DIABLO - scalar	2.84	3.8140×10^{-4}	NM	–
DIABLO - matrix	2.72	3.8141×10^{-4}	1.14	3.1848×10^{-3}
CFL3D	2.87	3.8131×10^{-4}	0.89	3.1888×10^{-3}
FUN3D	2.61	3.8177×10^{-4}	0.64	3.1956×10^{-3}

6.1.3 NACA0012 airfoil

The final two-dimensional case considered is the flow over the NACA0012 airfoil. This case provides an opportunity not only to compare the current algorithm to CFL3D on a case of practical interest, but also to compare to the experimental data of Gregory and O'Reilly [29] (albeit at a lower Reynolds number of 3×10^6). The flow conditions are

$$M = 0.15,\ Re = 6 \times 10^6,\ T_\infty = 540°\text{R},\ \alpha = 0°,\ 10°,\ \text{and}\ 15°.$$

The finest grid consists of 1793×513 nodes, with an off-wall spacing of 4×10^{-7} chord units. This grid represents the finest grid level available for this test case on the TMR website, and provides an average y^+ of approximately 0.1. Two additional coarser grid levels were created by removing every second node in each coordinate direction. The coarsest grid consists of 449×129 nodes.

This study not only provides additional grid convergence assessment for the algorithm, but also allows for the comparison of the distributions of C_p and C_f on the surface of the airfoil. Experimental data are provided for C_p, and the CFL3D C_f data provided is limited to the upper surface of the airfoil. Surface coefficient comparisons are made with the results from the finest grid level only.

Figure 6.6 presents the comparisons for all three angles of attack for the scalar dissipation model, as the two models produced nearly indistinguishable results for this grid. Due to the size of the grid, data points are omitted from the CFL3D data for increased clarity. The results of DIABLO provide excellent correspondence to those of CFL3D, and line up well with the experimental results. This case provides verification through comparison with CFL3D for a range of flow conditions and a validation of the solver against experimental results, including high angles of attack where boundary-layer separation is present.

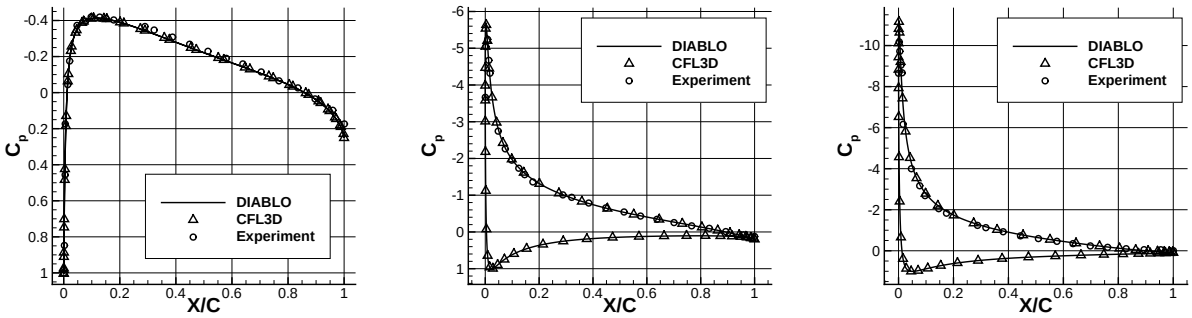
The lack of a complete set of numerical results from CFL3D precluded the use of Richardson extrapolation on the force coefficients for that solver; hence, only results from DIABLO are presented. Tables 6.3 to 6.5 present the grid convergence behaviour for both dissipation models on the three finest grid levels provided on the TMR website. Unlike the flat plate and bump-in-channel cases, this case allows for the use of the standard boundary condition implementation (SATs on all boundaries). The grid convergence results for the flow around the NACA0012 airfoil highlight the expected behaviour of the algorithm. With few exceptions, a grid convergence order of approximately 2 can be observed.

For completeness, Figure 6.7 presents the grid convergence of the coefficients of lift and drag for all three angles of attack. Partial data is also provided for CFL3D and FUN3D.

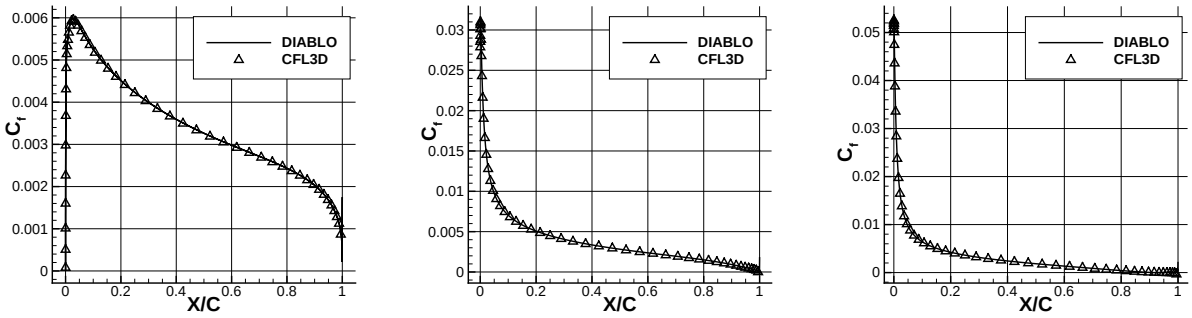
6.2 Three-Dimensional Steady-State Flow Solutions

6.2.1 ONERA M6 wing

Two cases were considered for the ONERA M6 wing. The first is the well known transonic flow over the wing, for which the experimental data of Schmitt and Charpin are available [88]. The flow conditions are



(a) C_p at $\alpha = 0°$ (b) C_p at $\alpha = 10°$ (c) C_p at $\alpha = 15°$



(d) C_f at $\alpha = 0°$ (e) C_f at $\alpha = 10°$ (f) C_f at $\alpha = 15°$

Figure 6.6: C_p and C_f comparison for NACA0012 flows

Table 6.3: Grid convergence characteristics for NACA0012 flow at $\alpha = 0°$

Algorithm	C_l		C_d	
	order, p	C_l^*	order, p	C_d^*
DIABLO - scalar	0.13	-1.1895×10^{-5}	2.28	8.1225×10^{-3}
DIABLO - matrix	NM	–	1.21	8.1248×10^{-3}
	$C_{d,p}$		$C_{d,f}$	
	order, p	$C_{d,p}^*$	order, p	$C_{d,f}^*$
DIABLO - scalar	2.25	1.3069×10^{-3}	2.28	6.8156×10^{-3}
DIABLO - matrix	2.84	1.3072×10^{-3}	0.96	6.8276×10^{-3}

$$M = 0.8395,\ Re = 11.72 \times 10^6,\ \alpha = 3.06°.$$

The Reynolds number is based on the mean aerodynamic chord (MAC) of the geometry. The grid used in this study consists of 128 blocks, with a total of 15.1 million nodes and an off-wall spacing of 2.3×10^{-7} root chord units, resulting in an average y^+ value of 0.4. The flow solution was computed using the scalar dissipation model. With 128 processors, the residual

Table 6.4: Grid convergence characteristics for NACA0012 flow at $\alpha = 10°$

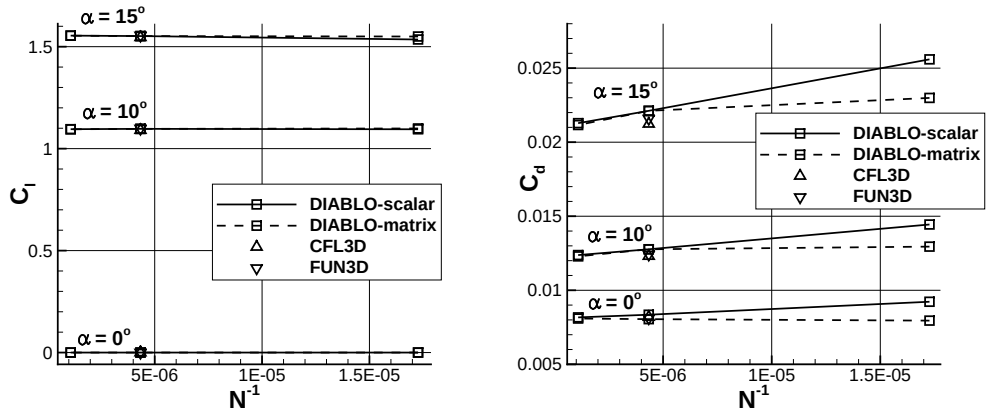
	C_l		C_d	
	order, p	C_l^*	order, p	C_d^*
DIABLO - scalar	NM	–	2.09	1.2257×10^{-2}
DIABLO - matrix	NM	–	2.19	1.2256×10^{-2}
Algorithm	$C_{d,p}$		$C_{d,f}$	
	order, p	$C_{d,p}^*$	order, p	$C_{d,f}^*$
DIABLO - scalar	2.00	6.0733×10^{-3}	2.24	6.1827×10^{-3}
DIABLO - matrix	1.94	6.0783×10^{-3}	1.16	6.1942×10^{-3}

Table 6.5: Grid convergence characteristics for NACA0012 flow at $\alpha = 15°$

Algorithm	C_l		C_d	
	order, p	C_l^*	order, p	C_d^*
DIABLO - scalar	2.96	1.5544	2.06	2.1020×10^{-2}
DIABLO - matrix	2.95	1.5540	2.10	2.1040×10^{-2}
	$C_{d,p}$		$C_{d,f}$	
	order, p	$C_{d,p}^*$	order, p	$C_{d,f}^*$
DIABLO - scalar	2.04	1.5838×10^{-2}	2.20	5.1818×10^{-3}
DIABLO - matrix	1.90	1.5855×10^{-2}	1.07	5.2071×10^{-3}

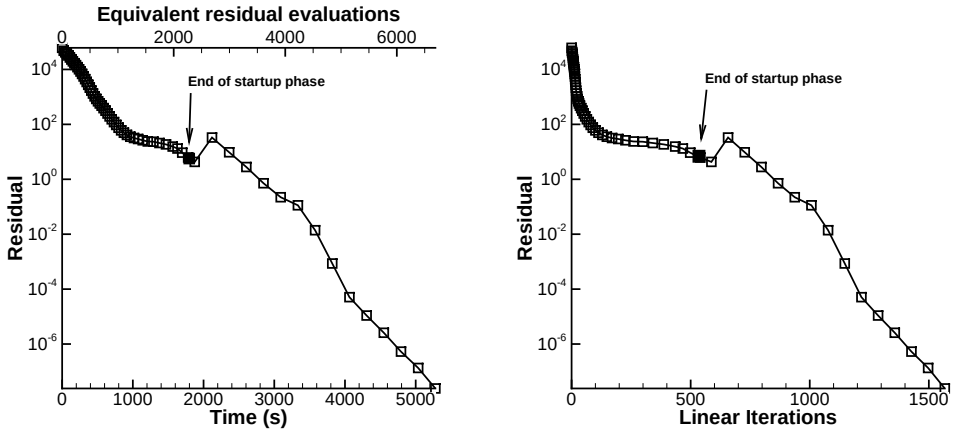
was reduced by 12 orders of magnitude in 86 minutes. Figure 6.8 presents the residual convergence plot for this case, in terms of both time and linear iterations, clearly highlighting the two phases of the solver. The residual includes all six equations. Convergence is also plotted versus "equivalent residual evaluations," normalizing the solution time by the time required to calculate the residual. Since the present algorithm spends substantial time in forming and factoring the preconditioner as well as solving the linear system at each nonlinear iteration, this CPU time measure provides a clearer comparison to other solution algorithms. The switch from the approximate-Newton phase to the inexact-Newton phase occurs at approximately 35 minutes. This plot represents a typical convergence history for DIABLO, with the symbols along the line showing individual nonlinear iterations. The simulation results in values of C_L and C_D of 0.26819 and 0.017145, respectively.

Figure 6.9 presents the C_p distributions at several spanwise sections of the wing, comparing them to experimental data. The comparison highlights the good correspondence of the computational and experimental results throughout the span of the wing. This case



(a) Lift (b) Drag

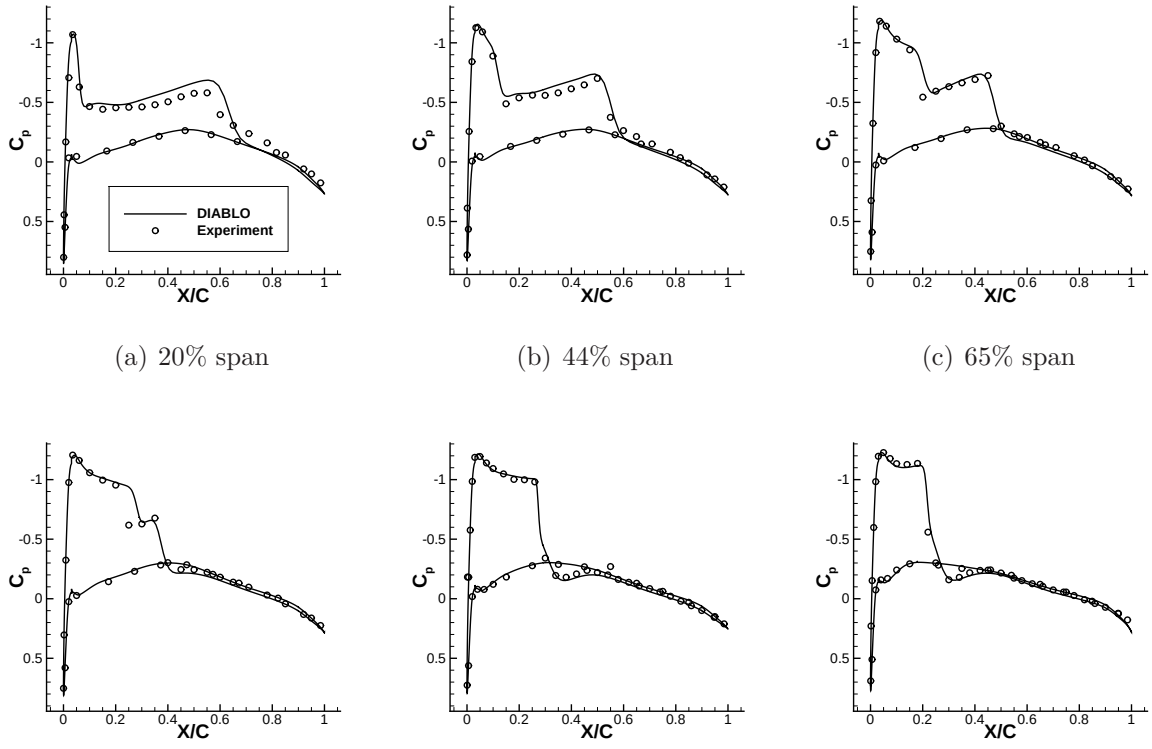
Figure 6.7: Force coefficients for NACA0012 flow at various angles of attack



(a) Time (b) Linear iterations

Figure 6.8: Residual convergence for transonic flow over the ONERA M6 wing with 128 processors

demonstrates the capability of the current algorithm to accurately and efficiently capture the complex transonic flow around the ONERA M6 wing. In order to determine the cause of the small discrepancies between the numerical and experimental results, the solution was computed on a finer, 35 million node grid, with an average y^+ value of 0.18. The values of C_L and C_D changed slightly to 0.27057 and 0.017004, respectively, but the C_p distributions remain virtually identical to that produced on the 15.1 million node grid. This suggests that a lack of grid refinement in the original flow solution is not responsible for the observed C_p distribution differences. Instead, the cause may lie in inherent differences in the problem set-up between the numerical and experimental cases, such as the use of free transition in



(a) 20% span (b) 44% span (c) 65% span

(d) 80% span (e) 90% span (f) 95% span

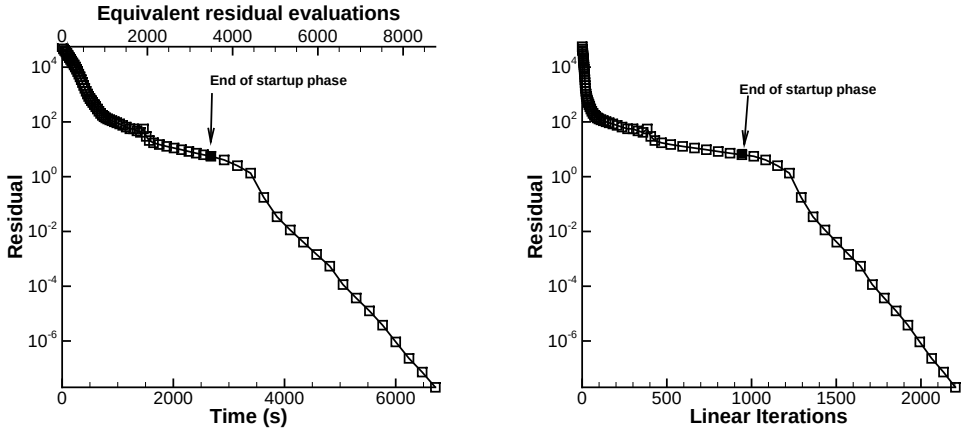
Figure 6.9: Experimental and numerical C_p distributions for ONERA M6 wing flow

the experimental measurements, a feature that is not reproducible in the current algorithm.

The second case for the ONERA M6 wing increases the Reynolds number significantly, with flow conditions of

$$M = 0.8395,\ Re = 40.0 \times 10^6,\ \alpha = 3.06^\circ.$$

The initial 15.1 million node grid was used here, resulting in an average y^+ value of 1.0. The flow solution was computed using the scalar dissipation model. With 128 processors, the residual was reduced by 12 orders of magnitude in 136 minutes. Figure 6.10 presents the residual convergence plot for this case, exhibiting the expected phases of convergence. The higher Reynolds number, and slightly different flow physics, causes slower convergence, with a larger number of total linear iterations required. Additionally, the simulation results in values of C_L and C_D of 0.27114 and 0.016780, respectively, producing higher lift and lower drag than the lower Reynolds number simulation. While no experimental data exist for this case, it serves as a demonstration of the high-Reynolds number solution capabilities of the current algorithm.



(a) Time (b) Linear iterations

Figure 6.10: Residual convergence for high-Re flow over the ONERA M6 wing with 128 processors

6.2.2 NASA Common Research Model wing-body configuration

The results presented in this section represent the participation of the current algorithm in DPW5.

Common grid study

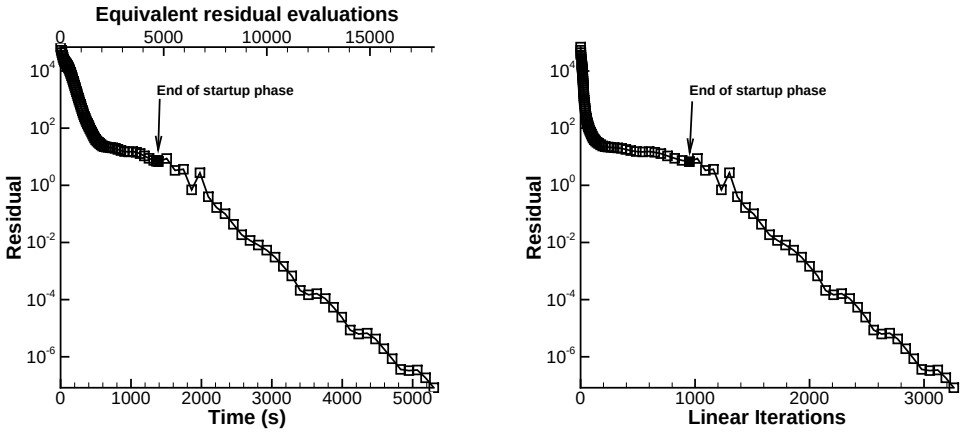
This case is a transonic flow around the CRM wing-body geometry. The O-O topology structured multi-block grids, obtained from the organizers of DPW5, contain between 750 thousand and 146 million nodes, with an off-wall spacing range of 2.4×10^{-6} to 1.9×10^{-7} MAC units, corresponding to y^+ values of 2.0 to 0.33. The original grid family was constructed using a 2-to-3-to-4 node refinement strategy, detailed by Vassberg [101]. The six grid levels effectively comprise two nested grid families, with the odd and even grid levels following the typical 1-to-2 node refinement strategy. The grids contain between 88 and 906 blocks, depending on the size of the grid, but were not load balanced perfectly due to the original grid construction. Table 6.6 provides a summary of the grid parameters. Of note, the wing of the CRM geometry possesses a blunt trailing edge. The flows were computed at flow conditions of

$$M = 0.85,\ Re = 5 \times 10^6,\ C_L = 0.500 \pm 0.001.$$

The Reynolds number is based on the MAC. Each grid level requires a different angle of attack to attain the specified value of C_L. Both scalar and matrix dissipation are examined.

Table 6.6: Grid characteristics for CRM geometry

Grid level	Blocks	Grid nodes	Off-wall distance
T - tiny	88	746,200	2.4×10^{-6}
C - coarse	704	2,647,232	1.6×10^{-6}
M - medium	704	5,969,600	1.2×10^{-6}
F - fine	704	19,150,016	6.5×10^{-7}
X - extra-fine	704	44,239,552	3.9×10^{-7}
S - super-fine	906	146,284,992	1.9×10^{-7}



(a) Time (b) Linear iterations

Figure 6.11: Convergence history for CRM flow on 19 million node grid (matrix dissipation model with 704 processors)

Solutions were obtained on six grid levels, converging the residual by 10 orders of magnitude on each grid. Figure 6.11 presents the convergence history for the 19 million node fine ("F") grid level with matrix dissipation. This particular solution was computed on 704 processors and converged in 70 minutes. A load balancing approach, as detailed by Apponsah and Zingg [6], could be used to substantially improve the computational efficiency of the flow solves, either by reducing the time required to compute the solution using the same computational resources, or by obtaining the solution in the same amount of time using roughly 50% of the processors. As with the previously presented ONERA M6 test case, both stages of the solution algorithm are clearly visible. The complexity of the flow (upper wing surface shocks and wing-body junction recirculation) contributes to substandard linear solver performance in the inexact-Newton phase, leading to lower residual drops between outer iterations. As a consequence, the solution requires more outer iterations in the inexact-Newton phase. Figure

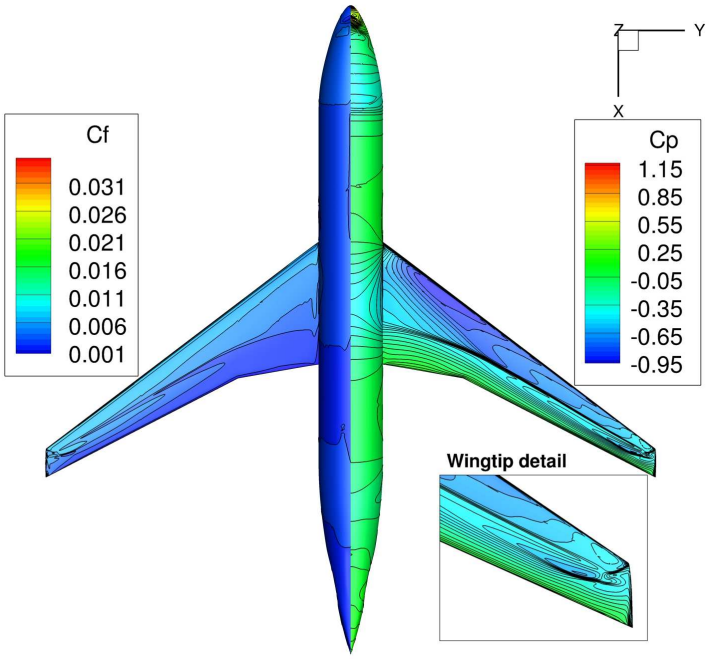
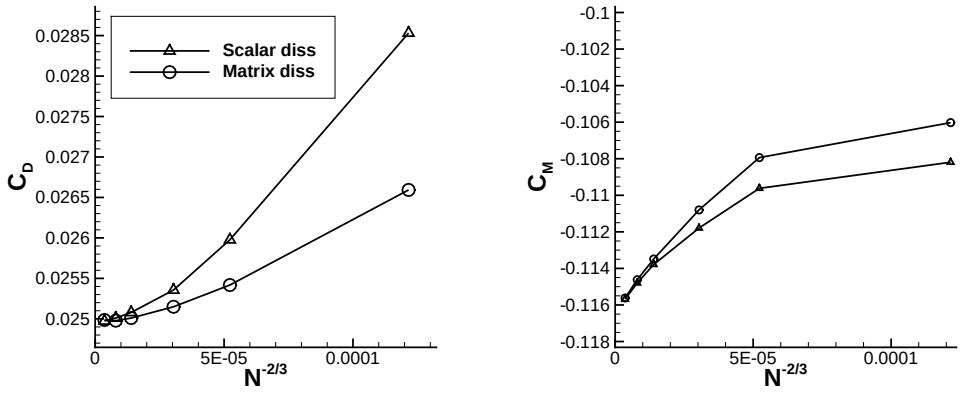


Figure 6.12: Surface skin-friction and pressure coefficients for CRM flow (upper surface)

6.12 shows the contours of C_p and C_f on the upper surface of the geometry. The pressure contours highlight the presence of a shock on the top surface of the wing, with a complex interplay of two shocks near the wingtip.

This case also allows us to observe the grid convergence of the algorithm. Figure 6.13 presents the grid convergence trends of both drag and moment coefficients, with the matrix dissipation model producing a flatter curve for drag convergence. This signifies that, as expected, the model is more accurate on coarser grids. Additionally, a second-order algorithm will tend toward a straight line between three consecutive grid levels when plotted versus $N^{-2/3}$. This behavior can be observed for the finer grid levels. Using values from the three finest grid levels, the order of convergence for C_D is calculated to be 1.60 and 3.32 for the scalar and matrix artificial dissipation models, respectively, exhibiting the expected, or better, grid convergence characteristics. These convergence rates are achieved despite the first-order treatment of the convective terms in the turbulence model, indicating that these terms do not dominate the discretization error and appear to play a small role. Both dissipation models tend towards a C_D of 0.0250 and a C_M of -0.119 as the grid is refined. Levy *et al.* [52], in their summary of the DPW5 results, report average and median values for C_D of 0.0251 and 0.0250, respectively, based on the 57 submitted datasets.



(a) C_D (b) C_M

Figure 6.13: Grid convergence of drag and moment coefficient for CRM flow

Buffet study

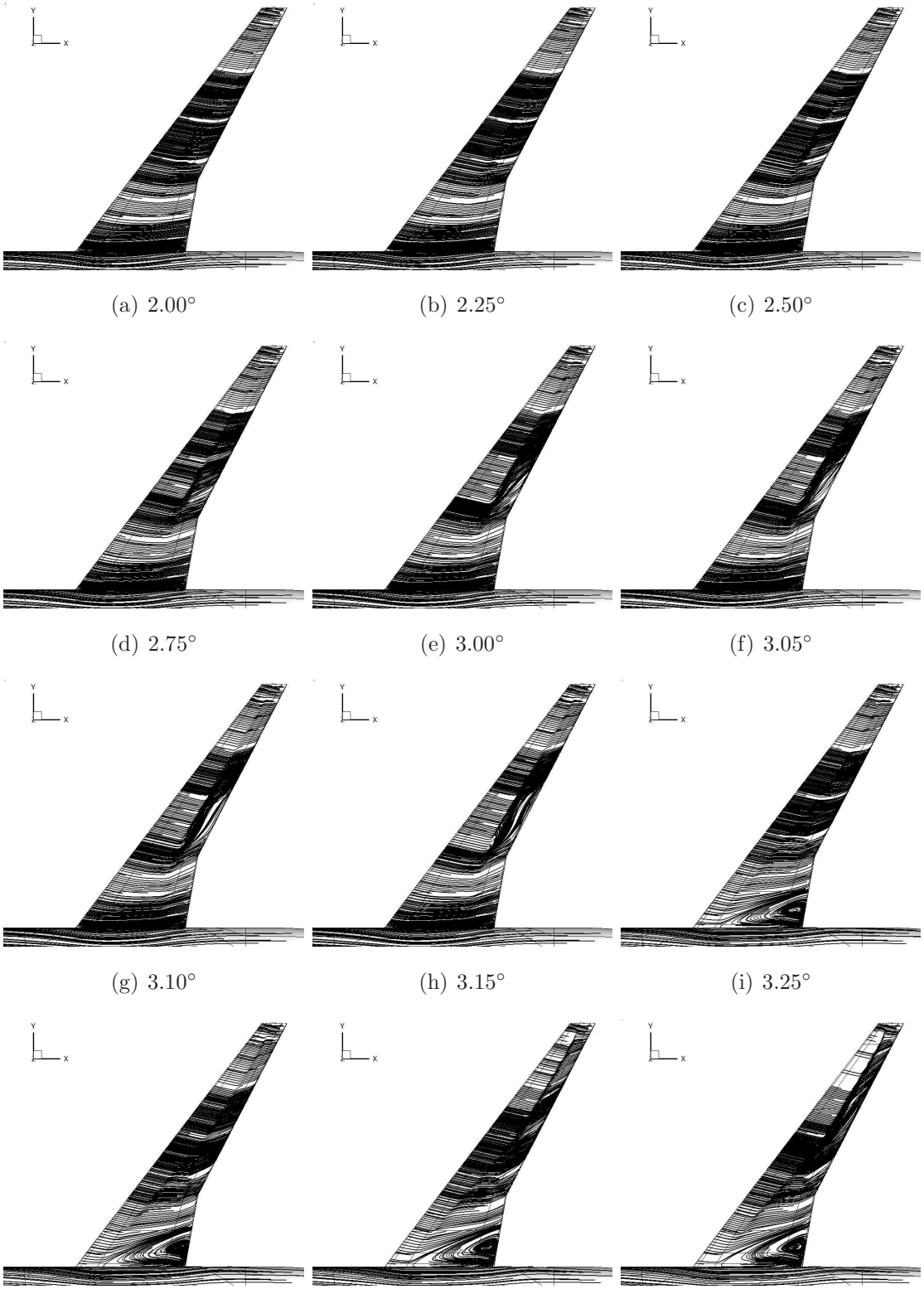
The second DPW5 case makes use of the medium grid level from the common grid family study, with 5.97 million nodes and an off-wall spacing of 5.3×10^{-6} chord units. The original grid has been subdivided into 704 blocks to leverage the parallel capabilities of the current algorithm.

To investigate the range of angles of attack during which buffet onset is expected to occur, the flow conditions used in this case are

$$M = 0.85,\ Re = 5.00 \times 10^6,\ \alpha = 2.00^\circ \text{ to } 4.00^\circ.$$

Solutions were computed with the scalar dissipation model for angles of attack up to 3.15°. The nonlinear residual is reduced by 12 orders of magnitude. Figure 6.14 shows the streamlines above the wing for solutions at all angles of attack.

At 3.25° and above, the steady solution algorithm fails to converge, with stalled nonlinear convergence. Unsteady flow features were assumed to be the cause of the convergence difficulty. Hence, time-accurate simulations were undertaken, using the unsteady solution algorithm developed by Boom and Zingg [13]. The unsteady solution algorithm was used to run solutions for all angles above 3.25°. After advancing all solutions 160 nondimensional time units, no oscillations were observed in the force coefficients, and the solution was converging to a steady flow. Using the final unsteady solution as a starting point, the steady solution algorithm was used to fully converge the solutions. Subsequently, further simulations with substantially reduced time-step ramping and reductions based on residual behavior allowed the steady-state solution algorithm to convergence without any use of the unsteady algorithm. At all angles above 3.25°, a substantially different flow pattern is



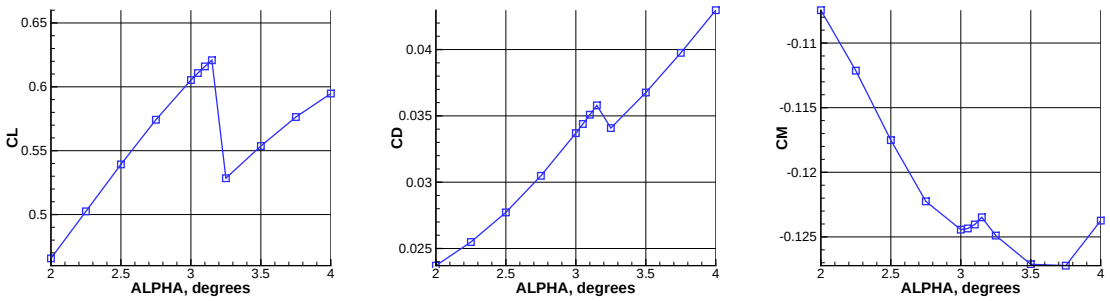
(a) 2.00° (b) 2.25° (c) 2.50°

(d) 2.75° (e) 3.00° (f) 3.05°

(g) 3.10° (h) 3.15° (i) 3.25°

(j) 3.50° (k) 3.75° (l) 4.00°

Figure 6.14: Surface-adjacent streamlines for transonic CRM flow solutions at various angles of attack



(a) Lift (b) Drag (c) Moment

Figure 6.15: CRM force coefficients for buffet study

observed, as can be seen in Figure 6.14. A substantial side-of-body separation bubble is present, originating at the wing-body junction. It is not presently known whether the large separation region represents a physically accurate phenomenon or is an artifact associated with the interplay of the grid resolution in the area and the turbulence model.

Figure 6.15 presents the lift, drag, and moment coefficients for all angles of attack. All coefficient values exhibit a discontinuity at an angle of attack of 3.25° due to the presence of the large recirculation bubble.

While the majority of participants in DPW5 did not show the separation region and the corresponding break in the force coefficient curves, a handful of results with the Spalart-Allmaras turbulence model did exhibit this feature to varying degrees. The occurrence of the large side-of-body separation bubble resulted in a recommendation of further study and development of turbulence models for corner flows [52].

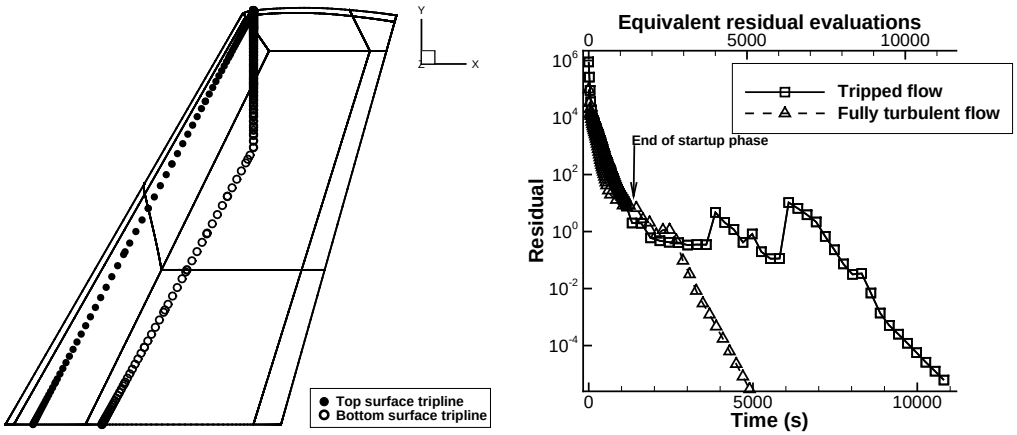
6.2.3 Explicitly-tripped flow solutions

Further extending the capability of the algorithm, the explicit trip location terms of the Spalart-Allmaras turbulence model are implemented. These terms require the user to specify an explicit location for laminar to turbulent transition to occur. Eventually, the ability to specify the transition location will be coupled with transition prediction, such that the flow solver will be incorporated into an optimization algorithm that can take advantage of natural laminar flow in improving the drag characteristics of an aerodynamic shape.

The case considered involves the ONERA M6 wing. The flow conditions for this case are

$$M = 0.30,\ Re = 1 \times 10^6,\ \alpha = 1.00°.$$

The grid used is identical to that in Section 6.2.1. The transition line is specified individually



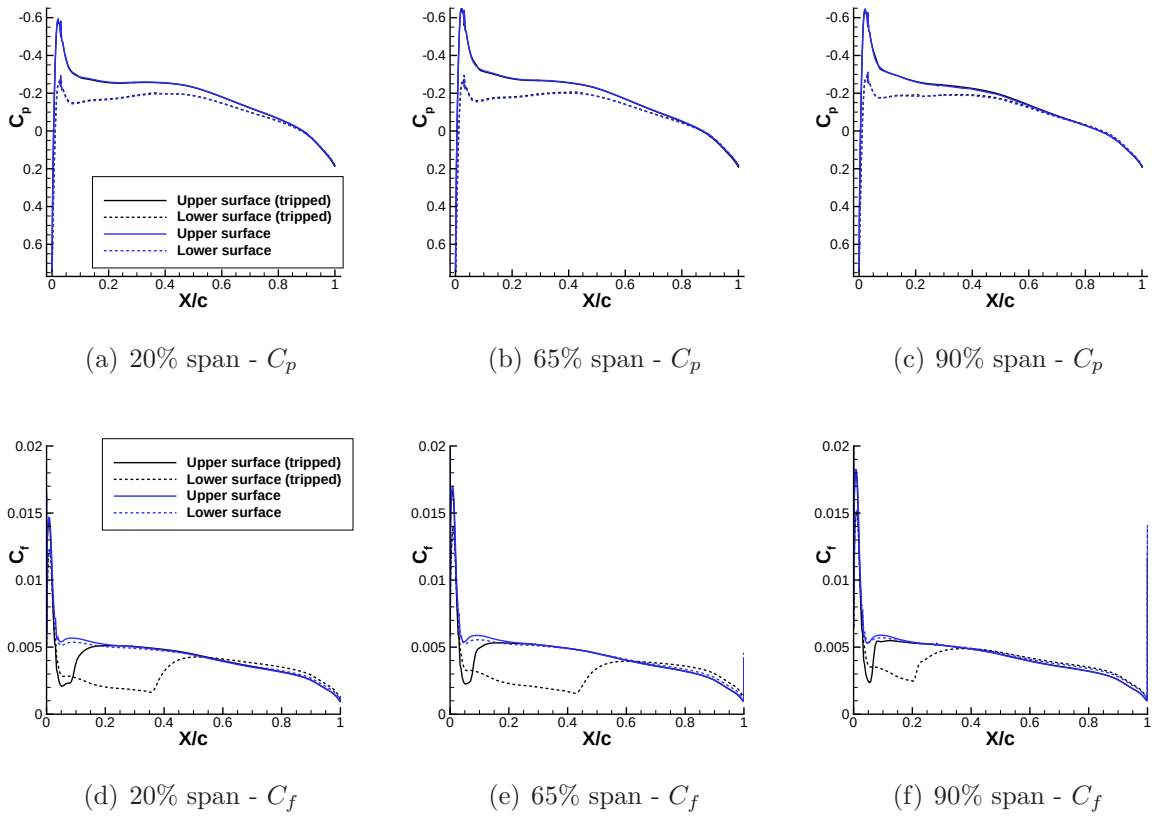
(a) Surface blocking and trip line definition (b) Convergence history

Figure 6.16: Explicitly tripped flow with ONERA M6 wing

on the top and bottom surfaces of the wing, as shown in Figure 6.16(a). Only the scalar dissipation model was used for this case.

The addition of the explicit trip terms increases the initial magnitude of the residual, so it is necessary to remain in the start-up phase for one additional order of magnitude until the residual drops by a factor of 1×10^{-5}. It is for this case that the time step modification discussed in Section 4.2.7 is critical; without it, the solution would diverge due to large changes in the turbulence quantity. Additionally, the complexity of the flow causes noticeable jumps in the residual, which results in longer convergence times than for fully turbulent solutions. This is likely due to the locations of the trip lines themselves, forcing the flow to remain laminar over large portions of the wing, especially on the lower surface. Other than these jumps, the convergence history in Figure 6.16(b) shows the typical convergence characteristics of the Newton-Krylov algorithm. For comparison, a fully turbulent solution at the same operating conditions was also computed, with the corresponding residual convergence included in Figure 6.16(b). As expected, the fully turbulent solution can be converged in substantially less time. Nevertheless, deep residual convergence is attained even for the more complex explicitly tripped flow.

Additionally, the C_f plots at selected span-wise sections, shown in Figure 6.17, demonstrate the effect of imposing specific trip locations. The C_f values along both surfaces of the tripped solution show the characteristic increase at the specified location of laminar to turbulent transition, which are not present for the fully turbulent solution. The corresponding C_p plots are also included, showing negligible sensitivity to the presence of the laminar-to-turbulent transition region.



(a) 20% span - C_p (b) 65% span - C_p (c) 90% span - C_p

(d) 20% span - C_f (e) 65% span - C_f (f) 90% span - C_f

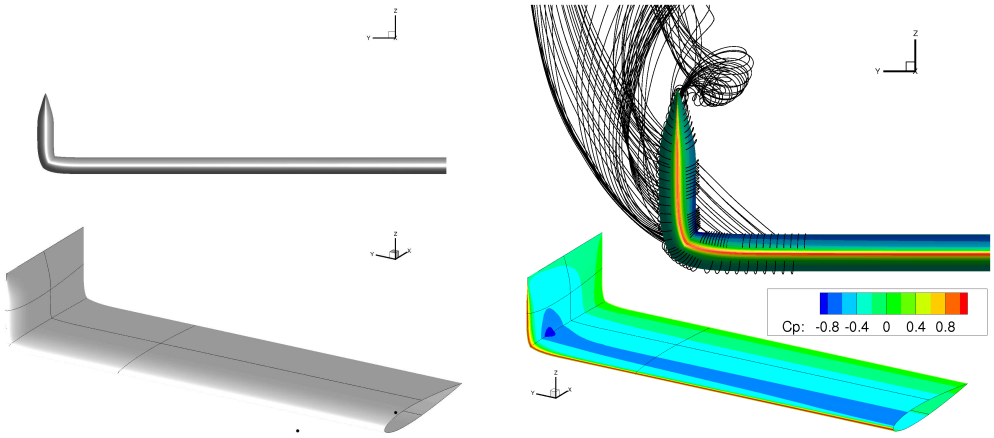
Figure 6.17: C_p and C_f values at selected span-wise sections for tripped and fully turbulent flow

6.2.4 Non-planar geometries

With computational analysis and optimization becoming more integrated in the design process of modern aircraft, flow solution algorithms have to be capable of handling more complex and unconventional geometries. For example, non-planar geometries are becoming commonplace and require computational analysis to fully understand their trade-offs relative to conventional wings. These cases may possess unique flow features, which may introduce numerical instabilities and hamper a solver's convergence to steady state. The test case considered here involves an unswept rectangular wing with a vertical winglet, as shown in Figure 6.18(a). Both the wing and winglet possess the NACA0012 airfoil cross-section. The flow conditions are

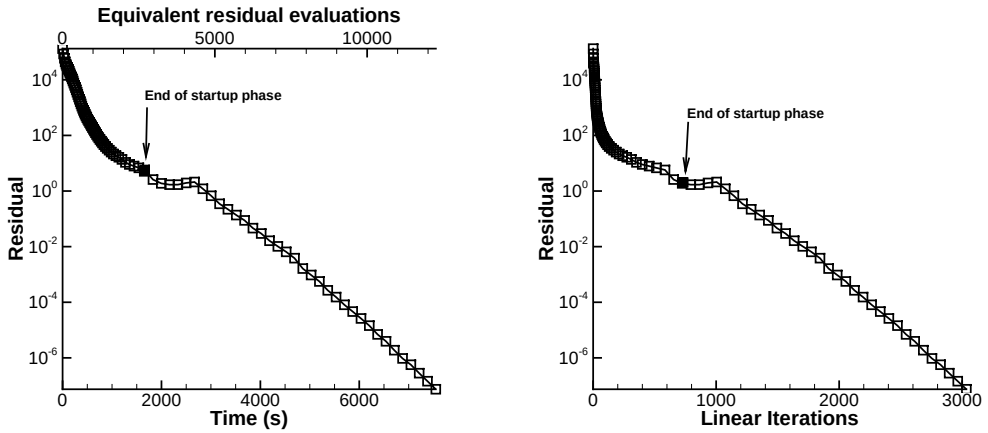
$$M = 0.40,\ Re = 7.48 \times 10^6,\ \alpha = 2.00^\circ.$$

The finest computational grid consists of 960 blocks, with a total of 113 million nodes and an off-wall distance of 4.0×10^{-7} chord units, giving an average y^+ of 0.18. Two additional grid levels were created by successively removing every second node in each computational



(a) Winglet geometry (b) Streamlines and $\boldsymbol{C_p}$ contours

Figure 6.18: Non-planar geometry and flow



(a) Time (b) Linear iterations

Figure 6.19: Non-planar wing convergence history on 133 million node grid (with 960 processors)

direction. On the finest grid, the solution was obtained using 960 processors, converging the residual by 12 orders of magnitude in 125 minutes, as shown in Figure 6.19. The results for this case illustrate the convergence capabilities of the current algorithm when dealing with non-planar geometries. Only the scalar dissipation model was used for this case. Figure 6.18(b) shows the C_p contours on the surface of the geometry, as well as some streamlines emanating from the winglet area.

The nested grid family allows us to use Richardson extrapolation to compute the order of convergence of lift and drag. Table 6.7 presents the force coefficients on all grid levels, in addition to the calculated orders of convergence, p, and grid-converged force values, F^*. The

Table 6.7: Grid convergence for C_L and C_D

Grid level	Grid nodes	C_L	C_D
coarse	2,109,120	0.1773	0.01676
medium	15,000,000	0.1820	0.01187
fine	112,943,040	0.1834	0.01087
order, p	-	1.80	2.32
F^*	-	0.1840	0.01064

orders of convergence for C_L and C_D are 1.80 and 2.32, respectively, exhibiting the expected grid convergence characteristics for the second-order spatial discretization. The projected area of 3.041 reference units squared is used as the reference area.

Non-planar geometries provide challenges in terms of grid generation compared to conventional wings, especially in the area of the winglet. The use of a multi-block gridding strategy allows us to place blocks and refine the grid around the geometry in a manner that resolves the boundary layer of both the wing and winglet without introducing excessive grid stretching. Additionally, the SBP-SAT discretization used in the current algorithm provides excellent flexibility in terms of the blocking strategy used, further extending the ease with which complex geometries can be accommodated.

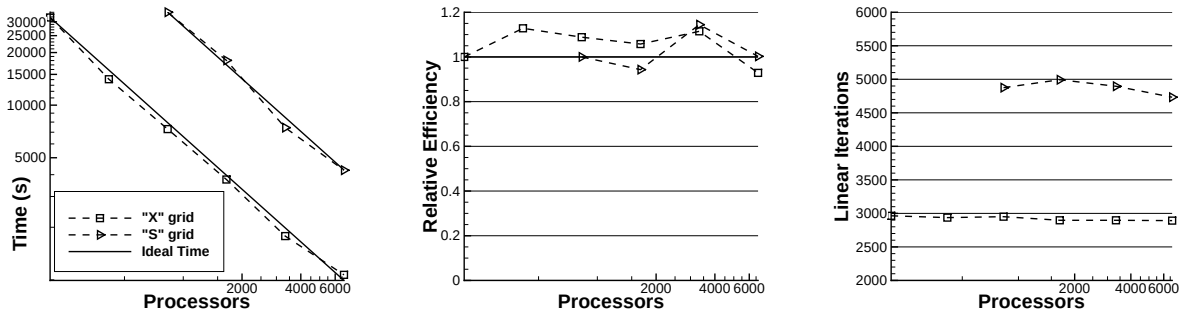
6.3 Parallel Scaling

In order to evaluate the parallel performance of the algorithm, a parallel scaling study was conducted for the solution of a transonic flow around the CRM wing-body geometry. The flow conditions are identical to those in Section 6.2.2.

The CRM wing-body case was run on two grids, with each grid consisting of a 6656 block O-O topology mesh obtained from the "X" and "S" grid levels used in DPW5. The "X" grid contains 48 million nodes and an off-wall spacing of 3.9×10^{-7} MAC units, while the "S" grid contains 154 million nodes and an off-wall spacing of 1.9×10^{-7} MAC units. The matrix dissipation model is used, and the residual is converged 10 orders of magnitude. All meshes are perfectly load-balanced, with an equal number of nodes in each block.

The results for both cases, presented in Figure 6.20, show that the code exhibits excellent parallel scaling characteristics up to 6656 processors*. The performance of the code is mea-

*The algorithm was not tested with more than 6656 processors due to computational system limitations, but is expected to scale well beyond this value.



(a) Time (b) Relative parallel efficiency (c) Linear iterations

Figure 6.20: Parallel scaling performance of DIABLO

sured by relative efficiency, which is based on the lowest possible number of processors that each case can be computed with. Due to memory requirements, the cases require a minimum of 208 and 832 processors, respectively. In the range of processors considered, the relative efficiency does not drop below 90%. In fact, many processor counts exhibit super-linear scaling, due partly to the changing form the the preconditioner, with different numbers of interface nodes contributing to the global Schur complement, and partly to the manner in which the parallel computing hardware manages parallel communication with varying numbers of processors. Additionally, the nearly constant number of linear iterations required to converge the solutions at different processor numbers highlights the effectiveness of the approximate-Schur preconditioner even when large numbers of processors (>6000) are used.

The scaling characteristics of the code, coupled with the ease with which the SBP-SAT approach can handle arbitrary numbers of blocks and their orientations, make this algorithm an attractive option for applications where fast turnaround times are required. Not only is the underlying parallel Newton-Krylov-Schur algorithm robust and efficient in obtaining converged steady-state solutions, it can easily make use of larger numbers of processors, when available, to further reduce wall-time required for computations without a significant loss of computational resource efficiency.

Chapter 7

Conclusions, Contributions, and Recommendations

This section presents a summary of the work completed as part of this thesis, with an emphasis on the contributions made to the solution algorithm and the understanding of the use of a Newton-Krylov-Schur method for the solution of the RANS equations. Recommendations for future work are presented in §7.2.

7.1 Conclusions and Contributions

A parallel Newton-Krylov-Schur algorithm was developed for the solution of the Reynolds-averaged Navier-Stokes equations, making use of the Spalart-Allmaras one-equation turbulence model. The governing equations are discretized using the SBP-SAT discretization, leading to efficient application of the approach on three-dimensional geometries. This is aided by the use of multi-block structured grids, which alleviate the difficulty of grid generation around complex geometries. The discrete equations for a steady-state solution are solved with a Newton-Krylov algorithm, making use of the GMRES linear solution method. Parallel preconditioning is achieved with the approximate-Schur preconditioner, which is demonstrated to remain effective with over 6500 processors.

The new "negative" turbulence model variant was investigated and shown to be effective at obtaining steady-state solutions, but did not provide any benefits over the standard model formulation. However, higher-order discretizations, which have been shown to produce negative turbulence quantities in the steady-state solutions on insufficiently refined grids, will likely benefit from the inclusion of this model variant. Several flow solutions with explicitly-specified laminar-to-turbulent transition locations were computed in two and three dimensions. The complexity of flow in three-dimensional transition regions can result in unpredictable residual increases during the convergence process, slowing overall convergence.

The overall solution algorithm was tuned through an extensive algorithm optimization process. The globalization strategy, using pseudo-transient continuation, was shown to be

effective at obtaining steady-state solutions to the RANS equation; optimal time stepping parameters were determined for both the approximate- (start-up) and inexact-Newton phase, along with the optimal drop tolerance for switching between the two phases. The importance of proper equation scaling in the linear system is shown through a study of several methods for the scaling of the turbulence model equation. The level of fill in the ILU factorization used in the preconditioner is shown to possess distinct optimal values for the two phases of the solution algorithm, with higher fill required for larger grids (larger linear systems) in the inexact-Newton phase. Node ordering with the reverse Cuthill-McKee method (with downstream root node selection and ordering bias) was demonstrated to be essential for the efficient solution of the linear system, especially for complex three-dimensional grids with arbitrary initial block orientations. Finally, the effect of turbulence model initialization was shown to be especially important for explicitly tripped flow solutions. If initialized to too high a value, the explicitly tripped flow solution would exhibit transition upstream of the specified trip locations.

The applicability of the algorithm to a wide range of flow solution problems was demonstrated through a number of results. In particular, the implementation of the algorithm is verified against the established CFL3D and FUN3D algorithms through a series of two-dimensional test cases. Correspondence with experimental data is highlighted with the solution of subsonic flow over the NACA0012 airfoil and transonic flow over the ONERA M6 wing. Additionally, the robustness of the algorithm is demonstrated through the solution of several two-dimensional high-lift cases. Solution of flow around a nonplanar configuration is shown with a vertical winglet geometry, and the explicit trip location definition is demonstrated in three dimensions with a subsonic solution over the ONERA M6 wing.

The development of this powerful three-dimensional solution algorithm allowed for participation in the Fifth AIAA Drag Prediction Workshop, held in New Orleans in June 2012. Converged steady-state flow solutions were obtained on grids with up to 150 million nodes, demonstrating the use of the algorithm on complex flows around wing-body configurations. The current algorithm produced solutions that compared very favourably with solutions obtained with well-established and widely-used flow solvers.

The primary contribution of this thesis is the development of a flow solver for the RANS equations based on the unique combination of the SBP-SAT discretization combined with a Newton-Krylov-Schur parallel implicit scheme and the demonstration of its characteristics in the context of practical problems. We can conclude that this combination provides an efficient and robust algorithm for the solution of the RANS equations for steady compressible aerodynamic flows over relatively clean geometries. The properties of the algorithm make it

well suited for use in aerodynamic shape optimization and MDO. Recent work by Osusky and Zingg [74] has already demonstrated the applicability of the current flow solver in the context of aerodynamic shape optimization.

The following provides an overview of specific contributions made during the course of this thesis work:

1. Developed and implemented SATs for use with the Spalart-Allmaras one-equation turbulence model, discretizing the model with the SBP-SAT approach. To the knowledge of the author, this is the first use of the SBP-SAT approach in this context. The turbulence model, along with the discretization of the viscous fluxes, was added to an existing solution algorithm for the Euler equations.

2. Conducted thorough investigation of optimum solution algorithm parameters that result in an efficient and robust Newton-Krylov-Schur solution strategy, with particular emphasis on the Spalart-Allmaras turbulence model.

3. Conducted extensive validation and verification of the current algorithm, increasing the confidence with which it can be applied to a wide range of flow solution problems, including non-planar geometries. This is especially important for the use of the flow solver on unconventional geometries, as the extent of computational and experimental data for these geometries is limited.

4. Participated in the Fifth Drag Prediction Workshop, demonstrating the accuracy, efficiency, and robustness of the current algorithm alongside a large number of established flow solution algorithms.

5. Developed discretization strategy for use on grids with degenerate edges, a grid type that was previously not suitable for use with the current algorithm. This addition greatly expands the number of grid topologies that can be used with the flow solver, easing the difficulty in cooperating with third parties.

6. Developed a fast and accurate off-wall distance calculation method. In addition, the importance of accurate values of off-wall distances for use with the turbulence model was demonstrated, especially on grids containing grid lines with high incidence angles to the solid surface boundary.

7. Developed novel discretization approach for the advective terms of the turbulence model, making use of fourth-difference dissipation operators. This provides a means

of extending the turbulence model discretization to higher-order schemes, which that was not possible with the original first-order upwinding-based discretization.

7.2 Recommendations

Based on the findings presented in this thesis, several recommendations can be made for future work. These include practical improvements to the algorithm, as well as future extensions to improve the efficiency and applicability of the solver.

Matrix dissipation model: While the current algorithm can make use of the matrix dissipation model, the robustness decrease that the use of this model currently imposes make its application to optimization runs limited. Hence, further investigation into the improvement of the robustness of the algorithm with the matrix dissipation model is recommended.

Trip location definition in three dimensions: In order to extend the use of the explicit trip terms for complex geometries, such as wing-body configurations, a more robust trip line definition strategy has to be developed. The current implementation assumes the use of simple three-dimensional shapes, as as wings.

Degenerate grid use: In order to further extend the capability of the current algorithm, an automated method for the identification of degenerate edges in grids should be investigated. The current approach requires the user to identify these edges manually and supply the information to the flow solver, a task that can become untenable for highly complex three-dimensional grids. Additionally, there is currently no capability in the mesh fitting and movement algorithm to handle the types of degenerate grids used by the flow solution algorithm. The incorporation of this capability would open the door to the use of grids which possess a better set of characteristics, such as orthogonality of grid lines, than the typical non-degenerate grids currently used.

General grid characteristics: The SBP-SAT method has been shown to be applicable to hybrid grids that make use of both structured and unstructured domains [28]. Furthermore, the discretization has the potential for removing the requirement of node-matching at interfaces; the number of nodes on one side of the interface does not have to correspond to the adjoining block, as demonstrated by Mattsson and Carpenter [58]. This capability could potentially open the way for the use of adaptive grid refinement, improving the efficiency of the algorithm by refining the computational mesh only in

regions that require it, such as in the turbulent boundary layer or in the vicinity of shocks.

Transition prediction: Even though the Spalart-Allmaras one-equation turbulence model allows for the specification of explicit locations for laminar-to-turbulent transition, it does not possess any transition prediction capability. Hence, it is up to the user to specify where transition should occur, a process that requires intimate knowledge of the flow characteristics; often, this can lead to unstable flow solutions, with transition locations specified in regions of flow that do not support laminar flow. Hence, the incorporation of a transition prediction capability, such as the e^N method, would greatly increase the accuracy and applicability of the solution algorithm to flows with both laminar and turbulent regions. Once implemented in the flow solution algorithm, transition prediction could be applied within the optimization process, allowing for the exploitation of regions of natural laminar flow for the reduction of drag.

Continuation strategies: Building on the success of the use of homotopy-based continuation strategies for the Euler and laminar Navier-Stokes equations [35, 36], these strategies should be investigated for the RANS equations, resulting in possible increases in robustness and efficiency. In particular, this may prove useful for some of the more problematic flows, especially those exhibiting nearly unsteady flow characteristics, leading to convergence problems for the PTC approach. This category includes the higher angle of attack cases studied as part of DPW5.

Higher-order spatial discretization: One way to improve the efficiency of the current approach to solving the RANS equations is by increasing the order of accuracy of the spatial discretization. This is achieved by obtaining higher levels of accuracy on a particular grid than the current second-order discretization, reducing the need for extremely fine grids. The SBP-SAT approach provides a rigorous method for the development of higher-order operators which could be incorporated into the current solution framework. The use of general artificial dissipation for the discretization of the advection terms in the Spalart-Allmaras one-equation turbulence model allows for straightforward extension to higher-order.

Unsteady flow analysis: The steady flow solution algorithm can also serve as the basis of an unsteady solver. For example, the inexact-Newton phase of the current algorithm can be applied to solve the nonlinear systems that arise within each time step of a time-accurate flow simulation with a Runge-Kutta method.

References

[1] *NASA Turbulence Modeling Resource*. http://turbmodels.larc.nasa.gov.

[2] K. Ajmani, W.-F. Ng, and M.-S. Liou, *Preconditioned conjugate gradient methods for the Navier-Stokes equations*, Journal of Computational Physics, 110 (1994), pp. 68–81.

[3] S. R. Allmaras, F. T. Johnson, and P. R. Spalart, *Modifications and clarifications for the implementation of the Spalart-Allmaras turbulence model*, in 7th International Conference on Computational Fluid Dynamics, ICCFD7–1902, Big Island, Hawaii, USA, July 2012.

[4] W. K. Anderson, J. C. Newman, D. L. Whitfield, and E. J. Nielsen, *Sensitivity analysis for Navier-Stokes equations on unstructured meshes using complex variables*, AIAA Journal, 39 (2001), pp. 56–63.

[5] W. K. Anderson, R. D. Rausch, and D. L. Bonhaus, *Implicit/multigrid algorithms for incompressible turbulent flows on unstructured grids*, Journal of Computational Physics, 128 (1996), pp. 391–408.

[6] K. P. Apponsah and D. W. Zingg, *A load balancing tool for structured multi-block grid CFD applications*, in 20th Annual Conference of the CFD Society of Canada, Canmore, Alberta, Canada, May 2012.

[7] T. J. Baker, *Mesh generation: Art or science?*, Progress in Aerospace Sciences, 41 (2005), pp. 29–63.

[8] B. S. Baldwin and H. Lomax, *Thin layer approximation and algebraic model for separated turbulent flows*, AIAA–78–257, 1978.

[9] G. E. Barter and D. L. Darmofal, *Shock capturing with higher-order, PDE-based artificial viscosity*, in 18th AIAA Computational Fluid Dynamics Conference, AIAA–2007–3823, Miami, Florida, June 2007.

[10] R. Beam and R. F. Waming, *An implicit finite-difference algorithm for hyperbolic systems in conservation law form*, Journal of Computational Physics, 22 (1977), pp. 87–110.

[11] M. Benzi, D. B. Szyld, and A. van Duin, *Orderings for incomplete factorization preconditionign of nonsymmetric problems*, SIAM Journal on Scientific Computing, 29 (1999), pp. 1652–1670.

[12] M. Blanco and D. W. Zingg, *Fast Newton-Krylov method for unstructured grids*, AIAA Journal, 36 (1998), pp. 607–612.

[13] P. D. Boom and D. W. Zingg, *Time-accurate flow simulations using an efficient Newton-Krylov-Schur approach and high-order temporal and spatial discretization*, in 51st AIAA Aerospace Sciences Meeting and Aerospace Exposition, AIAA–2013–0383, Grapevine, Texas, United States, Jan. 2013.

[14] A. Brandt, *A Guide to Multigrid Development*, Springer-Verlag, New York, NY, 1982.

[15] M. H. Carpenter, D. Gottlieb, and S. Abarbanel, *Time-stable boundary conditions for finite-difference schemes solving hyperbolic systems: methodology and application to high-order compact schemes*, Journal of Computational Physics, 111 (1994), pp. 220–236.

[16] M. H. Carpenter, J. Nordström, and D. Gottlieb, *A stable and conservative interface treatment of arbitrary spatial accuracy*, Journal of Computational Physics, 148 (1999), pp. 341–365.

[17] I. B. Celik, U. Ghia, P. J. Roache, C. J. Freitas, H. Coleman, and P. E. Raad, *Procedure for estimation and reporting of uncertainty due to discretization in CFD applications*, Journal of Fluids Engineering, 130 (2008).

[18] T. T. Chisholm, *A Fully Coupled Newton-Krylov Solver With a One-Equation Turbulence Model*, PhD thesis, University of Toronto, Toronto, Ontario, Canada, 2007.

[19] T. T. Chisholm and D. W. Zingg, *A Jacobian-free Newton-Krylov algorithm for compressible turbulent fluid flows*, Journal of Computational Physics, 228 (2009), pp. 3490–3507.

[20] S. C. Dias, *A high-order parallel Newton-Krylov flow solver for the Euler equations*, master's thesis, University of Toronto, Toronto, Ontario, Canada, 2009.

[21] S. C. Dias and D. W. Zingg, *A high-order parallel Newton-Krylov flow solver for the Euler equations*, in 19th AIAA Computational Fluid Dynamics Conference, AIAA–2009–3657, San Antonio, Texas, United States, June 2009.

[22] P. Diener, E. N. Dorband, E. Schnetter, and M. Tiglio, *Optimized high-order derivative and dissipation operators satisfying summation by parts, and application in three-dimensional multi-block evolutions*, Journal of Scientific Computing, 32 (2007), pp. 109–145.

[23] S. C. Eisenstat and H. F. Walker, *Choosing the forcing terms in an inexact Newton method*, SIAM Journal on Scientific Computing, 17 (1996), pp. 16–32.

[24] P. Eliasson, S. Eriksson, and J. Nordström, *The influence of weak and strong solid wall boundary conditions on the convergence to steady-state of the Navier-stokes equations*, in 19th AIAA Computational Fluid Dynamics Conference, AIAA–2009–3551, San Antonio, Texas, United States, June 2009.

[25] P. A. Forsyth and H. Jiang, *Nonlinear iteration methods for high speed laminar compressible Navier-Stokes equations*, Computers and Fluids, 26 (1997), pp. 249–268.

[26] D. Geer, *Chip makers turn to multicore processors*, Computer, 38 (2005), pp. 11–13.

[27] P. Geuzaine, *Newton-Krylov strategy for compressible turbulent flows on unstructured meshes*, AIAA Journal, 39 (2000), pp. 528–531.

[28] J. Gong and J. Nordström, *A stable and efficient hybrid scheme for viscous problems in complex geometries*, Journal of Computational Physics, 226 (2007), pp. 1291–1309.

[29] N. Gregory and C. L. O'Reilly, *Low-speed aerodynamic characteristics of NACA 0012 aerofoil sections, including the effects of upper-surface roughness simulation hoar frost*, Tech. Report NASA R&M 3276, 1970.

[30] M. Hafez, S. Palaniswamy, and P. Mariani, *Calculations of transonic flows with shocks using Newtons method and a direct solver, part II*, AIAA–88–0226, 1988.

[31] A. Hellsten, *New advanced k-omega turbulence model for high-lift aerodynamics*, AIAA Journal, 43 (2005), pp. 1857–1869.

[32] J. L. Hess, *Panel methods in cumputational fluid dynamics*, Annual Review Fluid Mechanics, 22 (1990), pp. 255–274.

[33] J. S. Hesthaven, *A stable penalty method for the compressible Navier-Stokes equations: III. multidimensional domain decomposition schemes*, SIAM Journal on Scientific Computing, 20 (1998), pp. 62–93.

[34] J. E. Hicken, *Efficient algorithms for future aircraft design: Contributions to aerodynamic shape optimization*, PhD thesis, University of Toronto, Toronto, Ontario, Canada, 2009.

[35] J. E. Hicken, H. Buckley, M. Osusky, and D. W. Zingg, *Dissipation-based continuation: a globalization for inexact-Newton solvers*, in 20th AIAA Computational Fluid Dynamics Conference, AIAA–2011–3237, Honolulu, Hawaii, United States, June 2011.

[36] J. E. Hicken, M. Osusky, and D. W. Zingg, *Comparison of parallel preconditioners for a Newton-Krylov flow solver*, in 6th International Conference on Computational Fluid Dynamics, St. Petersburg, Russia, July 2010.

[37] J. E. Hicken and D. W. Zingg, *A parallel Newton-Krylov solver for the Euler equations discretized using simultaneous approximation terms*, AIAA Journal, 46 (2008), pp. 2773–2786.

[38] ——, *Globalization strategies for inexact-Newton solvers*, in 19th AIAA Computational Fluid Dynamics Conference, AIAA–2009–4139, San Antonio, Texas, United States, June 2009.

[39] ——, *Aerodynamic optimization algorithm with integrated geometry parameterization and mesh movement*, AIAA Journal, 48 (2010), pp. 401–413.

[40] C. Hirsch, *Numerical Computation of Internal and External Flows*, John Wiley & Sons, Chichester, England, 1990.

[41] A. Jameson, W. Schmidt, and E. Turkel, *Numerical solution of the Euler equations by finite volume methods using Runge-Kutta time-stepping schemes*, in 14th Fluid and Plasma Dynamics Conference, AIAA–81–1259, Palo Alto, California, United States, 1981.

[42] D. Jespersen, T. Pulliam, and P. Buning, *Recent enhancements to OVERFLOW (Navier-Stokes code)*, in 35th AIAA Aerospace Sciences Meeting and Exhibit, AIAA–97–0644, Reno, Nevada, Jan. 1997.

[43] D. C. W. Kam, *A three-dimensional Newton-Krylov Navier-Stokes flow solver using a one-equation turbulence model*, master's thesis, University of Toronto, Toronto, Ontario, Canada, 2007.

[44] C. T. Kelley and D. E. Keyes, *Convergence analysis of pseudo-transient continuation*, SIAM Journal on Numerical Analysis, 35 (1998), pp. 508–523.

[45] D. E. Keyes, *Aerodynamic applications of Newton-Krylov-Schwarz solvers*, in Proceedings of the 14th International Conference on Numerical Methods in Fluid Dynamics, New York, 1995, Springer, pp. 1–20.

[46] D. B. Kim and P. D. Orkwis, *Jacobian update strategies for quadratic and near-quadratic convergence of Newton and Newton-like implicit schemes*, in 31st AIAA Aerospace Sciences Meeting and Exhibit, AIAA–93–0878, Reno, Nevada, 1993.

[47] W. Knight, *Two heads are better than one*, IEE Review, 51 (2005), pp. 32–35.

[48] D. Knoll and D. Keyes, *Jacobian-free Newton-Krylov methods: a survey of approaches and applications*, Journal of Computational Physics, 193 (2004), pp. 357–397.

[49] H.-O. Kreiss and G. Scherer, *Finite element and finite difference methods for hyperbolic partial differential equations*, in Mathematical Aspects of Finite Elements in Partial Differential Equations, C. de Boor, ed., Mathematics Research Center, the University of Wisconsin, Academic Press, 1974.

[50] S. L. Krist, , R. T. Biedron, and C. L. Rumsey, *CFL3D users manual (version 5.0)*, Tech. Report NASA TM 1998–208444, National Aeronautics and Space Administration, 1998.

[51] A. Krumbein, *Automatic transition prediction and application to high-lift multi-element configurations*, in 34th AIAA Fluid Dynamics Conference, AIAA–2004–2543, Portland, Oregon, June 2004.

[52] D. W. Levy, K. R. Laflin, E. N. Tinoco, J. C. Vassberg, M. Mani, B. Rider, C. L. Rumsey, R. A. Wahls, J. H. Morrison, O. P. Brodersen, S. Crippa, D. J. Mavriplis, and M. Murayama, *Summary of data from the Fifth AIAA CFD Drag Prediction Workshop*, in 51st AIAA Aerospace Sciences Meeting and Aerospace Exposition, AIAA–2013–0046, Grapevine, Texas, United States, Jan. 2013.

[53] D. W. Levy, T. Zickuhr, J. C. Vassberg, S. Agarwal, R. A. Wahls, S. Pirzadeh, and M. J. Hemsch, *Summary of data from the first AIAA CFD Drag Prediction Workshop*, AIAA–2002–0841, 2002.

[54] W.-H. Liu and A. H. Sherman, *Comparative analysis of the Cuthill-McKee and reverse Cuthill-McKee ordering algorithms for sparse matrices*, SIAM Journal of Numerical Analysis, 13 (1976), pp. 1998–213.

[55] H. Lomax, T. H. Pulliam, and D. W. Zingg, *Fundamentals of Computational Fluid Dynamics*, Springer–Verlag, Berlin, Germany, 2001.

[56] L. Martinelli, A. Jameson, and F. Grasso, *A multigrid method for the Navier-Stokes equations*, AIAA–86–0208, 1986.

[57] J. R. R. A. Martins, P. Sturdza, and J. J. Alonso, *The complex-step derivative approximation*, ACM Transactions on Mathematical Software, 29 (2003), pp. 245–262.

[58] K. Mattsson and M. H. Carpenter, *Stable and accurate interpolation operators for high-order multiblock finite difference methods*, SIAM Journal on Scientific Computing, 32 (2010), pp. 2298–2320.

[59] K. Mattsson, M. Svärd, and M. Shoeybi, *Stable and accurate schemes for the compressible Navier-Stokes equations*, Journal of Computational Physics, 227 (2008), pp. 2293–2316.

[60] D. J. Mavriplis, *Grid resolution of a drag prediction workshop using the nsu3d unstructured mesh solver*, in 17th AIAA Computational Fluid Dynamics Conference, AIAA–2005–4729, Toronto, Canada, June 2005.

[61] D. J. Mavriplis, J. C. Vassberg, E. N. Tinoco, M. Mani, O. P. Brodersen, B. Eisfeld, R. A. Wahls, J. H. Morrison, T. Zickuhr, D. Levy, and M. Murayama, *Grid quality and resolution issues from the drag prediction workshop series*, in 46th AIAA Aerospace Sciences Meeting and Exhibit, AIAA–2008–0930, Reno, Nevada, Jan. 2008.

[62] J. A. Meijerink and H. A. van der Vorst, *An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix*, Mathematics of Computation, 31 (1977), pp. 148–162.

[63] F. R. Menter, *Two-equation eddy-viscosity turbulence models for engineering applications*, AIAA Journal, 32 (1994), pp. 1598–1605.

[64] J. R. M. Moir, *Measurements on a two-dimensional aerofoil with high-lift devices*, Tech. Report 303, AGARD, 1994.

[65] W. A. Mulder and B. van Leer, *Experiments with implicit upwind methods for the Euler equations*, Journal of Computational Physics, 59 (1985), pp. 232–246.

[66] M. Nemec and D. W. Zingg, *A Newton-Krylov algorithm for aerodynamic design using the Navier-Stokes equations*, AIAA Journal, 40 (2002), pp. 1146–1154.

[67] N. C. Nguyen, P.-O. Persson, and J. Peraire, *RANS solutions using high order discontinuous Galerkin methods*, in 45th AIAA Aerospace Sciences Meeting and Exhibit, AIAA–2007–914, Reno, Nevada, Jan. 2007.

[68] J. Nichols and D. W. Zingg, *A three-dimensional multi-block Newton-Krylov flow solver for the Euler equations*, in 17th AIAA Computational Fluid Dynamics Conference, AIAA–2005–5230, Toronto, Canada, June 2005.

[69] E. J. Nielsen, R. W. Walters, W. K. Anderson, and D. E. Keyes, *Application of Newton-Krylov methodology to a three-dimensional unstructured Euler code*, in 12th AIAA Computational Fluid Dynamics Conference, AIAA–95–1733, San Diego, California, United States, June 1995.

[70] J. Nordström and M. H. Carpenter, *Boundary and interface conditions for high-order finite-difference methods applied to the Euler and Navier-Stokes equations*, Journal of Computational Physics, 148 (1999), pp. 621–645.

[71] ——, *High-order finite difference methods, multidimensional linear problems, and curvilinear coordinates*, Journal of Computational Physics, 173 (2001), pp. 149–174.

[72] J. Nordström, J. Gong, E. van der Weide, and M. Svärd, *A stable and conservative high order multi-block method for the compressible Navier-Stokes equations*, Journal of Computational Physics, 228 (2009), pp. 9020–9035.

[73] T. A. Oliver and D. L. Darmofal, *An unsteady adaptation algorithm for discontinuous Galerkin discretizations of the RANS equations*, in 18th AIAA Computational Fluid Dynamics Conference, AIAA–2007–3940, Miami, Florida, June 2007.

[74] L. Osusky and D. W. Zingg, *A novel aerodynamic shape optimization approach for three-dimensional turbulent flows*, in 50th AIAA Aerospace Sciences Meeting and Aerospace Exposition, AIAA–2012–0058, Nashville, Tennessee, United States, Jan. 2012.

[75] M. Osusky, P. D. Boom, D. C. Del Rey Fernández, and D. W. Zingg, *An efficient Newton-Krylov-Schur parallel solution algorithm for the steady and unsteady Navier-Stokes equations*, in 7th International Conference on Computational Fluid Dynamics, ICCFD7–1801, Big Island, Hawaii, USA, July 2012.

[76] M. Osusky, J. E. Hicken, and D. W. Zingg, *A parallel Newton-Krylov-Schur flow solver for the Navier-Stokes equations using the SBP-SAT approach*, in 48th AIAA Aerospace Sciences Meeting and Aerospace Exposition, AIAA–2010–116, Orlando, Florida, United States, Jan. 2010.

[77] M. Osusky and D. W. Zingg, *A parallel Newton-Krylov-Schur flow solver for the Reynolds-Averaged Navier-Stokes equations*, in 50th AIAA Aerospace Sciences Meeting and Aerospace Exposition, AIAA–2012–0442, Nashville, Tennessee, United States, Jan. 2012.

[78] R. P. Pawlowski, J. N. Shadid, J. P. Simonis, and H. F. Walker, *Globalization techniques for Newton-Krylov methods and applications to the fully coupled solution of the Navier– Stokes equations*, SIAM Review, 48 (2006), pp. 700–721.

[79] A. Pueyo, *An Efficient Newton–Krylov Method for the Euler and Navier–Stokes Equations*, PhD thesis, University of Toronto, Toronto, Ontario, Canada, 1998.

[80] A. Pueyo and D. W. Zingg, *Efficient Newton-Krylov solver for aerodynamic computations*, AIAA Journal, 36 (1998), pp. 1991–1997.

[81] T. H. Pulliam, *Efficient solution methods for the Navier-Stokes equations*, tech. report, Lecture Notes for the von Kármán Inst. for Fluid Dynamics Lecture Series: Numerical Techniques for Viscous Flow Computation in Turbomachinery Bladings, Rhode-Saint-Genèse, Belgium, Jan. 1986.

[82] T. H. Pulliam and J. L. Steger, *Implicit finite-difference simulations of three-dimensional compressible flow*, AIAA Journal, 18 (1980), pp. 159–167.

[83] Y. Saad, *Iterative Methods for Sparse Linear Systems*, SIAM, Philadelphia, PA, second ed., 2003.

[84] Y. Saad and M. H. Schultz, *GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM Journal on Scientific and Statistical Computing, 7 (1986), pp. 856–869.

[85] Y. Saad and M. Sosonkina, *Distributed Schur complement techniques for general sparse linear systems*, SIAM Journal of Scientific Computing, 21 (1999), pp. 1337–1357.

[86] Y. Saad and H. A. van der Vorst, *Iterative solution of linear systems in the 20-th century*, Journal of Computational and Applied Mathematics, 123 (2000), pp. 1–33.

[87] P. Sagaut and M. Germano, *Large Eddy Simulation for Incompressible Flows*, Springer-Verlag, New York, NY, second ed., 2004.

[88] V. Schmitt and F. Charpin, *Pressure distributions on the ONERA-M6-wing at transonic Mach numbers*, tech. report, Office National d'Etudes et Recherches Aerospatiales, 92320, Chatillon, France, 1979.

[89] C.-W. Shu, *High-order finite difference and finite volume WENO schemes and discontinuous Galerkin methods for CFD*, International Journal of Computational Fluid Dynamics, 17 (2003), pp. 107–118.

[90] A. M. O. Smith and T. Cebeci, *Numerical solution of the turbulent boundary layer equations*, Tech. Report DAC 33735, Douglar Aircraft Division, 1967.

[91] P. R. Spalart and S. R. Allmaras, *A one-equation turbulence model for aerodynamic flows*, in 30th AIAA Aerospace Sciences Meeting and Exhibit, AIAA–92–0439, Reno, Nevada, United States, Jan. 1992.

[92] P. R. Spalart and C. L. Rumsey, *Effective inflow conditions for turbulence models in aerodynamic calculations*, AIAA Journal, 45 (2007), pp. 2544–2553.

[93] J. L. Steger, *Implicit finite difference simulation of flow about arbitrary geometries with application to airfoils*, AIAA–77–665, 1977.

[94] B. Strand, *Summation by parts for finite difference approximations for d/dx*, Journal of Computational Physics, 110 (1994), pp. 47–67.

[95] M. Svärd, M. H. Carpenter, and J. Nordström, *A stable high-order finite difference scheme for the compressible Navier-Stokes equations, far-field boundary conditions*, Journal of Computational Physics, 225 (2007), pp. 1020–1038.

[96] M. Svärd and J. Nordström, *A stable high-order finite difference scheme for the compressible Navier-Stokes equations, no-slip wall boundary conditions*, Journal of Computational Physics, 227 (2008), pp. 4805–4824.

[97] R. C. Swanson and E. Turkel, *On central-difference and upwind schemes*, Journal of Computational Physics, 101 (1992), pp. 292–306.

[98] P. G. Tucker, *Assessment of geometric multilevel convergence robustness and a wall distance method for flows with multiple internal boundaries*, Applied Mathematical Modeling, 22 (1998), pp. 293–311.

[99] P. G. Tucker, C. L. Rumsey, P. R. Spalart, R. E. Bartels, and R. T. Biedron, *Computations of wall distances based on differential approaches*, AIAA Journal, 43 (2005), pp. 539–549.

[100] J. L. van Ingen, *A suggested semi-empirical method for the calculation of the boundary layer transition region*, Tech. Report VTH–74, Delft University of Technology, 1956.

[101] J. C. Vassberg, *A unified baseline grid about the Common Research Model wing-body for the Fifth AIAA CFD Drag Prediction Workshop*, in 29th AIAA Applied Aerodynamics Conference, AIAA–2011–3508, Honolulu, Hawaii, United States, June 2011.

[102] J. C. Vassberg, M. A. DeHann, S. M. Rivers, and R. A. Wahls, *Development of a Common Research Model for applied CFD validation studies*, in 26th AIAA Applied Aerodynamics Conference, AIAA–2008–6919, Honolulu, Hawaii, United States, Aug. 2008.

[103] J. C. Vassberg, E. N. Tinoco, M. Mani, B. Rider, T. Zickuhr, D. W. Levy, O. P. Brodersen, B. Eisfeld, S. Crippa, R. A. Wahls, J. H. Morrison, D. J. Mavriplis, and M. Murayama, *Summary of the fourth AIAA CFD Drag Prediction Workshop*, in 28th AIAA Applied Aerodynamics Conference, AIAA–2010–4547, Chicago, Illinois, United States, June 2010.

[104] V. Venkatakrishnan, *Newton solution of inviscid and viscous problems*, AIAA Journal, 27 (1989), pp. 885–891.

[105] V. Venkatakrishnan and T. J. Barth, *Application of direct solvers to unstructured meshes for the Euler and Navier-Stokes equations using upwind schemes*, AIAA–89–0364, 1989.

[106] F. M. White, *Viscous Fluid Flow*, McGraw–Hill Book Company, New York, 1974.

[107] D. C. Wilcox, *Formulation of the k-omega turbulence model revisited*, AIAA Journal, 46 (2008), pp. 2823–2838.

Appendices

Appendix A

Flow Solution Input Files

In order to execute a flow solution, several input files are required.

Optional input files used during non-standard flow solutions, such as flow solutions on degenerate grids, are described in §A.3.

A.1 `input.param` file

A typical input file for a transonic turbulent flow solution can be seen below. All required parameters, as well as their optimal values, are:

```
&OPTIMIZER
   opt_method = 'flowsolve'
&END
&OPPTS
   op_pts% nopt = 1,
   op_pts% fsmachs(1) = 0.8395d0,
   op_pts% alphas(1)  = 3.060d0,
   op_pts% reno(1)    = 1.462d7
&END
&FILES
   grid_file_prefix = 'grid',
   output_file_prefix = 'results'
&END
&MESH
   GRID% DEGEN = .false.
&END
&PLANFORM
&END
&SOLVER
   NHALO = 2,
```

```
   DIABLO% D_TYPE = 1,
   DIABLO% FF_MU_FRICTION = .false.,

   DIABLO% ISTREAM = 1,     DIABLO% IGROUND = 3,
   DIABLO% VISCOUS = .true.,  DIABLO% VISCROSS = .true.,
   DIABLO% TURBULNT = .true.,

   DIABLO% IDMODEL = 1,     DIABLO% P_SWITCH = .true.,
   DIABLO% DIS2(1) = 2.0d0, DIABLO% DIS4(1) = 0.04d0,
   DIABLO% DIS2(2) = 2.0d0, DIABLO% DIS4(2) = 0.04d0,
   DIABLO% DIS2(3) = 2.0d0, DIABLO% DIS4(3) = 0.04d0,
   DIABLO% Vl = 0.025d0,    DIABLO% Vn = 0.25d0,

   SAT_%Vl = 0.025d0,       SAT_%Vn = 0.25d0,

   DIABLO% STRTUP_PREC = 'schur',
   DIABLO% NEWTON_PREC = 'schur',
   DIABLO% FRCHT_EPS = 1.d-10,

   DIABLO% APP_LEV_FIL = 2,
   DIABLO% LEV_FIL = 3,     DIABLO% PREC_dlf = 6.d0,
   DIABLO% SCHUR_ITS = 5,   DIABLO% SCHUR_TOL = 1d-2,
   DIABLO% REORDER = 1,

   DIABLO% NK_TIME = 7,     DIABLO% DT_MIN = 0.001d0,
   DIABLO% A = 0.001d0,     DIABLO% B = 1.30d0,
   DIABLO% BETA = 1.5d0,

   DIABLO% NK_STRTUP = 2,   DIABLO% DROP_TOL = 1.d-4,
   DIABLO% UPDAT_FRQ = 1,   DIABLO% STRTUP_TOL = 5d-2,
   DIABLO% NEWTON_TOL = 1d-2,

   DIABLO% NK_ITS = 200,    DIABLO% KRYLV_MAX = 70,
   DIABLO% REL_TOL= 1.d-12, DIABLO% ABS_TOL= 1.d-12
&END
```

For subsonic flow solutions, the following changes have to be made to the `input.param` file:

```
DIABLO% P_SWITCH = .false.
DIABLO% DIS2(1) = 0.0d0
DIABLO% DIS2(2) = 0.0d0
DIABLO% DIS2(3) = 0.0d0
DIABLO% Vn = 0.025d0
SAT_%Vn = 0.025d0
```

A.2 Grid-related files

In addition the `input.param` file, several additional grid-related files are required.

`grid.g`: a binary file, in plot3D format, containing the coordinates of the computational grid, organized by individual blocks.

`grid.con`: a text file containing the connectivity information for the grid. This file describes the interface connections between the blocks of the grid, in addition to the boundary conditions, such as the locations of the solid surface, symmetry, and farfield boundaries.

A.3 Auxiliary input files

A.3.1 Explicit trip line definition

When computing turbulent flows with prescribed laminar to turbulent transition locations, it is necessary to define the trip locations on the surface of the aerodynamic shape. In the current implementation, this is accomplished by including an input file with the trip line definition. Below, the example contents of the file, which has the name `grid.trip`, are shown for a three-dimensional solution over the ONERA M6 wing. Figure A.1 shows the position of the trip line along the top and bottom surfaces. For two-dimensional cases, a single point is specified for both the top and bottom surfaces.

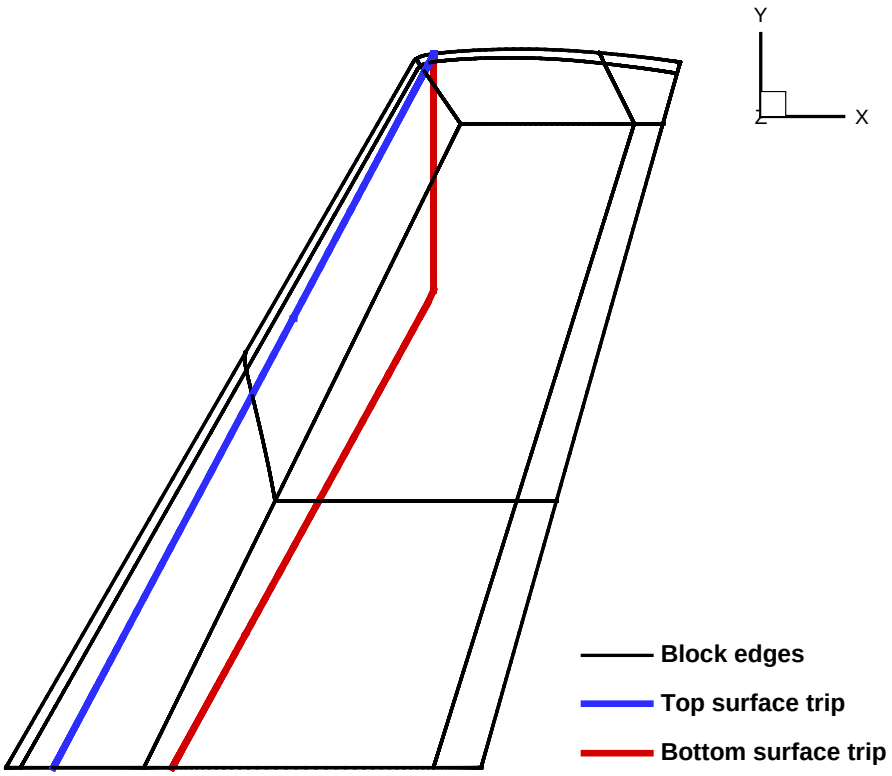


Figure A.1: Example trip line definition on ONERA M6 wing

`grid.trip` file contents:

```
 Turbulent trip line location file for Diablo
 number of definition points on upper surface
           2
 number of definition points on lower surface
           3

Upper surface point locations
------------------------------------------------------------
 xyz_stream       |       xyz_span
 -0.40            0.0
  0.40                    1.5

Lower surface point locations
------------------------------------------------------------
 xyz_stream       |       xyz_span
 -0.15                    0.0
  0.40                    1.0
  0.40                    1.5
```

A.3.2 Degenerate edge definition in 3D

When using degenerate grids, the following parameter has to be set in the `input.param` file:

```
GRID% DEGEN = .true.
```

In addition, a `grid.degen` file has to be included in the run directory. The file should be formatted as follows, defining the blocks, directions, and edge numbers of any degenerate edges:

`grid.degen` file contents:

```
 Grid degenerate edge information file for Jetstream
 number of edges
          12
---------------------
|  blk  | di | edge |
     1     3      1
     2     3      2
     5     3      1
     6     3      2
    66     3      3
    68     3      4
    78     3      3
    80     3      4
    83     3      3
    84     3      4
    87     3      3
    88     3      4
```

With the above specification of degenerate edges, it is necessary to define the direction, `di`, and edge numbering. Edge directions are defined based on the coordinate index that is non-constant along the edge. For example, a block edge with constant j- and k-indices, but a varying m-index, is taken as an edge in the ζ-direction. The three coordinate directions are are numbered as $(\xi, \eta, \zeta) = (1, 2, 3)$. The numbering of edges is based on the tangential directions for each edge direction, labelled as `it1` and `it2`, while the edge numbering is defined by the index values of the tangential directions, according to the conventions in Table A.1.

Table A.1: Grid direction and edge numbering conventions

(a) Directions

`di`	`it1`	`it2`
1	2	3
2	3	1
3	1	2

(b) Edge numbering

`edge`	`it1` index	`it2` index
1	1	1
2	1	max
3	max	1
4	max	max

Appendix B

Specialized Boundary Conditions

Several of the turbulence model verification cases required the use of specialized boundary conditions for temperature and pressure. These boundary conditions were not implemented using SATs, and were instead enforced using the "injection" method. This was done in order to fully match the inflow and outflow conditions implemented in the numerical algorithms used in the comparisons of Sections 6.1.1 and 6.1.2. The intent was to eliminate sources of discrepancy introduced by the inflow and outflow boundaries, focusing on the effectiveness of the SATs used for the solid surface boundary. The values of the residual, R, used at pertinent boundary nodes, overwrites the local SBP-SAT discretization with

$$R = J^{-1}\left(\mathbf{Q} - \mathbf{Q}_{\text{targ}}\right),$$

where $\mathbf{Q}$ is the local flow variable vector and $\mathbf{Q}_{\text{targ}}$ is a target flow variable vector based on the particular boundary condition. The metric Jacobian, J, is omitted for the turbulence model residual.

B.1 Total temperature/pressure boundary

The inflow condition requires the specification of total temperature, T_T, total pressure, p_T, and tangential velocity, V_t, values, with all remaining parameters extrapolated from the interior of the computational domain, taken here as the values from the boundary-adjacent node. For example, on a boundary with a j-index value of 1, a Riemann invariant from the $j+1$ node is required:

$$R_{1,j+1} = \left(V_n - \frac{2a}{\gamma - 1}\right)_{j+1},$$

where V_n is the velocity normal to the boundary.

The equation relating total and static temperature is

$$\frac{T_T}{T} = 1 + \frac{\gamma - 1}{2}M^2 = 1 + \frac{\gamma - 1}{2}\frac{u^2}{a^2},$$

into which we can substitute $u = V_n$ and $a^2 = T$. By making use of the Riemann invariant definition, and expanding all expressions, we obtain the following equation for $\sqrt{T}$:

$$\left(1 + \frac{2}{\gamma - 1}\right) T + 2R_{1,j+1}\sqrt{T} + \frac{\gamma - 1}{2}\left(R_{1,j+1}\right)^2 - T_T = 0.$$

Once a temperature value is obtained, the Mach number is calculated from

$$M^2 = \left(\frac{T_T}{T} - 1\right)\frac{2}{\gamma - 1},$$

which can be used to obtain the local value of pressure:

$$p = \frac{p_T}{\left(1 + \frac{1}{2}\gamma M^2\right)^{\frac{\gamma}{\gamma - 1}}}.$$

The local value of T also provides sound speed (in nondimensional form) as $a = \sqrt{T}$, leading to density, $\rho = \gamma p a^{-2}$.

Finally, we can obtain the value of the local normal velocity component, V_n, using the expression given above for the Riemann invariant, which remains constant between the boundary and boundary-adjacent nodes. With a prescribed value of the tangential velocity, V_t, typically set to 0, we can form the local velocity vector, $\langle u, v, w \rangle$, and the complete target vector for the Navier-Stokes equations as

$$\mathbf{Q}_{\text{targ}} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \frac{p}{\gamma - 1} + \frac{1}{2}\rho\left(u^2 + v^2 + w^2\right)^2 \end{bmatrix}.$$

The turbulence variable target is set to the free-stream value of $\tilde{\nu}$.

B.2 Static pressure boundary

The outflow boundary requires the specification of a static pressure value, p_S, with all remaining parameters extrapolated from the interior of the computational domain, again taken as values from the boundary-adjacent node. Here, we require two Riemann invariants from the boundary-adjacent node, namely

$$R_{2,j+1} = \left(V_n + \frac{2a}{\gamma - 1}\right)_{j+1}, \qquad R_{3,j+1} = \left(\frac{p}{\rho^\gamma}\right)_{j+1}.$$

The local density can now be obtained as

$$\rho = \left(\frac{p_S}{R_{3,j+1}} \right)^{\frac{1}{\gamma}},$$

which allows us to obtain a local value for the speed of sound

$$a = \sqrt{\frac{\gamma p_S}{\rho}}.$$

The tangential velocity, V_t, is extrapolated from the boundary-adjacent node, and the normal velocity is obtained from

$$V_n = R_{2,j+1} - \frac{2a}{\gamma - 1}.$$

With these values, the $\langle u, v, w \rangle$ vector can be obtained. We can finally form the target vector as

$$\mathbf{Q}_{\text{targ}} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \frac{p_S}{\gamma - 1} + \frac{1}{2}\rho \left(u^2 + v^2 + w^2 \right)^2 \end{bmatrix}.$$

The turbulence variable target is set to the local value of $\tilde{\nu}$.

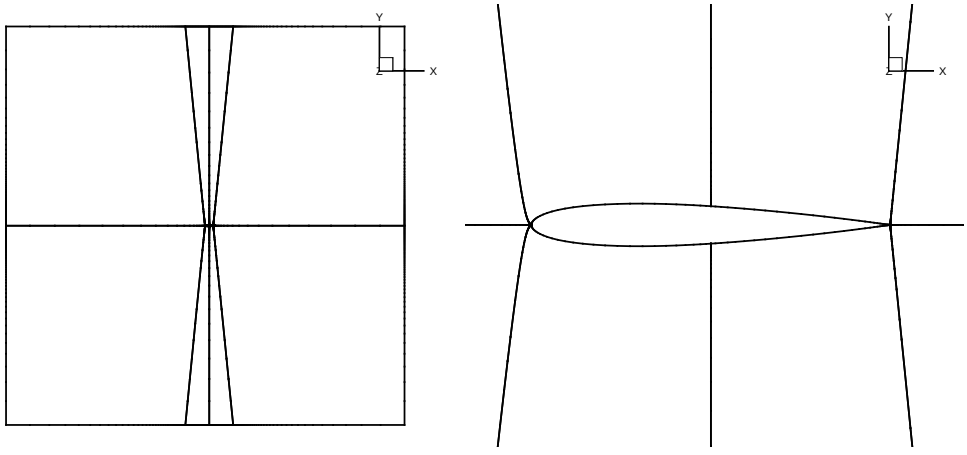
Appendix C

Grid Topology Examples

The following are examples of grid topologies. Their usage throughout this thesis is summarized in Table C.1.

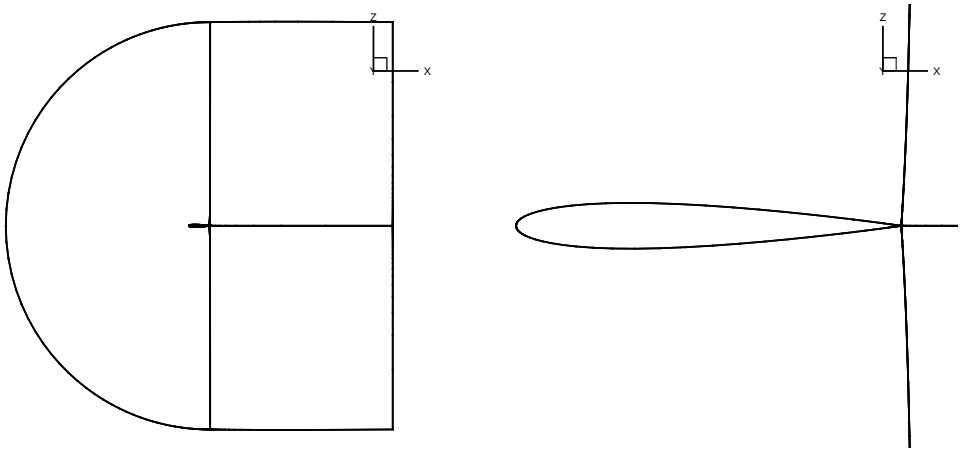
Table C.1: Grid topology usage

Topology	Usage
H	Algorithm optimization grids 1 and 2
C	§6.1.3
Y-H	Algorithm optimization grid 3, §6.2.1, §6.2.3
H-O	§6.2.4
O-O	Algorithm optimization grid 4, §6.2.2, §6.3



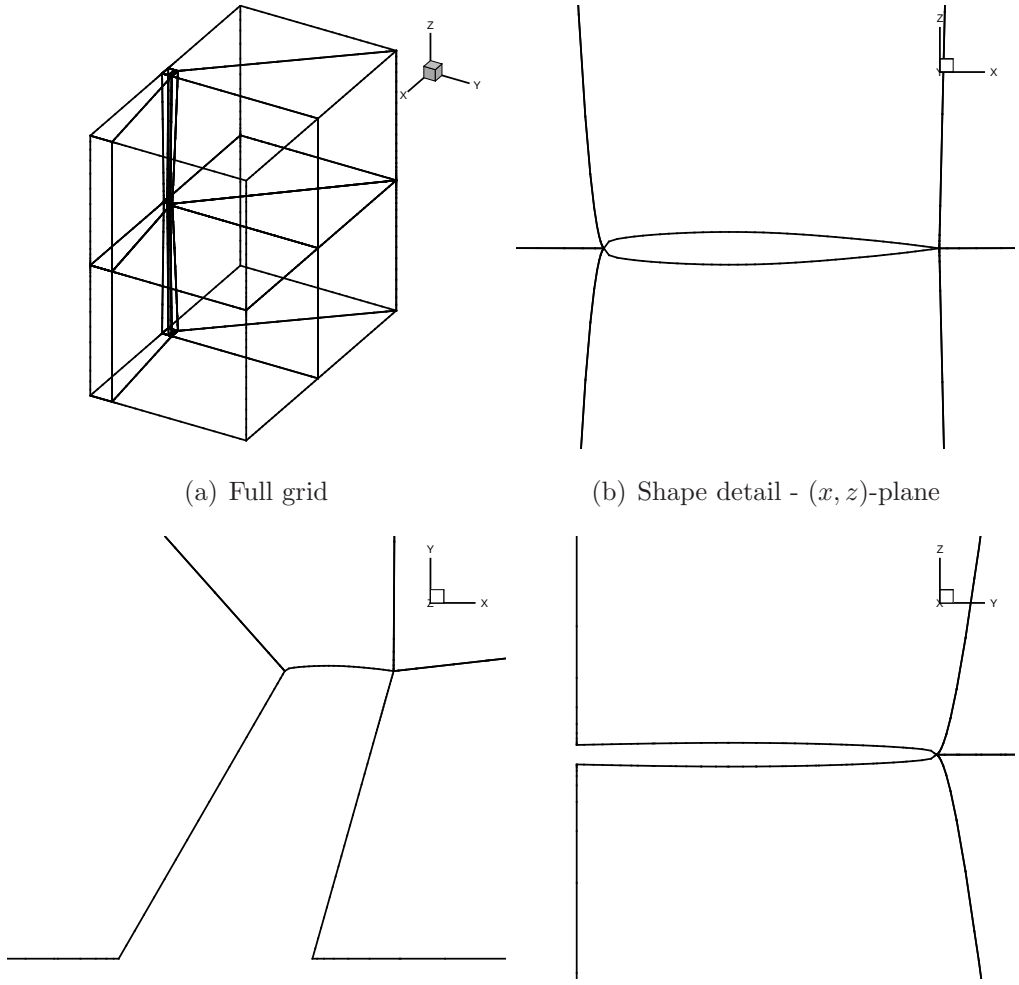
(a) Full grid (b) Airfoil detail

Figure C.1: 2D H topology grid (NACA0012 airfoil, 8 blocks)



(a) Full grid (b) Airfoil detail

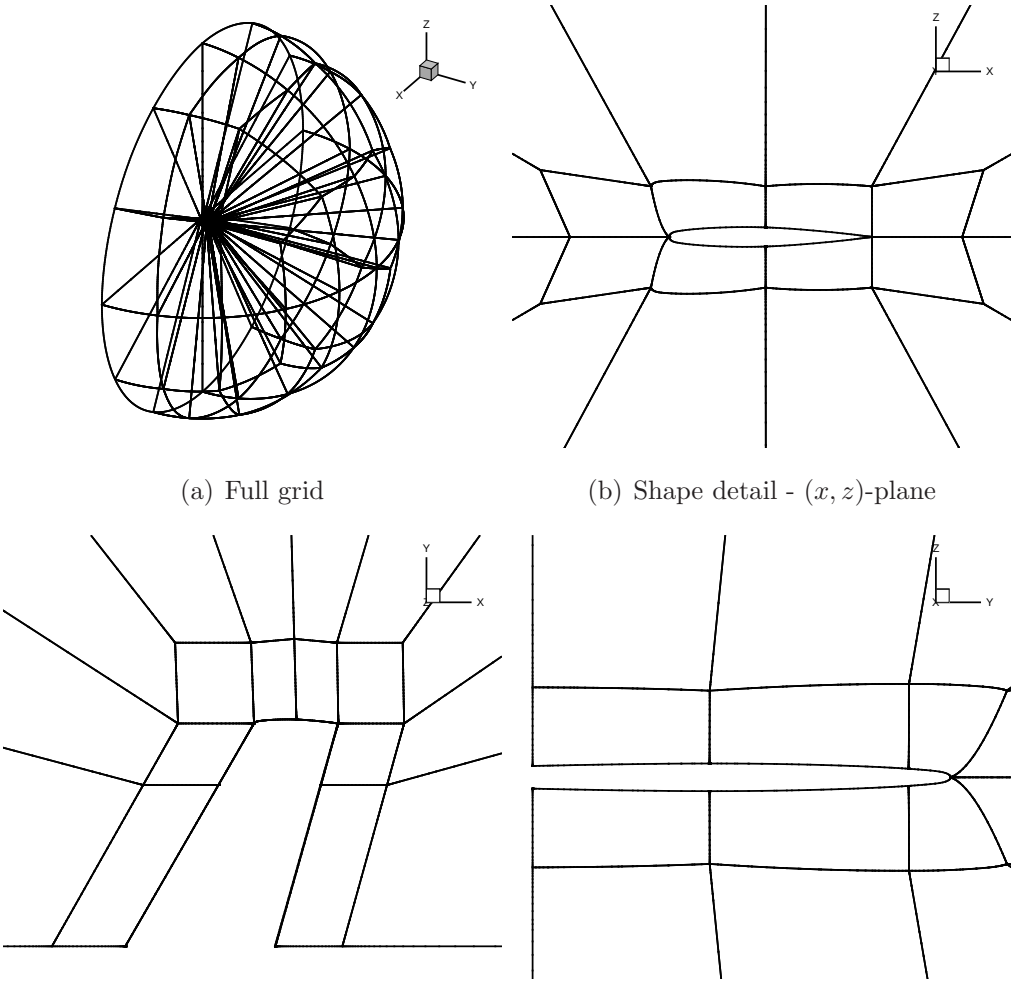
Figure C.2: 2D C topology grid (NACA0012 airfoil 3 blocks)



(a) Full grid (b) Shape detail - (x, z)-plane

(c) Shape detail - (x, y)-plane (d) Shape detail - (y, z)-plane

Figure C.3: 3D Y-H topology grid (ONERA M6 wing, 12 blocks)



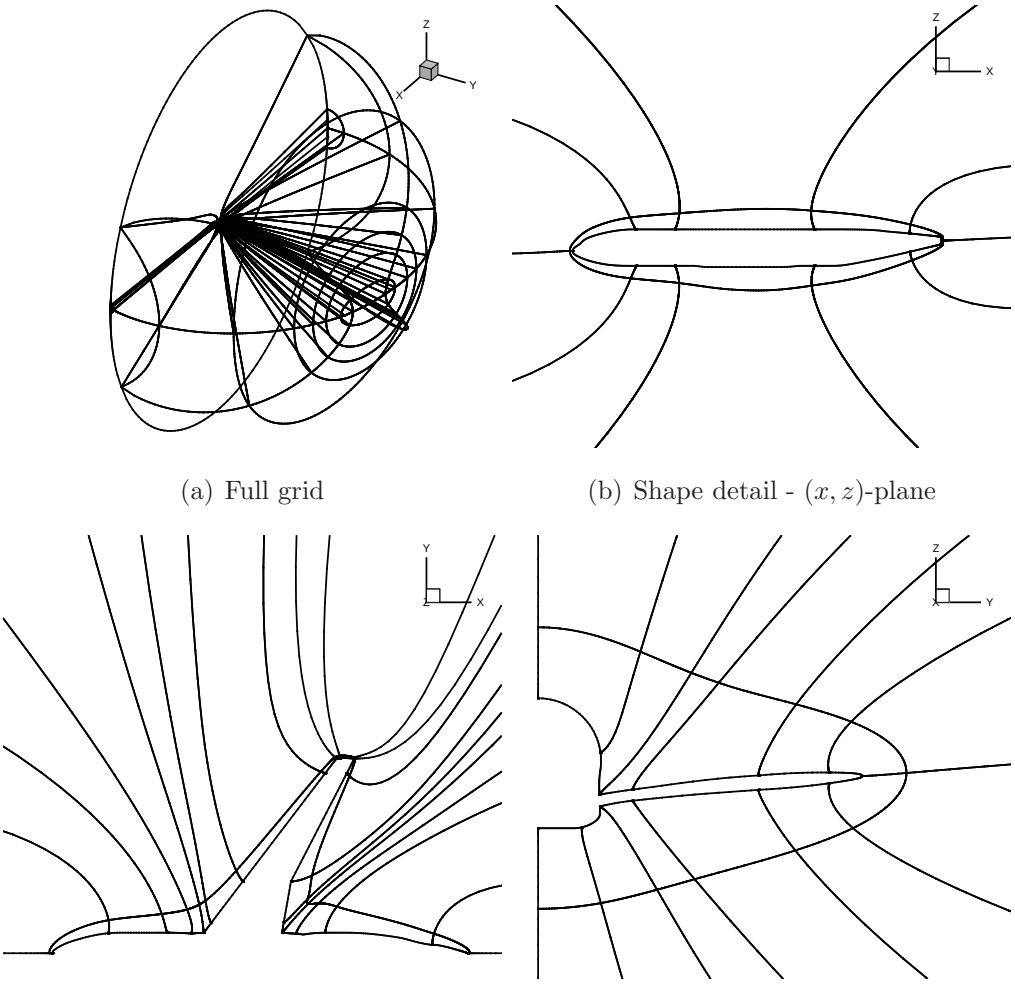
(a) Full grid

(b) Shape detail - (x, z)-plane

(c) Shape detail - (x, y)-plane

(d) Shape detail - (y, z)-plane

Figure C.4: 3D H-O topology grid (ONERA M6 wing, 68 blocks)



(a) Full grid

(b) Shape detail - (x, z)-plane

(c) Shape detail - (x, y)-plane

(d) Shape detail - (y, z)-plane

Figure C.5: 3D O-O topology grid (CRM wing-body, 88 blocks)

Appendix D

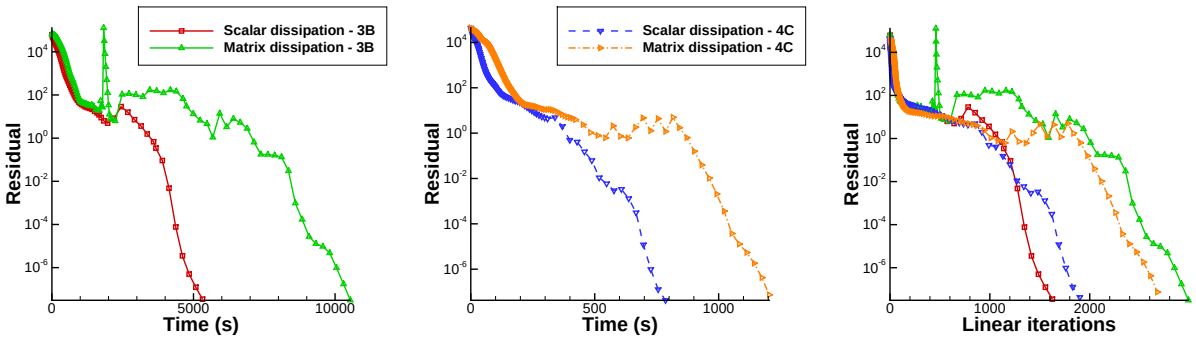
Numerical Dissipation Model

The purpose of adding artificial dissipation to the discretization of the governing equations is to damp high frequency modes that could result in the divergence of the algorithm. With this in mind, it is necessary to strike a balance in adding enough artificial dissipation to stabilize the convergence process, but not so much that it has a substantial impact on the converged steady-state solutions.

The effect of using the scalar or the matrix dissipation model on the accuracy of the converged solution can clearly be seen in the grid convergence results submitted to DPW5 (§6.2.2). While both models result in nearly identical grid-converged values of C_D (Figure 6.13), the profile of the matrix dissipation model curve is flatter, with coarser grid results possessing a drag coefficient closer to the grid-converged value. This is due to the lower amount of artificial dissipation the matrix dissipation model introduces into the discretization. Hence, the matrix dissipation model produces more accurate results for a given grid resolution, which is most noticeable on the coarser grid levels.

The use of the matrix dissipation model, however, results in decreased robustness for the solution algorithm. Steps need to be taken in order to prevent destabilization of the solution process in the early iterations. To this end, a slower ramping of the time-step (lower value of b) during the approximate-Newton phase, as was the case with the matrix dissipation solutions obtained for DPW5, is required. This in turn results in slower residual convergence and longer computational time.

Additionally, the reduced diagonal dominance of the linear system can result in destabilizing updates. This was demonstrated in the dissipation lumping factor study, presented in §5.3.3. Not only are smaller time step values required during the start-up phase, converged solutions are obtained for a narrower range of dissipation lumping factors. Figure D.1 presents a comparison of the convergence histories for both three-dimensional cases presented in §5.3.3 (3B and 4C), using the $\sigma_{\text{dlf}} = 5$ for the scalar dissipation model and $\sigma_{\text{dlf}} = 12$ for the scalar dissipation model. As expected, the scalar dissipation model converges faster, both in terms of time and linear iterations, but the extent to which the convergence differs is somewhat case- and algorithm-parameter-dependent. For example, running the scalar dis-



(a) Case 3B - Time (b) Case 4C - Time (c) Linear iterations

Figure D.1: Residual convergence for dissipation model comparison

Table D.1: Force coefficients for scalar and matrix dissipation model comparison

Case	C_L			C_D		
	scalar diss.	matrix diss.	rel. error	scalar diss.	matrix diss.	rel. error
3B	0.26796	0.26944	0.55%	0.017355	0.017130	1.31%
4C	0.50990	0.50883	0.21%	0.025896	0.025626	1.05%

sipation model with $\sigma_{\text{dlf}} = 12$ would result in slower convergence, with the total CPU time approaching the matrix dissipation results.

Table D.1 presents the values of the lift and drag coefficients obtained for scalar and matrix dissipation models, along with the relative error*. It should be noted that while the grids used for these cases do not represent particularly fine meshes, the maximum relative error does not exceed 1.5%. Since the solutions were computed at the same flow conditions, the results possess distinct lift coefficients. Therefore, the relative error would change for cases where a specific lift is required, such as for the DPW5 grid convergence study. For case 3B, a matching lift for scalar dissipation, and the corresponding increase in drag, would result in larger error, while for case 4C, matching the lift would decrease the error in drag.

While the use of the matrix dissipation model results in reduced robustness of the flow solution algorithm, it is possible to obtain converged solutions, albeit for a narrower range of solution algorithm parameters. For the application of the current flow solution algorithm within an optimization framework, the increased robustness of the scalar dissipation model is critical. However, the higher accuracy of the matrix dissipation model, particularly for coarser grids, provides incentive to investigate the robustness of the implementation and improve it such that the model can be used for optimizations with more confidence.

*Error calculated relative to the matrix dissipation model result

Appendix E

Two-Dimensional High-Lift Configurations

In order to highlight the high-lift calculation capabilities of the algorithm, I have focused on two airfoil geometries previously used in the verification of a two-dimensional Newton-Krylov algorithm [18]. With that particular solver, modifications were required to the turbulent time step in order to efficiently converge the high-lift configuration cases. The purpose of these results is to demonstrate the ability of the current algorithm to converge flows possessing substantial separation with the standard spatial discretization and solution algorithm parameters.

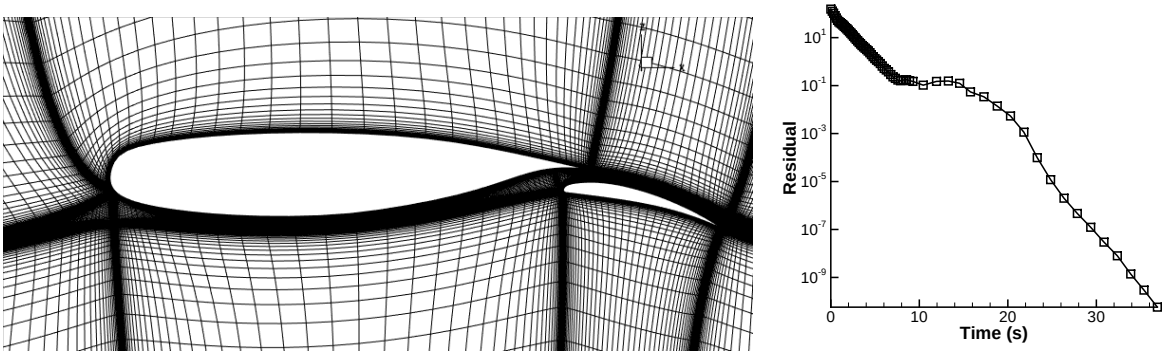
NLR two-element airfoil

The NLR two-element airfoil consists of an airfoil with a flap, as seen in Figure E.1(a). The following flow conditions are used:

$$M = 0.30,\ Re = 2.51 \times 10^6,\ \alpha = 6.0^\circ.$$

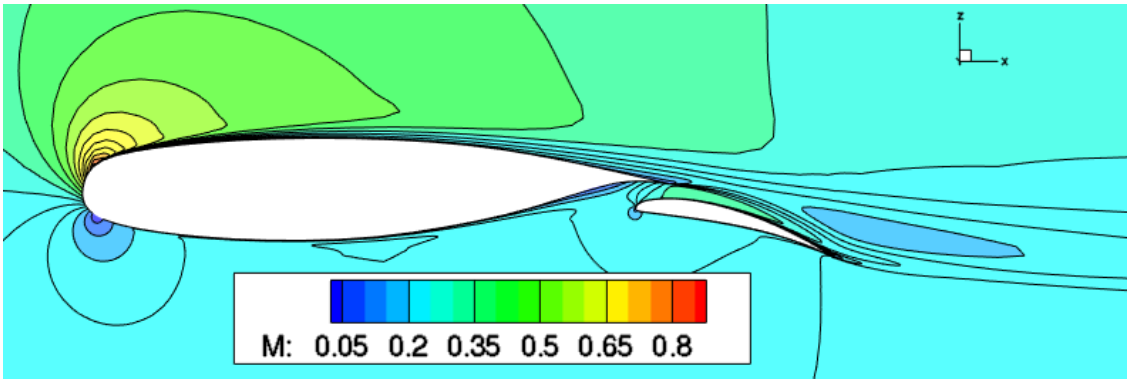
The Reynolds number is based on the chord length of the airfoil with flap retracted. The grid used for this case consists of 13 blocks, with a total of 33 thousand nodes. The off-wall distance of 2.0×10^{-5} results in an average y^+ value of 2.4. The flow solution was computed with the scalar dissipation model, converging 12 orders of magnitude in 37 seconds using 13 processors. The convergence history, plotted versus time, can be seen in Figure E.1(b).

This case provides a significant challenge for the solution algorithm due to the flow interaction between the main airfoil and flap, which can be seen in the Mach number contours of Figure E.2(a). Additionally, these flow conditions result in substantial turbulence, with turbulent viscosity values in excess of 1300. The contours of μ_t can be seen in Figure E.2(b).

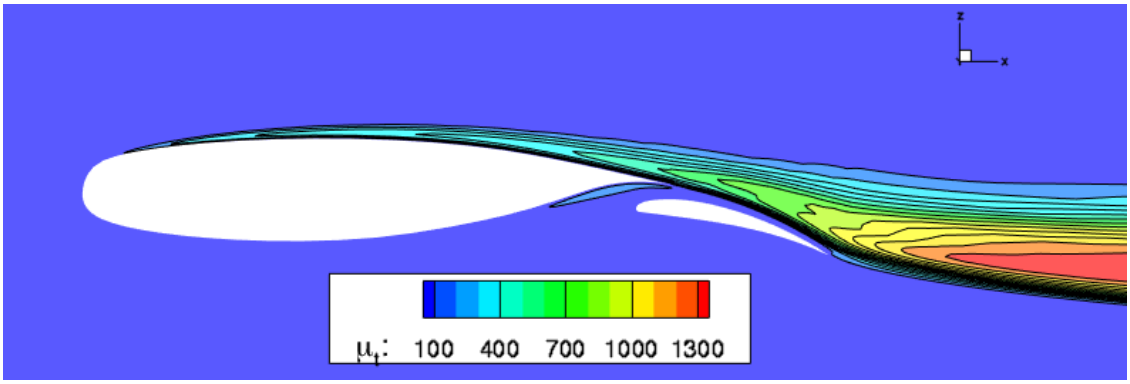


(a) Grid and geometry (b) Convergence history

Figure E.1: Grid and residual convergence for NLR airfoil with 13 processors



(a) Mach number

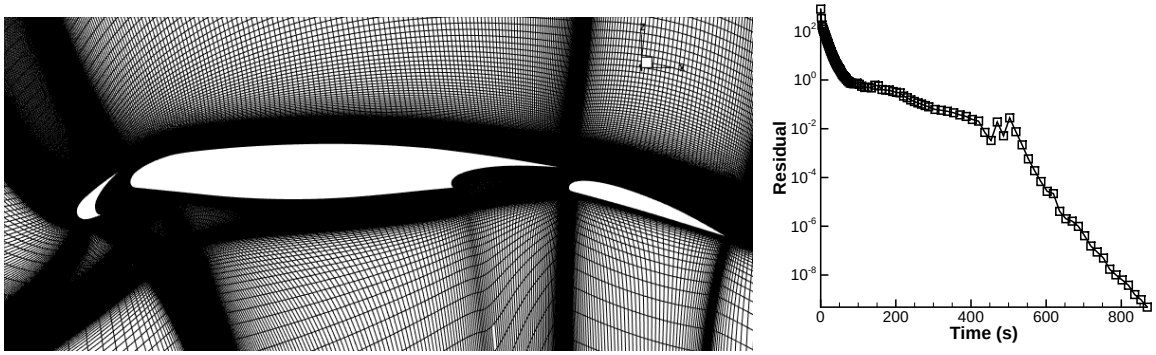


(b) Turbulent viscosity

Figure E.2: NLR airfoil flow-field contours

AGARD A2 three-element airfoil

The AGARD A2 airfoil, originally documented by Moir [64], provides a challenging case, with both a flap and slat included with the main airfoil body. The flow conditions are



(a) Grid and geometry (b) Convergence history

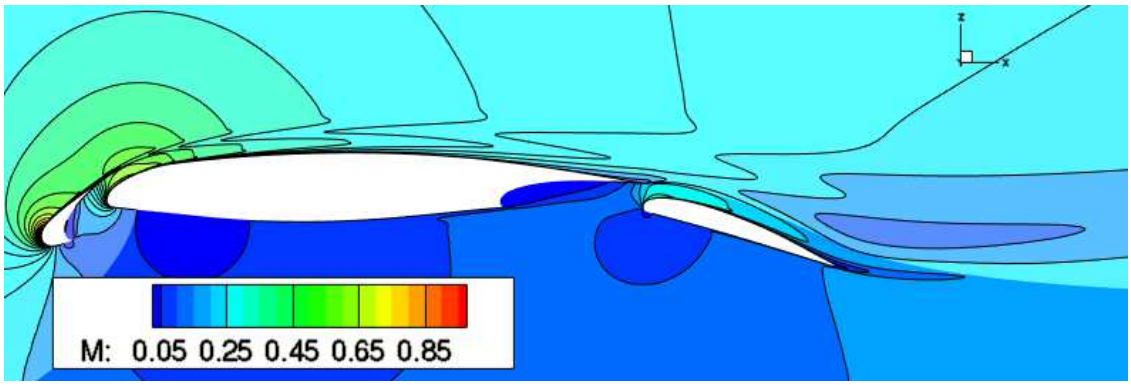
Figure E.3: Grid and residual convergence for AGARD A2 airfoil with 27 processors

$$M = 0.197,\ Re = 3.52 \times 10^6,\ \alpha = 20.0^\circ.$$

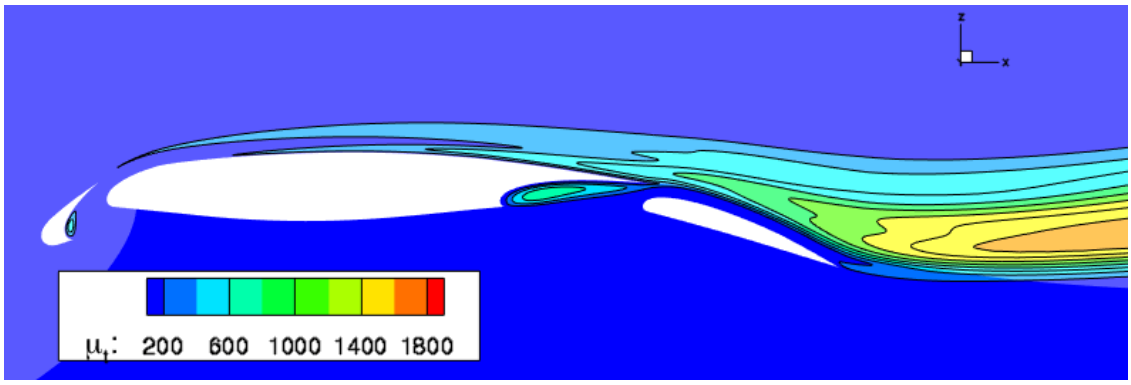
The Reynolds number is based on the chord length of the airfoil with flap and slat retracted The grid used for this case consists of 27 blocks, with a total of 255 thousand nodes, and an off-wall distance of 1.2×10^{-6}, resulting in an average y^+ value of 0.24. The flow solution was computed with the scalar dissipation model, converging 12 orders of magnitude in 869 seconds using 27 processors. The grid and convergence history for this case can be seen in Figure E.3, highlighting the effectiveness of the algorithm for difficult computational cases.

Figure E.4 shows the Mach number and turbulent viscosity contours for this case. As expected, the high angle of attack and the three-element geometry result in a complex flow pattern. Not only is there substantial interaction between the main airfoil and the flap, the slat located upstream of the main airfoil produces a small wake that interacts with the flow over the upper surface of the airfoil. Additionally, a substantial recirculation region is visible in the cut-out region on the lower surface of the airfoil near the trailing edge, along with a smaller separation in the cove of the slat.

It should be stressed that the AGARD A2 flow solution, along with the NLR two-element airfoil case, were run with default solver parameters, without any special time step modifications for the turbulence model. These cases demonstrate the robustness of the SBP-SAT discretization for challenging high-lift cases.



(a) Mach number



(b) Turbulent viscosity

Figure E.4: AGARD A2 airfoil flow field contours

Appendix F

Trip Location Enforcement Approaches

While not part of the goals of this thesis, transition prediction is an important capability of a solution algorithm, especially if used as part of an optimization tool. Accurately modeling laminar-to-turbulent transition and incorporating this information into the optimization process can leverage laminar flow regions in substantially decreasing the drag experienced by an aerodynamic shape.

The Spalart-Allmaras turbulence model has the capability of enforcing specific laminar-to-turbulent transition locations through the f_{t1} source terms. The terms seed the flow at specific nodes to trigger an increase in eddy viscosity. However, this approach does not provide any control over the transition length, which can have a substantial impact on flow characteristics and is often incorporated into transition prediction approaches.

An alternate approach to the use of the f_{t1} trip terms involves the implementation of an intermittency function that can be used to govern the transition properties. While not previously applied to the current turbulence model, the work of Krumbein [51] can be used to enforce transition without the f_{t1} terms. Instead, the intermittency function is used to ramp the turbulent viscosity over a specified percentage of the chord length. The turbulence model assumes the fully turbulent configuration, with $f_{t1} = 0$, while the intermittency function is applied in the calculation of μ_t and the production term, P:

$$\mu_t = \max\left[\gamma(x)\rho f_{v1}\tilde{\nu},\ \mu_{t,\text{ff}}\right], \tag{F.1}$$

$$P = \gamma(x)\frac{c_{b1}}{Re}[1 - f_{t2}]\tilde{S}\tilde{\nu}, \tag{F.2}$$

where

$$\gamma(x) = 1 - \exp\left(-0.412\xi^2\right), \tag{F.3}$$

$$\xi = \left(x - x_{\text{tr}}^{\text{beg}}\right)/\lambda, \tag{F.4}$$

$$\lambda = \left(x_{\text{tr}}^{\text{end}} - x_{\text{tr}}^{\text{beg}}\right)/3.36, \tag{F.5}$$

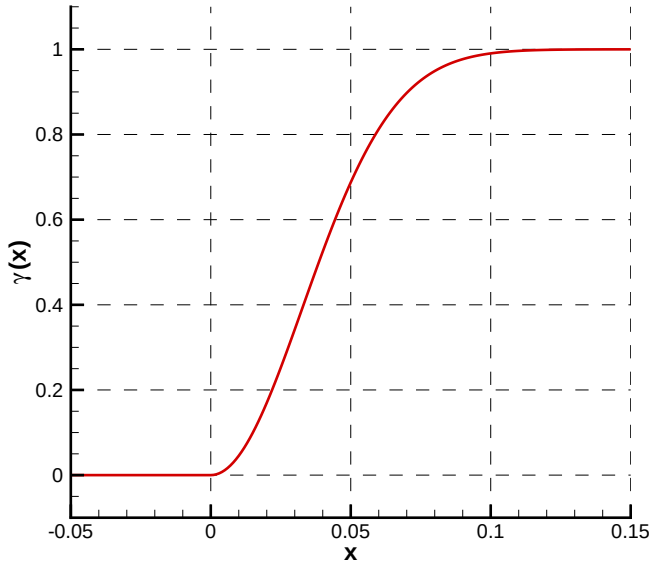
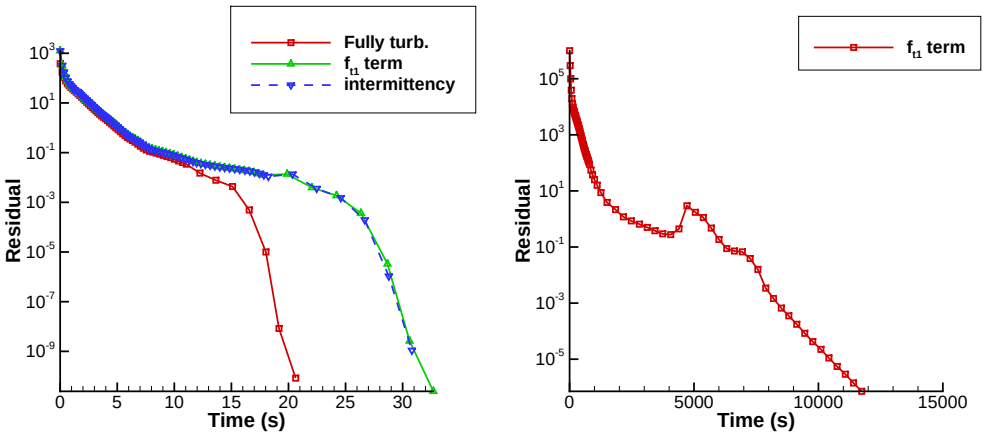


Figure F.1: S-curve intermittency function with transition length of 0.1

and $x_{\text{tr}}^{\text{beg}}$ and $x_{\text{tr}}^{\text{end}}$ define the beginning and end of the transition region. The values of x have been nondimensionalized by the chord length. The value of $\gamma(x)$ is zero upstream of the trip location, requiring the use of the "max" function to ensure a minimum turbulence value of the farfield turbulent viscosity, $\mu_{t,\text{ff}}$, is present, with the exception of solid surface nodes. No effective modification is applied to turbulent viscosity values downstream of the transition region, where the intermittency function takes on a value of 1.0 asymptotically. Figure F.1 provides an example of the form of the intermittency function for a trip length of 0.1 non-dimensional units.

This approach requires a supplied value of the transition length. While transition prediction approaches may inherently possess this information, no such capability exists within the current algorithm. Hence it is up to the user to specify the percentage of chord over which transition should occur. This relies on user expertise and knowledge of laminar-to-turbulent transition theory. However, future work on transition prediction can leverage the transition length capability of this approach, and the purpose of the current work is to investigate the use of the intermittency function trip location specification in the current algorithm. At the moment, transition length is specified as 10% of the chord length, starting at a user-specified trip location on the upper and lower surfaces of the aerodynamic shape being considered, either airfoils or wings.

The impact of the trip location enforcement approaches on convergence and steady-state solutions was analyzed for both two- and three-dimensional flows, using cases 2D and 3D,



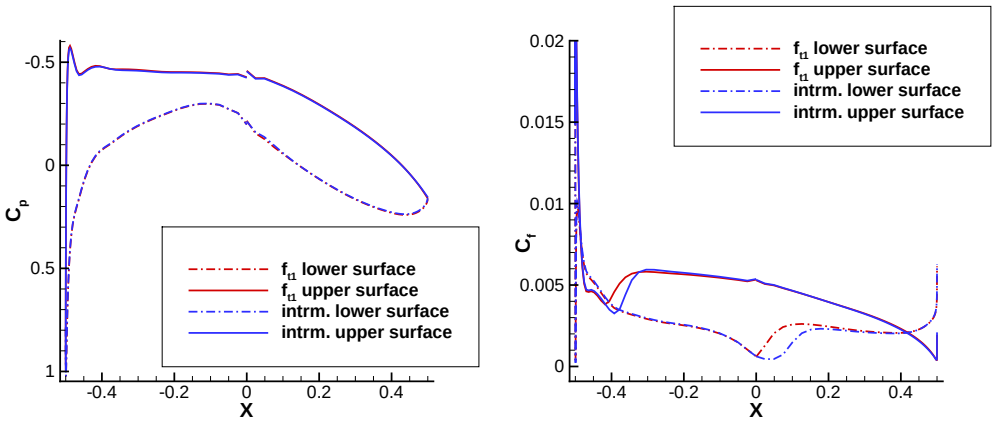
(a) Case 2D (b) Case 3D

Figure F.2: Convergence histories for turbulence model trip enforcement study

respectively, defined as part of the Algorithm Optimization chapter in §5.1. In the two-dimensional case, the trip locations were specified at 10% for the upper surface and 55% for the lower surface. The three-dimensional case uses straight trip location lines on the upper and lower surface, located at approximately 3 and 5% local chord, respectively, following the sweep of the leading edge of the ONERA M6 wing. Details of three-dimensional trip line definition can be found in Appendix A.3.1. For the intermittency function approach, the transition length was set to a constant of 10% root chord length.

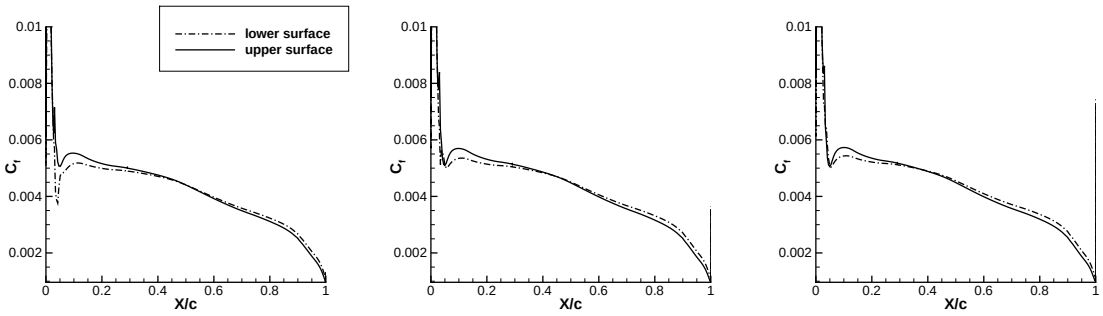
Figure F.2 presents the convergence histories for both cases, making use of either the f_{t1} trip term or intermittency function explicit trip location enforcement. The two-dimensional case was also computed with fully turbulent flow specified, and shows that explicitly tripped flow solutions do not add a significant amount of time to the solution process. While convergence for tripped flow solutions is not affected in an appreciable way for the two-dimensional flow (case 2D), the different means of enforcing transition can clearly be seen in the skin friction on the surface of the airfoil, shown in Figure F.3. The intermittency approach results in a more gradual shift in C_f from the laminar to turbulent regions, and is more precise in the enforcement of the transition point; the f_{t1} approach tends to trip the flow slightly upstream of the specified transition location. The coefficient of drag differs by less than 0.4% between the two approaches, and is approximately 20% lower than for the fully turbulent case. Minimal change is observed in the profiles of the coefficient of pressure.

Converging explicitly tripped three-dimensional solutions causes more difficulty, likely due to the complex flow features that are present in three-dimensions, particularly in regions of laminar-to-turbulent transition. In fact, the intermittency function approach failed to



(a) Coefficient of pressure (b) Coefficient of skin friction

Figure F.3: Coefficients of pressure and skin friction for case 2D



(a) 20% span (b) 60% span (c) 80% span

Figure F.4: Coefficients of skin friction for case 3D (with f_{t1} trip term)

result in a steady-state solution, with the residual diverging once the inexact-Newton stage was reached. While this approach shows great promise for two-dimensional flows, more work is necessary to understand its impact on three-dimensional flow. Figure F.4 presents sectional cuts of the distribution of C_f on the surface of the wing, clearly highlighting the regions of transition for the f_{t1} trip enforcement approach, near the leading edge of the wing.

Due to the current limitations of the intermittency approach, the f_{t1} trip term laminar-to-turbulent transition enforcement is the recommended approach for explicitly tripped flow.