MASSIVELY-PARALLEL SPECTRAL ELEMENT ALGORITHM DEVELOPMENT FOR HIGH SPEED FLOWS

A Dissertation

by

JOSHUA LANE CAMP

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

DOCTOR OF PHILOSOPHY

Chair of Committee,	Andrew Duggleby
Committee Members,	Devesh Ranjan
	Robert Handler
	Edward White
Head of Department,	Andreas Polycarpou

December 2013

Major Subject: Mechanical Engineering

ABSTRACT

The need to reduce both the time and cost of product design has allowed numerical analysis to play an ever-increasing role in design cycle analysis. This is particularly true in the aerospace industry, where the use of computational fluid dynamics can help reduce the need for costly prototype testing. Due to the extremely high computational costs associated with simulating complex industrial flows directly, most modern simulation tools employ solvers that rely heavily on turbulence modeling. However, the combination of modern supercomputers and algorithms that can take full advantage of them allows for higher fidelity solvers, with reduced dependence on turbulence modeling, to be included in design cycle analysis.

This work employs the discontinuous Galerkin spectral element method in a solver designed for high fidelity simulations in the subsonic and transonic flow regimes. The algorithm is implemented using NEK5000, an open-source incompressible spectral element solver, as a code base. Details of the algorithm are given, and the code is validated against several canonical inviscid and viscous test cases. The validation cases show that the code is accurate, stable, and a good performer on supercomputers. The new solver is then used to study the effectiveness of a cylindrical film cooling hole. The results show a much improved prediction capability of film cooling effectivness as compared to previous low-Mach simulation results. The algorithm is proven to produce quality large-eddy simulation data in a time frame accessible for design cycle analysis. At the end, a suggested direction for future development of the algorithm is discussed, with a focus on how to improve the stability and performance of the solver.

DEDICATION

To my loving wife

ACKNOWLEDGEMENTS

I would first like to thank my advisor, Dr. Andrew Duggleby, for the support he has given me throughout my tenure here at Texas A&M University. I know that the knowledge and the skills I have gained under your wing will benefit me greatly in attaining my future career goals. I can only hope to repay the great debt I owe to you someday. I would also like to thank Yuval "UV" Doran and Mike Martell. Although computational fluid dynamics and numerical analysis is not in your area of expertise, you both were willing to listen to my ideas. I cannot count how many times a major breakthrough in the development has come from one of you asking the right questions. I would like to thank Dr. Paul Fischer at Argonne National Labs for the many fruitful conversations over the years, whether that be in person or through email, that have significantly aided me in getting my code up and running. The support and love I have received from my family has been crucial to my graduate success, and I hope you know it has not gone unappreciated. Finally, I would like to thank my beautiful wife Kaleigh. This has been a long process for both of us, through the sleepless nights, frustrations, and triumphs. Your love and continued support has helped me in ways that cannot possibly be described in words. I love you, and I look forward the next phase in our lives.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

The engineering design cycle is an iterative process. A concept is tested against a set of constraints, and the design is adjusted and re-tested if the constraints are not met. A simple sketch of a design cycle process is shown in Figure 1.1; however, this does not give the complete picture. Due to a globally competitive market, it is no longer sufficient to simply meet design constraints. A successful product must also perform its duty in the most optimal fashion possible [1]. An example of this is a consumer vehicle. Most vehicles have the ability to transport passengers from one destination to another safely. Top sellers will also be optimal in other attribute(s) such as the price point or fuel economy. Thus, in addition to testing design variants against hard constraints, the designer must rank satisfactory designs according to an optimization objective (commonly known as a cost function [1]).

As a product's success ultimately depends on its level of optimization, it is beneficial for the product to undergo many iterations in the design cycle before advancing to the market. However, the role of time as a factor should not be understated; it is advantageous in most circumstances to bring a concept to market in a reasonably fast timeframe. One method of accelerating the design phase is to study design iterations on the computer using numerical analysis.

1.1 Numerical Analysis in the Design Cycle

With the advances in modern computing power, the testing phase of the design cycle has trended largely towards numerical modelling [2, 3]. Prototype testing can be an expensive and time consuming process; therefore, numerically exploring the design space can help the designer arrive at an optimal solution faster and at a lower cost. Although numerical analysis will never fully replace prototype testing, it can

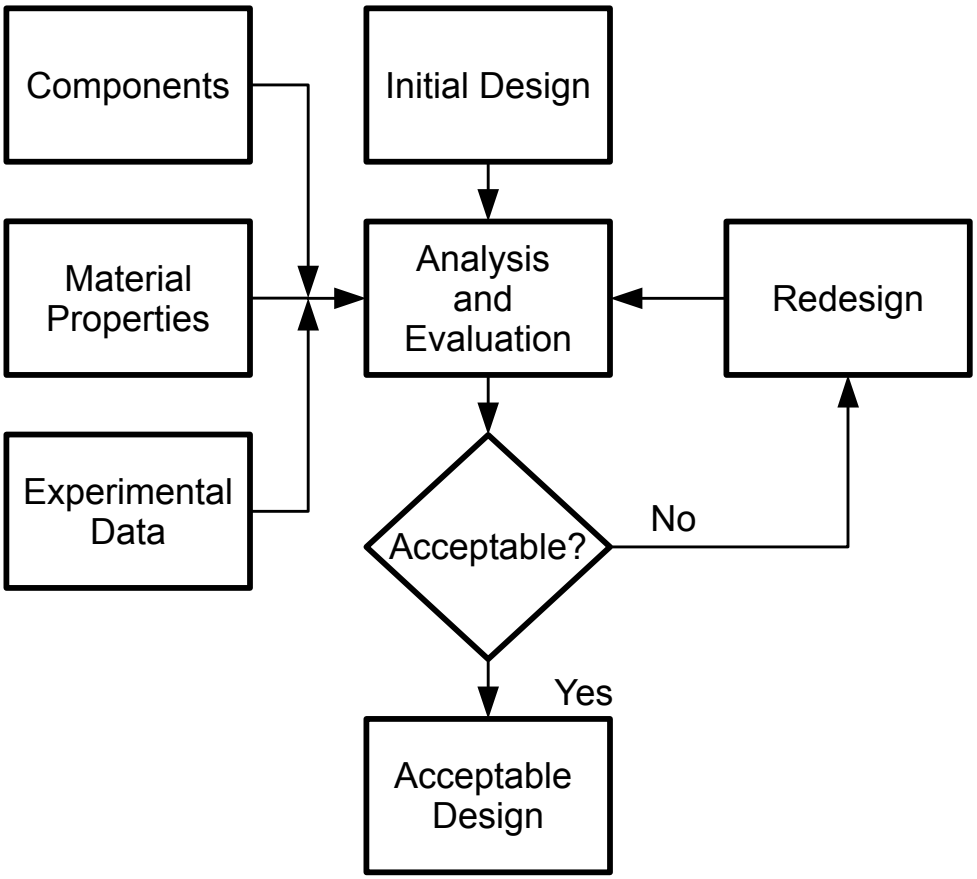


Figure 1.1: Flowchart of basic design cycle. The design cycle resembles a feedback loop, as evaluation of a design informs subsequent redesigns until the final forms meets all goals and contraints. Adapted from [1].

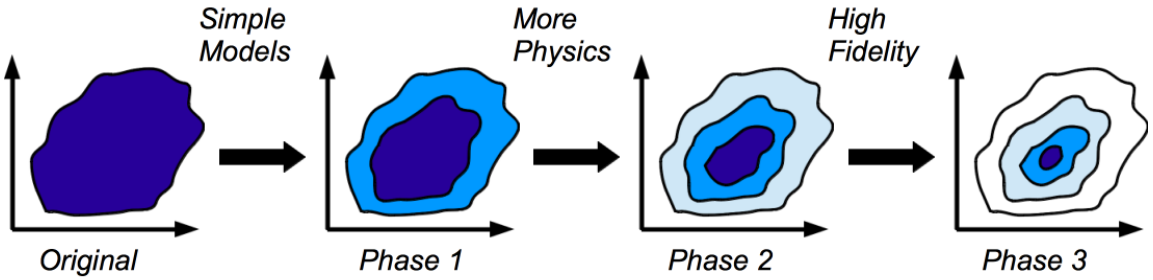


Figure 1.2: Narrowing of a design space through numerical analysis. As the space narrows towards an optimal design (left to right), the required sophistication of the numerical tools increases (and thus the cost).

be used to narrow the design space significantly, thus reducing the number of design variants that must be prototyped and tested.

Figure 1.2 shows an example of how a generic design space might be studied through numerical analysis. Throughout the analysis, trade-offs are continually made between simulation accuracy and simulation runtime. Initially, when there are potentially hundreds or thousands of design variants, simulation runtime needs to be on the order of minutes, even seconds, to narrow the design space in an efficient manner. This reduced runtime is achieved by easing the accuracy requirements of the results. Conversely, towards the end of the cycle when only a few design variants remain, simulations employing accurate physics are needed to differentiate between the designs at the cost of increased runtime (perhaps to days, even weeks).

1.2 Computational Fluid Dynamics in the Design Cycle

1.2.1 Nature of the Flow Equations and Turbulence

One area that benefits greatly from numerical analysis is fluid dynamics. Fluid dynamics is the study of fluid motions and their corresponding forces. If the full flow field is known, then other quantities of interest can be derived, such as drag or lift.

The Navier-Stokes equations (shown in Section 2) are a well-proven mathematical model for fluid dynamics. However, the equations are difficult to work with directly. All variables are strongly coupled, and thus must be solved for simultaneously. Also, the equations are highly non-linear. As a consequence, analytical solutions are only available for simple, textbook flows. In order to study flows of engineering consequence, the equations must be solved numerically in an area known as computational fluid dynamics (CFD), although this is not where the difficulties of fluid dynamics analysis end.

Almost all flows in real-world situations are turbulent. Turbulent flow fields are characterized by seemingly random fluctuations of the field variables in both space and time [4], making them difficult to predict. A defining (and simultaneously burdensome) feature of turbulent flow fields is a large range of length and time scales. For highly turbulent flow fields, there are large motions, or eddies, that are mostly unaffected by viscosity, known as integral length scales. Through the idea of an energy cascade, these largest eddies transfer energy to smaller and smaller eddies until the eddies are small enough for viscous forces to dissipate the energy as heat [4]. Another important feature of turbulent flow fields is enhanced mixing. This leads to both increased heat transfer and increased drag as compared to non-turbulent, or laminar, flows. As drag and heat transfer are important design parameters for gas turbine engines, it is crucial that the designer take into account how they are affected by the turbulence in the flow.

The most accurate method of simulation involves discretizing and solving the Navier-Stokes equations directly, resolving all spatial and time scales present in the flow. This is known as a direct numerical simulation, or DNS. Unfortunately, due to the large disparity of length and time scales in a turbulent flow, DNS has historically been impractical for flows of engineering interest. For example, the computing

power needed to perform a DNS of just one airplane wing is estimated to not be available until the year 2050, and a full airplane won't be possible until 2080. However, the numerical methodology presented in this work, in conjunction with modern supercomputing capabilities, opens up the possiblities of directly simulating smaller industrial components, such as a high pressure turbine blade in a gas turbine engine. In order to understand the motive behind the current development, the progression of CFD tools used in industry is now discussed.

1.2.2 Industrial CFD Use

One of the largest benefactors of CFD in design is the aerospace industry [5, 6, 7, 8, 9]. The earliest usage of CFD was to solve the equations assuming there are no viscous interactions. The modified equations are known as the Euler equations. The main advantage of this approach is the drastic reduction of length scales that need to be resolved (particular in smooth cases with no shocks). Thus, the stringent computational requirements seen for DNS are greatly relaxed, allowing for relatively complex Euler flows to be solved on desktop computers. For high speed flows, Euler solutions can provide fairly good approximations of certain items of engineering interest, such as lift for airfoils. However, conceptually, there is a large difference between a high speed turbulent flow, where the effects of viscosity on large scale motions approach zero, and an inviscid flow, where there are no viscous effects. For example, due to the absence of viscous effects, an Euler simulation incorrectly predicts zero drag around a symmetric object.

Since drag (and heat transfer) are important in the aerospace industry, viscous effects must be included. However, for many problems, direct resolution of all length and time scales is not important; rather, we are typically only interested in the large scale motions and their effect on engineering quantities such as heat transfer and drag.

In a DNS, most of the computational effort is spent on directly simulating the energy cascade; a relatively small portion of the modes are needed to capture the larger scales [4]. On this basis, the next logical approximation step is to time average the Navier-Stokes equations, resulting in the Reynolds Averaged Navier-Stokes (RANS) equations. The equations can then be solved for average flow quantities and average engineering effects (such as drag) can be computed directly. Due its ability to deliver results in a timely fashion on modest computing resources, RANS is the default method for most commercial solvers. The drawback of this approach is that the effects of turbulence on mean flow quantities are described entirely by a turbulence model. As the nature of a turbulent flow field can change drastically due to a variety of factors, creating turbulence models applicable to general flow fields is quite difficult. Turbulence modeling continues to be a large research area, and this focus continues to improve the effectiveness of RANS solvers. However, for the time being, it appears that higher-fidelity models are needed for areas where RANS solvers typically struggle (an example of this being combustor flows).

A relatively recent compromise between the stringent requirement of DNS and the reduced accuracy of RANS is large-eddy simulation, or LES. Although many variants exist, a common approach among LES models is to apply a spatial filter to the Navier-Stokes equations. Like the RANS approach, the filtering process creates additional terms in the equations that must be modeled. The key difference, however, is that the accuracy (or inaccuracy) of the model has a much less pronounced effect on the accuracy of the solver as a whole. LES models are also easier to extend to general turbulent flows, as they only attempt to model the smallest turbulent scales, which are generally thought to exhibit behaviors universal to all flows [4]. The main drawback is that, while reduced compared to DNS, the computational cost of LES is still far greater than RANS, which has thus far made it inaccessible for most

problems. However, the advent of modern supercomputing has made high-fidelity flow models, such as LES and DNS, a viable option for many industrial flows.

1.2.3 Massive-parallelism and CFD

The complexity of a numerical problem can generally be described in terms of its degrees of freedom (DOF). As the DOF increases, so does the amount of time needed to solve the problem. Modern supercomputing is the solution to tackling problems with impractically large DOF counts, such as those seen with high-fidelity CFD models. Supercomputers vary in size and power, but in general, they feature tens or hundreds of thousands of cores. In addition, supercomputers feature large, high-speed memory resources and efficient inter-processor communications. These characteristics combined allow supercomputers to effectively "spread" the problem out over hundreds of thousands of cores, allowing the cores to work on smaller segments of the problem in parallel. Ideally, as the number of cores employed in a parallel computation is doubled, the computation time should be cut in half. However, this is only possible if the method used is designed specifically for efficient parallel computations; using massively-parallel machines does not guarantee massively-parallel performance.

1.2.4 Towards Improving Design Cycle Analysis

In order to expand the usefulness of CFD tools to areas where RANS may not be as effective, an effort was made over the past decade to extend existing RANS solvers to LES. However, because the solvers weren't built specifically with efficient parallel computing in mind, they have not been able to take full advantage of modern supercomputing power, making the much increased DOF count a barrier to their success. In order to make more direct methods attractive for design cycle analysis, solvers employing them must be designed from the ground up to work well in a

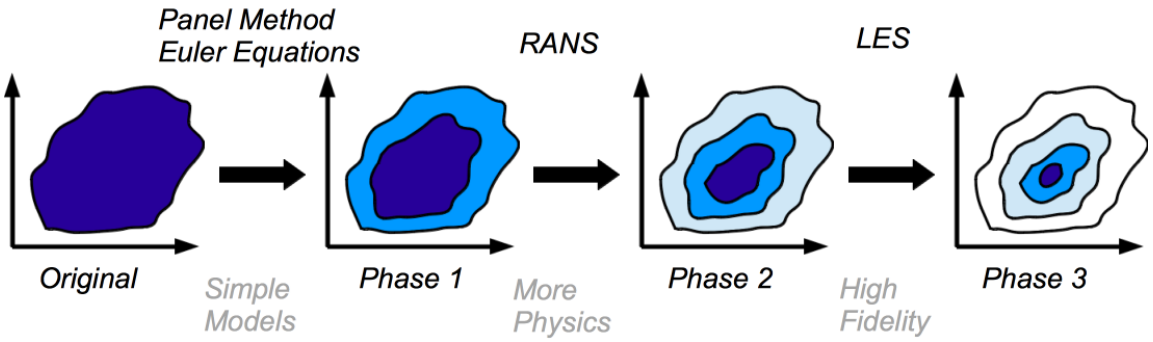


Figure 1.3: CFD levels ranked in a generic numerical framework. Historically, industrial CFD has limited to the RANS stage. However, advances in computing power have made high-fidelity models, such as DNS or LES, feasible as an additional stage in numerical design.

massively-parallel environment.

The need for high-fidelity CFD capabilities does not exclude RANS from continued use in design cycle analysis. Although RANS solvers currently see widespread use in the aerospace industry, the reduced order models that were once only used for post-design analysis are continuously improved upon and still see heavy use in design cycle analysis. Figure 1.3 shows how CFD tools form a natural hierarchy similar to what was seen generically in Figure 1.2. As discussed previously, the correct CFD tool at each phase is one which delivers the required accuracy to differentiate between design choices at a computational cost appropriate for the given stage. While industrial CFD usage currently ends with RANS, a tool that efficiently solves the Navier-Stokes equations with a reduced dependence on turbulence modeling, such as DNS or LES, will allow another stage to be added in the numerical analysis portion of the design cycle.

1.3 Keys for Effective High-Fidelity CFD

In order for a code employing LES or DNS to be successful in industrial applications, it must exhibit the following characteristics:

- High order/high accuracy
- High parallel efficiency
- Efficient geometric flexibility

1.3.1 Need for High Order

The main advantage of RANS solvers is the less stringent discretization requirements. However, the manner in which RANS solvers typically model turbulence makes it difficult to obtain a stable numerical solution (i.e., in a manner in which numerical errors are bounded). Thus, most RANS algorithms are implemented using low order methods that offer enhanced stability at the expense of resolution. This is an acceptable compromise for RANS solutions as the resolution needed to resolve the time-mean flow field accurately is not too great. However, the lack of resolution quickly becomes an issue once turbulent structures must be resolved. Consider the complex function shown in Figure 1.4. Although this function does not come directly from a turbulent flow, turbulent flows will exhibit similar high frequency characteristics.

A common measure of a method's accuracy characteristics is its convergence rate. Essentially, the convergence rate predicts how much the accuracy of a method improves if the degrees of freedom are increased. While the convergence rate of a method is somewhat problem dependent, in general a high order method will have a higher rate of convergence than a low order method–this is essentially why one would choose a high order method in the first place. To illustrate why this is important,

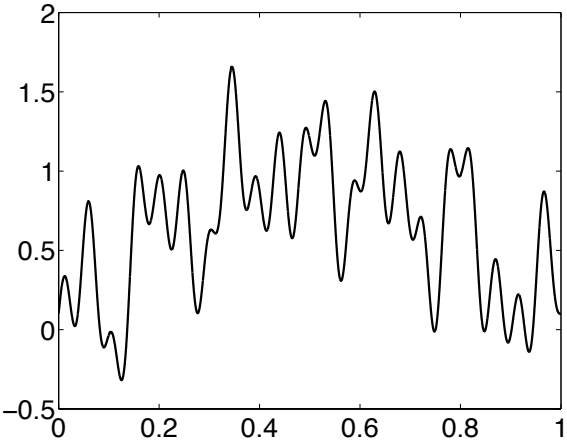


Figure 1.4: Sample waveform representative of turbulent signal. Low order methods require large DOF counts to capture this properly.

a fixed low order method is compared to a fixed high order method in terms of their abilities to represent the wave form in Figure 1.4 discretely. In both methods, the domain $x \in [0, 1]$ is broken down into equal segments, known as elements, and the resolution is increased by increasing the number of elements used. The low order method uses linear interpolation between points, while the high order method interpolates the solution within each element using a fifth order polynomial.

Figure 1.5 compares the errors of the low and high order methods as the resolution is increased. The important item to note is the rate at which the error drops as the resolution is increased, termed the convergence rate. The computed convergence rates for the low and high order methods are two and six, respectively. Simply put, the convergence rate for the high order method is greater because it can make more use out of the available grid points. The real practical advantage of a high accuracy method in general is that fewer grid points are needed to reach a desired accuracy level. For example, referring to Figure 1.5, the high order approximation needs approximately an order of magnitude fewer points than low order to achieve

an error level on the order of 10^{-6}. In the context of turbulent flows, this means that the stringent discretization requirements for LES or DNS can be lessened quite dramatically when using a high order method.

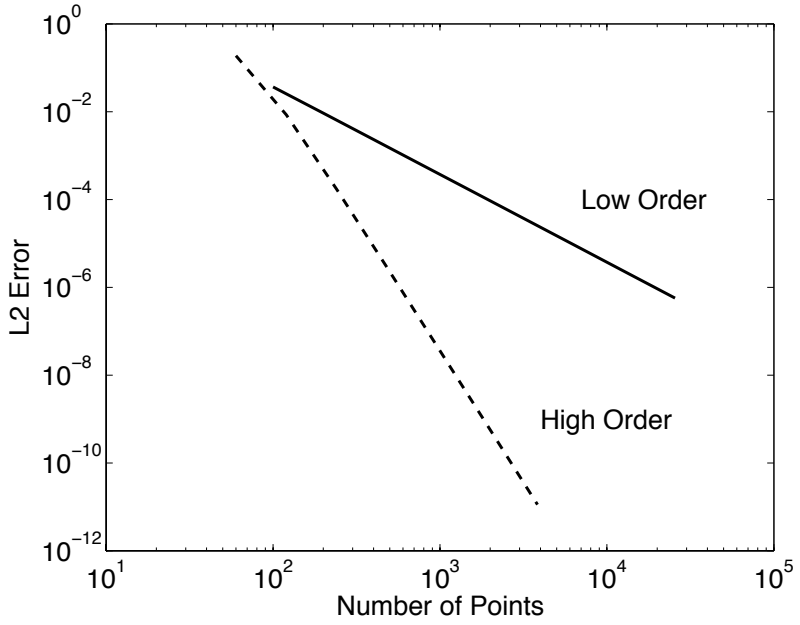


Figure 1.5: Comparison of convergence rates for low and high order methods. The larger convergence rate of the high order method is advantageous in that far fewer grid points are required to reach a certain error level.

While the preceding results are promising, even further gains can be made by implementing a method where the order used is variable. Figure 1.6 shows the vastly reduced error rates possible by leaving the number of elements fixed and instead increasing the order of each element. The errors below 1000 DOF are higher for the variable order method than the fixed, high order method. However, past 1000 DOF, the variable order method quickly reaches machine precision accuracy. The convergence rate seen is termed exponential convergence, and by definition, it is faster than what is realized by any fixed order method. It is important to note that

this behavior only occurs for solutions that are sufficiently smooth, such as the one used for this test. Actual CFD solutions will not typically exhibit such smoothness, and thus the convergence rate will be limited by the smoothness of the problem. For industrial flow applications, a method with the flexibility of varying both the number of elements used and the order of the elements can allow for a wider variety of flows to be analyzed.

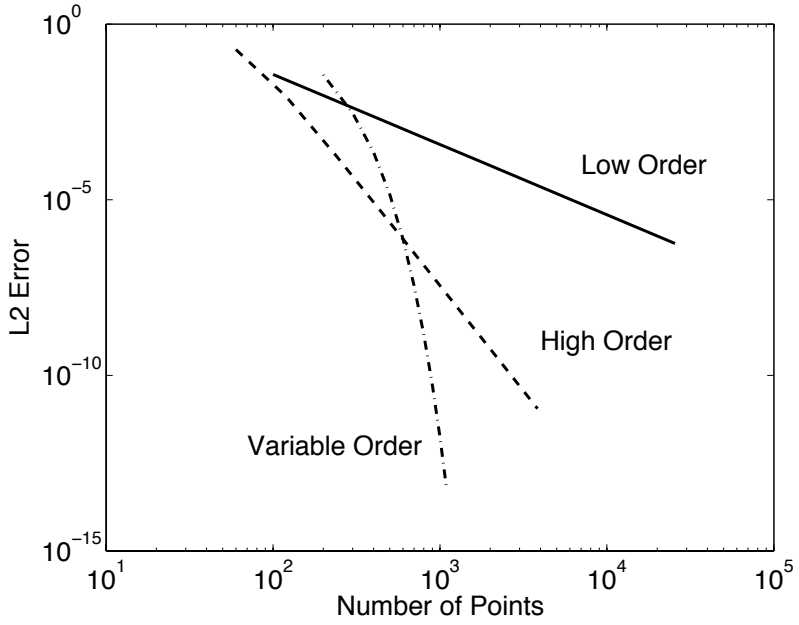


Figure 1.6: Example of exponential convergence for variable high order method. For smooth problems, variable order methods are often more accurate than fixed high order methods.

1.3.2 Need for High Parallel Efficiency

Modern supercomputers have the ability to transform an impossibly large problem into thousands, even millions, of small, more manageable problems. However, such a feat is not achieved automatically; only codes that are built with parallel

computing in mind can take full advantage of modern supercomputing capabilities.

In terms of a design cycle, prototype testing is much more expensive than numerical analysis. However, it should be understood that the use of supercomputers is not "free." Based on the amount of wattage needed to run a personal computer, one can imagine the massive amount of power consumed to run supercomputers that contain millions of computing cores. Supercomputer time should be considered a finite resource that must be used wisely and efficiently. One of the most common measures of the parallel efficiency of a code is termed scaling. There are a few different definitions of scaling, but we will use strong scaling for the purposes of the following discussion. For strong scaling, the total number of grid points is kept constant, and the speed of the code is measured as a function of the number of processors used. Ideal scaling is defined as a one-to-one relationship between the speedup of the simulation and the relative increase in computing cores used. For example, for a code with perfect scaling, a simulation that uses 1000 processors should run twice as fast as a simulation using only 500 processors. While a code will never exhibit perfect scaling for arbitrarily large processor counts, it should always be the goal. Once a code reaches the point where its scaling deviates too far from ideal, adding additional computing cores to the simulation would be considered wasteful.

Although it can vary from code to code, in general, the parallelization structure of CFD solvers follow a common formula. Once the domain is discretized, the total grid points are divided and distributed among the processors being employed. The operations of a parallel CFD code can be broadly categorized as

- local operations, where all the needed data is immediately available to the processor, and
- non-local operations, where a portion of the needed data currently resides on

other processors and must be retrieved.

The key to a highly-efficient code that scales well on massively-parallel machines is keeping the ratio of time of spent in local operations to time spent in non-local operation as high as possible. In general, local operations should exhibit near-ideal scaling. On the other hand, the time spent on non-local operations will at best stay flat, although the time would be expected to rise slightly as the number of processors used is increased. Therefore, the ratio of local to non-local operation time for a given processor count is a fairly good indicator of how far a code's scaling will reach, and this is more or less dependent on the method (but also dependent on the problem size).

Figure 1.7 shows an example of how a code's initial local to non-local operation ratio affects scaling. Using P processors, the code represented by the blue curve has a local to non-local ratio of 50, while the ratio for the code represented by the red curve is 2.5. Because communication time stays relatively flat in parallel operations (at best), this ratio quickly becomes less than one for the red code, and the result is that the speedup gains for the local operations have only a marginal effect on the code performance as a whole. On the other hand, the ratio for the blue code never dips below 5, and thus sizable speedups are achieved each time the processor count is doubled.

Even when choosing a high order method, the number of grid points needed for adequate solution resolution will still be high. Therefore, it is vital that the code be developed in such as way as to take full advantage of supercomputers, allowing the large problem size to be spread out to tens or hundreds of thousands of computing cores working in parallel.

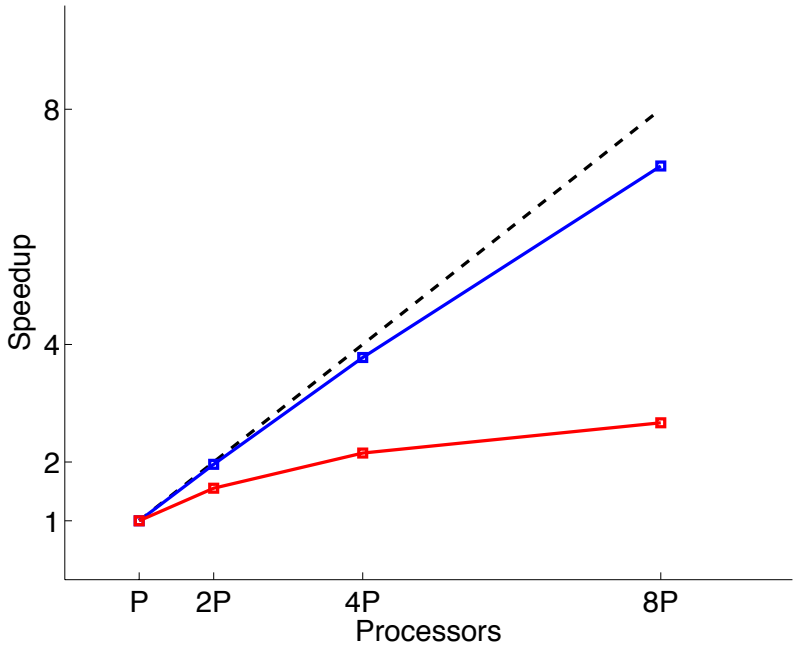


Figure 1.7: Example of good and poor scaling. The percentage of total time spent in non-local operations is a good indicator of well a code will scale. The curve in blue initially has a non-local time percentage of approximately 2%, while the curve in red is nearly 30%. Since the parallel communication time typically stays flat, the code represented by the red curve deviates quickly from ideal (dashed line).

1.3.3 Need for Geometric Flexibility

There are a few methods to choose from that are both high order and have demonstrated good scaling properties on massively-parallel supercomputers. However, a third requirement for industrial CFD is the ability to handle complex geometries while still maintaining high accuracy and high parallelism. The geometric flexibility of a code can be quantified by two characteristics: its ability to perform efficiently on unstructured discretizations, and the accuracy in which it discretizes domain boundaries.

In a structured grid, the elements are ordered in such a way that an element's neighbors are automatically identified by the unique index given to that element.

For example, in 2D, the neighboring element to the right of element $(4, 5)$ might be element $(5, 5)$, while the element immediately above element $(4, 5)$ would be $(4, 6)$. This type of structure greatly simplifies the coding of the numerical scheme. It also can also lead to performance gains, as a structured grid will minimize the number of elements connected to a given element (and therefore the number of elements that a given element must share data with). The downside of these grids is that they are difficult to adapt to geometries that are not logically a square (or a cube in three dimensions); thus, their utility for the complex geometries seen in industrial flows is fairly limited. Unstructured grids, on the other hand, work well in complex geometries, as the elements no longer have to be ordered in a certain way. However, greater care must be taken to ensure the performance of the code is not degraded by the use of unstructured grids. In the context of high-fidelity simulations, the use of methods which handle unstructured grids is a must; otherwise, more time may be spent in setting the problem up than actually simulating the flow.

Oftentimes, the accuracy of a simulation depends greatly on how well the discretization captures the boundary. This is particularly true in the simulation of turbulent flows, where much of the turbulent production occurs at walls [4]. This is why the grid is typically clustered near walls in CFD simulations. However, in many discretizations, more elements are used along domain boundaries than would be required by the physics of the problem. This is particularly true for low order methods, where the domain boundaries are discretized using linear interpolation. High order boundary representations are preferred in order to keep overall element counts low while still maintaining appropriate resolution.

1.4 Extension to NEK5000

The desired characteristics for a high-fidelity industrial CFD code are high accuracy, high parallel efficiency, and geometric flexibility. The spectral element method (SEM) is uniquely qualified for industrial LES/DNS use because it contains these characteristics. In the spectral element method, the domain is broken down into elements, and the solution is approximated within each element as a high order (typically 7th to 15th order) polynomial. The method is flexible in that higher resolution can be achieved by increasing the number of elements used or increasing the polynomial order (or both), thus leading to high, even exponential, convergence rates. The inherent data locality of the method lends itself well to highly efficient parallel computations, as the ratio of local work to non-local work is extremely high. Finally, the geometry in SEM is typically represented discretely using the same high order polynomials as the solution itself. Thus, high order boundary representations are available, reducing the number of elements required near walls while maintaining solution accuracy.

Fortunately, the characteristics of SEM are not simply theoretical niceties; they have been proven in practical CFD simulations. A particular code that our group has worked with extensively is NEK5000, an incompressible spectral element code developed out of Argonne National Lab [10, 11, 12, 13]. Recently, the code has shown strong scaling on over one million processes with a parallel efficiency of over 60% [10]. Using NEK5000, our group has simulated several gas turbine components using LES and DNS, including a high pressure turbine blade [14], low pressure turbine blade [15], and cold flow combustor [16].

While NEK5000 has the necessary attributes to run high-fidelity industrial CFD simulations on massively-parallel machines, there is one attribute remaining that

specifically pertains to the aerospace industry: compressiblity. Currently, NEK5000 is only set up to handle incompressible (constant property) or weakly compressible (low-Mach) codes. There are some areas in the aerospace industry where this is sufficient, but many flows are transonic or supersonic. In these areas, the standard NEK5000 algorithm struggles on two ends. First, using an incompressible or low-Mach formulation will produce incorrect physics when applied to a transonic flow; for one thing, pressure has a different meaning in an incompressible code and a compressible code. Also, extending NEK5000 to compressible flows is not as simple as changing the equations that are solved. Compressible flows also behave differently than incompressible flows numerically, and so the algorithm must be changed in order to handle compressible flows properly; otherwise, the simulation may be unstable. Therefore, in order to increase NEK5000's utility in the aerospace industry, it must be extended to fully compressible flows and its current algorithm must be modified in order to do so.

1.5 Statement of Purpose

Leveraging the proven performance of NEK5000, it is hypothesized that a compressible Navier-Stokes spectral element algorithm can be developed which meets the three requirements for successful industrial CFD use:

- High accuracy
- Efficient parallelism
- Geometric flexiblity

Such an algorithm will allow for larger, high-fidelity simulations to gain further traction in the aerospace community.

In section 2, after a review of the literature, the details of the new algorithm are given. In Section 3, the algorithm is validated against several canonical inviscid and viscous test cases in order to test various aspects of the code. Then the code is used to simulate film cooling heat transfer, with the results compared to both experimental data as well as a previous CFD study using a low-Mach incompressible solver; the details of this simulation are given in Section 4. Finally, in Section 5, concluding remarks are given.

2. LITERATURE REVIEW AND NUMERICAL METHODS

2.1 Literature Review

The most popular method for the solution of compressible flows, especially in commercial solvers, is the finite volume (FV) method. In a classical finite volume method, the domain is discretized into cells and the solution is approximated by piecewise constants, although improvements to the overall scheme have included linear reconstructions to increase the formal order. Much of the theory into appropriate numerical flux functions has started with the FV method [17]; further reference into the method and how it can be applied to the Navier-Stokes equations can be found in [18, 19]. Žaloudek used FV coupled with the Advection Upstream Splitting Method (AUSM) and linear reconstruction with slope limiters to study some simple Euler flows [20]. The effect of different limiters on solution accuracy was tested on a simple GAMM channel. Unsurprisingly, the standard FV first order scheme produced inferior results to higher order reconstructions. However, there was not a large difference between the different slope limiters. Also, for wall boundary conditions, using a simple, zeroth order extrapolation for wall pressure gave approximately the same results as higher order extrapolation.

The standard finite element method (FEM) at first is not a good fit for compressible flows, as it tends to produce spurious oscillations. However, methods have been developed to stabilize the scheme, such as Streamline-Upwind Petrov Galerkin [21]. Soulaimani uses an edge-based stability (EBS) method that borrows from SUPG as well as discontinuous Galerkin methods [22]. Tests are performed for both the Euler equations as well as the Navier-Stokes equations (coupled with the one equation Spallart-Almaras RANS model). The EBS method showed to be less diffusive than

SUPG, but also was more computationally expensive. Martinez uses FEM to compute solutions of the Boussinesq equations and the acoustically-filtered equations at large Rayleigh numbers [23]. The effort was in large part to study the differences between the methods for certain flows. Recently, Kirk used the SUPG scheme to simulate the compressible Navier-Stokes equations with the libMesh finite element library [24].

Don used spectral methods to solve flow around a cylinder in polar coordinates [25]. A Fourier series was used in the angular direction (due to the periodicity) and a Chebyshev series was used in the radial direction. Because spectral methods are sensitive to boundary conditions, care was taken to ensure the boundary conditions were implemented properly through characteristic analysis. Don also used multi-domain spectral methods in 2003 to study reactive compressible flows in a scramjet cavity [26]. The penalty method was used as interface conditions to provide a connection between domains, and filtering was used to stabilize the solution. Hesthaven also investigates spectral penalty methods, both single and multidomain, in a series of papers [27, 28, 29]. In the series, the question of proper boundary conditions is addressed, and as a unique contribution, both inviscid and viscous parts are patched simultaneously at element interfaces and boundaries.

Spectral collocation or spectral differencing formulations offer an interesting compromise between spectral element and finite differences. Kopriva used a staggered grid Chebyshev multidomain method in [30] for the Euler equations. The key point is that the flux computations were done on the Gauss quadrature points, not the Gauss-Lobatto points, which allows one to avoid the computation of multidimensional Riemann solutions. Kopriva also extended the method to non-conforming elements in [31] to compute solutions to the Navier-Stokes equations. This allowed the method greater flexibility in terms of local mesh refinement. Liu later formal-

ized the method as the spectral difference method [32], and extended it to triangles. Jameson's group has contributed much to the method, using it for Large Eddy Simulation [33], adapting it for discontinuity/shock capturing [34], and studying transitional flows over airfoils [35].

Discontinuous Galerkin finite element methods (DGFEM) and spectral element methods (DGSEM) relax the continuity requirements on the solution from their more classical counterparts (FEM and SEM), thus making it more amenable to convection dominated flows. In fact, some of the first works were on the Euler equations. Bassi used DGFEM to study a canonical test case of inviscid flow over a cylinder [36]. It is concluded that isoparametric elements must be used to receive any benefit from using higher order polynomials; in fact, the solution quality deteriorated as the polynomial order increased if the simulation was still using first order representations of the geometry. Bassi also developed one of the first schemes to handle the viscous operator in the Navier-Stokes equations; the scheme is dubbed the "BR1" scheme [37]. In it, the Navier-Stokes equations are recast as strictly first order equations. In [38], improvements are made to the original BR1 scheme, which include shortening the stencil for the viscous operator and including a mechanism for handling the viscous flux implicitly using a GMRES iterative scheme, and in [39], the scheme is extended to solve the RANS equations. Other schemes for handling the viscous operator have also been developed, including local discontinuous Galerkin (LDG) [40] and the Baumann-Oden scheme (BO) [41]. Warburton [42] and Kirby [43] extend the tensor product-based quadrilateral and hexahedral spectral elements to more general shapes, such as triangle and tetrahedrals, and give examples for Euler and Navier-Stokes. More information on these types of elements can be found in [40]. Nodal type DG schemes are discussed in [44] and also in the book by Hesthaven [45]. These schemes exhibit fairly high efficiency in some areas, but lose the performance gains the

tensor-product bases have to offer. Recently, Kopriva discussed the requirements on the contravariant metrics in order for DGSEM to be constant solution preserving [46], and Hindenlang showed great performance both in terms of accuracy and parallelism for their DGSEM code [47].

2.2 Conservation of Mass, Momentum, and Energy

As discussed in Section 1, the most common definition of CFD is the numerical solution of the Navier-Stokes equations. In index notation, the Navier-Stokes equations are

$$\begin{aligned} \frac{\partial \rho}{\partial t} + \frac{\partial \left(\rho u_j\right)}{\partial x_j} &= 0, \\ \frac{\partial \left(\rho u_i\right)}{\partial t} + \frac{\partial \left(\rho u_i u_j\right)}{\partial x_j} + \frac{\partial p}{\partial x_i} &= \frac{\partial \tau_{ij}}{\partial x_j}, \\ \frac{\partial \left(E\right)}{\partial t} + \frac{\partial \left[u_j \left(E+p\right)\right]}{\partial x_j} &= \frac{\partial \left(u_i \tau_{ij}\right)}{\partial x_j} - \frac{\partial q_j}{\partial x_j}, \end{aligned} \tag{2.1}$$

which represent conservation of mass, momentum, and energy. The primary, or conserved, variables are the density ρ, momentum ρu_i, and total energy E (note that the momentum and energy variables actually represent momentum and energy per unit volume). The transport terms are the stress tensor τ_{ij} and the heat flux q_j. Secondary, or primitive, variables that are derived from the primary variables are the pressure p and velocity u_i.

At this point, no assumptions have been made beyond continuum mechanics. However, common assumptions are that of a Newtonian fluid and Fourier's law, which affect the stress tensor and the heat flux, respectively. The stress tensor for a

Newtonian fluid, coupled with Stoke's approximation, is

$$\begin{aligned} \tau_{ij} &= 2\mu S_{ij} \\ S_{ij} &= \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) - \frac{1}{3}\frac{\partial u_k}{\partial x_k}\delta_{ij}, \end{aligned} \tag{2.2}$$

where μ is the dynamic viscosity of the fluid. Fourier's law for heat conduction is

$$q_j = -k\frac{\partial T}{\partial x_j}, \tag{2.3}$$

where k and T are the thermal conductivity and temperature, respectively, of the fluid. The dynamic viscosity and thermal conductivity are related through the Prandtl number as

$$\mathrm{Pr} = \frac{\mu c_p}{k}. \tag{2.4}$$

The dependence of viscosity on temperature for gas flows is most commonly defined through Sutherland's formula [48]:

$$\mu(T) = \mu_{\mathrm{ref}}\left(\frac{T}{T_{\mathrm{ref}}}\right)^{3/2}\frac{T_{\mathrm{ref}} + C}{T + C}, \tag{2.5}$$

where the quantities with the underscore "ref" denote reference quantities, and C is a constant depending on the gas. This work uses a simpler power law approximation [49],

$$\frac{\mu}{\mu_{\mathrm{ref}}} = \left(\frac{T}{T_{\mathrm{ref}}}\right)^{0.76}, \tag{2.6}$$

which gives reasonable results over the small temperature changes seen in the simulations in Sections 3 and 4.

The final assumption made in this work is that of a calorically perfect case. Through this assumption, the pressure, density, and temperature are related through

the ideal gas law as $p = \rho R_g T$, where R_g is the gas constant. Also, the total energy and pressure are related as $E = p/(\gamma - 1) + \rho u_i u_i/2$, where $\gamma = c_p/c_v$ is the ratio of specific heats, with c_p and c_v as the constant pressure specific heat and constant volume specific heat, respectively. In this work, the Prandtl number and the specific heats are all assumed to be constant, a reasonable assumption for subsonic flows. A summary of the variables, with their corresponding units in the SI system, are given in Table 2.1.

Table 2.1: Listing of variables in Navier-Stokes equations.

Conservative Variables		
ρ	kg/m^3	Density
ρu_i	N s/m^3	Momentum (per unit volume)
E	J/m^3	Total energy (per unit volume)
Transport Terms		
τ_{ij}	Pa	Stress tensor
q_j	W/m^2	Heat flux
Primitive Variables		
u_i	m/s	Velocity vector
p	Pa	Pressure
T	K	Temperature
Fluid Properties		
μ	Pa s	Dynamic viscosity
k	W/(m K)	Thermal conductivity
c_p	J/(kg K)	Constant pressure specific heat
c_v	J/(kg K)	Constant volume specific heat
R_g	J/(kg K)	Gas constant ($c_p - c_v$)

While the Navier-Stokes equations as given in Equation 2.1 are analytically sufficient to work with, potential problems arise when working with them numerically using physical units. This is because flow variables can have vastly different scales associated with them, leading to errors stemming from numerical precision. For ex-

ample, standard atmospheric pressure is on the order of 100,000 Pa, while the density is near 1 kg/m^3. In order to combat this, the equations are non-dimensionalized. Choosing a reference velocity (U_{ref}), density (ρ_{ref}), length scale (L_{ref}) and temperature (T_{ref}), reference quantities are derived for all other flow variables:

$$t_{\text{ref}} = L_{\text{ref}}/U_{\text{ref}} \qquad p_{\text{ref}} = \rho_{\text{ref}} U_{\text{ref}}^2 \qquad a_{\text{ref}} = \sqrt{\gamma R_g T_{\text{ref}}} \qquad E_{\text{ref}} = \rho_{\text{ref}} U_{\text{ref}}^2$$

$$\mu_{\text{ref}} = \mu(T_{\text{ref}}) \qquad \tau_{\text{ref}} = \mu_{\text{ref}} U_{\text{ref}}/L_{\text{ref}} \qquad q_{\text{ref}} = \mu_{\text{ref}} c_p T_{\text{ref}}/(L_{\text{ref}}\text{Pr})$$

Here, a_{ref} is the speed of sound at the reference temperature, and t_{ref} is the reference time interval.

Inserting these reference variables into the Navier-Stokes equations and performing some manipulations, we arrive at

$$\begin{aligned}
\frac{\partial \rho^*}{\partial t^*} + \frac{\partial \left(\rho^* u_j^*\right)}{\partial x_j^*} &= 0, \\
\frac{\partial \left(\rho^* u_i^*\right)}{\partial t^*} + \frac{\partial \left(\rho^* u_i^* u_j^*\right)}{\partial x_j^*} + \frac{\partial p^*}{\partial x_i^*} &= \frac{1}{\text{Re}} \frac{\partial \tau_{ij}^*}{\partial x_j^*}, \\
\frac{\partial E^*}{\partial t^*} + \frac{\partial \left[u_j^* \left(E^* + p^*\right)\right]}{\partial x_j^*} &= \frac{1}{\text{Re}} \frac{\partial \left(u_i^* \tau_{ij}^*\right)}{\partial x_j^*} - \frac{1}{\text{RePr}(\gamma - 1)\text{Ma}^2} \frac{\partial q_j^*}{\partial x_j^*}, \\
\tau_{ij}^* &= \mu^* \left(\frac{\partial u_i^*}{\partial x_j^*} + \frac{\partial u_j^*}{\partial x_i^*} - \frac{2}{3} \frac{\partial u_k^*}{\partial x_k^*} \right), \\
q_j^* &= -\mu^* \frac{\partial T^*}{\partial x_j^*},
\end{aligned} \tag{2.7}$$

where the * quantities denote variables normalized by their corresponding reference quantity. From here on, the * indicators will be dropped, and thus any flow variables will be assumed to refer to non-dimensional quantities unless stated otherwise.

The non-dimensional equations give rise to two new non-dimensional parameters that have yet to be discussed. The reference Mach number is defined as $\text{Ma} = U_{\text{ref}}/a_{\text{ref}}$. The reference Mach number is an indicator of the compressiblity of the flow. The higher the chosen reference Mach number is, the further the

flow will deviate from an incompressible (constant property) flow. The Reynolds number, defined as $\mathrm{Re} = \rho_{\mathrm{ref}} U_{\mathrm{ref}} L_{\mathrm{ref}} / \mu_{\mathrm{ref}}$, is an important non-dimensional flow parameter in fluid mechanics. It is a ratio of the inertial forces to viscous forces in the flow. In general, turbulent flow fields are described by a high Reynolds number, although what qualifies as "high" is dependent on the particular flow problem. The Reynolds number is also an indicator of how difficult the problem is to solve numerically in terms of required resolution. For example, for the DNS study of isotropic turbulence in a periodic box using a Fourier spectral method (currently the most accurate method available), the number of operations required scales as Re^3. While the actual operation count will change depending on the flow in question (and the method used), the takeaway is that the required resolution increases rapidly with the Reynolds number.

2.3 Numerical Methodology

The algorithm employed in the current work is the tensor product-based DGSEM. The method has shown great parallelism and accuracy [47] and robustness to obtain accurate solutions in realistic geometries [37, 38].

2.3.1 Discontinuous Galerkin Spectral Element Method: Fundamentals

For the purposes of clarity, a generic scalar conservation law is used as a means to introduce the discontinuous Galerkin spectral element method,

$$\frac{\partial u}{\partial t} + \nabla \cdot \mathbf{f} = 0, \tag{2.8}$$

where the flux $\mathbf{f} = \mathbf{f}(u)$ in general is non-linear in u. Similar to the standard Galerkin approach, Equation 2.8 is multiplied by a test function v from a suitable test space

and integrated over the domain, resulting in

$$\int_{\Omega} \left(\frac{\partial u}{\partial t} + \nabla \cdot \mathbf{f} \right) v d\mathbf{x} = 0. \tag{2.9}$$

Since the test function v is arbitrary, any solution u to Equation 2.9 will satisfy Equation 2.8.

Thus far, the formulation has represented a standard Galerkin formulation, where the function space is C^0 continuous. However, conservation laws tend to have solutions that are at best piecewise smooth, and thus may have discontinuities [40]. The function space typically used for standard Galerkin methods (such as the classical spectral element method) may not be the best choice for this particular problem. Instead, we choose a space of piecewise continuous functions for both our approximate solution and our test function.

To seek an approximate solution to Equation 2.9, the domain Ω is broken down into a tesselation of simpler domains, known as elements. For each element, the solution is approximated by a trial function u_h from a finite dimensional version of the function space described above. Choosing v_h from the same space and considering a single element, we have

$$\int_{\Omega_h^e} \left(\frac{\partial u_h^e}{\partial t} + \nabla \cdot \mathbf{f}(u_h^e) \right) v_h^e d\mathbf{x} = 0. \tag{2.10}$$

We now employ the divergence theorem to obtain

$$\int_{\Omega_h^e} \frac{\partial u_h^e}{\partial t} v_h^e d\mathbf{x} - \int_{\Omega_h^e} \nabla v_h^e \cdot \mathbf{f}(u_h^e) d\mathbf{x} + \oint_{\partial \Omega_h^e} \mathbf{f}(u_h^e) \cdot \mathbf{n} v_h^e ds = 0, \tag{2.11}$$

where $\mathbf{n}$ is the outward pointing normal along the surface of element e. However, the formulation as-is employs only information local to the element, and thus currently

there is no mechanism for information to propagate between elements. The necessary coupling is accomplished through the surface flux term. Consider one of the faces of element e and the neighboring element $e+1$ that shares that face with element e. As the approximation spaces are only piecewise continuous, in general the solutions contributed to the shared face by element e and $e+1$ will not be equal. Thus, the surface flux term $\mathbf{f}\cdot\mathbf{n}$ is replaced by a numerical flux function $\tilde{f}(u^-, u^+, \mathbf{n}^-)$ where the minus and plus superscripts refer to the contributions made by element e and $e+1$, respectively. There are many choices for the numerical flux function, although a good choice would be one that considers proper flow of information from upwinding [40]. Appropriate flux functions for the Navier-Stokes equations are detailed later; for now, it is assumed that a proper flux function has been chosen.

2.3.2 Lagrange Interpolating Polynomials and Semi-Discrete Operators

As of yet, the nature of the trial function and test function has not been specified. To keep the discussion concise, the h subscript will be dropped from the following discussion; it is now assumed that u refers to the approximate solutions we are seeking. In spectral element methods, both u^e and v^e are composed of a linear combination of basis functions from our test space on element e as

$$\begin{aligned} u^e(\mathbf{x}) &= \sum_{n=0}^{N} \hat{u}_n^e \psi_n^e(\mathbf{x}) \\ v^e(\mathbf{x}) &= \sum_{n=0}^{N} \hat{v}_n^e \psi_n^e(\mathbf{x}), \end{aligned} \tag{2.12}$$

where $\hat{u}_n^e$ are the coefficients to be determined. There are numerous choices for ψ_n^e. One choice is a set of orthogonal polynomials, such as the Legendre or Chebyshev polynomials, resulting in a modal scheme. If, instead, Lagrange interpolation polynomials are chosen, the scheme is nodal. While the choice of schemes will affect the

algorithm in some areas, ultimately the choice is not too important as one can easily convert back and forth using appropriate transforms. In this work, the nodal scheme is chosen due to some inherent advantages discussed later.

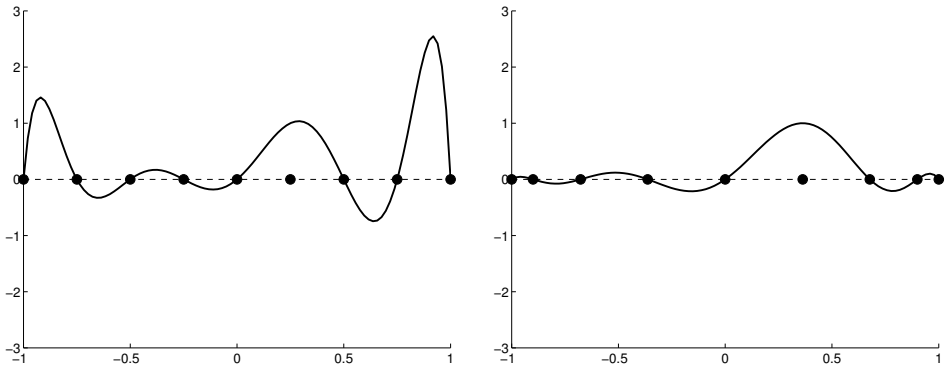
The Lagrange polynomials in one dimension are

$$\ell_n(x) = \prod_{i=0;i\neq n}^{N_x} \frac{x - x_i}{x_n - x_i}, \tag{2.13}$$

where x_n are the interpolation points. These polynomials have the property that $\ell_n(x_i) = \delta_{ni}$. Thus, when combined with the definition of our trial and test function (Equation 2.12), we see that our coefficients are in fact just the value of the approximation at the node point (i.e., $\hat{u}_n = u(x_n)$); hence the name "nodal." For clarity, then, we will represent $\hat{u}_n$ as simply u_n for the rest of the discussion. In order to completely define the Lagrange polynomials in Equation 2.13, the interpolation points must be chosen. One choice is to use evenly spaced points, as is done in finite elements. However, as the polynomial order increases, the quality of the interpolation decreases significantly, which has historically kept finite element methods at polynomial orders less than three or four. A good choice of points that minimizes this interpolation error are the Gauss Lobatto Legendre (GLL) points, which are defined on $x \in [-1, 1]$ as the zeros of

$$(1 - x^2)L'_N(x) = 0, \tag{2.14}$$

where $L'_N(x)$ is the derivative of the N_{th} order Legendre polynomial [50, 45]. These points keep the interpolation error to a minimum even for high order (> 12) interpolations. Figure 2.1 shows the increased quality of the interpolation using GLL points versus evenly spaced points.



(a) Evenly-spaced Points (b) Gauss Lobatto Legendre

Figure 2.1: Comparison of evenly-spaced and GLL points for interpolation. Evenly-spaced points a are typically used in finite elements, while GLL points b are seen in spectral elements. The use of evenly-spaced points for high order polynomials can introduce interpolation errors.

They are also beneficial in applying boundary conditions and in computing the numerical surfaces fluxes, as they include the end points (removing the need for extra interpolations). Higher dimensional variations typically use a tensor product of the 1D points, although the exact formulation differs for element types. This work uses a tensor product of the one dimensional Lagrange polynomials for higher dimensional formulations, which means that a tensor product of GLL grid points are also used to formulate the higher dimensional internal "grids." While the reasons for this choice are shown later, for now, the discussion will be kept general assuming a general family of Lagrange interpolants ψ_j with the exact nature of the corresponding interpolation points left undetermined.

With the form of the trial and test functions decided, we now look at the com-

ponents of Equation 2.11 in further detail. Examining the first component, we have

$$\begin{aligned}\int_{\Omega^e} \frac{\partial u^e}{\partial t} v^e d\mathbf{x} &= \int_{\Omega^e} \frac{\partial}{\partial t} \left(\sum_j^N u_j^e \psi_j(\mathbf{x}) \right) \left(\sum_i^N v_i^e \psi_i(\mathbf{x}) \right) d\mathbf{x} \\ &= \sum_i^N v_i^e \sum_j^N \frac{\partial u_j^e}{\partial t} \int_{\Omega^e} \psi_i(\mathbf{x}) \psi_j(\mathbf{x}) d\mathbf{x} \\ &= \underline{v^e}^T M^e \underline{u_t^e}, \end{aligned} \tag{2.15}$$

where the underline denotes an $N+1$ vector of coefficients, the subscript t refers to the time derivative, and $M_{ij}^e = \int_{\Omega^e} \psi_i(\mathbf{x}) \psi_j(\mathbf{x}) d\mathbf{x}$ is the elemental mass matrix. Inserting our trial and test functions into the second volume integral,

$$\begin{aligned}\int_{\Omega^e} \nabla v^e \cdot \mathbf{f}(u^e) d\mathbf{x} &= \int_{\Omega^e} \nabla \left(\sum_i^N v_i^e \psi_i(\mathbf{x}) \right) \cdot \left(\sum_j^N \mathbf{f}_j^e \psi_j(\mathbf{x}) \right) d\mathbf{x} \\ &= \sum_i^N v_i^e \sum_j^N \mathbf{f}_j^e \cdot \int_{\Omega_e} \nabla \psi_i(\mathbf{x}) \psi_j(\mathbf{x}) d\mathbf{x} \\ &= \underline{v^e}^T \mathbf{C}^e \cdot \underline{\mathbf{f}^e}, \end{aligned} \tag{2.16}$$

where $\mathbf{C}^e = [C_x^e, C_y^e, C_z^e]^T$ is the elemental weak divergence operator. The notation $\mathbf{C}^e \cdot \underline{\mathbf{f}^e}$ is meant to imply a dot product of matrix vector operations. For example, in 2D, this is

$$\mathbf{C}^e \cdot \underline{\mathbf{f}^e} = C_x^e \underline{f_x^e} + C_y^e \underline{f_y^e}. \tag{2.17}$$

For now, it is simply assumed that each component of the flux vector can also be represented as an N_{th} order polynomial; the details of this computation are left for

later discussion. Finally, examining the surface flux term, we have

$$\begin{aligned} \oint_{\partial\Omega_e} \tilde{f}(u^-, u^+, \mathbf{n}^-) v^e ds &= \oint_{\partial\Omega_e} \left(\sum_j^N \tilde{f}_j \psi_j(\mathbf{x}) \right) \left(\sum_i^N v_i \psi_i(\mathbf{x}) \right) ds \\ &= \underline{v^e}^T M_s^e \underline{\tilde{f}^e}, \end{aligned}$$

with M_s^e as the elemental surface mass matrix. Again, the details of how the coefficients $\tilde{f}_j^e$ are computed are left for later. However, because we are using the Lagrange interpolants through GLL points, it should be noted that M_s^e is quite sparse. Only nodes that lie on the faces of the element will contribute to the surface integral due the properties of Lagrange interpolation.

Putting it all together, the semi-discrete form of Equation 2.11 is

$$\underline{v^e}^T \left(M^e \underline{u_t^e} - \mathbf{C}^e \cdot \underline{\mathbf{f}^e} + M_s^e \underline{\tilde{f}^e} \right) = 0. \tag{2.18}$$

As the test function is arbitrary, Equation 2.18 represents $N+1$ equations for $N+1$ unknowns (on each element). From here, the equations could be advanced in time using an appropriate time-stepping algorithm. However, we have yet to specify how the individual operators are computed.

2.3.3 Transformation Metrics and Quadrature Rules

In theory, the operators M^e, $\mathbf{C}^e$, and M_s^e could be computed analytically for each element as a pre-processing step, as the Lagrange polynomials are known functions. However, since the primary interest in this work is to compute flows in complex geometries, it is highly likely that the elements used will be deformed (i.e., not regular), and computing analytical integrals and derivatives of high order polynomials in complex elements is non-trivial. As such, the computation of the operators is greatly simplified if the physical element is transformed to a simpler reference ele-

ment. As the method is already employing the GLL points for the interpolation, a good choice for a reference element in one dimension is the line segment $\xi \in [-1, 1]$. Higher dimensional elements also have corresponding master elements; for example, the corresponding master element for quadrilaterals is the square $\boldsymbol{\xi} \in [-1, 1]^2$. It is assumed from this point that all matrix operators and associated quantities refer to a single element; as such, the superscript e is dropped in the further discussion for clarity.

To make the transformation, we first assume a mapping $\boldsymbol{\xi} \rightarrow \mathbf{x}$ from the master element to the physical element, so that we have $\mathbf{x} = \mathbf{x}(\boldsymbol{\xi})$. This can be obtained numerically using any number of methods, such as transfinite interpolation [51] or the Gordon Hall method [50]. From this mapping, we can obtain what are termed as the covariant metrics [51], $\mathcal{H} = \frac{\partial \mathbf{x}}{\partial \boldsymbol{\xi}}$. However, it is desired to use derivative operators than act on the master element and then transform the results back to the physical element; as such, we will need the contravariant metrics, $\mathcal{G} = \frac{\partial \boldsymbol{\xi}}{\partial \mathbf{x}}$. From the chain rule, we have (in 3D)

$$\begin{pmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} \end{pmatrix} = \begin{pmatrix} \frac{\partial \xi_1}{\partial x} & \frac{\partial \xi_2}{\partial x} & \frac{\partial \xi_3}{\partial x} \\ \frac{\partial \xi_1}{\partial y} & \frac{\partial \xi_2}{\partial y} & \frac{\partial \xi_3}{\partial y} \\ \frac{\partial \xi_1}{\partial z} & \frac{\partial \xi_2}{\partial z} & \frac{\partial \xi_3}{\partial z} \end{pmatrix} \begin{pmatrix} \frac{\partial}{\partial \xi_1} \\ \frac{\partial}{\partial \xi_2} \\ \frac{\partial}{\partial \xi_3} \end{pmatrix} \tag{2.19}$$

and

$$\begin{pmatrix} \frac{\partial}{\partial \xi_1} \\ \frac{\partial}{\partial \xi_2} \\ \frac{\partial}{\partial \xi_3} \end{pmatrix} = \begin{pmatrix} \frac{\partial x}{\partial \xi_1} & \frac{\partial y}{\partial \xi_1} & \frac{\partial z}{\partial \xi_1} \\ \frac{\partial x}{\partial \xi_2} & \frac{\partial y}{\partial \xi_2} & \frac{\partial z}{\partial \xi_2} \\ \frac{\partial x}{\partial \xi_3} & \frac{\partial y}{\partial \xi_3} & \frac{\partial z}{\partial \xi_3} \end{pmatrix} \begin{pmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} \end{pmatrix}. \tag{2.20}$$

Comparing Equations 2.19 and 2.20 reveals that $\mathcal{G} = \mathcal{H}^{-1}$.

Now that we have a means for computing the divergence on the master element

and transforming the result back to physical space, we also need a similar manner to compute the integrals on the master element. To begin, we first need a manner to transform from $d\mathbf{x}$ to $d\boldsymbol{\xi}$. This can be done through $d\mathbf{x} = Jd\boldsymbol{\xi}$, where $J = |\mathcal{H}|$. We can define a similar "surface" Jacobian J_s for the surface integrals. The final piece is to prescribe how the integral on the master element is carried out. Using numerical quadrature rules is an efficient and accurate manner to compute an integral. With numerical quadrature, one approximates a generic integral on the master element as

$$\int y(\boldsymbol{\xi})d\boldsymbol{\xi} \approx \sum_{k=0}^{N_q} y(\boldsymbol{\xi_k})\omega_k, \tag{2.21}$$

where $\boldsymbol{\xi_k}$ are the quadrature points with corresponding weights ω_k. There are many choices for a quadrature rule. In fact, the GLL points described earlier come from the GLL quadrature rule, and thus have a corresponding set of weights. The GLL quadrature rule is exact for polynomials $y(\boldsymbol{\xi})$ up to $2N_q - 1$, where N_q is the order of the quadrature set. There is also the Gauss-Legendre (GL) rule. The points are similar to the GLL rule, except that no points are forced to be the end points. As such, the rule is more accurate, being exact up to $2N_q + 1$. For this work, we will only focus on the GLL and GL rules.

After a quadrature rule and order is chosen, we can now return to our core operators. The mass matrix becomes

$$\begin{aligned} M_{ij} &= \int_{\Omega} \psi_i(\mathbf{x}(\boldsymbol{\xi}))\psi_j(\mathbf{x}(\boldsymbol{\xi}))d\mathbf{x} \\ &= \int_{\hat{\Omega}} \psi_i(\boldsymbol{\xi})\psi_j(\boldsymbol{\xi})J(\boldsymbol{\xi})d\boldsymbol{\xi} \\ &\approx \sum_{k}^{N_q} \psi_i(\boldsymbol{\xi}_k)\psi_j(\boldsymbol{\xi}_k)J(\boldsymbol{\xi}_k)\omega_k. \end{aligned}$$

The above definition of the elemental mass matrix is true for any quadrature rule and order we choose, and in general is full. However, if we match the quadrature rule to our interpolation grid (with $N_q = N$), we get

$$M_{ij} \approx \sum_{k}^{N} \delta_{ik}\delta_{jk} J(\boldsymbol{\xi}_k)\omega_k = \text{diag}(J_k\omega_k), \tag{2.22}$$

again due to the property $\psi_i(\boldsymbol{\xi}_k) = \delta_{ik}$. Here, $J_k = J(\boldsymbol{\xi}_k)$. This choice results in the mass matrix being diagonal, and thus the computation of M^{-1} (needed once we discretize the equations in time) is trivial. It should be noted, however, that this particular choice in general is not exact, as the polynomial being integrated is at least of order $2N$ (it is higher for deformed elements [45]), while the GLL quadrature being used is only exact for polynomials up to $2N - 1$. Thus, we are incurring an error through this choice, although this error should be very small (and will vanish exponentially for a given problem as the polynomial order N is increased).

Before the operator $\mathbf{C}$ is addressed, some notation is given to aid in the following discussion. First, the Lagrange derivative operator is given as,

$$D_{ij}^{\xi_a} = \left.\frac{\partial \psi_j}{\partial \xi_a}\right|_{\boldsymbol{\xi}=\boldsymbol{\xi}_i}. \tag{2.23}$$

The interpretation of this operator is "the ξ_a derivative of the j_{th} Lagrange interpolant evaluated at the point $\boldsymbol{\xi}_i$". As we are going to be performing Gaussian integrations, it is efficient to choose $\boldsymbol{\xi}_i$ as the quadrature points from our Gaussian rule. We also define a discrete version of our contravariant metric matrix $\mathcal{G}$, named G and defined as

$$G_k^{ab} = \left.\frac{\partial \xi_b}{\partial x_a}\right|_{\boldsymbol{\xi}=\boldsymbol{\xi}_k} \omega_k J_k, \tag{2.24}$$

where w_k are the weights of our chosen quadrature rule. The operator G^{ab} represents

d^2 diagonal matrices, where d is the dimension of the problem. With these two operators in mind, we now analyze $\mathbf{C}$. To keep things concise, we concentrate first on C_x. Using numerical quadrature for the integral, we get

$$\begin{aligned} C_{x,ij} &= \int_{\Omega} \frac{\partial \psi_i(\mathbf{x}(\boldsymbol{\xi}))}{\partial x} \psi_j(\mathbf{x}(\boldsymbol{\xi})) d\mathbf{x} \\ &= \int_{\hat{\Omega}} \frac{\partial \psi_i(\boldsymbol{\xi})}{\partial x} \psi_j(\boldsymbol{\xi}) J(\boldsymbol{\xi}) d\boldsymbol{\xi} \\ &\approx \sum_{k}^{N_q} \left(\sum_{b=1}^{d} \left. \frac{\partial \psi_i}{\partial \xi_b} \right|_{\boldsymbol{\xi}=\boldsymbol{\xi}_k} \left. \frac{\partial \xi_b}{\partial x} \right|_{\boldsymbol{\xi}=\boldsymbol{\xi}_k} \right) \psi_j(\boldsymbol{\xi}_k) J(\boldsymbol{\xi}) \omega_k \\ &\approx \sum_{k}^{N_q} \left(\sum_{b=1}^{d} D_{ik}^{\xi_b,T} G_k^{xb} \right) \psi_j(\boldsymbol{\xi}_k) \omega_k \end{aligned} \tag{2.25}$$

If we again assume that we are matching the quadrature rule exactly to the solution points, this reduces further to

$$C_{x,ij} \approx \sum_{k}^{N_q} \left(\sum_{b=1}^{d} D_{ik}^{\xi_b,T} G_k^{xb} \right) \delta_{ij} \omega_k = \left(\sum_{b=1}^{d} D^{\xi_b,T} G^{xb} \right) \mathrm{diag}(\omega_k). \tag{2.26}$$

The form for C_y and C_z is similar. Expanding $C_x \underline{f_x} + C_y \underline{f_y} + C_z \underline{f_z}$ and collecting like terms, the operation $\mathbf{C} \cdot \underline{\mathbf{f}}$ can be written as

$$\begin{pmatrix} D^{\xi_1,T} & D^{\xi_2,T} & D^{\xi_3,T} \end{pmatrix} \begin{pmatrix} G^{x\xi_1} & G^{y\xi_1} & G^{z\xi_1} \\ G^{x\xi_2} & G^{y\xi_2} & G^{z\xi_2} \\ G^{x\xi_3} & G^{y\xi_3} & G^{z\xi_3} \end{pmatrix} \begin{pmatrix} \underline{f_x} \\ \underline{f_y} \\ \underline{f_z} \end{pmatrix}. \tag{2.27}$$

The computation of M_s follows along the same lines as M. A quadrature rule is chosen for the face (1D for 2D simulations, 2D for 3D simulations). If the chosen quadrature rule is chosen to match the nodes on the face, efficiency gains similar to those found in M are obtained. However, in this case, not only will M_s be diagonal,

it will only be non-zero when a node is on a face.

2.3.4 A Word on Aliasing

A case has been made for the efficiency gains by matching the order and type of the quadrature rule to the interpolation points used for the Lagrange polynomials (GLL in this case). Although this introduced an error for the elemental mass matrix, the error should be small. This error is known as aliasing, and is a well known problem from spectral methods [52]. The basic idea is that, by integrating a polynomial on a quadrature rule insufficient to be exact, information about the polynomial is lost. If the aliasing error is large enough, this information can, in some cases, energize the high frequency modes in the solutions, and eventually can cause the solution to become unstable.

Nothing has been specified about the flux function $\mathbf{f}(u)$ in Equation 2.8 thus far. If it is linear in u, then most likely a scheme using inexact quadrature will be sufficient. However, the same cannot be assumed if the flux is non-linear. Consider, for example, the Burger's flux, $\mathbf{f}(u) = \mathbf{V}u^2/2$, where $\mathbf{V}$ can be taken as the vector of ones for the purposes of discussion. To simplify matters, we will assume that the element we are computing the weak divergence operator on is an undeformed element, such that the contravariant metrics are constants. Take the case now where our approximation is marginally resolved such that all $N+1$ modes of u are non-zero. To compute the flux coefficients for the matching GLL quadrature scheme discussed previously, we would simply square the coefficients u_j and divide by two. Then, the weak divergence operator would be applied as discussed before. In this case, we are attemping to integrate a polynomial of order $3N$ with a quadrature only exact for $2N-1$; this constitutes a more egregious aliasing error than seen for the mass matrix. In Hesthaven's book [45], it is shown how such an error results in a

rapid Gibb's phenomenon near discontinuities and ultimately drives the simulation unstable. To counteract this, we can choose to use more quadrature points than we have nodal points (i.e., $N_q > N$). If we choose $N_q = 3/2N$, then our GLL rule is exact for $3N - 1$ order polynomials. Although this is not exact for our worst case scenario, it at least brings the aliasing error to a comparable level with the aliasing error of the mass matrix. If we instead use a GL rule, then the integral is exact for the choice $N_q = 3/2N$ in undeformed elements. For either choice, the modification made to the operator $\mathbf{C}$ is the same. To compute the flux on the N_q quadrature points, we first interpolate the solution to the N_q points by the matrix-vector operation $B\underline{u}$, where the operator $B_{ij} = \psi_j(\boldsymbol{\xi}_i)$ is a basis matrix that effectively interpolates the solution from the solution grid points to the quadrature grid points. The flux is then computed as before, instead using the interpolated solution this time. The generalized weak divergence operation is

$$\begin{pmatrix} D^{\xi_1,T} & D^{\xi_2,T} & D^{\xi_3,T} \end{pmatrix} \begin{pmatrix} G^{x\xi_1} & G^{y\xi_1} & G^{z\xi_1} \\ G^{x\xi_2} & G^{y\xi_2} & G^{z\xi_2} \\ G^{x\xi_3} & G^{y\xi_3} & G^{z\xi_3} \end{pmatrix} \begin{pmatrix} B\underline{f_x} \\ B\underline{f_y} \\ B\underline{f_z} \end{pmatrix}, \tag{2.28}$$

where the operation $B\underline{f_x}$, for instance, is shorthand for $f_x(B\underline{u})$. The points where D^{ξ_a} and G^{ab} are evaluated are also changed to the new quadrature points (as such, the matrices D^{ξ_a} are no longer square). The case where the N_{th} order GLL points are chosen is recovered by setting B to the identity matrix. The same problem applies for the surface flux terms. In this case, the surface nodal values are interpolated to the higher order quadrature rule on the faces, and then the numerical flux function is computed. The generalized surface mass matrix operation is

$$M_s^e \underline{\hat{f}} = B_s^T W_s B_s \underline{\hat{f}}, \tag{2.29}$$

where B_s is the surface basis matrix and $W_s = \text{diag}(J_{s,k} w_{s,k})$, $k = 0, 1, ..., N_{q,s}$. Similarly to the weak divergence operator, the operator $B_s \underline{\hat{f}}$ implies $\hat{f}(B_s \underline{u^-}, B_s \underline{u^+}, \mathbf{n}^-)$.

Although compelling evidence is shown in [50, 40, 45] that this act of "dealiasing" can help stabilize solutions, even severely under-resolved solutions, the cost for the interpolations can be quite large. A more cost effective method for marginal resolution is to apply a weak filter to the higher modes. It is shown in [45] that this approach can have the same stabilizing effect as true dealiasing, although the accuracy will take a slight hit because of it.

The filtering approach is used in this work, primarily because of the nature of equations being solved. The required N_q to compute exact integrals can be derived for the Burgers flux above because the flux can naturally be represented as a polynomial. However, as will be seen later in this section, most of the fluxes in the Navier-Stokes equations involve a division by density; as such, the Navier-Stokes fluxes are rational functions, and cannot be represented exactly as a polynomial. Because of this, N_q can never be chosen large enough to compute the integrals in the DG formulation exactly (and avoid aliasing errors), meaning that some filtering may be needed even if dealiasing is performed. The filter employed is the exponential filter seen in [45]. The filter has the form

$$f(\eta) = e^{-\alpha \eta^\beta}, \tag{2.30}$$

where $\eta = (i - N_c)/(N - N_c)$, $i = N_c, N_c + 1, ..., N$. The α parameter determines the strength of the filter, and β determines the shape. The cutoff parameter N_c can be adjusted to exclude as many lower modes from the filter as possible (although, as shown, the lowest mode is always left unaffected). While the proper choice of the filter parameters is part science, part art, the default values chosen for this work for smooth flows are $N_c = 0$, $\alpha = 0.01$, and $\beta = 30$. For marginally-resolved turbulent

flows, α should be increased ($\alpha = 20$ effectively wipes out the last mode). For under-resolved flows, β should also be decreased in order to filter more of the medium to high modes.

2.3.5 Performance Improvements using Tensor Products

As mentioned previously, the current algorithm uses a tensor product of one dimensional Lagrange interpolants to formulate approximate solutions for 2D and 3D variables. In order to understand the motivation behind this, we now examine in further detail the derivative operator shown in Equation 2.23.

In a general nodal formulation, the solution and test function are represented as shown in Equation 2.12; expanding what is seen there, we have

$$u(\xi_1, \xi_2, \xi_3) = \sum_{n=0}^{N} u_n \psi_n(\xi_1, \xi_2, \xi_3) \tag{2.31}$$

For a general formulation, the derivative operators given in Equation 2.23 are full $(N+1) \times (N+1)$ matrices, and thus, for example, the matrix vector product $D^{\xi_1}\underline{u}$ is an $(N+1)^2$ operation. If we specifically choose a tensor product formulation, the solution is represented as

$$u(\xi_1, \xi_2, \xi_3) = \sum_{i=0}^{N_1} \sum_{j=0}^{N_2} \sum_{k=0}^{N_3} u_{ijk} \ell_i(\xi_1) \ell_j(\xi_2) \ell_k(\xi_3) = \sum_{i,j,k=0}^{N_1,N_2,N_3} u_{ijk} \ell_i(\xi_1) \ell_j(\xi_2) \ell_k(\xi_3). \tag{2.32}$$

Because of this specific choice, the derivative in the ξ_1 direction is

$$\frac{\partial u}{\partial \xi_1} = \sum_{i,j,k=0}^{N_1,N_2,N_3} u_{ijk} \ell_i'(\xi_1) \ell_j(\xi_2) \ell_k(\xi_3). \tag{2.33}$$

In the context of the integrations seen in the discontinuous Galerkin formulation, we

can now make use of an alternative differential operator,

$$\hat{D}_{ij}^{\xi_1} = \left.\frac{\partial \ell_j}{\partial \xi_1}\right|_{\xi_1=\xi_{1,i}}, \tag{2.34}$$

which is an $(N_1+1)\times(N_1+1)$ matrix (with similar matrices for ξ_2 and ξ_3). Assuming that the i index runs fastest, the tensor product version of the operator in (2.23) for ξ_1 is a block diagonal matrix with $\hat{D}$ on the diagonals. The operational cost for $D^{\xi_1}\underline{u}$ using the tensor product formulation is $(N_1+1)^2(N_2+1)(N_3+1)$. To illustrate the differences between a general nodal formulation and the tensor product formulation, assume that $N_1 = N_2 = N_3$ and $N+1 = (N_1+1)^3$. From these assumptions, we see that the cost of $D^{\xi_1}\underline{u}$ is $(N_1+1)^6$ for the general nodal formulation and $(N_1+1)^4$ for the tensor product formulation. For the high order formulations used in this work, this represents a significant savings in computational work. As operations such as $D^{\xi_1}\underline{u}$ occur frequently in the DGSEM formulation, this savings greatly enhances the serial performance of the algorithm.

2.3.6 Time Advancement

Returning to Equation 2.18 and dropping the test function coefficients, we have (for a single element)

$$\underline{u_t^e} = (M^e)^{-1}\left(\mathbf{C}^e \cdot \underline{\mathbf{f}^e} - M_s^e \underline{\hat{f}^e}\right) = R(\underline{u^e}, t), \tag{2.35}$$

where R is termed the residual. The final step is to determine how advancement in time is handled. There are many choices for the time discretization [50], but this work chooses from the family of low storage explicit Runge-Kutta schemes. This is mostly due to the large non-linearities in the flux functions seen in the Navier-Stokes equations (even the viscous terms), making implicit schemes difficult to implement.

Specifically, this work uses the 3rd order RK scheme detailed in [51]:

$$
\begin{aligned}
u^0 &= u(t^n) \\
G^0 &= R\left(u^0, t^n\right) \\
u^1 &= u^0 + \frac{1}{3}\Delta t G^0 \\
G^1 &= -\frac{5}{9}G^0 + R\left(u^1, t^n + \frac{1}{3}\Delta t\right) \\
u^2 &= u^1 + \frac{15}{16}\Delta t G^1 \\
G^2 &= -\frac{153}{128}G^1 + R\left(u^2, t^n + \frac{3}{4}\Delta t\right) \\
u\left(t^{n+1}\right) &= u^2 + \frac{8}{15}\Delta t G^2
\end{aligned}
\tag{2.36}
$$

Here, Δt is the chosen timestep. It should be noted that the only a single vector u and G need be stored for each element; for example, u^0 is overwritten by u^1, which is overwritten by u^2. This particular scheme is chosen because it includes a portion of the imaginary axis in its stability region, allowing it to be useful for hyperbolic systems such as the Euler equations as well as mixed systems such as the Navier-Stokes equations [51].

2.4 DGSEM: Euler Equations

If we assume that the viscous forces are neglible compared to inertial forces (i.e., $\mathrm{Re} \to \infty$), then the terms on the right hand side of Equations 2.7 can be dropped resulting in the inviscid flow equation, or Euler equations. In vector form, these are

$$
\frac{\partial Q}{\partial t} + \nabla \cdot \mathbf{F}(Q) = 0, \tag{2.37}
$$

where $Q = [\rho, \rho\mathbf{u}, E]^T$ is the vector of conservative variables, and $\mathbf{F}(Q)$ is the flux vector, which is constructed using the inviscid terms from Equations 2.7. The Euler equations are very similar to the example conservation law given in Equation 2.8, with the exception that we are now dealing with a system of equations. Carefully following the discretization process seen in the previous section, we arrive at the semi-discrete form of the Euler equations:

$$\underline{Q_t^e} = (M^e)^{-1} \left(\mathbf{C}^e \cdot \underline{\mathbf{F}^e} - M_s^e \underline{\widetilde{F}^e} \right), \tag{2.38}$$

and these equations are advanced in time using the low storage Runge Kutta scheme given in Equations 2.36. To clarify the formulations above (and in the subsequent discussion), it should be noted that the bold face font will only be used for vectors in the Cartesian sense (e.g., a vector with x, y, and z components, such as $\mathbf{u}$). Thus, although Q is indeed a vector with five components, it will not use a bold face font in order to keep the notation between the scalar law in Equation 2.8 and a system such as the Euler equations consistent.

2.4.1 Numerical Fluxes

In general, any choice for a flux function should have the properties

$$\widetilde{F}(Q, Q, \mathbf{n}) = \mathbf{F}(Q) \cdot \mathbf{n}, \tag{2.39}$$

$$\widetilde{F}(Q^-, Q^+, \mathbf{n}^-) = -\widetilde{F}(Q^+, Q^-, \mathbf{n}^+), \tag{2.40}$$

with the former ensuring the flux is consistent and the latter ensuring the flux is conservative. An extremely simple flux function that follows these properties is

$$\widetilde{F}(Q^-, Q^+, \mathbf{n}^-) = \frac{1}{2}(\mathbf{F}^- + \mathbf{F}^+) \cdot \mathbf{n}^-, \tag{2.41}$$

where $\mathbf{F}^- = \mathbf{F}(Q^-)$. This numerical flux is simply the average of the fluxes using the two states at the element boundary. Unfortunately, while this is the simplest flux to use, it is not well suited for use in the Euler equations. This is due to the nature of hyperbolic systems. For example, if we assume a one dimensional flow moving from left to right at purely supersonic speeds, then the proper flow of information is also from left to right (upwind). Thus, the appropriate numerical flux at an element interface in this case should use the state on the left of the interface. For subsonic flows, the answer is not as simple, but the goal is the same; an appropriate numerical flux for the Euler equations is one the considers proper flow of information due to upwinding.

For linear equations, it is relatively simple to identify the correct state (or combination of states) to use. This is because the flux is not a function of the state variables; this is not true for the non-linear fluxes seen in the Euler equations. The essential idea behind constructing the flux at an element interface (or physical boundary) is the solution of the Riemann problem. The Riemann problem is a classical conservation law problem where the initial condition has a discontinuity at a single point. The computation of the numerical flux at in interface is essentially the same as solving a Riemann problem. The most accurate solution method is solving the Riemann problem exactly. However, this approach is unattractive because iteration is required to compute the solution. As the Riemann problem must be solved at each grid point on each face, this process would end up being very expensive. An alternative approach is to seek an approximate solution. The basic idea behind approximate solvers is to linearize the flux vector around a constant state that depends on the variables on the left (Q^-) and the right (Q^+) of the element surface. Further details on approximate Riemann solvers can be found in [17].

A simple, yet robust approximate Riemann flux is the Lax-Friedrichs flux. This

flux is formulated as [45]

$$\widetilde{F}_{\mathrm{LF}}(Q^-, Q^+, \mathbf{n}^-) = \frac{1}{2}\left(\mathbf{F}(Q^-) + \mathbf{F}(Q^+)\right) \cdot \mathbf{n}^- - \frac{|C|}{2}\left(Q^+ - Q^-\right), \tag{2.42}$$

where C is the maximum eigenvalue of the approximate state flux Jacobian $\hat{A} = \frac{\partial \mathbf{F}}{\partial Q}(\hat{Q}) \cdot \mathbf{n}$, where $\hat{Q}$ is a yet to be determined combination of the left and right states. The general flux Jacobian $A = \frac{\partial \mathbf{F}}{\partial Q} \cdot \mathbf{n}$ has the eigenvalues $\mathbf{u} \cdot \mathbf{n} + a$, $\mathbf{u} \cdot \mathbf{n} - a$ and $\mathbf{u} \cdot \mathbf{n}$ (with the latter having a multiplicity of three). Therefore, the maximum absolute eigenvalue for positive $\mathbf{u} \cdot \mathbf{n}$ is the first eigenvalue (the second eigenvalue if $\mathbf{u} \cdot \mathbf{n} < 0$). As a quick approximation, C can be chosen by choosing the maximum of the eigenvalues of $A(Q^-)$ and $A(Q^+)$. However, this could lead to the numerical flux being overly-diffusive. A better choice for $\hat{Q}$ is the Roe-averaged state. This is defined as [45]

$$\begin{aligned}
\hat{\rho} &= \sqrt{\rho^- \rho^+} \\
\hat{u}_1 &= \frac{\sqrt{\rho^-} u_1^- + \sqrt{\rho^+} u_1^+}{\sqrt{\rho^-} + \sqrt{\rho^+}} \\
\hat{u}_2 &= \frac{\sqrt{\rho^-} u_2^- + \sqrt{\rho^+} u_2^+}{\sqrt{\rho^-} + \sqrt{\rho^+}} \\
\hat{u}_3 &= \frac{\sqrt{\rho^-} u_3^- + \sqrt{\rho^+} u_3^+}{\sqrt{\rho^-} + \sqrt{\rho^+}} \\
\hat{H} &= \frac{\sqrt{\rho^-} H^- + \sqrt{\rho^+} H^+}{\sqrt{\rho^-} + \sqrt{\rho^+}},
\end{aligned} \tag{2.43}$$

where $H = \frac{E+p}{\rho}$ is the enthalpy. The velocities (with subscripts 1, 2, and 3) refer to a rotated coordinate system with x_1 aligned with $\mathbf{n}^-$ and x_2 and x_3 as the tangential components. The Roe speed of sound is

$$\hat{a}^2 = (\gamma - 1)\left(\hat{H} - \frac{\hat{\mathbf{u}} \cdot \hat{\mathbf{u}}}{2}\right). \tag{2.44}$$

Finally, $C = \max(\hat{u}_1 + \hat{a}, \hat{u}_1 - \hat{a})$. While this approach is a bit more computational intensive, it should reduce the diffusion at the element interfaces.

While the Lax flux is sufficient for the flows studied in this work, a more accurate choice that is still computationally efficient is from Roe's approximate Riemann solver. This flux is given as [17]

$$\widetilde{F}_{\text{Roe}}(Q^-, Q^+, \mathbf{n}^-) = \frac{1}{2}\left(\mathbf{F}^- + \mathbf{F}^+\right) \cdot \mathbf{n} - \frac{1}{2}\sum_{i=1}^{5} \hat{\alpha}_i |\hat{\lambda}_i| \hat{K}^i, \tag{2.45}$$

where $\hat{\lambda}_i$ is the i^{th} eigenvalue of the Roe-averaged flux Jacobian $\hat{A}(\hat{Q})$ with corresponding right eigenvector $\hat{K}^i$, and $\hat{\alpha}_i$ is the corresponding wave strength. The details of this flux can be found in [17]. Other choices for the numerical flux are possible, such as the Harten-Lax-van Leer (HLL) and Harten-Lax-van Leer Contact (HLLC) fluxes. However, these approaches are computationally more expensive, and did not give any noticeable improvements for the test cases shown in Section 3. As such, the Roe flux is chosen for the purposes of this work.

2.5 DGSEM: Navier-Stokes

The Navier-Stokes equations in vector form are

$$\frac{\partial Q}{\partial t} + \nabla \cdot \mathbf{F}(Q) = \nabla \cdot \mathbf{F}_v\left(Q, \nabla Q\right), \tag{2.46}$$

with $\mathbf{F}_v$ as the viscous flux vector. The subsequent semi-discrete DGSEM form is

$$\underline{Q_t^e} = (M^e)^{-1}\left(\mathbf{C}^e \cdot \left(\underline{\mathbf{F}^e} - \underline{\mathbf{F}_v^e}\right) - M_s^e\left(\underline{\hat{F}^e} - \underline{\hat{F}_v^e}\right)\right), \tag{2.47}$$

with $\underline{\mathbf{F}_v^e} = \mathbf{F}_v(\underline{Q^e}, \underline{\nabla Q^e})$ and $\underline{\widetilde{F}_v^e} = \widetilde{F}_v\left(\underline{Q}^-, \underline{\nabla Q}^-, \underline{Q}^+, \underline{\nabla Q}^+, \mathbf{n}^-\right)$ as the viscous numerical flux. The proper formulation for $\widetilde{F}_v$ is not immediately obvious; we use the

heat equation to help formulate an appropriate numerical flux for diffusion operators.

The heat equation is

$$\frac{\partial u}{\partial t} = \nabla \cdot \mathbf{q}, \tag{2.48}$$

where $\mathbf{q} = k\nabla u$ is the heat flux vector. The first attempt at solving this equation in a discontinuous Galerkin framework is to use $\mathbf{q}$ in a manner similar to how $\mathbf{f}$ was used in the scalar conservation law (Equation 2.8). The resulting semi-discrete scheme is

$$\underline{u_t^e} = -\left(M^e\right)^{-1}\left(\mathbf{C}^e \cdot \underline{\mathbf{q}^e} + M_s^e \underline{\widetilde{q}^e}\right), \tag{2.49}$$

with $\widetilde{q}^e(u^-, u^+, \mathbf{n}^-) = \frac{1}{2}\left(\mathbf{q}(u^-) + \mathbf{q}(u^+)\right)$ as the numerical flux (similar to average numerical flux in Equation 2.41). Unfortunately, this scheme does not perform as desired; at low polynomial orders N, the scheme does not converge by increasing the number of elements used, and the scheme is unstable at high N [45]. Thus, an alternative approach is needed.

Equation 2.48 is first split into a first order system of equations as

$$\frac{\partial u}{\partial t} = \nabla \cdot (k\mathbf{S}) \tag{2.50}$$

$$\mathbf{S} = \nabla u, \tag{2.51}$$

with S denoted as auxiliary variables. Multiplying Equation 2.50 by the test function v and Equations 2.51 by the vector valued test function $\boldsymbol{\phi}$ and applying the divergence theorem where appropriate, the DG formulation is obtained as

$$\int_{\Omega^e} \frac{\partial u}{\partial t} v d\mathbf{x} = -\int_{\Omega^e} \nabla v \cdot \mathbf{q} d\mathbf{x} + \oint_{\partial\Omega^e} \widetilde{q} v ds, \tag{2.52}$$

$$\int_{\Omega^e} \mathbf{S} \cdot \boldsymbol{\phi} d\mathbf{x} = -\int_{\Omega^e} u \nabla \cdot \boldsymbol{\phi} d\mathbf{x} + \oint_{\partial\Omega^e} \widetilde{u} \boldsymbol{\phi} \cdot \mathbf{n} ds, \tag{2.53}$$

with $\mathbf{q} = k\mathbf{S}$. The new numerical fluxes are

$$\widetilde{q}(\mathbf{S}^-, \mathbf{S}^+, \mathbf{n}^-) = \frac{1}{2}\left(\mathbf{q}(\mathbf{S}^-) + \mathbf{q}(\mathbf{S}^+)\right) \cdot \mathbf{n}^-, \tag{2.54}$$

$$\widetilde{u}(u^-, u^+) = \frac{1}{2}\left(u^- + u^+\right). \tag{2.55}$$

A mathematically equivalent formulation for Equation 2.53 is obtained by integrating by parts again:

$$\int_{\Omega^e} \mathbf{S} \cdot \boldsymbol{\phi} d\mathbf{x} = \int_{\Omega^e} \nabla u \cdot \boldsymbol{\phi} d\mathbf{x} + \oint_{\partial\Omega^e} [[u]] \boldsymbol{\phi} \cdot \mathbf{n} ds, \tag{2.56}$$

with $[[u]] = (u^+ - u^-)/2$ the jump in the solution variable.

The semi-discrete form for Equations 2.52 and 2.53 or 2.56 are then obtained by inserting approximations for u and $\mathbf{S}$. This formulation is known as the BR1 scheme [37]. Other types of schemes are possible, but for the simulations done in this work, the BR1 scheme gives accurate and stable results.

Applying the BR1 scheme to the Navier-Stokes equation, we get

$$\underline{Q_t^e} = (M^e)^{-1}\left(\mathbf{C}^e \cdot \left(\underline{\mathbf{F}^e} - \underline{\mathbf{F}_v^e}\right) - M_s^e\left(\underline{\hat{F}^e} - \underline{\hat{F}_v^e}\right)\right), \tag{2.57}$$

$$\underline{\mathbf{S}^e} = (M^e)^{-1}\left(\mathbf{D}^e \underline{Q^e} + M_s^e \underline{[[Q]]}\right), \tag{2.58}$$

with $\underline{\mathbf{F}_v^e} = \mathbf{F}_v\left(\underline{Q^e}, \underline{\mathbf{S}^e}\right)$ and

$$\underline{\hat{F}_v^e} = \hat{F}_v(\underline{Q^-}, \underline{\mathbf{S}^-}, \underline{Q^+}, \underline{\mathbf{S}^+}, \mathbf{n}^+) = \frac{1}{2}\left(\mathbf{F}_v^- + \mathbf{F}_v^+\right) \cdot \mathbf{n}^-. \tag{2.59}$$

It should be noted that while the viscous flux vector originally involved the primitive variables (velocity and temperature), they must be computed numerically using the conservative variables Q as well as $\mathbf{S} = \nabla Q$. For example, the gradient of the

velocity in the x direction, u, is computed as

$$\nabla u = \frac{1}{\rho}\left(\nabla m - \frac{m}{\rho}\nabla\rho\right), \tag{2.60}$$

where $m = \rho u$. In the above formulation, the $\mathbf{S}$ equivalents of ∇m and $\nabla\rho$ are used; it is not sufficient to simply use the gradient of m and ρ, as this would not take into account the discontinuous nature of the solution at element boundaries.

2.6 Boundary Conditions

In the DG framework, boundary conditions are weakly imposed through the use of the numerical fluxes on the boundary. In general, at a boundary, the numerical fluxes become

$$\widetilde{F} = \widetilde{F}(Q^-, Q^{\mathrm{BC}}, \mathbf{n}^-), \tag{2.61}$$

$$\widetilde{F}_v = \widetilde{F}_v(Q^-, \mathbf{S}^-, Q^{\mathrm{BC}}, \mathbf{S}^{\mathrm{BC}}, \mathbf{n}^-), \tag{2.62}$$

$$[[Q]] = Q^{\mathrm{BC}} - Q^-. \tag{2.63}$$

The exact form of Q^{BC} and $\mathbf{S}^{\mathrm{BC}}$ depends on the type of boundary condition being specified.

2.6.1 Inviscid Boundary Conditions

The inviscid boundary conditions are applied through the inviscid numerical flux, $\widetilde{F}$. As such, they are applicable to both the Euler equations and the Navier-Stokes equation.

One of the most important boundary conditions is a wall. The only condition that can be stipulated at the wall is no penetration; i.e., $\mathbf{u}\cdot\mathbf{n} = 0$. This is accomplished numerically by copying the interior density and total energy to the boundary state

vector, and reflecting the momentum vector. That is, we set Q^{BC} to

$$Q^{\mathrm{BC}} = \begin{bmatrix} \rho^- \\ (\rho\mathbf{u})^- - 2((\rho\mathbf{u})^- \cdot \mathbf{n}^-)\mathbf{n}^- \\ E^- \end{bmatrix} \tag{2.64}$$

Inlets and outlets can be set in multiple ways. The simplest inlet/outlet boundary condition is the far field boundary condition. This can be applied if the full state of the boundary is known. This is typically applied in external flow problems if the external boundaries are placed far enough away from the object being studied. The boundary state is simply set using the external flow variables. The advantage of this boundary condition is that the question of whether a particular point on the boundary is an inlet or outlet does not need to be settled *a priori*; the Riemann solver will compute the flux correctly either way.

If all variables are not known prior to the simulation (as is common for internal flows), then other approaches can be taken using the variables that *are* known. A common method for inlets is to specify the total pressure, total temperature, and flow direction [53]. The total pressure and temperature of a flow (also known as the stagnation pressure and temperature) is the pressure and temperature of the flow if it is brought isentropically to rest; that is, in the absence of viscous forces or shocks. Using this these quantities, and assuming that the flow across the boundary is adiabatic and isentropic, the appropriate boundary state can be set. First, the total enthalpy, as well as the outgoing Riemann invariant [19], is computed using the

internal variables as

$$H_t = \frac{p^-}{\rho^-}\left(\frac{\gamma}{\gamma-1}\right) + \frac{1}{2}\mathbf{u}^- \cdot \mathbf{u}^- \tag{2.65}$$

$$R^+ = \mathbf{u}^- \cdot \mathbf{n} + \frac{2a^-}{\gamma-1} \tag{2.66}$$

From the assumption of an adiabatic flow, the total enthalpy is conserved across the boundary [53], giving

$$H_t = \frac{a^{\mathrm{BC}}, 2}{\gamma-1} + \frac{1}{2}\left(\mathbf{u}^{\mathrm{BC}} \cdot \mathbf{n}\right)^2. \tag{2.67}$$

Also, by definition, the outgoing Riemann invariant gives

$$R^+ = \mathbf{u}^{\mathrm{BC}} \cdot \mathbf{n} + \frac{2a^{\mathrm{BC}}}{\gamma-1}. \tag{2.68}$$

Combining Equations 2.68 and 2.67 results in a quadratic equation for the boundary speed of sound (not shown here for brevity). The normal velocity at the boundary is computed using Equation 2.68, and then the boundary Mach number follows. The Mach number is used to compute the boundary temperature and pressure using the specified stagnation temperature and pressure. Finally, the boundary velocity is rotated using the specified flow direction. Further details can be found in [53].

An outlet boundary condition commonly used for internal flow problems is the specifiction of a back pressure. The density and velocity at the outlet are set as [53]

$$\rho^{\mathrm{BC}} = \frac{p_b \gamma \mathrm{Ma}^2}{T^-},$$

$$\mathbf{u}^{\mathrm{BC}} = \mathbf{u}^-,$$

where p_b is the back pressure being set. From here, the conservative variables at the boundary can be constructed.

2.6.2 Viscous Boundary Conditions

As seen in Equation 2.62, the specification of boundary conditions through the viscous flux can potentially have two parts; a Dirichlet condition through Q^{BC}, and a Neumann condition through $\mathbf{S}^{\mathrm{BC}}$. For inlets and outlets, $\mathbf{S}^{\mathrm{BC}}$ is simply set to $\mathbf{S}^{-}$, and the numerical flux is computed according to Equation 2.62.

There are two types of wall boundary conditions employed in this work: an isothermal wall and an adiabatic wall. For both walls, a no slip condition is applied ($\mathbf{u}^{\mathrm{BC}} = 0$). For an isothermal wall, the temperature is also set as $T^{\mathrm{BC}} = T_{\mathrm{wall}}$. The pressure at the wall is set equal to the internal pressure, and the density at the wall is computed according to the ideal gas law. The boundary values for the auxiliary variables are set to the values of the internal state. For the adiabatic wall, the pressure and temperature (and thus, density) are copied from the internal state. The auxiliary variables for density and momentum are set to the interior values, and the auxiliary variable for energy is modified such that $\nabla T \cdot \mathbf{n} = 0$. For both of these cases, the numerical flux is computed directly from the specification of Q^{BC} and $\mathbf{S}^{\mathrm{BC}}$ as opposed to averaging with the internal viscous flux (as suggested by Equation 2.62).

The symmetry condition is another wall-type boundary condition, and it behaves in a similar fashion as a wall in the Euler equations. Like the inviscid wall, the component of the velocity normal to the boundary is set to zero; the rest of the boundary state is copied from the internal state. In addition to specifying zero heat flux (as in an adiabatic wall), the shear stress is also set to zero. This can be done numerically in two steps (using Einstein index notation):

- Compute $\tau_{w,i} = \tau_{ij} n_j$
- Set $\tau_{w,i} t_{1,i} = 0$ and $\tau_{w,i} t_{2,i} = 0$, where $\mathbf{t}_1$ and $\mathbf{t}_2$ are the two tangential vectors

at the wall.

As in the wall boundary conditions specified above, the numerical flux is computed using the specified boundary values for Q and $\mathbf{S}$.

In all cases, the jump vector $[[Q]]$ is computed as in Equation 2.63.

2.7 Implementation Details

As mentioned in the introduction, the algorithm in question is implemented using NEK5000 as a framework. This allows us to focus only on the implementation of the method itself, leveraging the existing NEK5000 pre- and post-processing infrastructure, as well as NEK5000's excellent communication library.

A current limitation of the code is that the polynomial order must be the same in the three master element coordinate directions; that is, $N_1 = N_2 = N_3$ (except for 2D simulations, where N_3 is set to zero). The same polynomial order must also used for all elements; the algorithm is currently not set up for variable polynomial orders. These two limitations greatly simplify the coding of the algorithm, and also help guarantee that the load balancing on each processor is optimal. However, as the discontinuous Galerkin method is particular well suited for non-conformal meshes, this is an obvious improvement in the future development of the code. From hereon, the polynomial order N will be taken to mean the polynomial order in each direction.

In the standard Galerkin formulation (which NEK5000 uses in normal operation), a solution that is continuous throughout the domain is sought. Thus, while the domain is still discretized into individual elements in the spectral element formulation, the problem is still ultimately a strongly global problem. As such, the discrete representation of Equation 2.9 is

$$M_g \underline{u} + \mathbf{C}_g \cdot \underline{\mathbf{f}} = 0, \tag{2.69}$$

where M_g and $\mathbf{C}_g$ are global versions of the discrete mass matrix and advection operators discussed previously. In order to avoid explicit formation of the global system, the idea of a gather-scatter operation is exploited.

To simplify the following discussion, we will consider the one-dimensional form of the conservation law. Consider a solution point u_j (in the global sense) is shared by elements e and $e+1$. This solution data (which by definition is the same for elements e and $e+1$) is stored locally for each element as u_N^e and u_0^{e+1}. In order to compute, for example, the global matrix vector product $M_g\underline{u}$, one first computes locally $\underline{h}^e = M^e\underline{u^e}$ and $\underline{h}^{e+1} = M^{e+1}\underline{u^{e+1}}$. To form the global product at the global solution point j, we simply sum the contributions h_N^e and h_0^{e+1} together (gather), and then the resulting sum is stored back in the local representations (scatter). In summary, the gather operation forms the global representation from the local operations, and the scatter operation ensures that the local representations have the same data.

As this is essentially the only communication operation in a Galerkin spectral element formulation, it is vital that the gather-scatter operation be efficient in order for the code to perform well in parallel. As such, a focus of the NEK5000 development has been to optimize this portion of the code, leading to unprecedented parallel performance on massively-parallel machines. In the discontinuous Galerkin formulation, only local problems are solved which are weakly tied together through numerical fluxes; as such, the gather-scatter operation does not play a direct role. Instead, information needs to be swapped between neighboring elements. However, rather than rewrite NEK5000's communication library, the gather-scatter operation is manipulated mathematically allowing the performance of the existing communication library to be leveraged.

A simple example of this is in the inviscid flux computations. For these numerical fluxes, assuming the same 1D simulation as above, element e needs the data at u_0^{e+1}

and element $e+1$ needs u_N^e. This swap is performed in parallel as

- Element e copies u_N^e to a temporary variable h^e, and element $e+1$ copies u_0^{e+1} to h^{e+1}.
- Gather-scatter is performed on h, giving the result $h^e = h^{e+1} = u_N^e + u_0^{e+1}$.
- Element e extracts u_0^{e+1} by subtracting u_N^e from h^e, and element e similarly extracts u_N^e.

While some extra operations are involved than a simple swap, they are miniscule compared to the scheme as a whole. Finally, the flux is computed locally for each element using the newly gathered neighbor data, relying on the conservation property of the chosen numerical flux in order to ensure that the flux contributions are in fact equal and opposite. This simple leveraging of the existing NEK5000 communication framework ensures that the current algorithm will also see the same excellent parallel performance.

3. VALIDATION TESTS

The current algorithm is now validated against a variety of test flows, using both the Euler equations and the Navier-Stokes equations. The tests for the Euler equations are subsonic flow in a channel with a bump, over a cylinder, and in converging-diverging nozzle. The test cases for the Navier-Stokes equations include a validation case with a manufactured solution, subsonic, transonic, and supersonic flow over a viscous cylinder, weakly turbulent flow over a sphere, and turbulent flow in a channel.

3.1 Euler Cases

Although the ultimate purpose of this code is the simulation of turbulent flows, the Euler equations are an important test in the development of the code. In high Reynolds number flows, the most important terms are the inviscid terms; simulating the Euler equations allows these terms to be tested in isolation, ensuring they are performing as expected. Additionally, high order methods are more prone to instabilities than low order methods due to the lack of numerical dissipation. Simulating the Euler equations, with their inherent lack of diffusion, is a strong test of the stability of the method. If the method can be proven stable for the Euler equations, then it will most likely be stable for viscous simulations as well.

3.1.1 Subsonic Flow in Channel with Bump

The flow in question is a subsonic (Ma = 0.3) flow in a channel with a semicircular bump obstruction. This particular case is chosen to test the convergence properties of the algorithm. The flow is isentropic; thus, the accuracy of the algorithm for this case can be judged by tracking any changes in entropy. As the problem is refined,

any deviations from the initial entropy value should vanish.

The geometry (along with a sample mesh) is shown in Figure 3.1. The limits of the domain are $x \in [-3, 3]$ and $y \in [0, 2]$. The semi-circular bump located at $x = 0$ has a radius of 0.5. The domain is discretized with an unstructured mesh consisting of 14 elements. The polynomial order is varied between $N = 3$ and $N = 13$ in increments of two. The left and right boundaries of the domain are set as far field boundaries, with the specification $[\rho^{\text{BC}}, u^{\text{BC}}, v^{\text{BC}}, T^{\text{BC}}] = [1, 1, 0, 1]$. The top and bottom boundaries are set as inviscid slip walls. The flow is initialized using the far field values, and solution is advanced until a steady state is reached.

Figure 3.2 shows contours of Mach number for the $N = 7$ case. The plot shows the relatively large acceleration seen at the top of the bump. The results compare well qualitatively with those seen in [54]. As there are no shocks present in the flow, the flow is isentropic. Therefore, any deviations from isentropic error are due to errors from the discretization. The entropy error is defined as

$$S_{\text{err}} = \frac{p/p_{\text{ref}}}{(\rho/\rho_{\text{ref}})^{\gamma}} - 1, \tag{3.1}$$

where p_{ref} and ρ_{ref} can be chosen as in the inlet values, for convenience. Figure 3.3 shows the convergence (in the L_2 norm) of the error in entropy, exhibiting the exponential convergence property of spectral elements.

3.1.2 Subsonic Flow over a Cylinder

The next test case is 2D inviscid flow over a cylinder, a test case detailed in [55]. The flow variables are similar to the bump case, except that the reference Mach number is 0.38. While the steady state solution is purely subsonic, the flow experiences a weak shock near the cylinder surface during the transition period. As such, this is a good test of the code's stability properties in the presence of discontinuities.

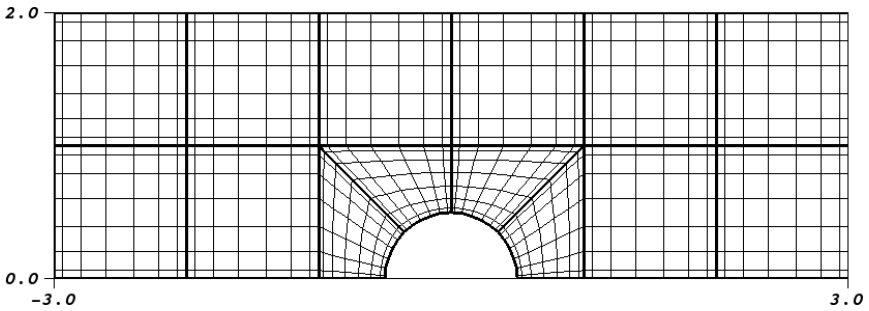


Figure 3.1: Mesh for channel flow with semicircular bump. The mesh consists of 14 elements, which are shown with the thicker lines. The lighter lines represent the Gauss-Lobatto-Legendre points for polynomial order $N = 7$.

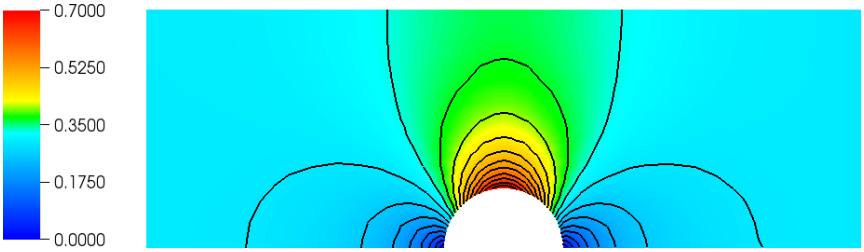


Figure 3.2: Contours of Mach number for channel with semicircular bump. As the flow progresses over the bump, the flow accelerates, causing a drop in pressure. For the chosen flow variables, no shocks are seen; as such, the entropy in the flow should remain constant everywhere.

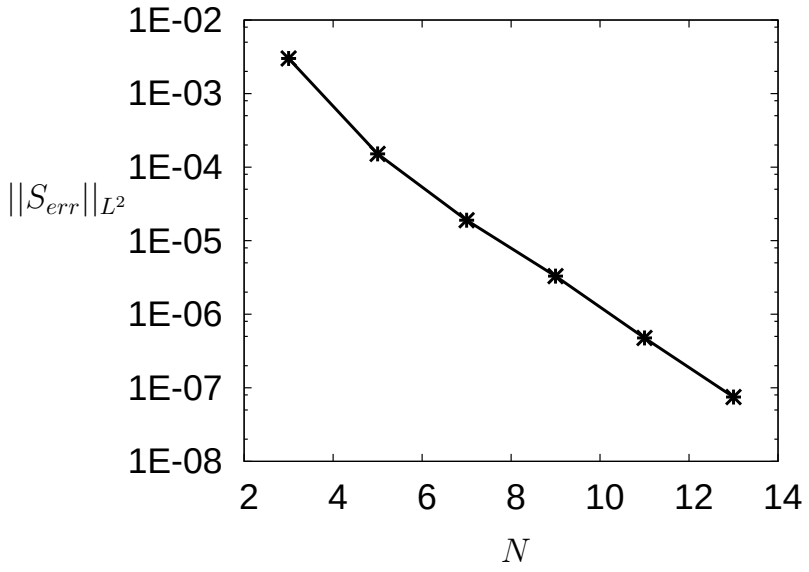


Figure 3.3: Convergence of L_2 norm of entropy error for channel problem. As the polynomial order is increased, the entropy error approaches zero exponentially fast.

Far field boundaries are set at $R/D = 20$ using the same conditions seen for the bump case. The initial conditions are set to the far field boundary conditions, and the simulation is run until a steady state is reached. Simulations were run on 3 different meshes (4×16, 8×32, and 16×64) and with polynomial orders ranging from $N = 3$ to $N = 7$. The coarse mesh at $N = 5$ is shown in Figure 3.4 for the window $[-2, 2]^2$. For each mesh, the elements are spaced evenly in the circumferential direction, and the elements in the radial direction are grown with a geometric rato of approximately 3.3, 1.8, and 1.3 respectively.

Figure 3.5 shows contours of Mach number at $N = 3$ for the 8×32 mesh, which compares qualitatively well to the results shown by Bassi and coworkers [55]. An important thermodynamic variable for aerodynamic calculations is the coefficient of pressure,

$$C_p = \frac{p - p_{\text{ref}}}{\frac{1}{2}\rho_{\text{ref}} U_{\text{ref}}^2}, \tag{3.2}$$

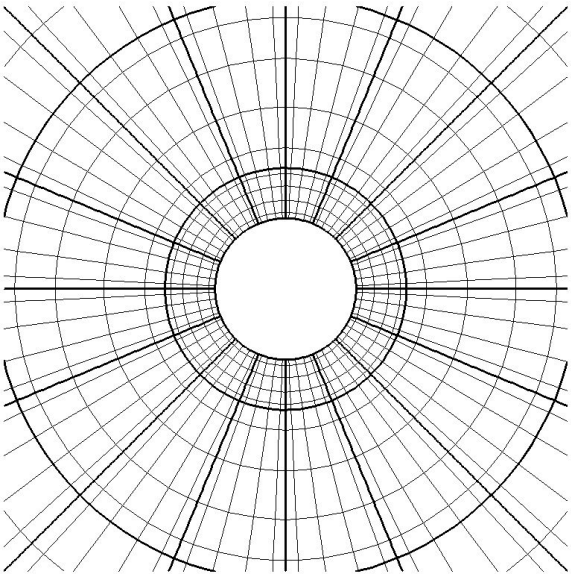


Figure 3.4: Coarse mesh for inviscid subsonic cylinder. The mesh uses 16 equally spaced elements circumferentially and 4 elements radially with a geometric growth factor of approximately 3.3. The element boundaries (darker lines) are shown along with the GLL points ($N = 5$).

where the reference variables are taken as the inlet quantities. To give a quantitative measure of accuracy, Figure 3.6 shows the coefficient of pressure for the coarse mesh at $N = 7$ as compared to the data given in the work by Bassi [55], showing good agreement with the previous CFD results.

Figure 3.7 shows convergence of the error in entropy for both h-type (increasing the number of elements) and p-type (increasing the polynomial order) refinement. The h-type refinement is done on the three meshes at $N = 3$, while the p-type refinement is done on the coarsest mesh. As the figure shows, achieving an error on the order of 10^{-6} requires approximately 4 times fewer points using p-type refinement over h-type refinement, and the trend of the two refinements show that the this gap would be greater if the desired error is lower. This is a practical example of how a method with a variable order discretization, such as spectral elements, can quickly

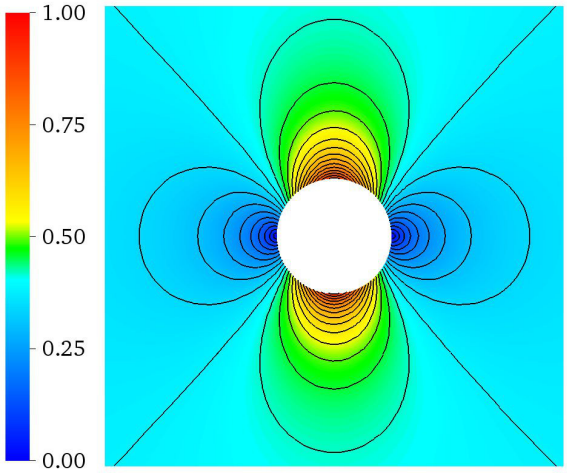


Figure 3.5: Contours of Mach number of subsonic inviscid flow over cylinder. The results shown are for medium mesh at $N = 3$. Contour lines shown are $\mathrm{Ma}_0 + i\Delta\mathrm{Ma}$, with $\mathrm{Ma}_0 = 0$ and $\Delta\mathrm{Ma} = 0.038$.

achieve convergence for smooth problems.

Due to the start up shock, the polynomial degree could not be increased beyond $N = 7$ without significantly increasing the filter strength (which effectively reduces the accuracy of the scheme back to a lower order). While the goals of this work are to simulate subsonic and transonic viscous flows, where the inherent dissipation of the viscous terms would stabilize weak shocks such as the one seen here, more robust shock-handling schemes will be developed in future work to expand the usefulness of the code.

3.1.3 Subsonic Flow through Converging-Diverging Nozzle

The final Euler test case is a subsonic converging-diverging nozzle verification test found on the NPARC Alliance CFD Verification and Validation Archive [56]. Although the case is axisymmetric, it is solved with the current algorithm as a test

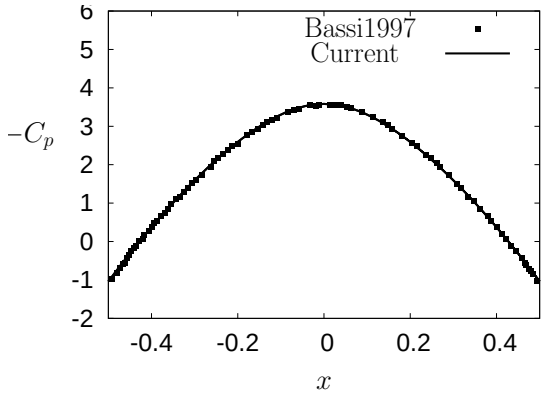


Figure 3.6: Coefficient of pressure along cylinder surface. Data from current algorithm taken from coarse mesh at $N = 7$, which shows good agreement with data from [55].

for the 3D solver. The profile of the nozzle is given by

$$A(z) = \begin{cases} 1.75 - 0.75\cos((0.2z - 1.0)\pi), & z < 5.0 \\ 1.25 - 0.25\cos((0.2z - 1.0)\pi), & z > 5.0 \end{cases}, \tag{3.3}$$

where z is the axial coordinate in inches and A is the cross-sectional area in squared inches. The mesh for the duct is shown in Figure 3.8.

At the inlet, the total temperature and pressure are specified as $P_t = 1$ psi and $T_t = 100°\text{R}$, with the flow direction given as $\mathbf{n} = [0, 0, 1]$. At the outlet, the back pressure is specified at $p_b = 0.89$ psi. The initial conditions are set by assuming an inlet Mach number of 0.2. The flow variables are non-dimensionalized using the stagnation flow properties (the chosen reference velocity in this case is the speed of sound at the stagnation properties), and the geometry is non-dimensionalized by the duct length, $L_{\text{ref}} = 10$ in. The simulation is advanced in time until the solution stops oscillating and a steady state has been reached.

Figure 3.9 shows the static pressure and Mach number variation along the axial

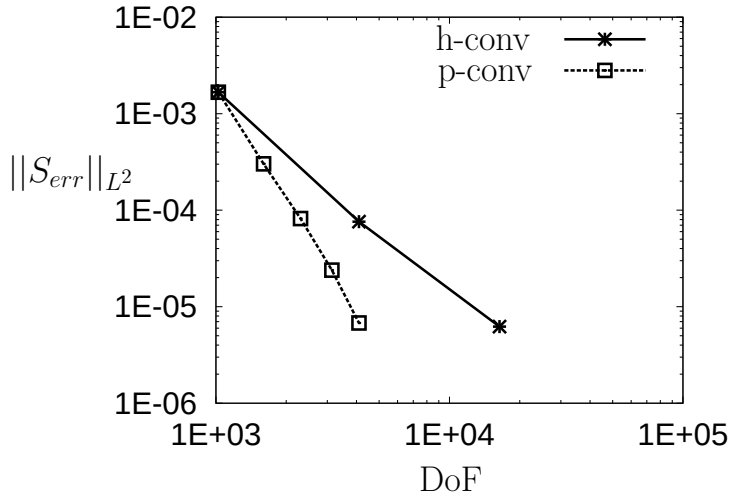


Figure 3.7: Convergence (h- and p-type) plot for inviscid subsonic cylinder. The h-type convergence is shown on a sequence of three successively refined meshes using a relatively low polynomial order of $N = 3$, while the p-type convergence is demonstrated on the coarsest mesh. For this smooth problem, convergence is achieved in a more efficient manner by varying the polynomial order than refining the physical mesh.

midline as compared to the analytical solution given by the validation archive [56]. Overall, the current results show good agreement with the analytical solution.

Although NEK5000 has many desirable traits for a competitive CFD code, its true advantage is its extreme scalability. Thus, an important goal for this work is to maintain the inherent parallel performance of the base NEK5000 code. Figure 3.10 shows speedup as a function of processor count for the converging-diverging nozzle case at $N = 7$. For this case, optimal performance is seen up to 64 processors, translating to around 2 elements per processor or 1000 points per processor. At one element per processor, we still see a performance increase, although the parallel efficiency (a ratio between actual scaling and ideal scaling) drops from over 90% at 64 processors to under 70% at 120 processors.

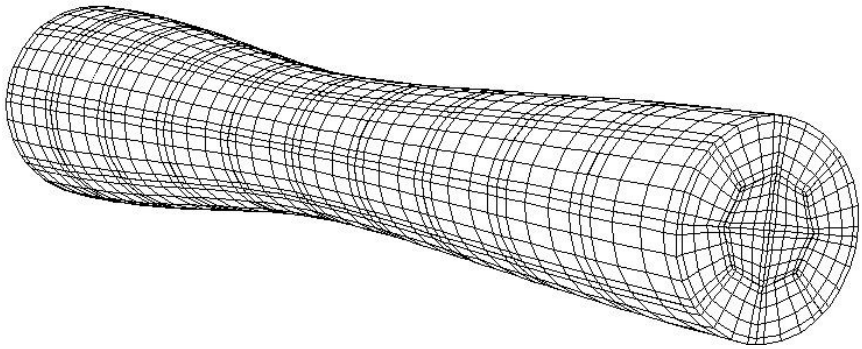
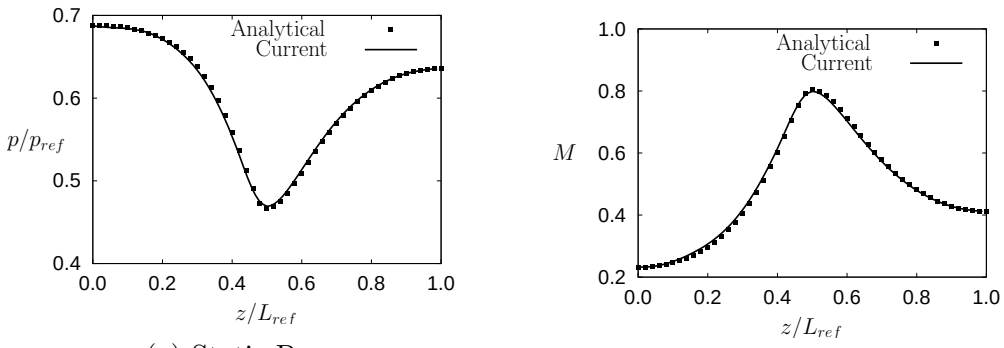


Figure 3.8: Mesh for the converging-diverging verification nozzle test case. The cross sectional area is meshed with 12 elements and is extruded in the axial direction with 10 evenly spaced elements. The mesh is shown with the GLL points at polynomial order $N = 5$.



(a) Static Pressure

Figure 3.9: Variation of static pressure and Mach number along axial midline for nozzle. The results shows good agreement to the analytical solution from [56].

3.2 Navier-Stokes Cases

Simulation of turbulent, compressible flows is the true goal of this work. With the inviscid portion of the code validated, the viscous terms are now tested.

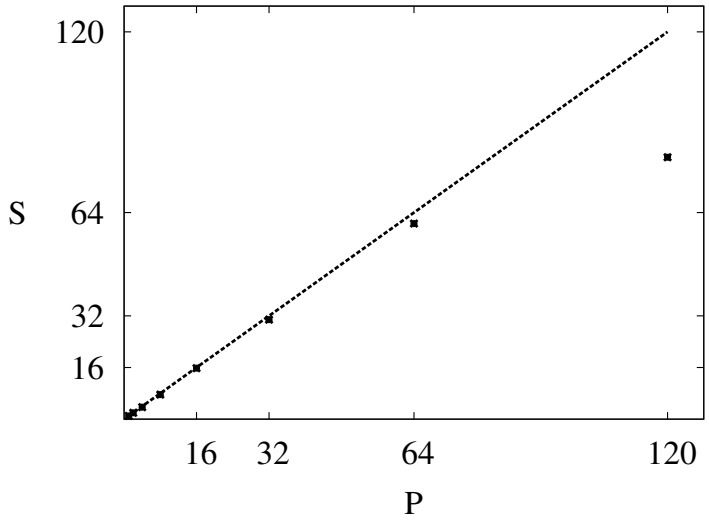


Figure 3.10: Strong scaling plot for converging-diverging nozzle at $N = 7$. The algorithm maintains greater than 90% parallel efficiency up to 64 processors (≈ 2 elements per processor, ≈ 1000 points per processor).

3.2.1 Manufactured Solution

A common method to test particular aspects of a code is through the use of manufactured solutions. The main idea here is to derive forcing terms that will force the equations into a pre-determined solution. As an example, assume that a code is developed to numerically solve the simple linear wave equation

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = 0. \tag{3.4}$$

A test solution is chosen arbitrarily to be $v = x + 2t$. Inserting this into Equation 3.4 does not result in the right hand side being zero. Instead, the new equation is

$$\frac{\partial v}{\partial t} + \frac{\partial v}{\partial x} = 3. \tag{3.5}$$

Thus, in order to test correctness of the code, one solves Equation 3.5, applying the correct boundary conditions, and tests whether the chosen solution is obtained. While this choice was arbitrary, the test solution can be constructed in order to test particular aspects of the code.

The manufactured solution chosen to test the implementation of the 3D Navier-Stokes solver is $Q = [r, r, r, r, r^2]^T$, with $r = c_1 \sin(c_2(x + y + z) - c_3 t) + 2$. For brevity, the resulting forcing terms are not shown here; they can be found in [44]. The chosen parameters are $c_1 = 0.1$, $c_2 = \pi$, and $c_3 = 2$, and the domain used is the box $[-1, 1]^3$ with periodic boundary conditions. The reference Mach number and Reynolds number are 1.0 and 1000, respectively. The exact solution at $t = 0$ is used as the initial condition, and the simulation is run until $t = 1$. The accuracy of the simulation is measured by comparing the numerical solution to the exact solution in the L_2 norm. An example solution for ρ at $t = 0.5$ is shown in Figure 3.11.

Both h-type and p-type convergence runs are performed. For h-type convergence, a sequence of evenly spaced meshes are used. Starting with three elements in each direction, each subsequent simulation increases the number of elements in each direction by one until the number of elements in each direction is ten. The initial $3 \times 3 \times 3$ mesh is chosen for p-type convergence tests, and simulations are run from $N = 3$ to $N = 11$. Convergence to the correct solution is shown for both runs in Figure 3.12. Comparing the finest mesh used for the h-convergence run to the highest polynomial used for the p-convergence run, we see a difference of accuracy of nearly two orders of magnitude. The large boost in accuracy is not the only benefit of using p-refinement. Both runs were performed using 8 cores to test the performance differences between the two. The run using the finest mesh with a moderate polynomial order needed 100 seconds to complete, consuming approximately 0.2 core-hours, whereas the run using the highest polynomial on a coarse mesh used up only 0.04 core-hours, taking

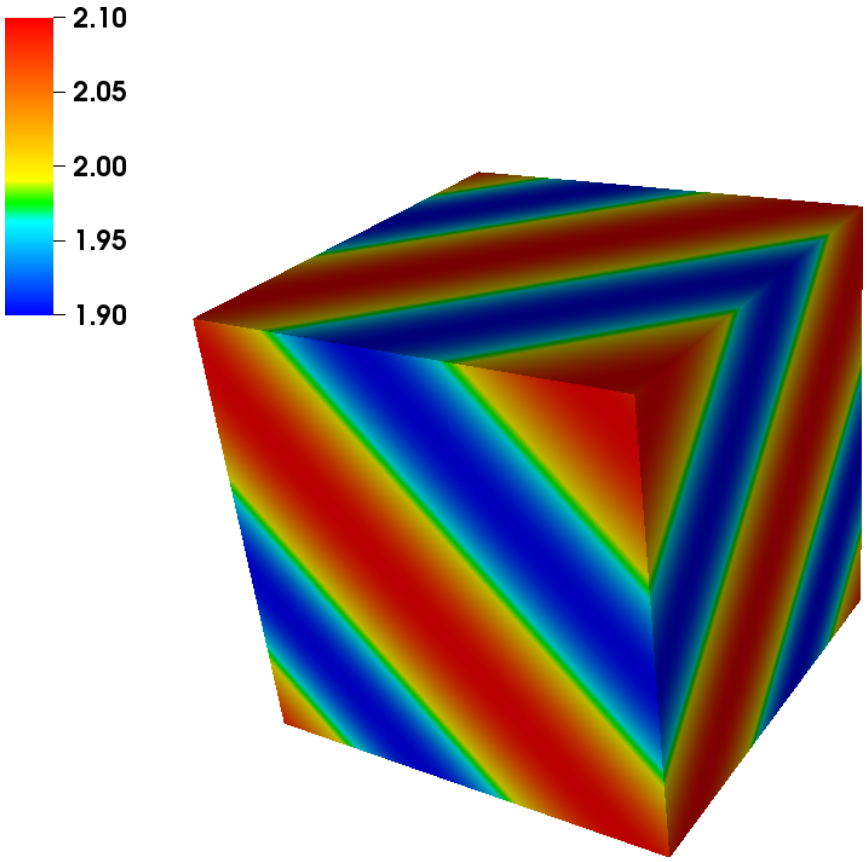


Figure 3.11: Density for manufactured solution case at $t = 0.5$. This solution is taken from the $E = 27$, $N = 11$ case, with an L_2 error of 3×10^{-9} as compared to the exact solution.

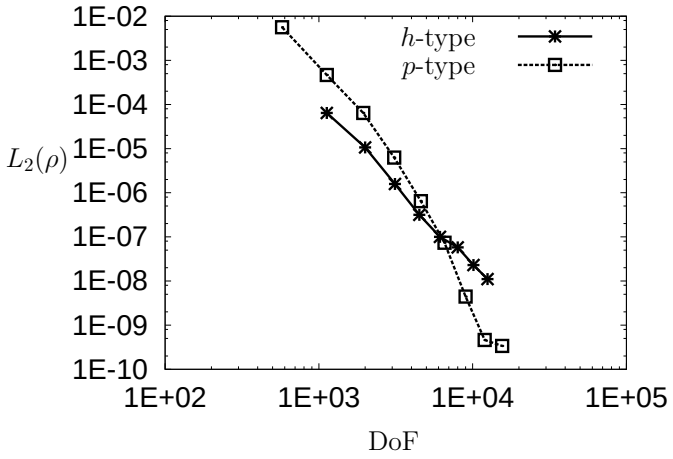


Figure 3.12: Convergence plot for manufactured solution case. Convergence is demonstrated by either refining the mesh (h-type, solid line) or increasing the polynomial order (p-type, dashed line).

approximately 16 seconds to complete. Thus, not only does fixing the mesh size and increasing the polynomial order result in higher convergence rates, it also performs better in the context of spectral element methods.

The ability to obtain accurate results by either refining the mesh or increasing the polynomial order is an important attribute of the code. Although, as shown here, p-refinement typically results in faster convergence rates for smooth problems, real Navier-Stokes simulations will most often not be infinitely smooth. In these cases, mesh refinement near areas with large solution gradients is most likely the better course of action.

3.2.2 Viscous Cylinder

A classic test case is viscous flow over a cylinder. As the Reynolds number of the flow changes (based on the freestream properties and the cylinder diameter), the flow undergoes several transistions. For extremely low Reynolds numbers, Stokes flow is seen, but above $\mathrm{Re} = 6$ or so, the flow separates on the dowstream end of the

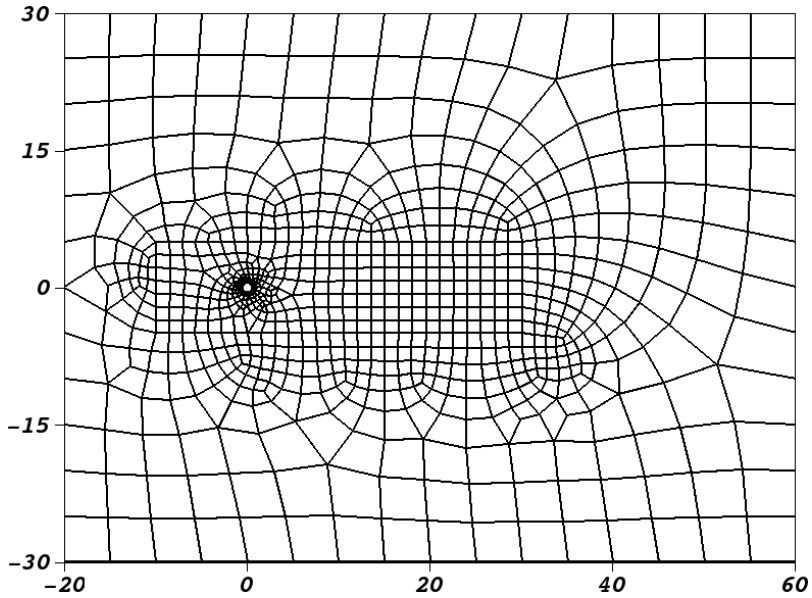


Figure 3.13: Unstructured mesh for viscous cylinder cases. The mesh contains 883 elements, with elements clustered near the cylinder and in the wake to properly capture the physics in these regions.

cylinder, forming two attached eddies. Above Re = 40, an instability in the wake causes the vortices to detach in an alternating fashion in what is commonly known as the von Kármán vortex street. Beyond Re = 150, the vortex street becomes turbulent and the flow can no longer be considered two dimensional.

However, these trends are for incompressible flow. This next study examines the flow over a cylinder at Re = 100 for a subsonic , transonic, and supersonic flow (Ma = 0.2, 0.7, and 1.0, respectively). The subsonic and transonic cases are simulated in the domain $x \in [-20D, 60D], y \in [-30D, 30D]$ with the mesh shown in Figure 3.13. The supersonic case uses a mesh with the width extended to $y \in [-60D, 60D]$.

The subsonic case should compare well to incompressible data, as the Mach number is low enough for compressibility effects to be neglible. As discussed at the beginning, incompressible flow at this Reynolds number exhibits an instability in the wake, causing a vortex street. The frequency of the vortex street is computed by ex-

amining the oscillations of the lift coefficient. The Strouhal number ($\mathrm{St} = fD/U_{\mathrm{ref}}$), of this flow is measured to be 0.165, and the computed mean drag coefficient, defined as $C_d = 2F_d/(\rho U_{\mathrm{ref}}^2 A)$, is 1.366. Both values are in good agreement with previous simulations and experimental data [57], which reported the Strouhal number as 0.165 and the drag coefficient as 1.375.

In the transonic case, the drag coefficient increases quite dramatically to 1.82. The reason for this is shown in Figure 3.14; the pressure drag (or form drag) increases by almost 50%. This is due to the increased speed of the flow. The higher speed flow will experience a relatively larger pressure increase as it stagnates on the front of the cylinder, leading to an overall larger differential pressure from the front to the back of the cylinder. On the other hand, the viscous portion of the drag is approximately the same for each case. The Strouhal number is slightly less for this case than the $\mathrm{Ma} = 0.2$ case, measured at 0.158. Both the drag coefficient and the Strouhal number compare well to previously published compressible simulation data [57].

In the sonic case, the vortex street disappears entirely, and is instead replaced by a rather large recirculation zone, causing the drag coefficient to climb to 2.1. It is theorized that the stabilizing stratification term, $\nabla\rho \times \nabla p$, is sufficiently large at $\mathrm{Ma} = 1.0$ to squash the viscous instabilities, resulting in a steady flow. Contours of velocity magnitude are shown in Figure 3.15, showing the unsteady nature of the first two cases and the long recirculation zone of the sonic case.

3.2.3 Weakly Turbulent Sphere

To test the code's ability to correctly simulate bulk 3D fluid motions, the next validation case chosen is subsonic ($\mathrm{Ma} = 0.3$) flow over a sphere at a moderate Reynolds number, $\mathrm{Re} = 1000$, based on the diameter of the sphere, D, and the free stream flow properties.

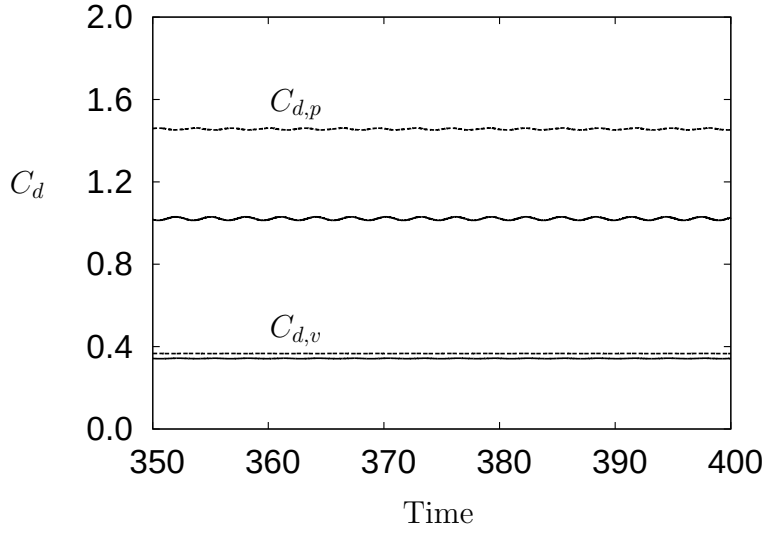
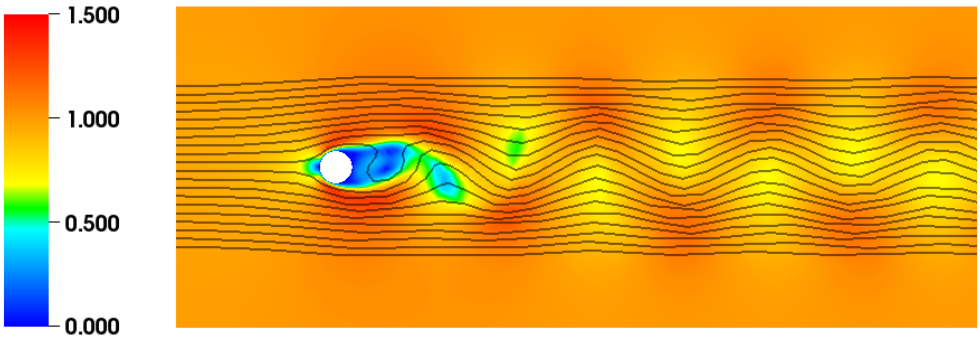


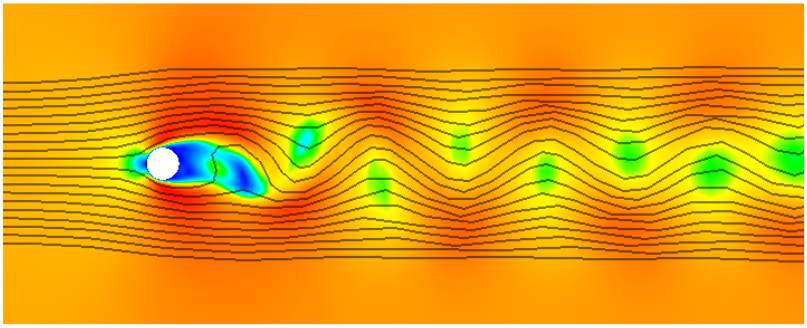
Figure 3.14: Comparison of drag coefficients for subsonic and transonic viscous cylinder. The Ma = 0.7 case is shown with the dashed line, and the solid line represents the Ma = 0.2 case. While there are no major increases in the viscous drag, the pressure drag increases by approximately 50% at Ma = 0.7 as compared to Ma = 0.2.

The domain extends $25D$ downstream of the sphere, and $4.5D$ upstream and circumferentially. To discretize the domain, a block structured mesh of 17,920 elements is used with a polynomial order of $N = 5$, resulting in 3.87 million grid points. A slice of the domain at $z = 0$ is shown in Figure 3.16, showing the block structured nature of the mesh. A closeup of the mesh near the sphere is given in Figure 3.17, showing the details of the surface mesh and the boundary layer refinement. It should be noted that although the mesh is block-structured, it is still used as a general unstructured mesh in the simulation.

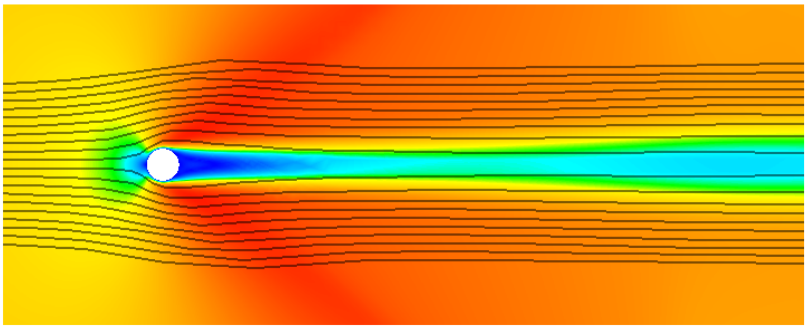
Far field boundary conditions are set for the boundaries away from the sphere, and isothermal no-slip walls are used at the sphere surface. The initial condition is set to the freestream properties. The simulation is first run for 300 non-dimensional time units to ensure a statistically steady state is reached. Then the results are



(a) Re = 100, Ma = 0.2



(b) Re = 100, Ma = 0.7



(c) Re = 100, Ma = 1.0

Figure 3.15: Contour of velocity magnitude with streamlines for viscous cylinder cases. The subsonic and transonic cases feature a vortex street, while the increased speed in the supersonic case prevents the wake instability from growing.

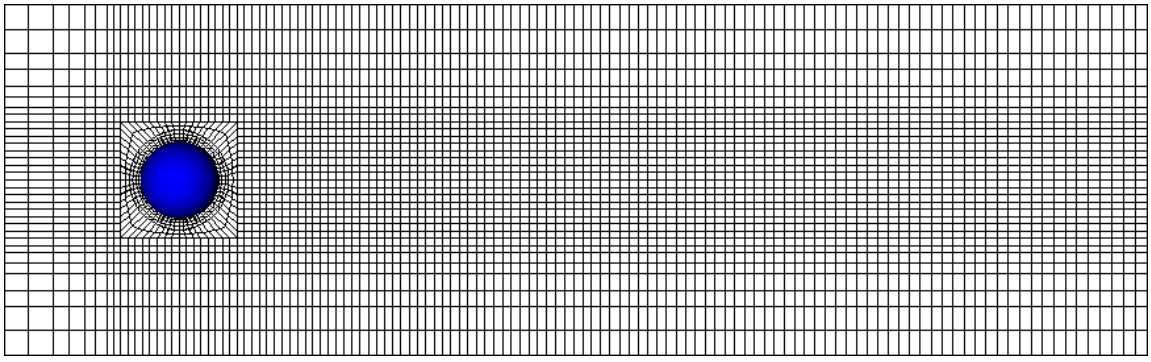


Figure 3.16: Mesh used for simulation of sphere at Re = 1000. The mesh contains 17,920 elements, and polynomial order $N = 5$ is used (GLL points not shown for clarity). Pictured is a slice at $z = 0$, showing the semi-structured nature of the mesh.

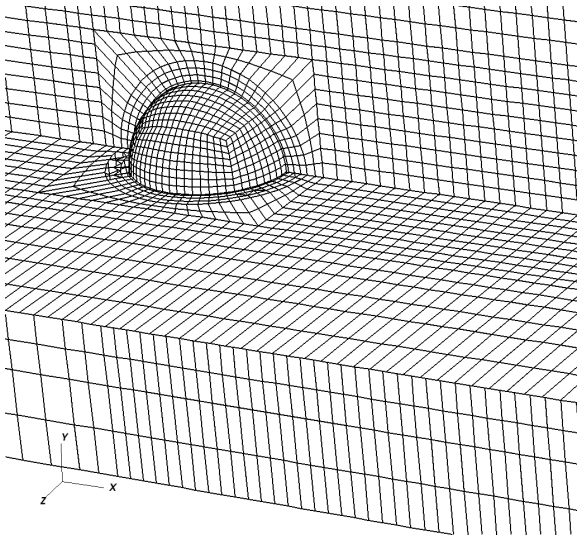


Figure 3.17: Details of surface mesh for subsonic sphere simulation. The discretization employs 1536 surface elements, and elements are clustered near the sphere surface to properly capture the boundary layer.

time-averaged for another 100 time units.

The computed drag coefficient is 0.48, which is in good agreement with previous DGSEM results [47] and with incompressible empirical correlations. There are two Strouhal numbers reported in [58]; the main frequency of 0.195 and a secondary frequency of 0.35. The current results give frequencies of 0.19 and 0.38, although the signal is quite noisy and thus it was difficult to define clear dominating frequencies.

At this Reynolds number, small scale structures appear due to an instability in the shear layer, rendering the wake turbulent [58]. One method of identifying vortical structures indicative of turbulent flow is the λ_2 criterion given by Jeong and Hussan [59]. Essentially, λ_2 is the second eigenvalue of the matrix $S_{ik}S_{kj} + \Omega_{ik}\Omega_{kj}$, where Ω_{ij} is the antisymmetric portion of the velocity strain rate tensor defined as

$$\Omega_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} - \frac{\partial u_j}{\partial x_i}\right). \tag{3.6}$$

The symmetric portion S_{ij} is as defined before in Section 2. Isosurfaces of λ_2, colored by Mach number, are shown in Figure 3.18. Small scale structures show up in the wake of the sphere, and further downstream, vortical structures with a size on the order of the sphere are seen.

The current results are compared to published incompressible DNS results [58] and experiment data [60] down the streamwise axis in Figure 3.19. The current simulation shows good agreement with the DNS and experimental results except in the region very close to the sphere, where the current results undershoot the previous data. The rms component of the current simulation overshoots the incompressible DNS and experiment, but the slope of the curve is similar.

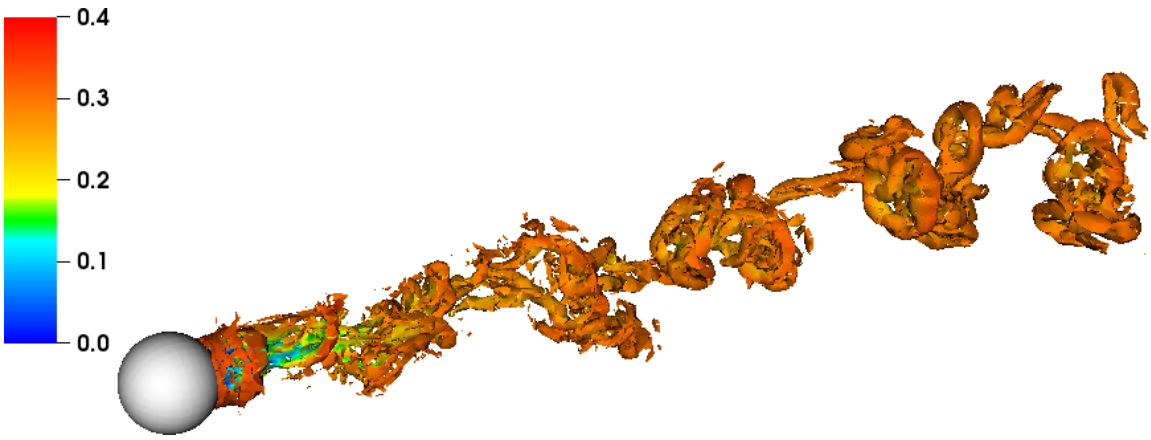
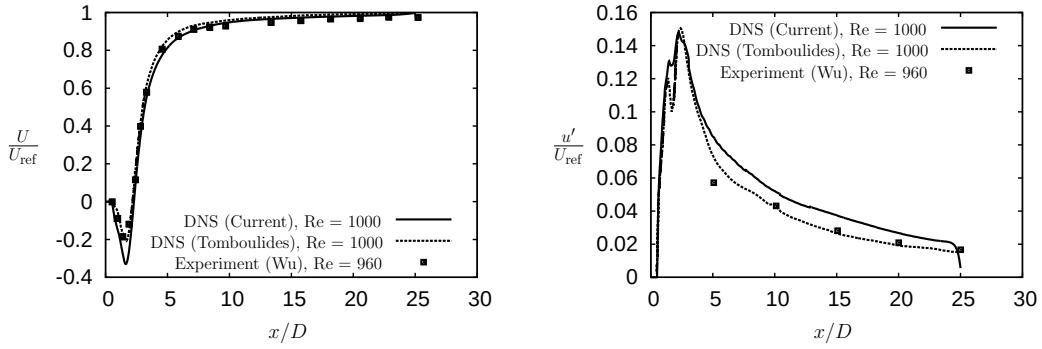


Figure 3.18: Small scale structures in subsonic sphere flow at Re = 1000. Isosurfaces of $\lambda_2 = -0.001$ are shown, colored by Mach number.



(a) Mean streamwise velocity

(b) Streamwise rms velocity

Figure 3.19: Mean and rms streamwise velocity for sphere simulation. The mean component agrees well with previous incompressible DNS [58] and experimental [60] data. The rms velocity is overpredicted downstream of the sphere, but the slope is similar.

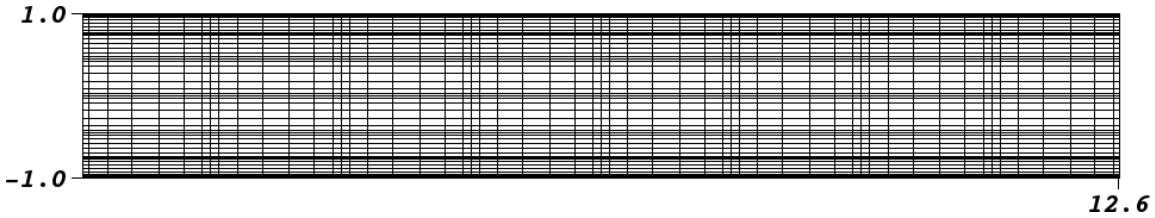
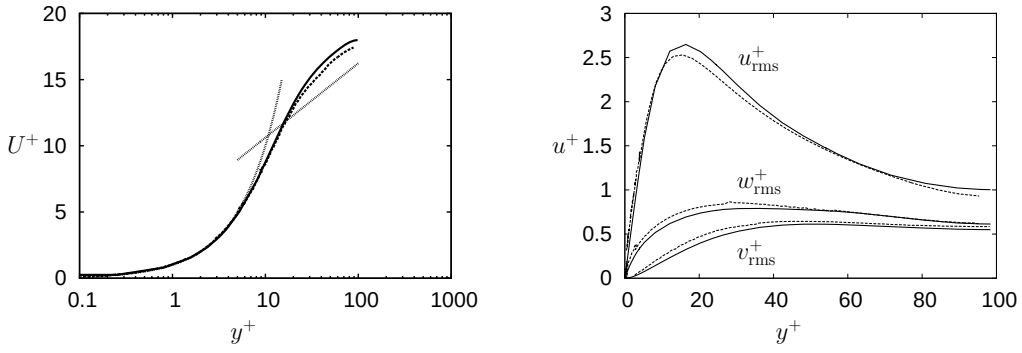


Figure 3.20: Mesh for turbulent channel case. Eight elements are used in each cardinal direction, resulting in 512 elements. The element spacing in the wall normal direction varies from 0.04δ at the wall to 0.45δ at the interior. The GLL points for $N = 7$ are also shown in the figure.

3.2.4 Turbulent Channel

The final validation test is a DNS study of turbulent channel flow. This is a classical flow case, both experimentally and numerically, and thus there is a wealth of information to compare to. The domain in question has the dimensions $[L_x, L_y, L_z] = [4\pi, 2, 4\pi/3]$. The Reynolds number based on the mean shear velocity is $\mathrm{Re}_\tau = \rho u_\tau \delta/\mu = 100$, where $u_\tau = \sqrt{\tau_w/\rho}$ is the friction velocity and δ is the channel half height. The Mach number at the centerline is kept to approximately 0.3 in order to facilitate comparison to incompressible DNS results reported in [61].

The mesh used for the simulation is shown in Figure 3.20. The domain is meshed using 8 evenly spaced elements in the both the x and z directions, and a stretched mesh is used in the y direction. The polynomial order is set to $N = 7$. In terms of wall units ($\Delta^+ = \rho u_\tau \Delta/\mu$), the minimum wall-normal spacing used is $\Delta y^+ \approx 0.5$, and the spacing in the flow and spanwise directions are $\Delta x^+ \approx 22$ and $\Delta z^+ \approx 7.5$. As such, the simulation may be slightly under-resolved in the flow direction for the purposes of a DNS, although this does not appear to have an adverse effect on the results. The solution is initialized using a turbulent mean profile with random perturbations, and the simulation is run until the flow reaches a statistically steady state.



(a) U^+ (b) $\mathbf{u}_{\mathrm{rms}}$

Figure 3.21: Mean and rms velocity profiles for turbulent channel DNS. The current DGSEM results (solid lines) are in good agreement with incompressible DNS results (dashed lines) reported in [61] The theoretical law of the wall also shown in (a) for reference using dotted lines.

The results of the simulation are shown in Figure 3.21 as solid lines, with the data from [61] shown with dashed lines. Figure 3.21a shows the mean velocity profile, normalized by the shear velocity as $U^+ = U/u_\tau$. For reference, the theoretical "law of the wall" profile is shown with dotted lines, which is

$$U^+ = y+ \quad y^+ < 5, \tag{3.7}$$

$$U^+ = 2.44 \ln(y^+) + 5.1 \quad 30 < y^+ < 150. \tag{3.8}$$

Figure 3.21b shows the root-mean-square (rms) turbulent quantities, defined as $u_{\mathrm{rms},i} = \sqrt{\langle u_i' \rangle^2}$, with u_i' the deviation from the mean velocity U_i. Good agreement is seen for both the mean velocity profile and the rms quantities with the previous DNS results [61].

4. FILM COOLING SIMULATION

4.1 Background on Film Cooling

One of the predominant design problems in the gas turbine community is the task of increasing the efficiency of gas turbine engines while also ensuring safety and reliability factors are met. A common route towards increasing gas turbine performance is by increasing the inlet temperature into the turbine section (usually by burning more fuel). However, turbine inlet temperatures are already far beyond the melting temperature of the materials used in the turbine [62]. In order to prevent melting, relatively cooler air is extracted from the compressor section and is used to cool the turbine parts, such as the endwall and the blades themselves. This extraction results in an overall loss of efficiency, so the amount of cooling air and how it is used must be optimized.

Turbine blades are cooled through many mechanisms, often simultaneously. The coolant air enters the blade through where it is attached to the endwall. The air flows through a serpentine-type passage, cooling the metal from the interior. Ribs or pins are often used to increase the internal heat transfer. To protect the external metal surface, the coolant air is diverted through tiny holes that are drilled from the blade surface to the internal hollow passages of the blade. The objective of these holes is to eject a layer of cooling air over the blade surface, acting as a buffer between the extreme temperatures of the air exiting the combustor and the vulnerable materials of the blade. This buffer is termed "film cooling," and its optimization has been a major subject of research in gas turbine.

The performance of film cooling is most commonly determined, at least in labo-

ratory experiments, by the film cooling effectiveness, defined as

$$\eta = \frac{T_\infty - T_{aw}}{T_\infty - T_c}, \tag{4.1}$$

where T_∞ is the freestream temperature, T_{aw} is the adiabatic wall temperature, and T_c is the temperature of the coolant at the exit of the hole. Film cooling performance is affected by a variety of parameters, such as freestream turbulence and the hole geometry used [63]. The density ratio and blowing ratio, defined as

$$DR = \frac{\rho_c}{\rho_\infty}, \tag{4.2}$$

$$BR = \frac{\rho_c U_c}{\rho_\infty U_\infty}, \tag{4.3}$$

are two important parameters in predicting film cooling effectiveness. As noted by Bogard [63], an increase of BR gives an initial increase in η, but if this is increased too high, the coolant will instead jet from the hole and into the freestream, negating its effectiveness. For a given BR, a decrease in DR increases the velocity ratio, and therefore the momentum ratio. However, the density ratio appears to have less of an effect on η overall than the blowing ratio.

The accurate simulation of film cooling is a difficult task. Even for one hole, there is a large difference between length scales associated with the cooling hole and the blade, making direct simulation impractical. RANS models are able to capture time mean large scale structures, but they often struggle to predict the spreading and diffusion of the coolant [64]. Large eddy simulation has been the next step towards the accurate simulation of film cooling at reasonable computational cost, and has been used in recent studies that saw results superior to typical RANS simulations [65, 66].

4.2 Overview of Large Eddy Simulation

The goal behind LES is the same as RANS; in order to reduce the prohibitive computational costs of DNS, we seek to instead only resolve a portion of the scales, and model the rest. In RANS, this means applying a time average to the Navier-Stokes equations, and thus computing only the time mean variables (and modeling all time fluctuations). In LES, a spatial filtering approach is used instead. The flow variables are decomposed into a filtered quantity and an unresolved residual. Thus, for density, we have $\rho = \overline{\rho} + \rho'$, where the overbar denotes the filtering operation. A similar decomposition is performed for pressure. For compressible flows, it is convenient to consider the density weighted, or Favre-filtered, velocities and temperatures:

$$\widetilde{\mathbf{u}} = \frac{\overline{\rho \mathbf{u}}}{\overline{\rho}} \tag{4.4}$$

$$\widetilde{T} = \frac{\overline{\rho T}}{\overline{\rho}}. \tag{4.5}$$

For brevity, we will only consider the momentum equations. After applying the filter to the momentum equations, the result is

$$\frac{\partial \left(\overline{\rho}\widetilde{u}_i\right)}{\partial t} + \frac{\partial}{\partial x_j}\left(\widetilde{u}_i \overline{\rho}\widetilde{u}_j\right) + \frac{\partial \overline{p}}{\partial x_i} = \frac{\partial \overline{\tau}_{ij}}{\partial x_j} - \frac{\partial \overline{\tau}_{ij}^{SGS}}{\partial x_j}, \tag{4.6}$$

with $\overline{\tau}_{ij}^{SGS} = \overline{\rho}\left(\widetilde{u_i u_j} - \widetilde{u}_i \widetilde{u}_j\right)$ known as the sub-grid scale (SGS) stress tensor [67]. While the resolved filtered stress tensor, $\overline{\tau}_{ij}$, has its own issues, the focus is on the SGS tensor. The first thing to point out is that, while it has the units of stress, the SGS stress tensor is not actually a stress. It is merely an artifact from the filtering operation performed on the equations; a similar term appears in RANS. For both LES and RANS, this residual term is unknown, and thus the SGS tensor must be

modeled.

The earliest developments of LES employed the Smagorinksy model. In the Smagorinsky model, the SGS stress tensor is related to the resolved strain rate tensor as

$$\overline{\tau}_{ij}^{SGS} - \frac{1}{3}\overline{\tau}_{kk}^{SGS}\delta_{ij} = -2\left(C_s\Delta\right)^2 |S|\,\overline{S}_{ij}, \tag{4.7}$$

where Δ is the filter width, C_S is the Smagorinksy coefficient, and $|S| = \sqrt{2\overline{S}_{ij}\overline{S}_{ij}}$ [4]. Various improvements have been made to this model in continued development of LES, such as dynamic Smagorinsky [68] and the high-pass filtered eddy-viscosity model [69]. The desired effect for any LES model is to add diffusion in areas where small scale structures are not resolved directly and to do nothing in areas where the resolution is adequate to DNS standards. Thus, a much simpler model, made practical through high order schemes such as spectral elements, is to simply apply a spectral filter to the resolved flow variables. In regions that are well resolved, the filter will have little to no effect, as the high frequency spectral coefficients will have low weights. However, in marginally-resolved or under resolved regions, the filter emulates the effects that an additional diffusion term in the equations would have. The benefit of the model is that no additional terms for the Navier-Stokes equations need be computed, saving on computational cost. At the moment, the model still needs to put through rigorous validation tests. However, the model has seen promising results in previous studies, such as in [70]. The simplified model is used in this work for the simulation of film cooling.

4.3 Problem Setup

The problem at hand is the large-eddy simulation of a single film cooling hole on a flat surface. The pertinent details of the geometry are given in Figure 4.1. Although the figure shows the dimensions in millimeters, the simulation is performed

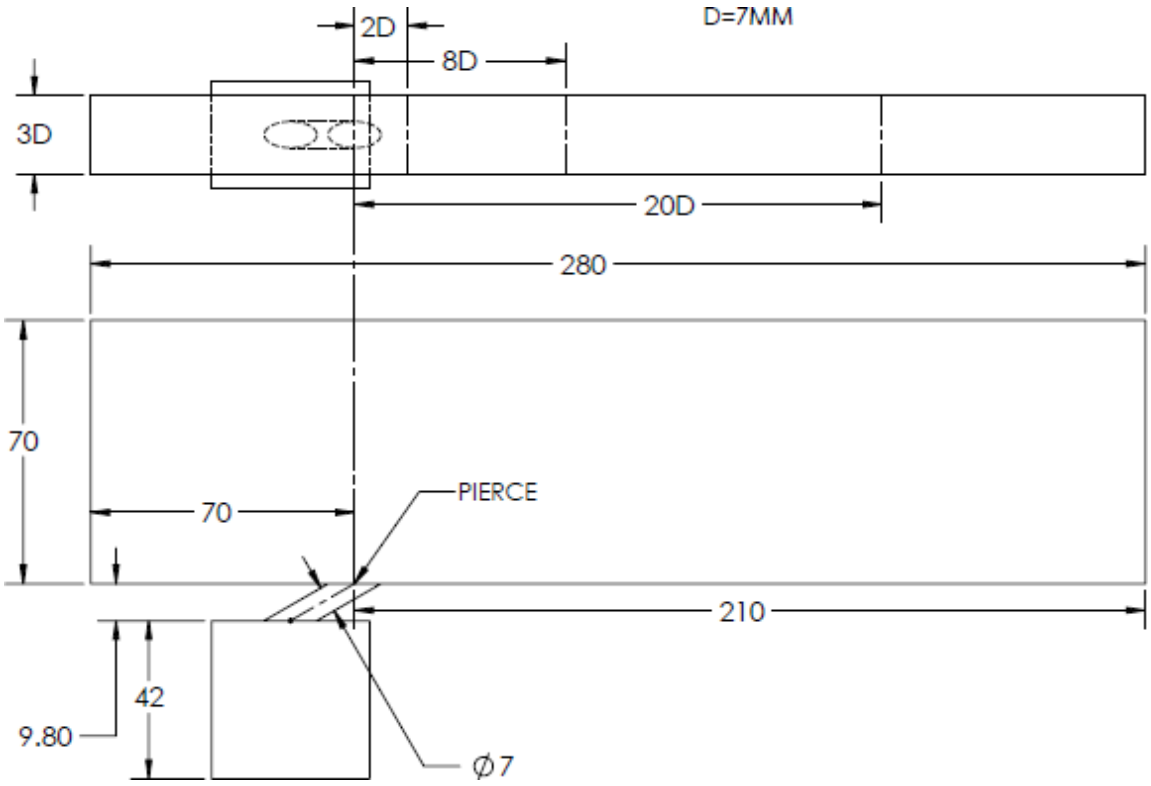


Figure 4.1: Details of geometry used for film cooling simulation. A simple, cylindrical film cooling hole of diameter $D = 7$ mm is used. The domain extends $10D$ upstream and $30D$ downstream of the hole, and domain is truncated vertically at $10D$. The simulated pitch is $3D$.

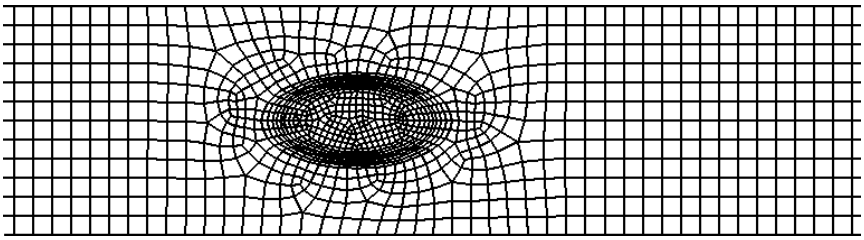


Figure 4.2: Closeup of mesh near film cooling hole. In total, 85,048 elements are used at a polynomial order of $N = 5$, resulting in 18.3 mio grid point.

on a geometry non-dimensionalized by the diameter of the film cooling hole. The blowing ratio and density ratio are $BR = 1.0$ and $DR = 1.5$, respectively. The inlet Mach number is 0.3, and the Reynolds number based on the hole diameter and free stream properties is $Re_D = 32,000$. A closeup of the mesh near the film cooling hole is shown in Figure 4.2. The mesh contains $E = 85,048$ elements at $N = 5$, resulting in 18.3 million grid points.

A key difficulty with high-fidelity CFD models, such as LES or DNS, is the accurate imposition of turbulent inflow boundary conditions. In RANS, prescribing a turbulent inflow is more or less accomplished by inputting various mean parameters, such as turbulence intensity. Because LES and DNS solve the time dependent equations, the inflow must also be time dependent. One method of generating realistic inflow conditions is to extend the inflow boundary far enough upstream to allow for natural development of the turbulent boundary layer. However, this method is prohibitively expensive in the context of LES/DNS, where the resolution requirements are very high. Another method is to artificially prescribe a turbulent inflow. However, this is not as simple as randomizing the velocity inputs, as turbulent flows are not truly random. Many sophisticated methods have been proposed to fulfill the task of turbulent inflow generation; for the purposes of this work, the recycling and rescaling method (RRM) proposed by Urbin and Knight for compressible flows is

used [71]. The essentials of the RRM method are to take the velocity and temperature profile at some point downstream from the inlet, modify them according to theoretical wall scaling laws, and reintroduce them at the inlet. A realistic turbulent boundary layer profile develops over time, without the need for a large inlet flow length. For brevity, the details of the method are not given here; further details can be found in the original paper [71].

Previously, a large-eddy simulation of this geometry was performed using the low-Mach formulation in NEK5000 (Duggleby *et al*, [64]). The pressure is split into a thermodynamic and hydrodynamic component, $p = p_{th} + p_h$. In the low-Mach formulation, the density is allowed to change due to temperature changes or transient thermodynamic pressure changes, but is independent of hydrodynamic pressure. Allowing for density changes with respect to temperature, the normal divergence free condition on incompressible flows is instead replaced with

$$\frac{\partial u_j}{\partial x_j} = \frac{1}{T}\frac{DT}{Dt}, \tag{4.8}$$

where $D(\cdot)/Dt = \partial(\cdot)/\partial t + u_j \partial(\cdot)/\partial t$ is the material derivative. The divergence term acts as a source in the pressure Poisson equation that arises in incompressible simulations [64].

The low-Mach formulation is technically only supposed to be used for Mach numbers below 0.3; beyond this, the variation of density with respect to hydrodynamic pressure changes begins to become too significant to ignore. Unfortunately, in the low-Mach simulation, the Mach number in hole exceeds this limitation, as shown in Figure 4.3. Because the low-Mach formulation does not account for compression due to hydrodynamic forces, the density remains constant in the hole, causing the velocity to have an unphysically high values and detach from the plate. The con-

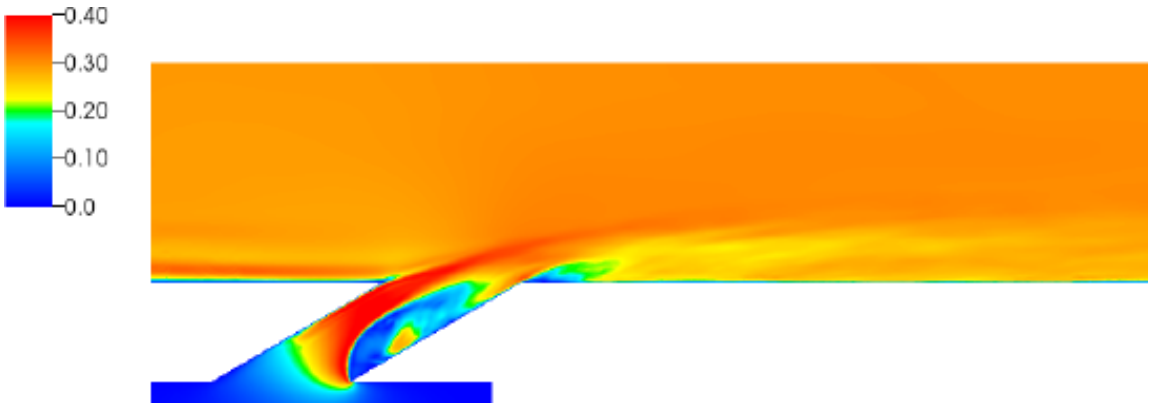


Figure 4.3: Average Mach number in cooling hole from low-Mach simulation in Duggleby *et al* [64]. The Mach number exceeds 0.4 in the hole in violation of the low-Mach assumption. The lack of real compressibility in the low-Mach formulation cause the coolant to accelerate too much, leading to the coolant lifting off the surface of the flat plate.

sequences of this are shown in Figure 4.4. The high velocity in the hole causes the coolant to jet from the hole, where it is quickly mixed into the turbulent freestream instead of laying on the surface. This leads to largely different film cooling behavior as compared to experimental results [72], resulting in a false prediction of poor film cooling performance. This behavior was observed even with when the mesh was refined in and around to the hole to DNS quality, suggesting that a fully compressible formulation is needed to accurately simulate this case.

4.4 Current Results

The current compressible LES simulation was run for 280 time units, with the time averaging taking place over the last 40 units. The total computational cost was 35,000 CPU hours, spread out over 256 processors.

In the experimental rig described in Aga [72], the inlet momentum thickness and displacement thickness are $\delta_2 = 0.05$ and $\delta_1 = 0.12$, giving a shape factor of $H = 2.31$. Thus, the velocity profile is not fully developed at the inlet in the experiment; to trip

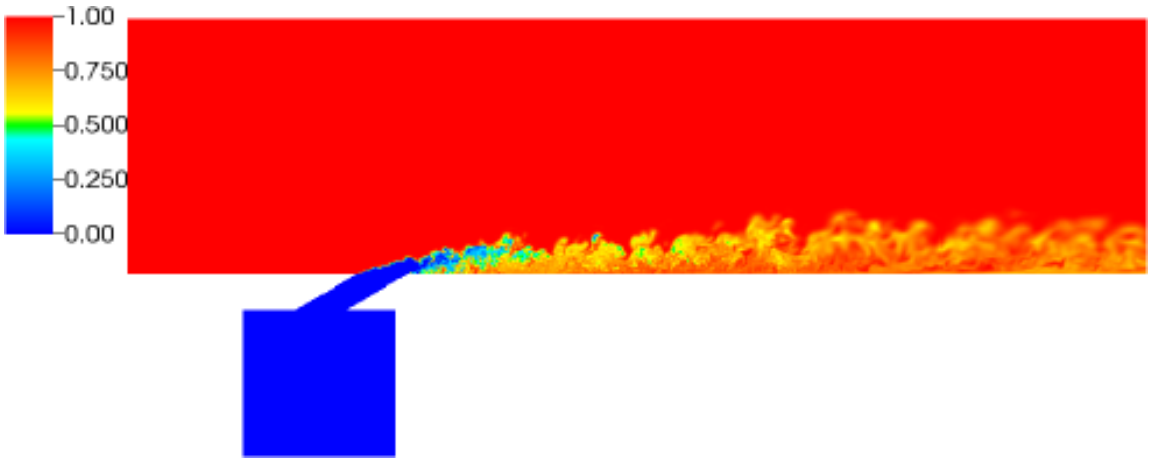


Figure 4.4: Instantaneous contour of non-dimensional temperature in low-Mach simulation in Duggleby *et al* [64]. Because the Mach number in the hole exceeds the low-Mach limit of 0.3, the velocity in the hole is too high. This leads to the coolant being ejected into the freestream, where it is quickly mixed and dissipated due to turbulence. The end result is that the film cooling effectiveness is grossly underpredicted.

the boundary layer, the cooling insert was pushed up slightly above the flat plate. The recycling procedure in the simulation produces $\delta_2 = 0.17$ and $\delta_1 = 0.22$, resulting in a shape factor of $H = 1.35$. Thus, while the momentum and displace thicknesses are not matched to the experiment, the inlet is a fully developed turbulent profile. The profile in wall units is given in Figure 4.5, along with the theoretical viscous sublayer profile showing good agreement in the sublayer.

Figure 4.6 shows contours of instantaneous and time-mean Mach number in the film cooling hole. As compared to what was seen in Figure 4.3 for the low-Mach LES [64], the Mach number in the compressible simulation does not exceed 0.3 (at certain instances in time, the Mach number may reach 0.4-0.5, but this is most likely a numerical issue than a physical one). Based on this plot alone, it is expected that the film cooling effectiveness will be higher than what was predicted in the low-Mach simulation, as the lower speed flow will better allow for the film to lay on the surface.

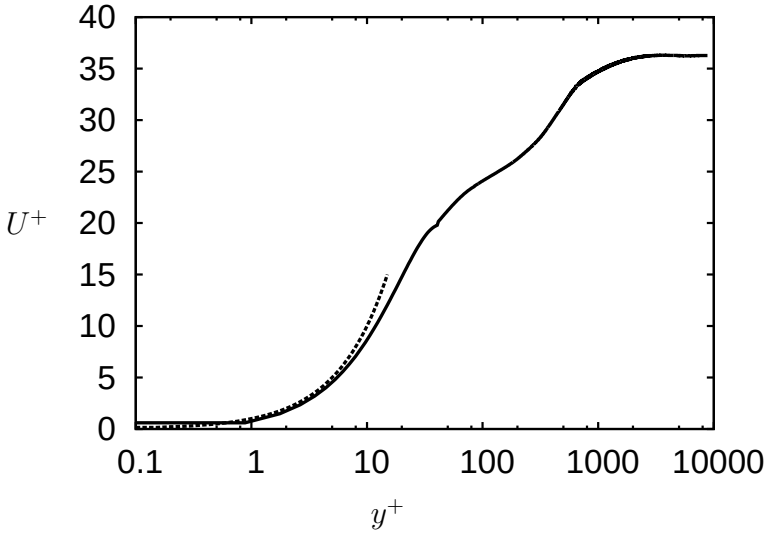
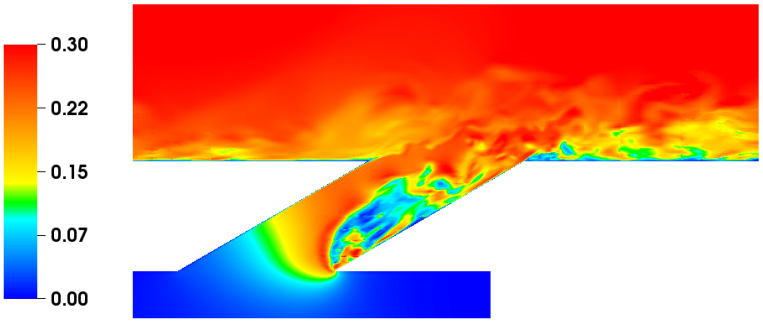


Figure 4.5: Inlet velocity profile for film cooling case. The simulated velocity profile is given by the solid line, and the theoretical viscous sublayer profile is given by the dashed line for reference. The momentum and displacement thicknesses are $\delta_2 = 0.17$ and $\delta_1 = 22$, giving a shape factor of 1.35 that is indicative of a fully developed turbulent boundary layer.

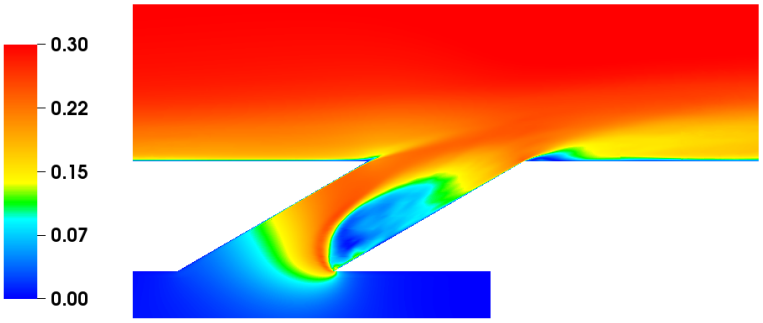
Figures 4.7 through 4.10 show mean temperature $\overline{\theta}$, rms temperature θ_{rms}, and turbulent heat transfer $\overline{v'\theta'}$ for slices at z/D =0, 0.25, 0.5, and 0.75. Here, θ is the non-dimensional temperature,

$$\theta = \frac{T_\infty - T}{T_\infty - T_c}. \tag{4.9}$$

In Figure 4.7 ($z/D = 0$), the contour for $\overline{\theta}$ shows evidence of a slight liftoff at the exit of the cooling hole, although the coolant reattached fairly quickly. Compared to the instantaneous contour from the low-Mach LES shown in Figure 4.4, it appears that the reduced Mach number in the hole indeed leads to less of a jetting effect, as expected. The rms temperature shows strong turbulent fluctuations in the shear layer where the free stream and the upstream side of the coolant meet. Numerically, this can be an issue for high order methods, as this represents a discontinuity. The



(a) Instantaneous



(b) Average

Figure 4.6: Instantaneous and average Mach number in cooling hole for compressible film cooling simulation. Due to the fully compressible nature of the code, the Mach number does not exceed 0.3, unlike the low-Mach results from Duggleby *et al* [64].

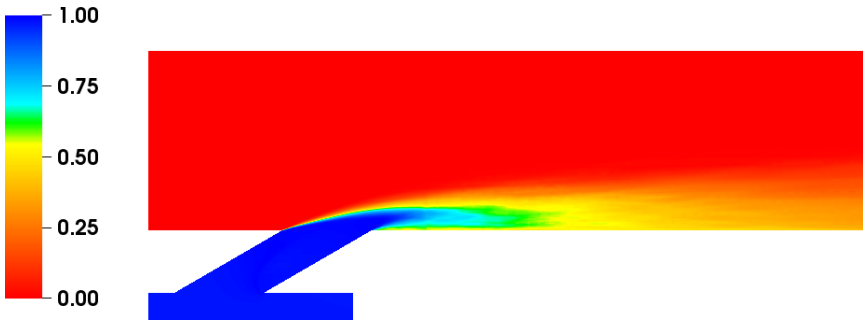
current algorithm, with some dissipation from the spectral filter, proves to handle this contact in a stable manner.

Halfway across the radius of the hole, there is again evidence of liftoff, as shown in Figure 4.8a. In the low-Mach case, this complete separation remains until approximately $1.5D$ downstream of the hole. However, the flow reattaches much closer to the hole in the fully compressible results. The turbulent mixing, as evidenced by θ_{rms} in Figure 4.8b, is spread out over a larger area at this location, due to its proximity to the outer edge of the film hole.

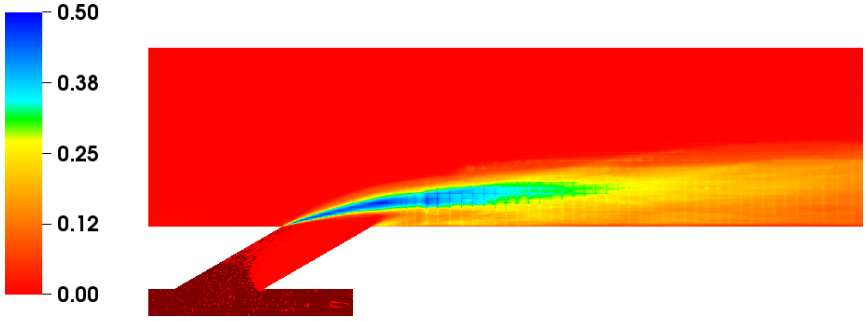
At the outer edge of the hole ($z/D = 0.5$), the coolant is still being spread from the hole, although at a reduced capacity. This contrasts with the previous low-Mach results, where the coolant has already been mixed into the freestream at this location [64]. Although the geometry does not appear in the figure, the turbulent quantities, particularly the turbulent heat transfer, are most pronounced right at the wall at $x/D = 0$. This would result in large amounts of heat transfer into the wall, although its difficult to say whether this is positive or negative, as it depends on the fluid temperature at the wall.

Finally, at a quarter pitch ($z/D = 0.75$), the coolant is mostly absent, although there is still evidence of turbulent fluctuations and heat transfer even at this location (albeit located off the wall). At this location for the low-Mach results, there is no coolant present and almost no evidence of fluctuations, indicating that the turbulent mixing due to the increased liftoff has prevented the coolant from spreading in the spanwise direction.

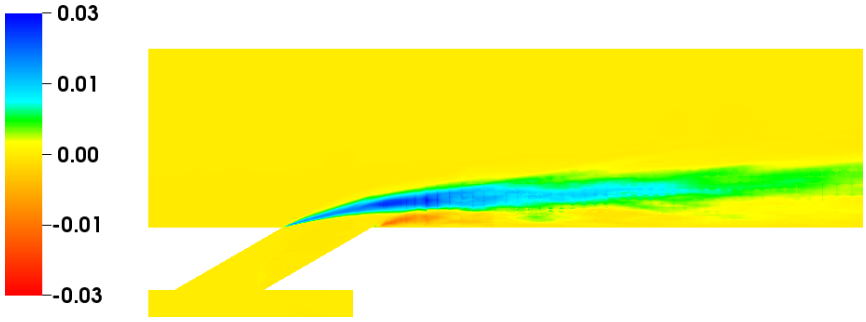
As discussed before, the performance of the film cooling hole is measured by the film cooling effectiveness (Equation 4.1). A contour of average film cooling effectiveness at the wall is shown in Figure 4.11. Overall, the film appears to spread across the span well except for near the holes at $z/D = \pm 0.25$, where evidence of coolant



(a) $\overline{\theta}$

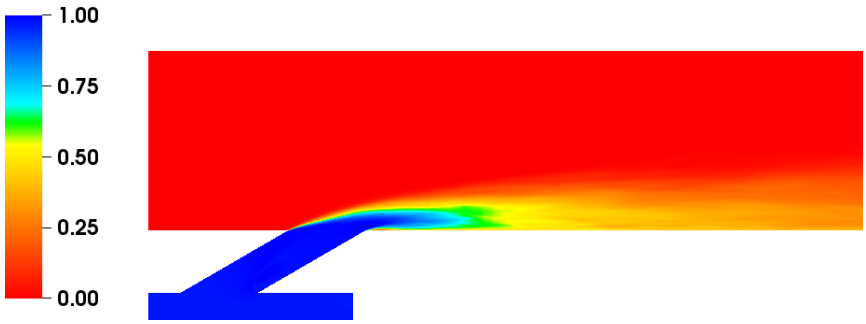


(b) θ_{rms}

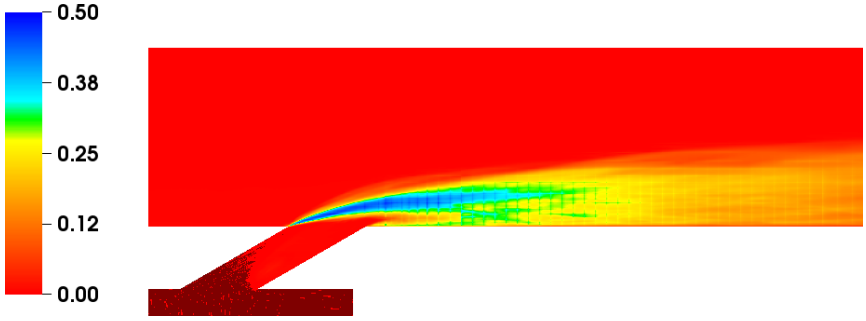


(c) $-\overline{v'\theta'}$

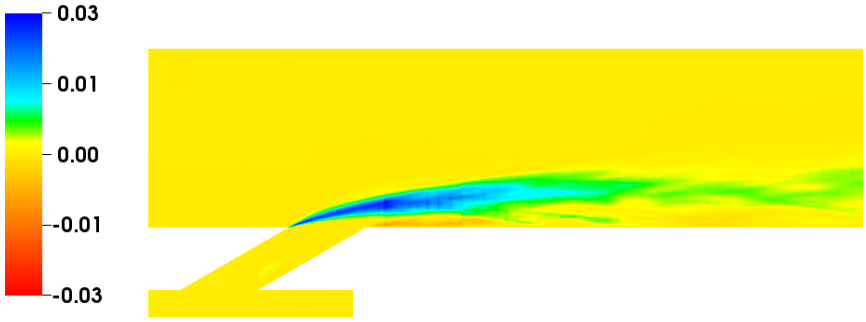
Figure 4.7: Contours of $\overline{\theta}$, θ_{rms} and $\overline{v'\theta'}$ at $z/D = 0.00$. The average temperature contour shows a slight liftoff near the trailing edge of the film cooling hole. The rms temperature and the cross term show strong levels of turbulence in the shear layer.



(a) $\overline{\theta}$

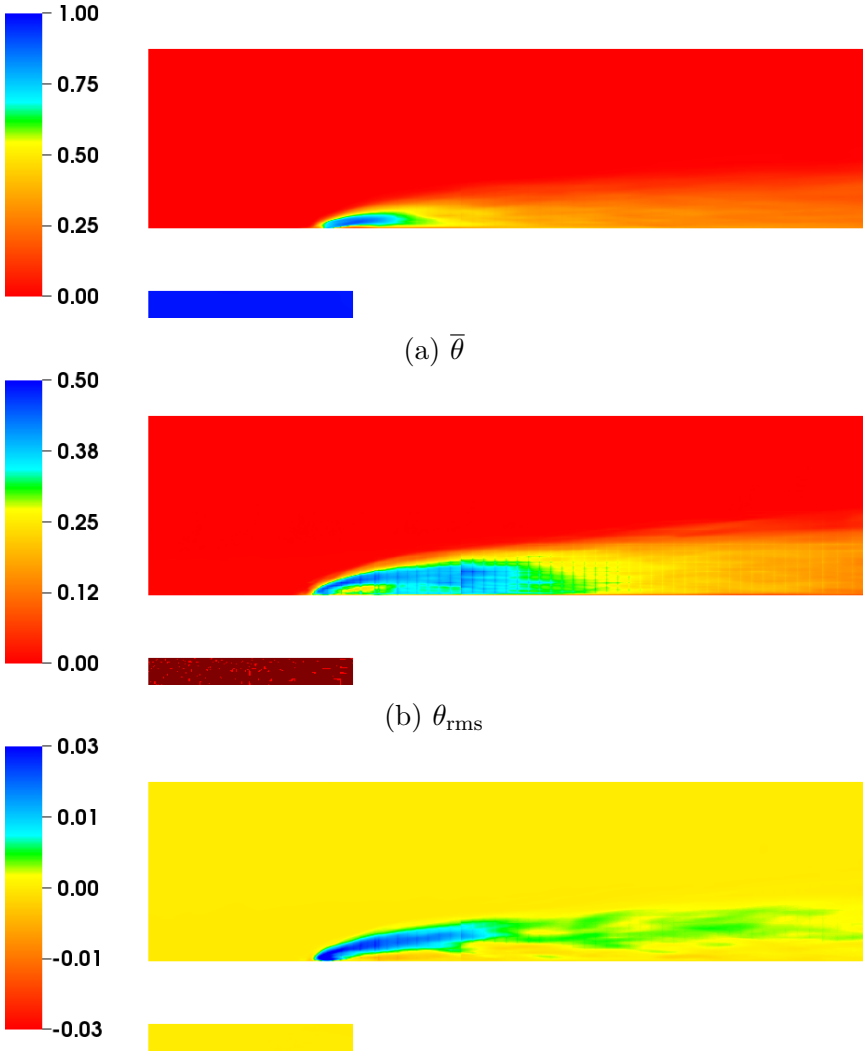


(b) θ_{rms}



(c) $-\overline{v'\theta'}$

Figure 4.8: Contours of $\overline{\theta}$, θ_{rms} and $\overline{v'\theta'}$ at $z/D = 0.25$. Evidence of complete liftoff is seen near the hole exit, but the coolant quickly reattaches. The turbulence mixing appears to be more pronounced, encompassing a larger area.

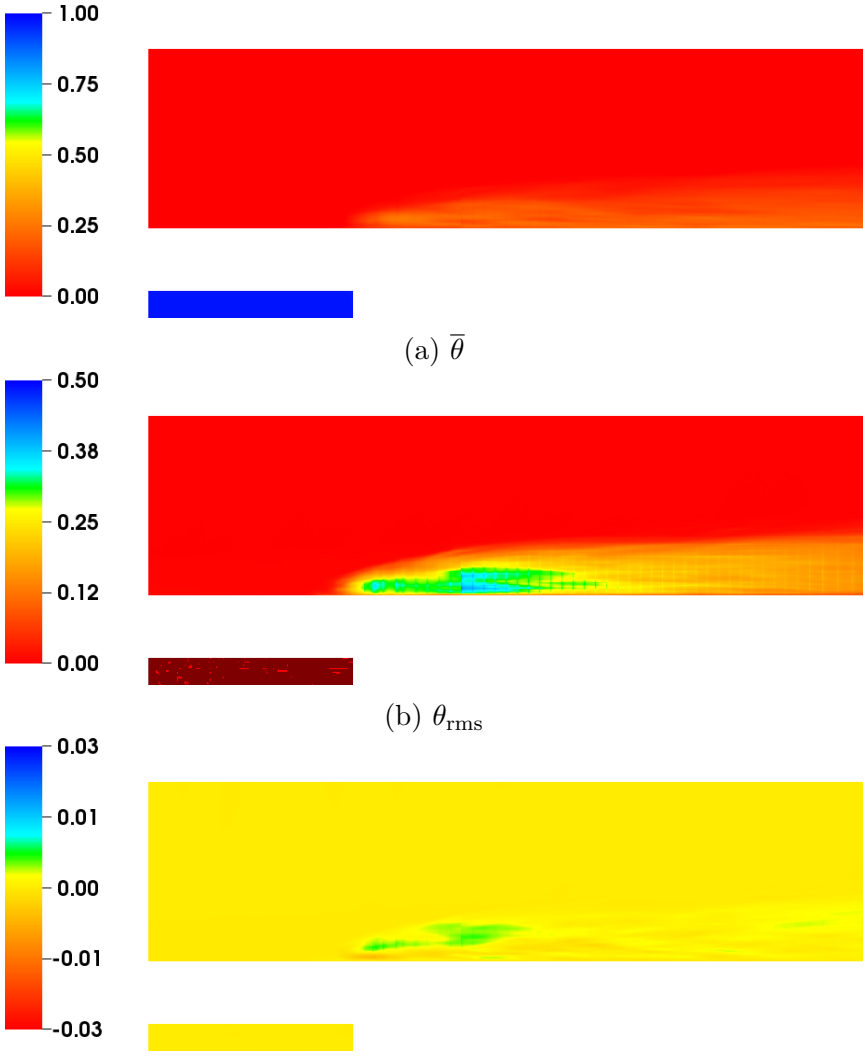


(a) $\overline{\theta}$

(b) θ_{rms}

(c) $-\overline{v'\theta'}$

Figure 4.9: Contours of $\overline{\theta}$, θ_{rms} and $\overline{v'\theta'}$ at $z/D = 0.50$. At the outer rim of the cooling hole, the coolant is still active, although in a reduced capacity.



(a) $\overline{\theta}$

(b) θ_{rms}

(c) $-\overline{v'\theta'}$

Figure 4.10: Contours of $\overline{\theta}$, θ_{rms} and $\overline{v'\theta'}$ at $z/D = 0.75$. Very little cooling has spread this far, although evident of fairly strong temperature fluctuations is still present.

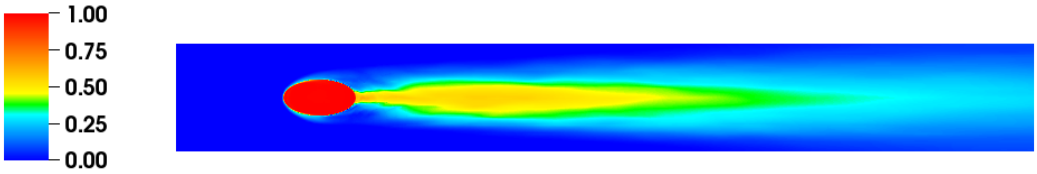
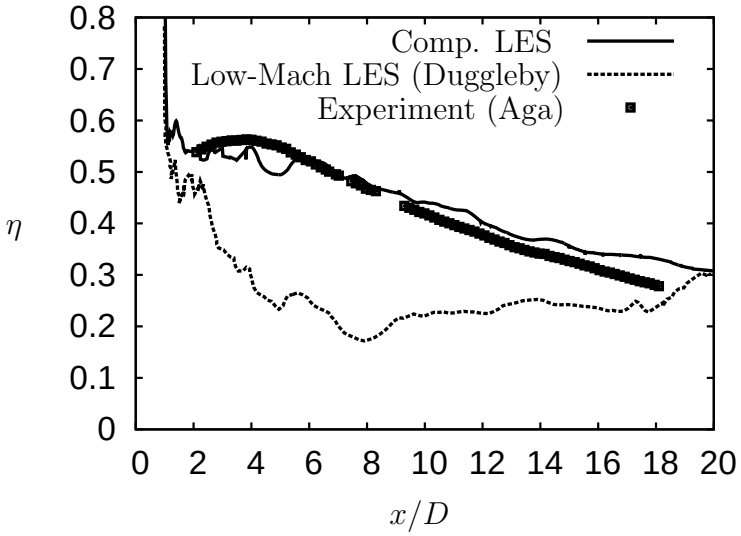


Figure 4.11: Contour of average film cooling effectiveness for film cooling case. The lift off and subsequent reattachment near $z/D = \pm 0.25$ can be seen in the figure. Overall, the spreading fo the film cooling is much improved as compared the previous low-Mach simulation in Duggleby *et a* [64].

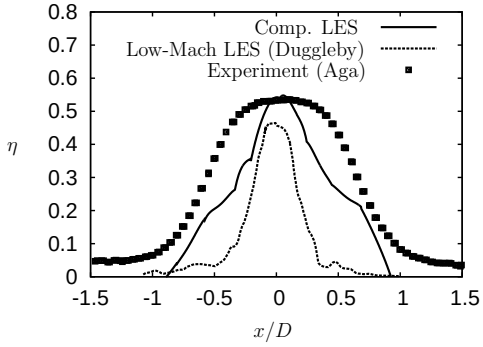
separation is seen. Based on what was seen at $z/D = 0.25$ in Figure 4.8a, this is not surprising. Quantatively, the centerline film cooling effectivness and spanwise film cooling effectiveness at $x/D = 8$ from the current simulation compares well to the experimental data from Aga [72], as shown in Figure 4.12. The prediction is not as good at $x/D = 2$, which is precisely the location where the coolant begins to reattach in Figure 4.11. Thus, the current simulation appears to overpredict the precise location of the reattachment, rather than underpredict the actual film cooling performance.

While the current compressible film cooling LES results are promising, there are a few outstanding questions that must be addressed in future work with the algorithm. The first item that could be improved is the inflow turbulence generation technique. While the RRM method appears to be effective in generating a realistic turbulent inflow, an unintended side effect is that unphysical cross-correlations appear, as evidenced in Figure 4.13 by the pulses in Mach number. While the pulses are faint, they effectively increased the amount of time required for time averaging. In a larger simulation, this increased time requirement can be problematic.

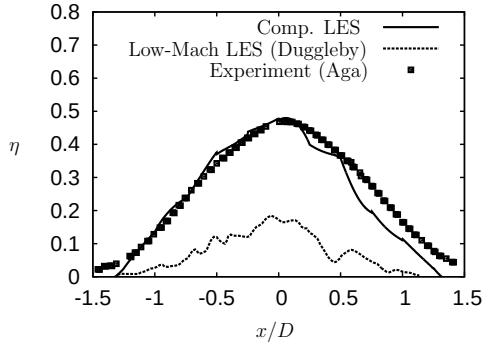
The simplified LES model also still needs to be quantified more rigorously, par-



(a) Centerline η



(b) Spanwise η, $x/D = 2$



(c) Spanwise η, $x/D = 8$

Figure 4.12: Plots of centerline and spanwise film cooling effectiveness for film cooling hole. There is good agreement between the compressible LES and the experimental data from Aga [72] for the centerline effectivness and the spanwise effectiveness at $x/D = 8$. The effectiveness at $x/D = 2$ for the compressible LES is underpredicted compared to the experiment, but still provides a better prediction than the low-Mach LES (Duggleby *et al* [64]).

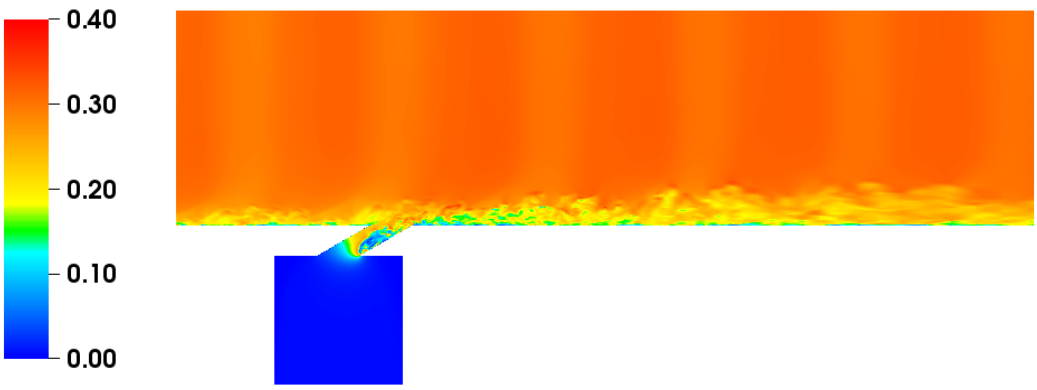


Figure 4.13: Contour of instantaneous Mach number for entire film cooling domain. While the recycling and rescaling method does well in generating a turbulent profile, the downside is that unphysical large scale motions are introduced, which are evident by the faint pulses of Mach number seen in the freestream. These pulses have a periodic length that is the same as the recycling length, suggesting that the recycling technique could use improvement.

ticularly in the context of the current compressible algorithm. The issue at hand is that, because of the lack of dealiasing in the current algorithm, a small filter is needed even for DNS studies to stabilize the flow. As such, it is difficult to differentiate between the amount of filtering needed for stability and the amount needed to represent the unresolved scales in the flow. A proper characterization of the LES filter is needed to quantify these differences.

5. CONCLUSIONS

The globally competitive market has led to increased importance of design optimization. Additionally, there is a need for a design cycle analysis to be fast and efficient, and in conjunction with the vast increases in available computing power, this has led to an increased use of numerical analysis in the design cycle.

This design shift is very much prevalent in the aerospace community, where computational fluid dynamics plays a large role in everything from aircraft design to film cooling optimization. Because of the large inherent computational costs of direct numerical simulation, industrial CFD usage has been limited mostly to the use of solving the time-averaged equations which rely heavily on turbulence models. The use of RANS solvers will continue to see heavy use in the aerospace industry, but RANS models are ineffective in many areas of the industry, necessitating the use of higher fidelity models such as DNS or LES. The modern supercomputing paradigm, with increased focus on massively-parallel machines employing hundreds of thousands or millions of processors, allows for LES or DNS to be practical alternatives for these cases. However, methods that work well for RANS are typically not built for success on massively-parallel machines, and thus newer methods must be explored.

A successful LES/DNS solver for industrial use must contain three core characteristics: it must be high order, it must be efficient on parallel machines, and it must be geometrically flexible. In a survey of modern methods, the spectral element method appears to be the best candidate for such a method. The high order, geometrically flexible nature of the discretization allows for high convergence rates in geometries of industrial interest, and the high data locality of the method lends itself well to massively-parallel computations. The open source SEM code, NEK5000, has proven

these characteristics in practical usage, but thus far has seen relatively limited usage in the aerospace industry due to its incompressible formulation. This project took the NEK5000 code base and implemented a new method allowing for the simulation of compressible turbulent flows of industrial relevance.

The DG compressible spectral element algorithm developed herein was heavily validated against a variety of inviscid and viscous test cases. These cases were specifically chosen to test certain attributes of the code, such as accuracy, stability and geometric flexiblity. With the code validated, a large-eddy simulation of film cooling heat transfer was undertaken to demonstrate the power of the formulation in practical usage. The new method demonstrated the ability to accurately predict film cooling effectiveness in an case where a previous low-Mach LES was not sufficient and RANS struggles. This algorithm which is highly accurate, massively-parallel, and geometricaly flexible is useful in terms of future design exploration, and will hopefully reduce the need for costly prototype testing in design cycle analysis.

REFERENCES

[1] Jaluria, Y., 2007. *Design and Optimization of Thermal Systems.* CRC Press, Boca Raton, FL.

[2] Thomas, V., 2009. "Virtual modeling helps build better cars.". *Machine Design,* **81**(23), pp. 32 – 37.

[3] Thierry Marchal, 2012. Simulate first; build to last. http://www.rdmag.com/articles/2012/04/simulate-first-build-last.

[4] Pope, S. B., 2000. *Turbulent Flows.* Cambridge University Press, New York, NY.

[5] Fujii, K., 2005. "Progress and future prospects of CFD in aerospace–wind tunnel and beyond". *Progress in Aerospace Sciences,* **41**(6), pp. 455 – 470.

[6] Jameson, A., and Ou, K., 2011. "50 years of transonic aircraft design". *Progress in Aerospace Sciences,* **47**(5), pp. 308 – 318.

[7] Jameson, A., Martinelli, L., and Vassberg, J., 2002. "Using computational fluid dynamics for aerodynamics–a critical assessment". In Proceedings of ICAS, pp. 2002–1.

[8] Imlay, S., 2012. "The role of parametric CFD analysis in design". *Machine Design,* **84**(3), pp. 68 – 70.

[9] Razinksy, E., 2010. Perspective on R&D needs for gas turbine power generation. UTSR Workshop. http://www.netl.doe.gov/publications/proceedings/11/Utsr/.

[10] Paul F. Fischer, J. W. L., and Kerkemeier, S. G., 2008. NEK5000 Web page. http://nek5000.mcs.anl.gov.

[11] Fischer, P., Lottes, J., Pointer, D., and Siegel, A., 2008. "Petascale algorithms for reactor hydrodynamics". In *Journal of Physics: Conference Series*, Vol. 125, IOP Publishing, p. 012076.

[12] Fischer, P. F., Ho, L. W., Karniadakis, G. E., Ronouist, E. M., and Patera, A. T., 1988. "Recent advances in parallel spectral element simulation of unsteady incompressible flows". *Comput. & Struct.,* **30**, pp. 217–231.

[13] Tufo, H. M., and Fischer, P. F., 1999. "Terascale spectral element algorithms and implementations". In Proc. of the ACM/IEEE SC99 Conf. on High Performance Networking and Computing, IEEE Computer Soc. Gordon Bell Prize paper.

[14] Schwaenen, M., Meador, C., Camp, J., Jagannathan, S., and Duggleby, A., 2011. "Massively-parallel direct numerical simulation of turbine vane endwall horseshoe vortex dynamics and heat transfer". *ASME Paper GT2011-45915.*

[15] Jagannathan, S., Schwänen, M., and Duggleby, A., 2011. "Low pressure turbine relaminarization bubble characterization using massively-parallel large eddy simulations". *J. Fluids Engineering.* In Review.

[16] Camp, J., 2011. "Massively-parallel spectral element large eddy simulation of a ring-type gas turbine combustor". Master's thesis, Texas A&M University, May 2011.

[17] Toro, E., 2009. *Riemann Solvers and Numerical Methods for Fluid Dynamics: A Practical Introduction.* Springer-Verlag Berlin Heidelberg, New York, NY.

[18] LeVeque, R., 2002. *Finite Volume Methods for Hyperbolic Problems*. Cambridge Texts in Applied Mathematics. Cambridge University Press, New York, NY.

[19] Hirsch, C., 2007. *Numerical Computation of Internal and External Flows: The Fundamentals of Computational Fluid Dynamics*, Vol. 1. Elsevier Science, Burlington, MA.

[20] Žaloudek, M., Fořt, J., and Fürst, J., 2006. "Numerical solution of compressible flow in a channel and blade cascade". *Flow, Turbulence and Combustion,* **76**(4), pp. 353–361.

[21] Brooks, A. N., and Hughes, T. J., 1982. "Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations". *Computer Methods in Applied Mechanics and Engineering,* **32**(1), pp. 199–259.

[22] Soulaimani, A., Saad, Y., and Rebaine, A., 2001. "An edge based stabilized finite element method for solving compressible flows: formulation and parallel implementation". *Computer Methods in Applied Mechanics and Engineering,* **190**(49), pp. 6735–6761.

[23] Martinez, M. J., and Gartling, D. K., 2004. "A finite element method for low-speed compressible flows". *Computer Methods in Applied Mechanics and Engineering,* **193**(21), pp. 1959–1979.

[24] Kirk, B. S., and Carey, G. F., 2008. "Development and validation of a SUPG finite element scheme for the compressible Navier-Stokes equations using a modified inviscid flux discretization". *International Journal for Numerical Methods in Fluids,* **57**(3), pp. 265–293.

[25] Don, W.-S., and Gottlieb, D., 1990. "Spectral simulations of an unsteady compressible flow past a circular cylinder". *Computer Methods in Applied Mechanics and Engineering,* **80**(1), pp. 39–58.

[26] Don, W.-S., Gottlieb, D., and Jung, J.-H., 2003. "A multidomain spectral method for supersonic reactive flows". *Journal of Computational Physics,* **192**(1), pp. 325–354.

[27] Hesthaven, J., and Gottlieb, D., 1996. "A stable penalty method for the compressible Navier-Stokes equations: I. Open boundary conditions". *SIAM Journal on Scientific Computing,* **17**(3), pp. 579–612.

[28] Hesthaven, J., 1997. "A stable penalty method for the compressible Navier–Stokes equations: II. one-dimensional domain decomposition schemes". *SIAM Journal on Scientific Computing,* **18**(3), pp. 658–685.

[29] Hesthaven, J., 1998. "A stable penalty method for the compressible Navier–Stokes equations: III. multidimensional domain decomposition schemes". *SIAM Journal on Scientific Computing,* **20**(1), pp. 62–93.

[30] Kopriva, D. A., and Kolias, J. H., 1995. A conservative staggered-grid chebyshev multidomain method for compressible flow. Tech. rep., DTIC Document.

[31] Kopriva, D. A., 1998. "A staggered-grid multidomain spectral method for the compressible Navier–Stokes equations". *Journal of Computational Physics,* **143**(1), pp. 125–158.

[32] Liu, Y., Vinokur, M., and Wang, Z., 2006. "Spectral difference method for unstructured grids I: basic formulation". *Journal of Computational Physics,* **216**(2), pp. 780–801.

[33] Liang, C., Premasuthan, S., Jameson, A., and Wang, Z., 2009. "Large eddy simulation of compressible turbulent channel flow with spectral difference method". *AIAA Paper, AIAA-2009-402, Orlando, FL.*

[34] Premasuthan, S., Liang, C., and Jameson, A., 2009. "A spectral difference method for viscous compressible flows with shocks". *AIAA Paper,* **3785**, p. 2009.

[35] Castonguay, P., Liang, C., and Jameson, A., 2010. "Simulation of transitional flow over airfoils using the spectral difference method". *AIAA Paper,* **4626**, p. 2010.

[36] Bassi, F., and Rebay, S., 1997. "High-order accurate discontinuous finite element solution of the 2D Euler equations". *Journal of Computational Physics,* **138**(2), pp. 251–285.

[37] Bassi, F., and Rebay, S., 1997. "A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier–Stokes equations". *Journal of Computational Physics,* **131**(2), pp. 267–279.

[38] Bassi, F., and Rebay, S., 2000. "GMRES discontinuous Galerkin solution of the compressible Navier-Stokes equations". In *Discontinuous Galerkin Methods*, B. Cockburn, G. E. Karniadakis, and C.-W. Shu, eds. Springer, New York, NY, pp. 197–208.

[39] Bassi, F., and Rebay, S., 2000. "A high order discontinuous Galerkin method for compressible turbulent flows". In *Discontinuous Galerkin Methods*, G. E. Cockburn, Bernardo andKarniadakis and C.-W. Shu, eds. Springer, New York, NY, pp. 77–88.

[40] Karniadakis, G. E., and Sherwin, S., 2005. *Spectral/hp Element Methods for Computational Fluid Dynamics*. Oxford University Press, New York, NY.

[41] Oden, J. T., and Baumann, C. E., 2000. "A conservative DGM for convection-diffusion and Navier-Stokes problems". In *Discontinuous Galerkin methods*, B. Cockburn, G. E. Karniadakis, and C.-W. Shu, eds. Springer, New York, NY, pp. 179–196.

[42] Warburton, T., Lomtev, I., Du, Y., Sherwin, S., and Karniadakis, G., 1999. "Galerkin and discontinuous Galerkin spectral/hp methods". *Computer Methods in Applied Mechanics and Engineering,* **175**(3), pp. 343–359.

[43] Kirby, R., Warburton, T., Lomtev, I., and Karniadakis, G., 2000. "A discontinuous Galerkin spectral/hp method on hybrid grids". *Applied Numerical Mathematics,* **33**(1), pp. 393–405.

[44] Gassner, G. J., Lörcher, F., Munz, C.-D., and Hesthaven, J. S., 2009. "Polymorphic nodal elements and their application in discontinuous Galerkin methods". *Journal of Computational Physics,* **228**(5), pp. 1573–1590.

[45] Hesthaven, J., and Warburton, T., 2008. *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications*. Texts in Applied Mathematics. Springer, New York, NY.

[46] Kopriva, D. A., 2006. "Metric identities and the discontinuous spectral element method on curvilinear meshes". *Journal of Scientific Computing,* **26**(3), pp. 301–327.

[47] Hindenlang, F., Gassner, G. J., Altmann, C., Beck, A., Staudenmaier, M., and Munz, C.-D., 2012. "Explicit discontinuous galerkin methods for unsteady problems". *Computers & Fluids,* **61**, pp. 86–93.

[48] Sutherland, W., 1893. "LII. the viscosity of gases and molecular force". *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science,* **36**(223), pp. 507–531.

[49] Dussauge, J.-P., and Smits, A. J., 1996. *Turbulent Shear Layers in Supersonic Flow*. AIP press, New York, NY.

[50] Deville, M. O., Fischer, P. F., and Mund, E. H., 2002. *High-order methods for incompressible fluid flow*, Vol. 9. Cambridge University Press, New York, NY.

[51] Kopriva, D. A., 2009. *Implementing Spectral Methods for Partial Differential Equations*. Springer, New York, NY.

[52] Canuto, C., Hussaini, Y., Quarteroni, A., and Zang, T., 2010. *Spectral Methods: Fundamentals in Single Domains*. Scientific Computation. Springer, New York, NY.

[53] Carlson, J.-R., 2011. Inflow/outflow boundary conditions with application to FUN3D. Tech. Rep. NASA/TM-2011-217181, Langley Research Center, Hampton, Virginia, October.

[54] Lomtev, I., Quillen, C., and Karniadakis, G., 1998. "Spectral/hp methods for viscous compressible flows on unstructured 2d meshes". *Journal of Computational Physics,* **144**(2), pp. 325–357.

[55] Bassi, F., and Rebay, S., 1997. "High-order accurate discontinuous finite element solution of the 2D Euler equations". *Journal of Computational Physics,* **138**(2), pp. 251 – 285.

[56] Slater, J. W., 2008. Converging-diverging verification (CDV) nozzle, July. http://www.grc.nasa.gov/WWW/wind/valid/cdv/cdv.html.

[57] Lomtev, I. L., 1999. "A discontinuous Galerkin method for the compressible Navier-Stokes equations in stationary and moving 3D domains". PhD thesis, Brown University, Applied Mathematics.

[58] Tomboulides, A. G., and Orszag, S. A., 2000. "Numerical investigation of transitional and weak turbulent flow past a sphere". *Journal of Fluid Mechanics,* **416**(1), pp. 45–73.

[59] Jeong, J., and Hussain, F., 1995. "On the identification of a vortex". *Journal of Fluid Mechanics,* **285**(69), pp. 69–94.

[60] Wu, J.-S., and Faeth, G., 1993. "Sphere wakes in still surroundings at intermediate Reynolds numbers". *AIAA Journal,* **31**(8), pp. 1448–1455.

[61] Chang, Y., 2000. "Reduced order methods for optimal control of turbulence". PhD thesis, Rice University, Mechanical Engineering and Materials Science.

[62] Han, J.-C., Dutta, S., and Ekkad, S., 2000. *Gas Turbine Heat Transfer and Cooling Technology.* Taylor & Francis, Boca Raton, FL.

[63] Bogard, D., 2006. "Airfoil film cooling". *The Gas Turbine Handbook.* NETL, http://www.netl.doe.gov/technologies/coalpower/turbines/refshelf/handbook/4.2.2.1.pdf.

[64] Duggleby, A., Camp, J., and Laskowski, G., 2013. "Evaluation of massively-parallel spectral element algorithm for LES of film-cooling". *ASME Paper GT2013-94281*.

[65] Liu, K., and Pletcher, R. H., 2005. "Large eddy simulation of discrete-hole film cooling in a flat plate turbulent boundary layer". *AIAA Paper,* **4944**.

[66] Guo, T., Li, S., and Liu, J., 2007. "Large eddy simulation of film cooling". In *Challenges of Power Engineering and Environment*, K.-f. Cen, Y. Chi, and J. Yan, eds. Springer, New York, NY, pp. 1419–1422.

[67] Gatski, T. B., and Bonnet, J.-P., 2013. *Compressibility, Turbulence and High Speed Flow*. Academic Press, Amsterdam, The Netherlands.

[68] Germano, M., Piomelli, U., Moin, P., and Cabot, W. H., 1991. "A dynamic subgrid-scale eddy viscosity model". *Physics of Fluids A: Fluid Dynamics,* **3**, p. 1760.

[69] Stolz, S., 2005. "High-pass filtered eddy-viscosity models for large-eddy simulations of compressible wall-bounded flows". *Journal of Fluids Engineering,* **127**(4), pp. 666–673.

[70] Duggleby, A., and Camp, J., 2011. "Massively-parallel computational fluid dynamics with large eddy simulation in complex geometries". *ASME Paper IMECE2011-62489*.

[71] Urbin, G., and Knight, D., 2001. "Large-eddy simulation of a supersonic boundary layer using an unstructured grid". *AIAA Journal,* **39**(7), pp. 1288–1295.

[72] Aga, V., 2009. "Experimental investigation of the influence of flow structure on compound angled film cooling performance". PhD thesis, ETH Switzerland.