Adjoint-Based Optimization of U-Bend Channel Flow Using a Multi-Fidelity Eddy Viscosity Turbulence Model

by

Michael Elia Hayek

B.S., University of Massachusetts Amherst (2009)

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2017

Signature redacted

Author .....
Department of Aeronautics and Astronautics
May 25, 2017

Signature redacted

Certified by .....
Qiqi Wang
Associate Professor of Aeronautics and Astronautics
Thesis Supervisor

Signature redacted

Accepted by .....
Youssef M. Marzouk
Associate Professor of Aeronautics and Astronautics
Chair, Graduate Program Committee

Adjoint-Based Optimization of U-Bend Channel Flow Using a Multi-Fidelity Eddy Viscosity Turbulence Model

by

Michael Elia Hayek

Submitted to the Department of Aeronautics and Astronautics
on May 25, 2017, in partial fulfillment of the
requirements for the degree of
Master of Science

Abstract

Many fluid flows in engineering are turbulent and require the use of computational fluid dynamics (CFD) for design purposes. Optimization with CFD has largely been limited to low-fidelity simulation methods, such as Reynolds Averaged Navier Stokes (RANS), due to current computational capabilities. However, RANS has been shown to lack sufficient accuracy for certain flows. This thesis presents CFD simulation of a 180 degree U-bend square duct using low-fidelity steady RANS and high-fidelity wall-resolved Large Eddy Simulation (LES) models. The LES solution is shown to match experimental results, whereas the RANS solution is not sufficiently accurate. A process for training a RANS eddy viscosity field using the LES solution is provided. This approach is based on solving an inference problem by comparing the RANS calculations to the LES solution and tuning cell-based turbulent viscosity values. This multi-fidelity framework is intended to highlight that high-fidelity solutions can be used to improve even the simplest RANS turbulence models. The adjoint method is used for efficient gradient-based optimization of the turbulent viscosity on a U-bend channel to minimize the velocity solution error. Other objective functions are explored to check the uniqueness of the optimized turbulent viscosity. Sensitivity of the optimized result to the numerical convection scheme is presented to help provide insight for future optimization of turbulence models. The optimized turbulent viscosity is also used on a modified U-bend channel to demonstrate the applicability of the method on new geometries.

Thesis Supervisor: Qiqi Wang
Title: Associate Professor of Aeronautics and Astronautics

Acknowledgments

First and foremost, I would like to thank my advisor Professor Qiqi Wang. No challenge is too grand for Professor Wang. He has always made himself available and provided invaluable ideas and insight. This work would not have been possible without his guidance.

I would like to thank General Electric Aviation for their support. This research was funded by GE through the Advanced Courses in Engineering, under the supervision of Kenneth Gould and Paul Patoulidis. I would also like to thank GE management for allowing me to pursue this degree while employed at GE Aviation. This specifically includes Tyler Hooper, Marcus Ottaviano, Spiro Harbilas, Dan Barber, and Dan Daigle. They have been patient and accommodating during my extended time at MIT.

I would also like to thank Greg Laskowski, who has been my GE research advisor and has generously offered his time to review my work. Our regular meetings have provided valuable understanding and direction to the research.

Additional appreciation goes to current and former ACDL group members, namely Chaitanya Talnikar and Han Chen for enduring countless questions and sharing their knowledge.

I would like to thank my parents, Jay and Nora, and the rest of my family for their constant support throughout my time at MIT and my academic career. Finally, I would like to thank my wife Jenna, who has always been very loving, patient, and understanding. The completion of this thesis would not have been possible without her continual encouragement.

Contents

List of Figures

List of Tables

Nomenclature

Subscripts

b	Bulk
c	Centerline
eff	Effective
∞	Far-field
t	Turbulent
th	Threshold
w	Wall

Symbols

d_h	Hydraulic diameter
H	Full channel width
I	Turbulence intensity
J	Objective function
k	Turbulent kinetic energy
Pe	Peclet number
p	Pressure
Re	Reynolds number
s_{ij}	Strain rate tensor
u	Velocity
u_τ	Friction velocity
u',v',w'	Fluctuating velocity in x, y, z directions
$\overline{u}$,$\overline{v}$,$\overline{w}$	Mean velocity components in x, y, z directions

Greek

γ	Upwind / central differencing blend factor
η	Kolmogorov length scale
μ	Molecular viscosity
ν	Kinematic viscosity
ν_t	Kinematic eddy, or turbulent, viscosity
ρ	Density
τ	Shear stress
Δ	Change in quantity; LES filter width
Δx	Grid size
Δt	Timestep size

Abbreviations

ACDL	Aerospace Computational Design Laboratory at MIT
AD	Automatic Differentiation
CFD	Computational Fluid Dynamics
CFL	Courant-Friedrichs-Lewy
DOF	Degrees of Freedom
FD	Finite Difference
FTU	Flow Through Unit
HiFi	High-fidelity
HIFIR	High-fidelity trained RANS
LES	Large Eddy Simulation
PDE	Partial Differential Equation
PIV	Particle Image Velocimetry
RANS	Reynolds Averaged Navier-Stokes
SGS	Sub-grid Scale
SST	Shear Stress Transport
TKE	Turbulent Kinetic Energy

Chapter 1

Introduction

1.1 Motivation

Most fluid flows in nature and engineering applications involve turbulence. Fluid flows can be accurately described by the Navier-Stokes equations with proper boundary conditions. However, there is no general solution to the Navier-Stokes equations and thus most flow solutions must be obtained through numerical simulation. Fully resolving turbulence using the Navier-Stokes equations requires unsteady simulations with extremely fine spatial and temporal discretization. This method is known as direct numerical simulation (DNS) and is computationally very expensive. Many computational methods have been introduced to reduce the computational cost by modeling part or all of the turbulence as additional terms to the Navier-Stokes equations. Large eddy simulations (LES) directly solve for the larger turbulent structures but model the smallest scales of turbulence and therefore do not require spatial discretization as fine as that required for DNS. LES, by nature, is also unsteady and three-dimensional, and still relatively expensive. Methods based on solving the Reynolds averaged Navier-Stokes (RANS) equations fully model the effect of turbulence. RANS can be run steady-state to provide a statistically-averaged flow field for a significant reduction in computational cost versus LES and DNS.

The price of reduced computational cost is often reduced accuracy. For simpler flows, such as channel flow and free shear flow that is inhomogeneous in one direction,

even the most basic RANS turbulence models can produce good results. However, it is known that for more complex flows the mean flow fields obtained with RANS turbulence modeling show substantial discrepancies to experimental data, especially in cases with flow separation [32]. Although these limitations are known, RANS is the most popular Computational Fluid Dynamics (CFD) method in industry. This is due to the relatively inexpensive cost in which simulations may take only several hours. LES has been shown to produce more accurate results compared to RANS on complex three-dimensional flows and can provide solutions that agree well with experimental data without the additional cost of performing DNS [1].

Optimization with CFD is a fast growing area and is being embraced by the design community to improve existing products and aid engineers in design space exploration to find improved solutions to problems involving fluid flows. Optimization with CFD requires many design iterations and using LES or DNS for the flow field solution for each design iteration is prohibitively expensive today due to computational resource and schedule constraints. Recently, due to continued growth in computer capabilities, LES has become more common but is still mostly used for research and single-run applications. Thus, most CFD optimization has been performed with RANS models despite the inaccuracies with complex flows.

The motivation for this work is to improve the accuracy of RANS models for use in optimization of complex flows. The desire is to obtain accuracy similar to time-averaged LES solutions with computational cost closer to that of RANS simulations. This would allow for more accurate flow solutions which would in turn lead to improved designs with minimal impact to cost and schedule. The approach is to train a RANS model with a higher fidelity time-averaged solution (LES, DNS, or experimental data). The turbulent, or eddy, viscosity field of the low-fidelity (RANS) simulation is modified in order to match the velocity field of the high fidelity simulation.

1.2 Background and Approach

1.2.1 Applicable Scenarios

The multi-fidelity approach briefly discussed in Section 1.1 is applicable to complex flows where RANS predictions break down. This includes wall-bounded flows or flows with high curvature containing significant secondary flows as well as flows with separation or strong rotation. A higher fidelity simulation, such as LES or DNS, can accurately represent the physical flow field of these complex flows. One such flow scenario where RANS can breakdown is in the internal flow of cooled gas turbine airfoils, further discussed in the following section. The internal cooling circuit of many turbine airfoils have 180° bends which feature large streamline curvature, secondary flow, and separation and recirculation zones. This specific case is used as the test case for this research.

1.2.2 Turbine Airfoil Internal Cooling

Gas turbine engines, either land-based or aero-purposed, operate based on the Brayton cycle. The cycle involves compression, constant pressure combustion, and expansion. The turbine is responsible for expansion of the high-pressure, high-temperature gases. Turbines often contain multiple stages, especially in turbofan, turboprop, and turboshaft configurations. Each turbine stage consists of stationary airfoils called vanes or nozzles and rotating airfoils called blades. Through each stage, energy is extracted and converted to rotational, mechanical energy and the pressure and temperature of the gases are reduced. One method to improve overall cycle efficiency is to increase peak cycle temperature. Internal cooling of airfoils in the early turbine stages allows for significant turbine inlet temperature increases and therefore improved efficiency over uncooled turbines, despite the use of high-pressure air taken from the cycle after compression as the source of cooling air. The goal is to maximize blade life while minimizing cooling flow and therefore reducing the penalty to engine performance. Air-cooled turbine airfoils can have complex internal flow networks which

work the cooling air to maintain airfoil durability. Many times, the cooling air is routed to the airfoil external flow through cooling holes. This cooling also helps envelope areas of the blade external surface with a film of cooler air, providing reduced exposure to the extreme gas-path temperatures. Much research and design effort is invested in internal cooling circuit configurations to maximize the effectiveness of the cooling flow and minimize the cooling flow demand from the primary gas-path flow to minimize the impact to engine performance.

Figure 1-1 shows an example of a turbine blade. The blade is also shown transparent exposing the internal cooling circuit. These cooling configurations are known as "serpentine" cooling, due to the winding nature of the cooling cavities. The serpentine cooling configurations contain tip and root-turns which are essentially 180° tight radius bends. These turns, or bends, are a significant contribution to pressure drops within the cooling circuit. In order to ensure the cooling flow exits through cooling holes properly, the pressure internally must be greater than that of the external local pressure. Excessive pressure drop from the turn can restrict the positioning of film cooling holes or minimize the amount of additional heat transfer possible. A miscalculation of the internal flow pressures could also lead to hot-gas ingestion, which can quickly fail a turbine airfoil and perhaps the engine.

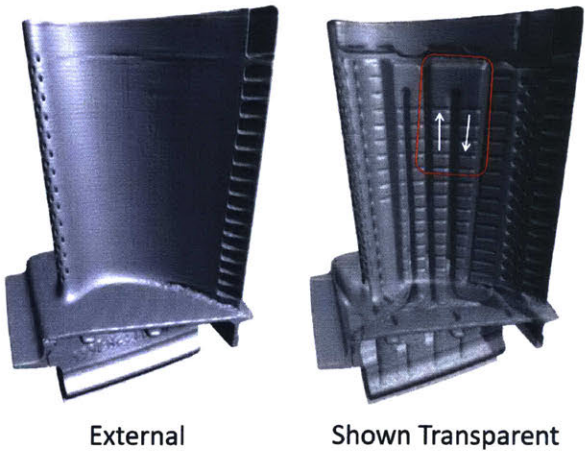
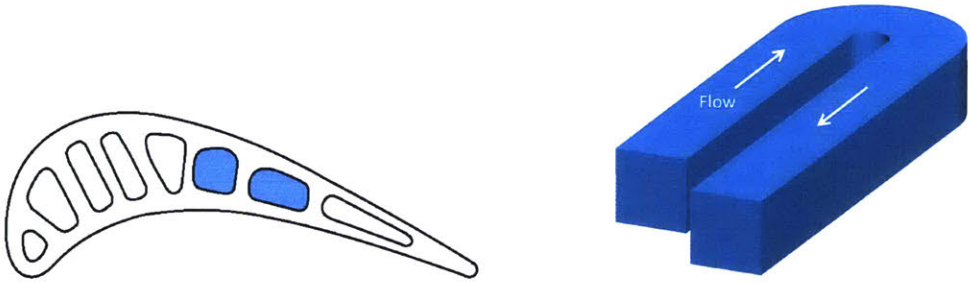


Figure 1-1: Example turbine blade shown with internal passages.
Original figure courtesy of Lindstrom [19]

A component of this research to demonstrate the accuracy of standard RANS and LES approaches on an idealized smooth-wall square duct with a 180° U-bend to motivate the need for improved RANS turbulence modeling. The idealized U-bend square duct is used to model the highlighted serpentine turn in Figure 1-1, which is typical of cooled turbine airfoils. The square duct models cavities near the mid-chord region of the airfoil, where the chord-wise length of the cavity is approximately the same as the airfoil thickness as shown in the blade section view of Figure 1-2a, highlighted in blue. Figure 1-2b displays the idealized square duct model domain, which is the same geometry used by Verstraete et al. [39].



(a) Airfoil cross-section (b) Idealized duct model

Figure 1-2: Section of turbine airfoil and idealized duct model

The effect of rotation is ignored to reduce complexity of the flow simulations. Including the effect of rotation would not change the process of optimization presented in this thesis. The duct is also smooth-walled. Heat transfer enhancement features, such as turbulators, were not included as these features can be added to increase heat transfer as necessary and again would not change the process presented in this thesis.

1.2.3 Approach

As mentioned in Section 1.1, the approach taken in this research to demonstrate accurate and fast results for complex flow is to train a low-fidelity model with a higher-fidelity solution. The low fidelity model used is a RANS based solver. The high-fidelity solution can be obtained from LES and DNS solutions. For this research, time-averaged wall-resolved LES is used as the high-fidelity solution.

In Chapter 3, the LES vs RANS performance is compared to experimental data on a U-bend square duct to show the accuracy of the two methods. The optimization is then performed on a U-bend channel, in which the time-averaged results can be collapsed to 2D. The channel is used instead of the square duct to reduce the computational cost and allow for a 2D RANS simulation.

A generic eddy viscosity RANS turbulence model following the Boussinesq approximation is used for the low-fidelity simulations. The Boussinesq approximation relates the turbulence stress to the mean flow, and introduces the concept of eddy viscosity. The eddy viscosity in the low-fidelity solver is inferred through optimization instead of being computed by additional transport equations performed in other popular turbulence models, such as the k-ϵ or k-ω models. This approach is discussed in detail in Section 2.3.4.

In this optimization, the eddy viscosity field is the design variable. The objective is to minimize the velocity field difference between the low and high fidelity simulations. The eddy viscosity is allowed to vary throughout the domain with each cell having a unique value. This creates a large number of design variables with a single objective. For this reason, adjoint-based gradient methods are used for optimization of eddy viscosity. Adjoint-based methods have the ability to handle a very large number of design variables efficiently. This optimization will be known as "training" of the low-fidelity model.

In the future, more sophisticated eddy viscosity field definition - perhaps based on local flow parameters and non-dimensional spatial position relative to key geometry features - could be used in conjunction with shape optimization. This is further discussed in Section 6.2.

1.3 Previous Research

The square duct U-bend geometry in this thesis is modeled after the geometry used by Verstraete et al. [39]. Verstraete et al. optimized the shape of the U-bend region to minimize the pressure drop across the bend using a metamodel-assisted differential

evolution algorithm and the k-ϵ RANS turbulence model. Experimental measurements were performed on the baseline and optimized geometry in part two of their research [3]. It was observed that the simulation did not adequately capture the turbulence generated by the U-bend, but matched the overall experimental pressure drop measurements well. Cheah et al. [2] used laser-Doppler anemometry to investigate the impact of rotation on 180° U-bend ducts. Schabacker et al. [35] and Son et al. [36] used particle image velocimetry (PIV) to characterize the flow field. The experiments by Son et al. also investigated the impact of smooth vs turbulated ducts. All studies showed that for sharp bends, thus high curvature, separation occurs along the inner wall in the post-bend region.

Saha and Acharya [33, 34] explored several geometry modifications to reduce the pressure drop across the bend, such as increasing the inner bend radius by creating a bulb, or the use of turning vanes to reduce the separation. Metzger et al. [24] varied the duct width, the outer wall corner radii, and the duct height at the bend peak. The latter, called the clearance height, was found to have a large impact on the pressure drop. Liou et al. [20] explored the impact of the inner wall thickness and found that a thicker wall reduced the separation zone due to reduced turbulence levels.

Turbulence modeling has been a highly active area of research since the introduction of computers. More recently, the use of higher fidelity results, such as DNS or experimental data, to improve RANS models has gained interest. Some approaches have used the data directly to replace the Reynolds stress term to fully close the RANS equation without the use of additional modeling. Poroseva et al. [30] showed that the use of DNS data directly to represent all unknown terms in the RANS equations for channel flow and boundary layer simulations can lead to nonphysical results. The error was attributed to uncertainty in the statistical data collected from DNS, as the erroneous results were reproduced on multiple reliable RANS solvers. In fact, this process was proposed as a tool for uncertainty quantification in DNS data [29]. Poroseva et al. also investigated the impact of time-averaging of the DNS data and has shown there is a systematic error in the DNS independent of time-averaging [31]. This indicates that direct prescription of unknown terms in the RANS equation may

never produce an accurate result. Similar solution errors using DNS results were found by Wang [43], in which the direct usage of the Reynolds stress from DNS for RANS channel flow did not produce the DNS time-averaged velocity profile. However, when a simple eddy viscosity model was used in which the spatial distribution of the eddy viscosity was derived from the DNS Reynolds stress and DNS velocity gradient, the resulting RANS velocity profile was in excellent agreement with the DNS profile. This suggests there is some error in the DNS statistical data and that the RANS solution is very sensitive to these errors.

Another approach to improving RANS models is the use of high fidelity results or data to reduce the modeling error or solution error. Using a simple eddy-viscosity model, Dow [4] used adjoint-based optimization to minimize the solution error to infer the error of the turbulent viscosity field for turbulent channel flow. Tracey et al. [38] used machine-learning to replace parts of the Spalart-Allmaras (SA) turbulence model source terms using flat plate simulations as the training data. The machined-learned SA produced fairly good results on low angle of attack airfoils and channel flow when compared to higher fidelity solutions. However, when using a learned model to replace all the source terms, the model struggled with some new channel flow cases. Duraisamy et al. [5] showed the potential of inverse modeling and machine learning techniques by creating functional forms of the RANS closure, rather than tuning model parameters, to improve predictive models. Wang et al. [41] used machine learning techniques to reconstruct the Reynolds stress by utilizing DNS of a periodic hill. The learned model was used on different hill domains and showed very good agreement of the Reynolds stress with higher fidelity solutions and showed improvement over standard RANS models.

1.4 Thesis Objectives and Contributions

The objective of this research is to show that high fidelity simulations can be used to reduce the solution error of RANS models through the use of optimization to estimate the Reynolds stress. As discussed earlier, using the higher fidelity data directly to replace model unknowns can lead to instabilities and erroneous solutions. Using the high fidelity data to reduce modeling error has also been shown to lead to significant solution errors. The method utilized in this research is more stable, as the high fidelity data is used to reduce the solution error by tuning the existing turbulence model.

Primary contributions of this thesis:

- An assessment of the accuracy of RANS and LES methods is provided for complex internal flows. The square duct test case used has many complex flow features in which the LES was shown to provide adequate results. In contrast, common RANS models are not sufficiently accurate. Both the k-ϵ and k-ω SST RANS models are shown to miss on the flow field and turbulent kinetic energy for the test case.

- High fidelity simulations can be used to tune even the simplest turbulence models to significantly reduce the solution error. In this work, LES solutions are used to train an eddy viscosity turbulence model by tuning a single RANS turbulence model parameter: the turbulent viscosity. The turbulent viscosity is inferred to minimize the mismatch between the RANS and time-averaged LES velocity field instead of direct prescription of the Reynolds stress term.

- The optimized turbulent viscosity can be dependent on the optimization framework. The use of the velocity mismatch as the solution error is shown to be the most successful approach. It is recommended to use higher order convective schemes and to use the log of the turbulent viscosity as the control parameter to allow for unbounded optimization.

1.5 Thesis Structure

The remainder of this thesis is presented in the following manner:

Chapter 2:

Chapter 2 focuses on the methods used to perform the research presented in Chapter 3 and 4. Gradient based optimization, using the adjoint method, is presented for the RANS training procedure. Turbulence modeling background, along with the specific RANS training procedure, is also provided.

Chapter 3:

Chapter 3 investigates the numerical results of a 180° U-bend square duct simulation using LES and RANS modeling methods. The results are also compared to available experimental data of the same geometry. LES is shown to be an accurate computation tool, whereas standard RANS models are not adequate tools to predict the complex dynamics present in this example.

Chapter 4:

A 180° U-bend channel is used as a new test case to demonstrate a method of improving a RANS turbulence model using LES solutions. The time-averaged LES solution is used to train an eddy viscosity RANS turbulence model. The results of the optimized model are compared to standard k-ω SST RANS model and the time-averaged LES solution. Sensitivity of the optimization results to various objective functions is provided. The method of bounding the turbulent viscosity is also explored to help identify areas of the simulation where the eddy viscosity model could be breaking down.

Chapter 5:

The U-bend geometry from Chapter 4 is modified by reducing the middle wall thickness to create a smaller bend radius. The optimized turbulent viscosity from

Chapter 4 is mapped onto the new geometry and rerun to explore the applicability of the HIFIR model to new geometries. The results are compared to LES and k-ω SST RANS solution obtained with the new geometry.

Chapter 6:

Chapter 6 provides a summary of the thesis and offers suggestions for future research.

Chapter 2

Methods

2.1 Optimization Methods

2.1.1 Gradient Based Optimization

Optimization is the minimization or maximization of some function relative to some set. The process results in the selection of a set of input, or design, variables that provide an optimal solution. Often the function is non-linear and the optimization procedure is iterative.

Gradient based optimization methods are more efficient than gradient-free methods for smooth, differentiable functions when the gradient information is available. The gradient information aids in the selection of new input variables for the subsequent iterations. The iterative Newton method is an effective gradient-based method that uses the gradient and Hessian information to drive the objective to a minimum. The Newton method is centered around a quadratic approximation for a function $f(x)$ for inputs around x_i at iteration i. If f is assumed to be twice differentiable, the Taylor expansion can be written as

$$f(x+\delta x) = f(x) + \delta x^{\mathrm{T}} \nabla f(x) + \frac{1}{2}\delta x^{\mathrm{T}} \big(\nabla^2 f(x)\big)\delta x \tag{2.1}$$

where it is desired that $f(x+\delta x) < f(x)$ for a minimization problem. The gradient ∇f at iteration i is denoted as g_i and the Hessian $\nabla^2 f$ at iteration i is rewritten as

H_i. The quadratic approximations Q at iteration i is rewritten in Equation 2.2.

$$Q_i(\delta x) = f(x_i) + \delta x^{\mathrm{T}} g_i + \frac{1}{2}\delta x^{\mathrm{T}} H_i \delta x \tag{2.2}$$

It is desirable to minimize the local quadratic approximation, thus the first derivative of Q is set to zero as shown in Equation 2.3. This can be rearranged to obtain the value of δx shown in Equation 2.4.

$$\frac{\partial Q_i(\delta x)}{\partial \delta x} = g_i + H_i \delta x = 0 \tag{2.3}$$

$$\delta x = -H_i^{-1} g_i \tag{2.4}$$

This provides a good search direction to select the next set of inputs, $x_{i+1} = x_i + \delta x$. Since the gradient and Hessian can change at the next iteration, the step size is scaled by α as shown in Equation 2.5. The value of α is set to obtain a $f(x_{n+1})$ that is sufficiently smaller than $f(x_i)$ by using line search methods.

$$x_{i+1} = x_i - \alpha\left(H_i^{-1} g_i\right) \tag{2.5}$$

For large problems, it is often impractical to obtain the Hessian. In this research, only the gradient is obtained using the adjoint method and is further discussed in Section 2.2. Therefore, a quasi-Newton method, in which approximations of the Hessian are calculated, is a good option for performing the optimization. The Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm is a popular qausi-Newton optimization method.

2.1.2 L-BFGS method

The BFGS method is named after the four authors who independently developed the algorithm in 1970. As mentioned, the BFGS algorithm only requires gradient information at every iteration. The Hessian approximation starts as the identity matrix and is updated using previous step gradient and position information. An

alternative method is to use only a small number of recent position and gradient information to recreate and update the Hessian to reduce memory requirements and is known as the L-BFGS algorithm (L for limited memory). This method avoids storage of the dense n by n Hessian matrix, where n is the number of degrees of freedom. With these modifications, the L-BFGS algorithm is a very efficient algorithm for large-scale problems. For this research the L-BFGS-B method from the Python Scipy.minimize package is used to perform the optimization.

For the turbulent viscosity training optimization, the number of degrees of freedom (DOF) is the number of cells in the mesh since each cell's turbulent viscosity is tuned to minimize the objective function. A 2D problem could have over 10,000 cells and a 3D problem can have over a million cells, making the L-BFGS method a necessity.

2.2 The Adjoint Method

2.2.1 Introduction

Often, simulations are performed where the solution is used to calculate a desired quantity, such as the pressure drop or the flow-rate in a fluid simulation. A discretized partial differential equation (PDE) is solved of the form $F(x, s)$, where x is the solution vector and s is the design parameter or control variable. An objective function $J = J(x)$ is then computed based on the solution of the PDE. Many times, the gradient $\partial J/\partial s$, the gradient of the objective function with respect to the design or control parameters, is also desired for sensitivity studies or for gradient-based optimization.

One method of approximating this gradient is to evaluate the objective functions by perturbing each design variable independently and approximating the gradient using finite differences as shown in Equation 2.6.

$$\frac{\partial J}{\partial s_i} \approx \frac{J((x(s_i + \delta s_i) - J(x(s_i))}{\delta s_i} \tag{2.6}$$

With n design variables, computing the sensitivity gradient using finite differences requires n+1 evaluations of the objective function. Each objective function requires

solving the PDE, which can take considerable amount of time for fluid simulations. The number of degrees of freedom, n, can also be very large depending on the problem.

In this research, the number of design variables for the turbulent viscosity training matches the number of cells, which is approximately 40,000 for the 2D case. Each simulation can vary between 10 and 60 minutes depending on initial conditions. Using finite differences, it could take over a year to compute the sensitivity gradient $\partial J/\partial s$.

The adjoint method provides an efficient way to compute the gradient that is independent of the number of design variables. In fact, the cost is usually comparable to the cost of solving the PDE for x once, which would take less than an hour in the above example. The basic process is discussed in the next section and the implementation in the code is discussed in Section 2.2.3.

2.2.2 Formulation

In this research, the physics are governed by PDEs. The governing equation is solved iteratively and has the form shown in Equation 2.7, where s is a set of control variables or parameters, x is the corresponding solution to the PDE using s, and k is the iteration.

$$x_{k+1} = F(x_k, s) \tag{2.7}$$

When the solution has converged, the output is essentially equal to the previous solution, $x_{k+1} = x_k = x$. The governing equation can be rewritten as

$$x - F(x, s) = 0 \ . \tag{2.8}$$

The objective function J is a function of the solution x, as shown below.

$$J = J(x) \tag{2.9}$$

In order to obtain the sensitivity of the objective function to the control parameters, the above non-linear equations must be linearized. It is known a small change

in parameter s causes a small change in the solution x, which in turns causes a small change in the objective J. Using linearization, these small changes are characterized using the chain rule in Equations 2.10 and 2.11 for the governing equation and the objective function, respectively.

$$\delta x - \frac{\partial F}{\partial x}\delta x - \frac{\partial F}{\partial s}\delta s = 0 \tag{2.10}$$

$$\delta J = \frac{\partial J}{\partial x}\delta x \tag{2.11}$$

Since the variation of the linearized governing equation is zero, then

$$\phi^T\left(\delta x - \frac{\partial F}{\partial x}\delta x - \frac{\partial F}{\partial s}\delta s\right) = 0 \tag{2.12}$$

is also zero for any vector ϕ. Equation 2.12 can then be added to the perturbed linearized objective function and rearranged to produce

$$\begin{aligned}\delta J &= \frac{\partial J}{\partial x}\delta x + \phi^T\left(\delta x - \frac{\partial F}{\partial x}\delta x - \frac{\partial F}{\partial s}\delta s\right) \\ &= \left(\frac{\partial J}{\partial x} + \phi^T - \phi^T\left(\frac{\partial F}{\partial x}\right)\right)\delta x - \phi^T\left(\frac{\partial F}{\partial s}\right)\delta s\end{aligned} \tag{2.13}$$

which is valid for any ϕ. The key is to select a particular ϕ so that the first term on the right hand side of Equation 2.13 is zero. This equation, shown in Equation 2.14, is known as the adjoint equation.

$$\frac{\partial J}{\partial x} + \phi^T - \phi^T\left(\frac{\partial F}{\partial x}\right) = 0 \tag{2.14}$$

Solving the adjoint equation for ϕ reduces Equation 2.13 to

$$\delta J = -\phi^T\left(\frac{\partial F}{\partial s}\right)\delta s \tag{2.15}$$

which can be rearranged to obtain the sensitivity gradient shown below.

$$\frac{\partial J}{\partial s} = -\phi^T\left(\frac{\partial F}{\partial s}\right) \tag{2.16}$$

As shown above, once the adjoint equation is solved for ϕ, the gradient can be directly computed. This overall process requires one original PDE solve and one additional solve of the adjoint equation. The computational cost of solving the adjoint equation is roughly the same as the cost of the original PDE. The adjoint method is a powerful tool for obtaining sensitivities of large systems as the cost of obtaining the gradient information is not dependent on the number of control parameters.

There are a number of ways to implement the above process in practice. In the continuous adjoint method, the adjoint equation can be written out in continuous form, then discretized similar to the governing PDE and solved numerically. In the discrete adjoint, the adjoint equation is formulated using the already discretized governing equations. In this research, an automatic differentiation tool is used to compute the discrete adjoint automatically by computing derivatives to functions in the computer code and following a path of dependencies to obtain the sensitivity of one variable to another. This method is explained in the following section.

2.2.3 Automatic Differentiation

Automatic differentiation (AD) is a technique to evaluate the derivative of functions automatically by a computer program using the chain-rule repeatedly. This is possible since all computer programs are sequences of elementary arithmetic operations, such as addition, subtraction, multiplication, and division, and elementary functions, such as log, exp, sin, and cos.

There are two main multiple AD methods: source code transformation and operator overloading. Source code transformation requires minimal changes to the original code as a compiler is used to transform the source code of mathematical operations to a code of automatic differentiation operations. This method can result in faster performance yet is generally more difficult to implement compared to operator overloading. In addition, source code transformation may not be possible in some programming languages, whereas operator overloading is theoretically possible in any language. In operator overloading, each variable is assigned to a new class or object type in which AD can be performed along with the original source code operations.

A Python utility called numpad written by Wang [42] utilizes operator overloading to perform the automatic differentiation to compute the gradients needed for optimization. The numpad tool automatically determines if forward or reverse mode AD should be used based on the dimensionality of the system. The reverse mode is much more efficient when the number of input parameters are greater than the quantity of output variables.

In this research, the reverse mode is used. The source code of interest is the iterative flow solver, which may require a significant amount of iterations to converge. The reverse mode requires the storage of the full primal flow solver simulation operations in memory which could prohibit the calculation of the gradient when many iterations of the flow solver are needed. In order to work-around this limitation, only one iteration of the primal solver is stored and used to compute the gradient after the solution is converged. Knowledge of the adjoint equation is then used to remove dependencies to other input variables iteratively, akin to solving the full reverse mode of a fully stored primal solution, to obtain the true gradient of interest. This low-memory process is shown in detail in Appendix A.

2.3 Turbulence Modeling

2.3.1 Introduction

As mentioned in Chapter 1, there is no general solution to the Navier-Stokes equations. Specific solutions to simple flow problems can be obtained analytically. However, theoretical analysis and prediction of turbulence has been the fundamental problem of fluid dynamics. The major difficulty of turbulent flow is the chaotic nature of turbulence phenomena which also has an infinite amount of scales, or degrees of freedom. Turbulent flow is also rotational and three-dimensional.

Turbulent flow can often be characterized by different sized eddies, or local swirling motion whose characteristic dimension is the local turbulence scale. The range of turbulence scales in the flow is bounded by dimensions of the flow field (such as

the channel width in duct or channel flow) and by the diffusive action of molecular viscosity [37]. This leads to a wide and continuous spectrum of scales. Only at the smallest scales can the turbulent velocity fluctuations be smoothed out by molecular interaction, in which turbulent energy is dissipated into heat. The smallest scales are usually many orders of magnitude smaller than the largest scales, and the ratio of small to large scales decreases rapidly as the Reynolds number is increased. It is observed that kinetic energy is transferred from larger to smaller eddies, creating an energy cascade in which the energy is ultimately dissipated into heat.

The small scale motions occur on a smaller time scale than that of the relatively slow dynamics of the larger eddies, and therefore can be assumed to be independent of the mean flow characteristics [44]. Thus, the rate of energy transferred from larger scale turbulence structures should be equal to the energy dissipated into heat and is key to Kolmogorov's universal equilibrium theory introduced in 1941. As discussed by Tennekes and Lumley [37], the small scale motion is governed by the dissipation rate at which large structures supply energy to the smaller structures, ϵ, and the kinematic viscosity ν. By dimensional analysis, the following length and time scales of the smallest turbulence structures can be formed and are known as the Kolmogorov microscales.

$$\eta \equiv \left(\nu^3/\epsilon\right)^{1/4} \tag{2.17}$$

$$\tau \equiv \left(\nu/\epsilon\right)^{1/2} \tag{2.18}$$

For over a century, a mathematical description of turbulence has been an area of active research. Boussinesq sought to approximate the turbulent stresses by mimicking the molecular gradient diffusion process and introduced the concept of eddy, or turbulent, viscosity in 1877. Many of today's most common turbulence models are built on the Boussinesq approximation.

In 1895, Reynolds used a statistical approach to express all quantities of the governing equations in mean and fluctuating components. This is the basis of the

Reynolds-averaged Navier-Stokes (RANS) modeling approach. The time-averaged momentum equations are identical to the instantaneous equations except for the addition of a new term which is the correlation of fluctuating velocities. This term is known as the Reynolds stress tensor or turbulent stress. In order to compute all mean flow properties, a prescription for computing the Reynolds stress is needed. This is known as the closure problem and is discussed in Section 2.3.2. Many models exist to approximate the Reynolds stress.

The RANS equations can be solved steady-state to approximate the statistically-averaged flow field. RANS simulations model all scales of turbulence. Since all unsteady terms are either removed or approximated by use of Reynolds averaging and closure models, RANS approaches are relatively fast to converge and inexpensive. Thus, RANS modeling has become the most popular method of approximating turbulent flows in industry. The use of mean properties to approximate the Reynolds stress is not a physical but mathematical relationship; therefore this representation of turbulence is liable to produce inaccurate results.

The RANS turbulence models are used to model all scales of turbulence. However, since majority of the turbulent energy is tied to the larger eddies, a more accurate representation of the flow can be obtained by resolving these larger eddies. By low-pass filtering the Navier-Stokes equations the larger scales of turbulence are directly solved, or resolved. The smallest scales, which are the most expensive to solve due to the required spatial discretization, are modeled similar to RANS. This method is known as large eddy simulation and is further discussed in Section 2.3.3. The concept was first introduced by Smagorinsky in 1963 and further explored by Deardorff. The turbulence models for the small scales are known as sub-grid scale (SGS) models. Similar to RANS turbulence modeling, many SGS models exist and provide various levels of accuracy and cost.

The most accurate and expensive form of turbulence simulation is direct numerical simulation (DNS). As the name of the method suggests, all aspects of the flow are derived by directly solving the full unsteady Navier-Stokes equation for all turbulent length scales larger than the Kolmogorov scale. The simulation mesh must be made

fine enough to capture the small scale structures. This requires extremely fine spatial and temporal discretization resulting in substantial increases in computational cost.

An actual comparison between LES and DNS costs on a backward facing step at a Reynolds number of 5100 showed that the LES required 2.7% of the number of grid points needed for the DNS (1.72e3 vs 9.4e6 cells) [17, 25, 44]. This corresponded to computer time of only 2% compared to the DNS. Both methods resulted in similar solutions and had good agreement with experimental data.

A more recent evaluation between LES and DNS performed on a periodic hill at a Reynolds number of 10,600 showed similar grid size differences [1]. The fully wall-resolved LES case required 5.58 million cells, only 2.8% of the 200 million cells required by DNS. A wall-modeled LES case - in which the cells at the wall can be made larger in conjunction with wall functions - further reduced the grid size to 1.18 million cells. The DNS and wall-resolved LES compared well to experimental results. The wall-modeled LES did not compare as well as the wall-resolved LES, but was also reasonably accurate.

DNS has many benefits but is primarily used for research purposes due to the computational costs. Precise details of turbulence parameters can be calculated at any point in the flow. This aids in development and validation of LES and RANS turbulence models. In addition, instantaneous results can be generated that may not be measurable by experimentation. When properly set-up, DNS can be used in place of experimentation for low Reynolds number flows and is often regarded as being as accurate as results of well-designed experiments. In general, both LES and DNS are considered high fidelity CFD methods.

2.3.2 Reynolds Averaged Navier Stokes (RANS)

The RANS equations can be derived by substituting instantaneous terms with mean and fluctuating components, as shown in Equation 2.19 for a velocity component, where the overbar denotes the mean quantity and the prime symbol denotes the fluctuating quantity. The mean quantity can be obtained using spatial, time, or ensemble averaging, depending on the flow. For this paper, time averaging is appropriate since

the turbulent flow field, on average, does not vary with time.

$$u_i(\mathbf{x},t) = \overline{u_i}(\mathbf{x}) + u_i'(\mathbf{x},t) \tag{2.19}$$

The incompressible Navier-Stokes equations are in shown in Equations 2.20 and 2.21,

$$\frac{\partial u_i}{\partial x_i} = 0 \tag{2.20}$$

$$\rho\frac{\partial u_i}{\partial t} + \rho\frac{\partial}{\partial x_j}(u_i u_j) = -\frac{\partial p}{\partial x_i} + \frac{\partial \tau_{ij}}{\partial x_j} \tag{2.21}$$

where t is time, the vectors u_i and x_i are velocity and position, p is pressure, ρ is density, and τ_{ij} is the viscous stress tensor. The viscous stress tensor is defined in Equation 2.22 where μ is the molecular viscosity and s_{ij} is the mean strain-rate tensor, defined in Equation 2.23.

$$\tau_{ij} = 2\mu s_{ij} \tag{2.22}$$

$$s_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \tag{2.23}$$

By substituting in mean and fluctuating terms in place of the instantaneous values, the Reynolds-Averaged Navier-Stokes equations are obtained and shown in Equation 2.24 and 2.25.

$$\frac{\partial \overline{u_i}}{\partial x_i} = 0 \tag{2.24}$$

$$\rho\frac{\partial \overline{u_i}}{\partial t} + \rho\frac{\partial}{\partial x_j}\left(\overline{u_i}\,\overline{u_j} + \overline{u_i'u_j'}\right) = -\frac{\partial \overline{p}}{\partial x_i} + \frac{\partial \overline{\tau}_{ij}}{\partial x_j} \tag{2.25}$$

It can be seen that the RANS equations are identical to the instantaneous Navier-Stokes equations with mean terms in place of instantaneous terms except for a new correlation term $\overline{u_i'u_j'}$. This term is the time-averaged rate of momentum transfer due to the turbulence. Equation 2.25 can be rearranged to move the correlation term

to the right hand side with the viscous stress tensor. Dividing through by density and substituting for the viscous stress tensor, the incompressible RANS momentum equation is rewritten in Equation 2.26.

$$\frac{\partial \overline{u_i}}{\partial t} + \frac{\partial}{\partial x_j}\left(\overline{u_i}\ \overline{u_j}\right) = -\frac{1}{\rho}\frac{\partial \overline{p}}{\partial x_i} + \frac{\partial}{\partial x_j}\left(2\nu\overline{s_{ij}} - \overline{u_i' u_j'}\right) \tag{2.26}$$

The correlation term $-\overline{u_i' u_j'}$ is known as the specific Reynolds stress tensor. In order to compute all mean flow properties, a prescription of the Reynolds stress tensor is required since the fluctuating velocity components do not exist. This is where turbulence modeling comes into play. RANS turbulence models use mean flow values to estimate the Reynolds stress tensor in order to close the equations and provide a numerical solution.

There are many RANS turbulence models that approximate the Reynolds stress with a wide range of computational complexity in existence today. The most popular models are based on the Boussinesq approximation which is shown in Equation 2.27, where ν_t is known as the turbulent, or eddy, viscosity, and k is the turbulent kinetic energy.

$$-\overline{u_i' u_j'} = 2\nu_t\overline{s_{ij}} - \frac{2}{3}k\delta_{ij} \tag{2.27}$$

The turbulence-generated eddy viscosity is modeled to alter the local property of the fluid and mimic the relationship of stress and rate of strain similar to that found in laminar flows. It is important to keep in mind that molecular viscosity is a property of the fluid whereas turbulence is a characteristic of the fluid flow. The family of models that use this approximation are known as eddy viscosity models.

The most popular eddy viscosity models use additional transport equations to calculate the turbulent viscosity. Most industrial RANS codes are based on two-equation models, such as the k-ω and k-ϵ models. Both of these models have a transport equation for the turbulent kinetic energy k. The k-ϵ model also has a differential equation for ϵ which is the turbulence dissipation rate per unit mass. The turbulent viscosity is then obtained using Equation 2.28, where C_μ is one of the

model coefficients. The full k-ϵ model can be found in Wilcox's book on turbulence modeling [44].

$$\nu_t = C_\mu k^2/\epsilon \tag{2.28}$$

The k-ω model has a differential equation for k similar to that of the k-ϵ model. The second differential equation is for ω, which is the specific dissipation rate per unit turbulent kinetic energy. The turbulent viscosity is then obtained using Equation 2.29, where $\tilde{\omega}$ is the dissipation limited by an expression based on the mean strain-rate tensor. The full k-ω model can also be found in Wilcox's book.

$$\nu_t = \frac{k}{\tilde{\omega}} \tag{2.29}$$

In 1994, Mentor introduced a two-equation model that essentially blends the k-ω and k-ϵ models called the k-ω shear stress transport (SST) model [23]. Due to superior near-wall behavior of the k-ω model, the SST model takes the form of the k-ω model in the inner parts of a boundary layer. This also allows the use of the model without the need for wall functions. The SST model behaves more like the k-ϵ model away from the walls and is thought to avoid the sensitivity to free-stream turbulence seen in the k-ω turbulence model. The model is more robust and intended for engineering applications due to improved handling of adverse pressure gradients compared to other inexpensive turbulence models. The k-ω SST model is the baseline RANS turbulence model used in this research.

2.3.3 Large Eddy Simulation (LES)

As mentioned earlier, the larger scale turbulent motions are typically much more energetic than the small scale motions. In order to improve the accuracy of simulations, these large structures must be resolved. The small scale turbulence is relatively weak and has small contribution to the overall Reynolds stress. Due to this, the small scale structures can be filtered out and modeled instead of directly resolving their motion. Since the large eddies are computed, this method is known as Large Eddy Simulation.

Due to the fact that the smallest structures are modeled, the minimum cell size can be made much larger than the Kolmogorov length scale leading to smaller overall cell count and larger allowable timestep sizes. The LES governing equations can be obtained by filtering the Navier-Stokes equations to compute only large scale components of the velocity field. A spatial filter of the velocity is shown in Equation 2.30 for a general filter kernal $G(x, \xi)$,

$$\overline{u_i}(x) = \int G(x, \xi) u_i(\xi) d\xi \tag{2.30}$$

which leads to

$$u_i = \overline{u_i} + u_i' \tag{2.31}$$

where $\overline{u_i}$ is the resolved velocity and u_i' is the sub-filter velocity. There are several common forms of the filter kernel, such as the Gaussian and Fourier cut-off filters. A simple and popular filter kernel is the top-hat or box filter shown in Equation 2.32, which is a local volume average.

$$G(x, \xi, \Delta) = \begin{cases} 1/\Delta^3, & |x_i - \xi_i| < \Delta x_i/2 \\ 0, & \text{otherwise} \end{cases} \tag{2.32}$$

As shown, the filter is also a function of the filter width, Δ. Since the filter size Δ can be associated with the local grid size, the sub-filter quantities are commonly known as "sub-grid" features, regardless of the filter type used. The filtered incompressible Navier-Stokes equations can be obtained using Equation 2.31 to substitute resolved and sub-grid quantities in place of the total (all scale) values. The resulting equation has very similar form to the RANS equations.

$$\frac{\partial \overline{u_i}}{\partial x_i} = 0 \tag{2.33}$$

$$\frac{\partial \overline{u_i}}{\partial t} + \frac{\partial}{\partial x_j}\left(\overline{u_i u_j}\right) = -\frac{1}{\rho}\frac{\partial \overline{p}}{\partial x_i} + 2\nu \frac{\partial}{\partial x_j}\overline{s_{ij}} \tag{2.34}$$

The difficulty arises in dealing with the $\overline{u_i u_j}$ term. Following Leonard [18], the filtered advection term is split into two terms as shown in Equation 2.35.

$$\overline{u_i u_j} = \overline{u_i}\ \overline{u_j} + \tau_{ij}^{\mathrm{SGS}} \tag{2.35}$$

The filtered Navier-Stokes equation can be rewritten as

$$\frac{\partial \overline{u_i}}{\partial t} + \frac{\partial}{\partial x_j}\left(\overline{u_i}\ \overline{u_j}\right) = -\frac{1}{\rho}\frac{\partial \overline{p}}{\partial x_i} + 2\nu \frac{\partial}{\partial x_j}\overline{s_{ij}} - \frac{\partial \tau_{ij}^{\mathrm{SGS}}}{\partial x_j} \tag{2.36}$$

where τ_{ij}^{SGS} is the sub-grid scale stress tensor and is comprised of several components shown in Equation 2.37.

$$\begin{aligned} \tau_{ij}^{\mathrm{SGS}} &= L_{ij} + C_{ij} + R_{ij} \\ &\text{where} \\ L_{ij} &= \overline{\overline{u_i}\ \overline{u_j}} - \overline{u_i}\ \overline{u_j} \\ C_{ij} &= \overline{\overline{u_i} u_j' + u_i' \overline{u_j}} \\ R_{ij} &= \overline{u_i' u_j'} \end{aligned} \tag{2.37}$$

The L_{ij}, C_{ij}, and R_{ij} are known as the Leonard stress, cross-term stress, and SGS Reynolds stress. The Leonard stress can be computed from the resolved scales, but the remaining two terms must be modeled. The most common SGS models model the deviatoric part of SGS stress tensor while using an eddy viscosity model. Smagorinsky first proposed an eddy viscosity SGS model in the form of

$$\tau_{ij}^{\mathrm{SGS}} - \frac{1}{3}\tau_{kk}^{\mathrm{SGS}} = 2\nu_t \overline{s_{ij}} \tag{2.38}$$

$$\nu_t = C_s^2 \Delta^2 \sqrt{\overline{s_{ij}}\ \overline{s_{ij}}} \tag{2.39}$$

where $\overline{s_{ij}}$ is the resolved mean strain rate and C_s is the Smagorinsky constant which is usually set to 0.2. It is known that C_s is not constant and may be a function of the Reynolds number and other parameters dependent on the type of flow [6]. For complex flows, a single constant may not be sufficient. Another issue with the

Smagorinsky model is that the turbulent viscosity does not tend to zero approaching a wall boundary. One work around is to use the Van Driest damping function to reduce the near-wall turbulent viscosity by adjusting the near-wall Smagorinsky constant as shown in Equation 2.40,

$$C_s^2 = C_{s0}^2\left(1 - \mathrm{e}^{-y^+/A^+}\right)^2 \tag{2.40}$$

where $y+$ is the distance from the wall in viscous wall units and $A+$ is a constant usually set to 25. The distance from the wall in wall units is defined in Equation 2.41. The friction, or shear, velocity is defined in Equation 2.42, where τ_w is the shear stress at the wall.

$$y^+ = \frac{yu_\tau}{\nu} \tag{2.41}$$

$$u_\tau = \sqrt{\frac{\tau_w}{\rho}} \tag{2.42}$$

Despite these issues, the Smagorinsky model has been widely used. The inlet boundary condition simulation discussed in Section 3.4 utilized the standard Smagorinsky model with the Van Driest wall damping function.

In 1991, Germano [8] built on the Smagorinsky model by introducing a dynamic model that adjusts the model parameter automatically in different areas of the flow field. The procedure essentially filters the resolved velocity $\overline{u_i}$ using a "test filter" broader than that used in the LES SGS model to obtain a very large scale field $\overline{\overline{u_i}}$. By comparing the two resolved scales, the model parameter can be determined at every spatial point at any timestep, allowing the LES to automatically compute C_s as the simulation evolves. The dynamic model allows the near wall turbulent viscosity to asymptotically approach zero without the use of wall functions. Dynamic models have been shown to be more accurate than the standard Smagorinsky model when compared to DNS [44]. The test filter dynamic modeling concept can be used with any sub-grid scale model and has been adapted to more complicated SGS models.

The dynamic methods add additional computation time due to the test filtering

and computation for the model parameters. In 2004, Vreman introduced an eddy viscosity model suitable for engineering applications that is computationally similar to the Smagorinsky model. The model has been shown to be more accurate than the Smagorinsky model and as good as the standard dynamic model for turbulent mixing layers and turbulent channel flow [40]. The model uses the filter width and first-order derivatives of the resolved velocity field and has the ability to reduce the turbulent viscosity near the wall without the use of wall functions. Unlike the Smagorinsky model, the Vreman SGS model is also able to handle transitional flows.

Due to the improvements over the Smagorinsky model without significant additional computational cost or wall functions, the Vreman SGS model is used for all U-bend simulations in this thesis.

2.3.4 Low Fidelity (RANS) Training

In this research a simple eddy viscosity model based on a prescribed turbulent viscosity scalar field is introduced. There are no additional transport equations and thus this eddy viscosity model can be considered a zero-equation model. The last term in the Boussinesq approximation from Equation 2.27 is ignored because k is not readily available, similar to the Spalart-Allmaras turbulence model. The Reynolds stress tensor approximation is shown in Equation 2.43.

$$-\overline{u_i' u_j'} = 2\nu_t \overline{s_{ij}} \tag{2.43}$$

Substituting Equation 2.43 into the incompressible RANS equation produces Equation 2.44. It can be seen that the turbulent viscosity is treated in similar fashion as the kinematic viscosity, which creates an effective viscosity shown in Equation 2.45.

$$\frac{\partial \overline{u_i}}{\partial t} + \frac{\partial}{\partial x_j}\left(\overline{u_i}\ \overline{u_j}\right) = -\frac{1}{\rho}\frac{\partial \overline{p}}{\partial x_i} + \frac{\partial}{\partial x_j}\left(2(\nu + \nu_t)\overline{s_{ij}}\right) \tag{2.44}$$

$$\nu_{eff} = \nu + \nu_t \tag{2.45}$$

The turbulent viscosity is treated as a scalar field and can have a unique value for each cell of the mesh. The turbulent viscosity is obtained through optimization with a higher fidelity simulation. The high fidelity simulation output mean velocity field is the desired quantity to match in the lower fidelity RANS simulation. Therefore, the objective function is based on the mismatch between the velocity fields of the two simulations and the optimization is to minimize the objective function by adjusting the spatial turbulent viscosity field. The objective function is shown in Equation 2.46.

$$J(u(\nu_t)) = \|u(\nu_t) - u_{\text{HiFi}}\|^2_{\text{L2}} \tag{2.46}$$

The minimization can then be written as

$$\min \|u(\nu_t) - u_{\text{HiFi}}\|^2_{\text{L2}} \qquad \text{s.t.} \quad \nu_t \geq 0. \tag{2.47}$$

For physical reasons, the turbulent viscosity must be non-negative, which is equivalent to non-negative turbulent kinetic energy and dissipation rate. Following Dow [4], the optimization can be simplified by updating $\log(\nu_t)$. This leads to an unconstrained optimization since $\log(\nu_t)$ automatically enforces the non-negativity requirement. The L-BFGS optimization algorithm is used in concert with gradient information obtained by the adjoint method. Automatic differentiation is used to obtain the sensitivity gradient. The gradient used to drive the optimization is calculated by simply multiplying the turbulent viscosity by the adjoint sensitivity gradient as shown in Equation 2.48

$$\frac{\partial J}{\partial \log(\nu_t)} = \nu_t \frac{\partial J}{\partial \nu_t} \tag{2.48}$$

The training optimization is demonstrated on a 180° U-bend channel in Section 4.3 with time-averaged LES as the high-fidelity solution.

Chapter 3

Simulation of a 180° U-bend Square Duct

3.1 Introduction

As discussed in Chapter 1, the square duct 180° U-bend is an approximation of turbine airfoil serpentine cooling passages. This geometry has also been selected due to existing experimental data by Coletti and Verstraete [3]. The objectives are to highlight areas where RANS methods break-down and show why LES can be considered an accurate representation. This supports the current motivation, which is to use high fidelity simulations, such as LES, to guide designs containing complex internal flows. This chapter discusses the details of the LES and RANS simulations and compares the solutions to experimental data to validate the use of LES to capture the proper fluid dynamics.

3.2 Large Eddy Simulation

3.2.1 Fluid Solver

The large eddy simulation results are generated using OpenFOAM's pisoFoam incompressible LES solver [26]. This solver is based on the "pressure-implicit with splitting

of operators" (PISO) algorithm developed by Issa [14]. This pressure-based algorithm is suitable for unsteady simulations and can maintain a stable calculation without the use of under-relaxation.

A sub-grid scale (SGS) model is implemented based on the Vreman eddy-viscosity SGS model [40]. The Vreman SGS model has desirable properties and is well-suited for engineering applications. One such property is that the sub-grid turbulent viscosity tends to zero approaching a wall without the use of wall damping functions. Another desirable property is that the model is less complicated than dynamic models and similar in complexity to the standard Smagorinsky SGS model. On several test cases the Vreman model has been shown to provide more accurate results over the standard Smagorinsky model and is comparable to the accuracy obtained with dynamic Smagorinsky models [40, 45].

The SGS filter cut-off used for the LES is OpenFOAM's cubeRootVolDelta. This is a top-hat filter based on the inverse of the cube root of the local cell volume and is a common filter used in finite volume implementations of LES.

The divergence scheme is based on a second order limited linear differencing method. The gradient and Laplacian operators utilize a second order central differencing Gaussian integration method. All interpolation is based on second-order central differencing. The time discretization scheme is second order implicit backward Euler.

The momentum equations are solved using OpenFOAM's preconditioned bi-conjugate gradient (PBiCG) solver with a diagonal incomplete LU (DILU) preconditioner. The same solver is used for SGS turbulence parameters. It was found that the most effective method of solving the pressure correction equation was with the use of the generalized geometric-algebraic multigrid (GAMG) solver with a diagonal-incomplete Cholesky (DIC) smoother.

The bulk velocity, U_b, through the duct and the fluid density are set to unity and the Reynolds number of 40,000 is achieved by setting the kinematic viscosity to 2.5e-5. Boundary conditions used in the simulation are shown in Table 3.1.

$$Re = \frac{\rho U_b d_h}{\mu} = \frac{U_b d_h}{\nu} \tag{3.1}$$

Table 3.1: Boundary conditions for LES duct

Parameter	Inlet	Outlet	Walls
velocity, u	mapped	zero-gradient	zero
pressure, p	zero-gradient	zero	zero-gradient
SGS turbulent kinetic energy, k_{SGS}	0.004	zero-gradient	zero
SGS turbulent viscosity ν_{SGS}	zero-gradient	zero-gradient	zero-gradient

The inlet velocity boundary condition is mapped from an unsteady pre-simulation and is discussed in detail in Section 3.4. The sub-grid scale turbulent kinetic energy of 0.004 at the inlet corresponds to 5% turbulence intensity converted to turbulent kinetic energy based on the fully developed smooth duct relation shown in Equation 3.2 below, where I is the turbulence intensity. A 5% turbulence intensity was measured experimentally in Coletti and Verstraete's rig [3].

$$k = \frac{3}{2}(U_b I)^2 \tag{3.2}$$

The flow field is initialized with zero velocity. The simulation is run sufficiently long to ensure well converged time-averaged fields. The initial start-up transient is not included in the time averaging. The time-averaging technique is explained in detail in Section 3.2.3.

3.2.2 Spatial and Temporal Discretization

The duct cross-section is a square with a side length and thus hydraulic diameter, d_h, of 1. The domain consists of a straight inlet section of 5 d_h. The bend section has an inner radius of 0.26 d_h. The straight exit section has a length of 6 d_h. Figure 3-1 shows multiple views of the domain.

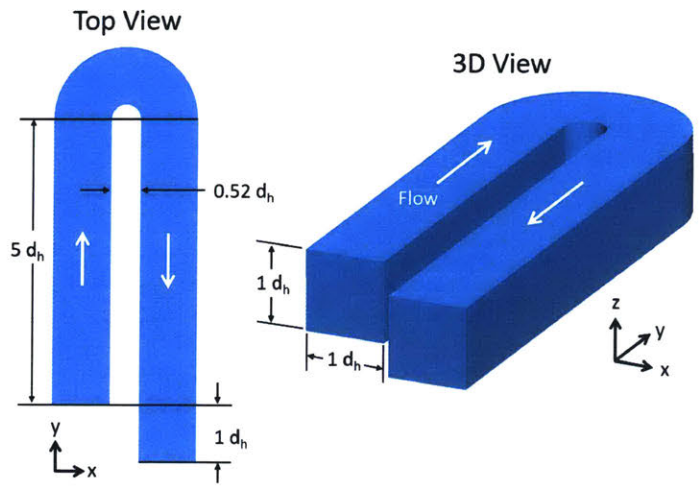


Figure 3-1: Domain geometry used for square duct U-bend LES

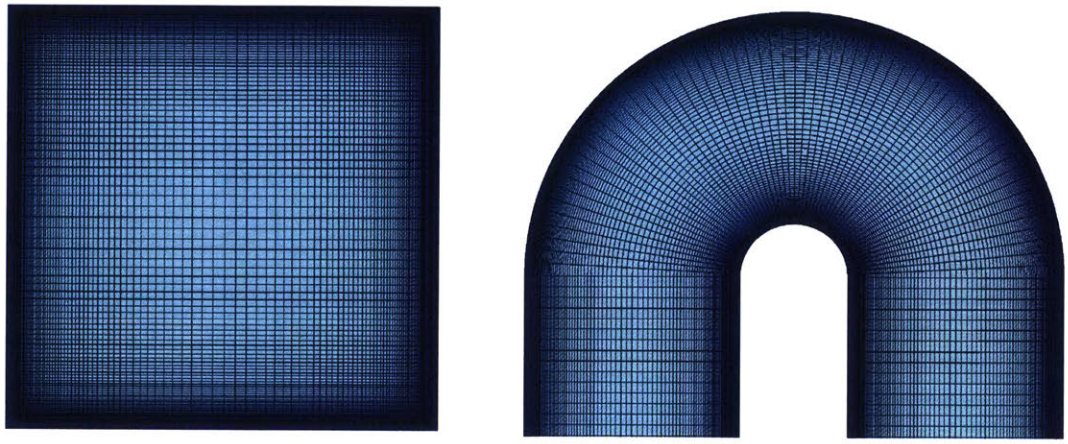
A structured hex mesh of size 150 x 150 x 265 is created using OpenFOAM's blockMesh utility. The cell sizes are selected to meet LES best practices. In order to sufficiently resolve the wall velocity profile, a y+ of 1, or one wall unit from the wall boundary, is the first cell target size. It is also desirable to have 3 cells within 5 wall units to resolve the viscous sublayer at the walls. The wall unit, defined in Equation 3.3, is based on the mean flow solution in the inlet section of the U-bend where τ_w is the wall shear, ρ is the fluid density, and ν is the fluid kinematic viscosity.

$$\text{wall unit} = \frac{\nu}{u_\tau} = \nu\sqrt{\frac{\rho}{\tau_w}} \tag{3.3}$$

A graded mesh is used in the spanwise direction from all four walls. A grading ratio (ratio of largest to smallest cell) of 50 over a span of 75 cells results in a cell expansion factor of 1.054, below the target maximum of 1.15. The largest spanwise cell size in the middle of the duct is approximately 26 wall units square. The resulting spanwise cell count is 150 x 150.

The cell size in the streamwise direction is approximately 50 wall units in the straight sections. The streamwise cell size in the bend section ranges from 13 to 66 from the inner to the outer wall. The last d_h of the exit section consists of a sponge region. The sponge region contains expanding cells in the streamwise direction which allow for out-going flow features to leave the computation domain without reflecting

any signature back into the domain. A grading ratio of 5 is used to produce a cell expansion factor of 1.53. The total streamwise cell count is 265, with 100 cells in each straight section, 60 cells in the bend section, and 5 cells in the sponge region. The overall mesh cell count is 5,962,500. Multiple views of the mesh are in shown in Figures 3-2 and 3-3.



(a) Spanwise mesh topology (b) top view of mesh in bend region

Figure 3-2: Views of the U-bend mesh

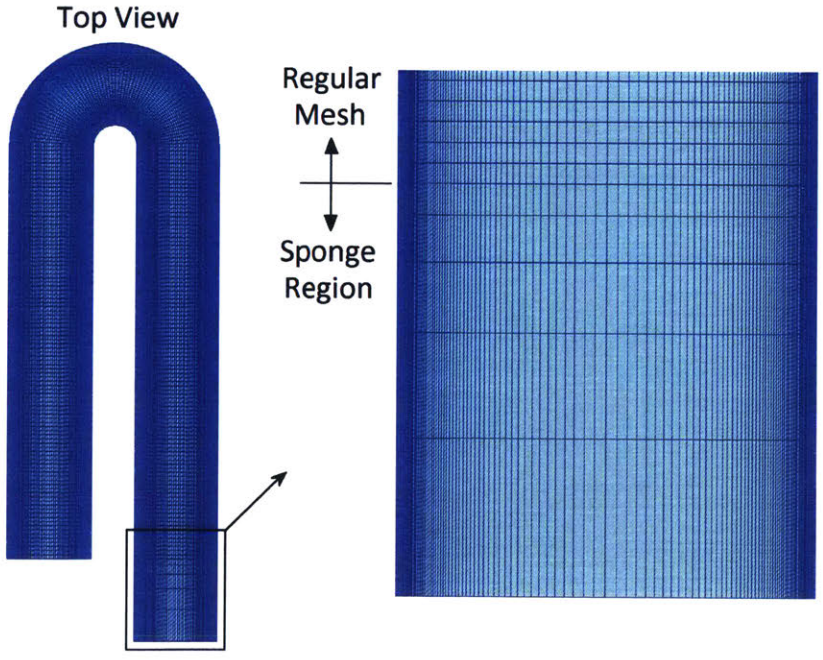


Figure 3-3: Top view of LES mesh sponge region

The mesh results in an average first cell size of approximately y+ = 0.5 and the

first three cells are within a y+ of 4 for majority of the duct. As shown in Figure 3-4, the largest instantaneous y+ is near the start of the bend at the inner wall where y+ reaches approximately 2 to 3 due to the local acceleration.

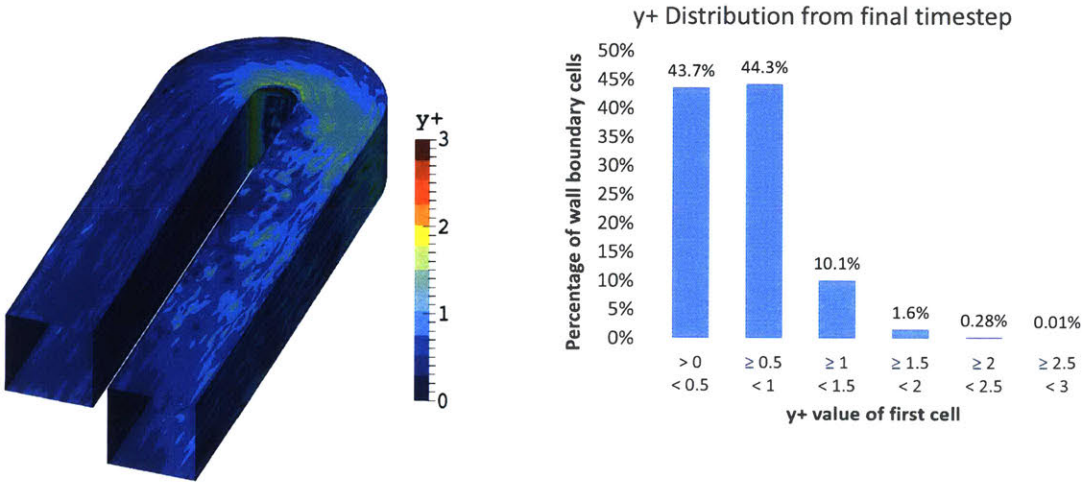


Figure 3-4: Instantaneous y+ from final timestep of LES

It is worth noting that as part of a mesh sensitivity study, an alternate mesh was made roughly twice as coarse and compared to the fine mesh. The coarse mesh had a size of 76 x 76 x 210, or about 1.2 million cells, for an 80% reduction in overall size. The coarse mesh solution displayed excess low-momentum fluid at the outer wall, increased post-bend velocities, and well over-predicted turbulent kinetic energy in the bend region. The fine mesh compared well to available experimental data and the results are further discussed in Section 3.6.

The spatial discretization is performed first, then the timestep size, Δt, is set to ensure accuracy and stability. The CFL, or Courant-Friederichs-Levy, number is shown for a one dimensional case in Equation 3.4 and is often used as a check of the stability. If the CFL number is larger than 1, fluid particles move across multiple cells in one timestep and can cause instability for explicit schemes. In this case, despite the use of an implicit scheme which can handle a CFL > 1, it is desired to have the maximum CFL number around unity.

$$\mathrm{CFL} = u\frac{\Delta t}{\Delta x} \tag{3.4}$$

A constant timestep size of $\Delta t = 0.001$ seconds is used for the simulation. The instantaneous CFL number is calculated for all cells at every timestep. The peak CFL number occurs at the inner wall of the bend due to high fluid velocity and the small cell size. On average, the max instantaneous CFL number at each timestep is approximately 1.6, but does periodically reach values as large as 2.2. The full domain average CFL is approximately 0.04.

3.2.3 Mean Field: Time-Averaging

The initial condition for the flow field is set to zero velocity, creating a start-up transient. Time-averaging from the start of the simulation would lead to erroneous results due to the start-up. Therefore, monitors are used throughout the domain to track local flow parameters and determine when time-averaging can begin. Figure 3-5 shows the monitoring point locations on a slice at $z/d_h = 0.5$ and the associated data from the first 30 seconds of the simulation. Figure 3-6 shows the velocity field during the first 15 seconds of the start-up.

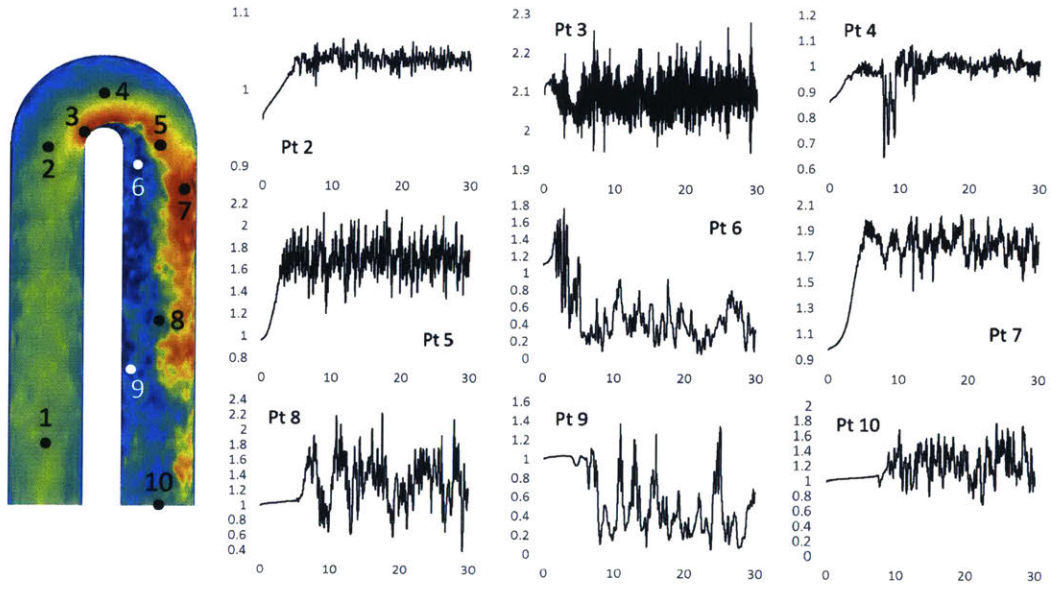


Figure 3-5: Instantaneous velocity monitor point data over the first 30 seconds of simulation

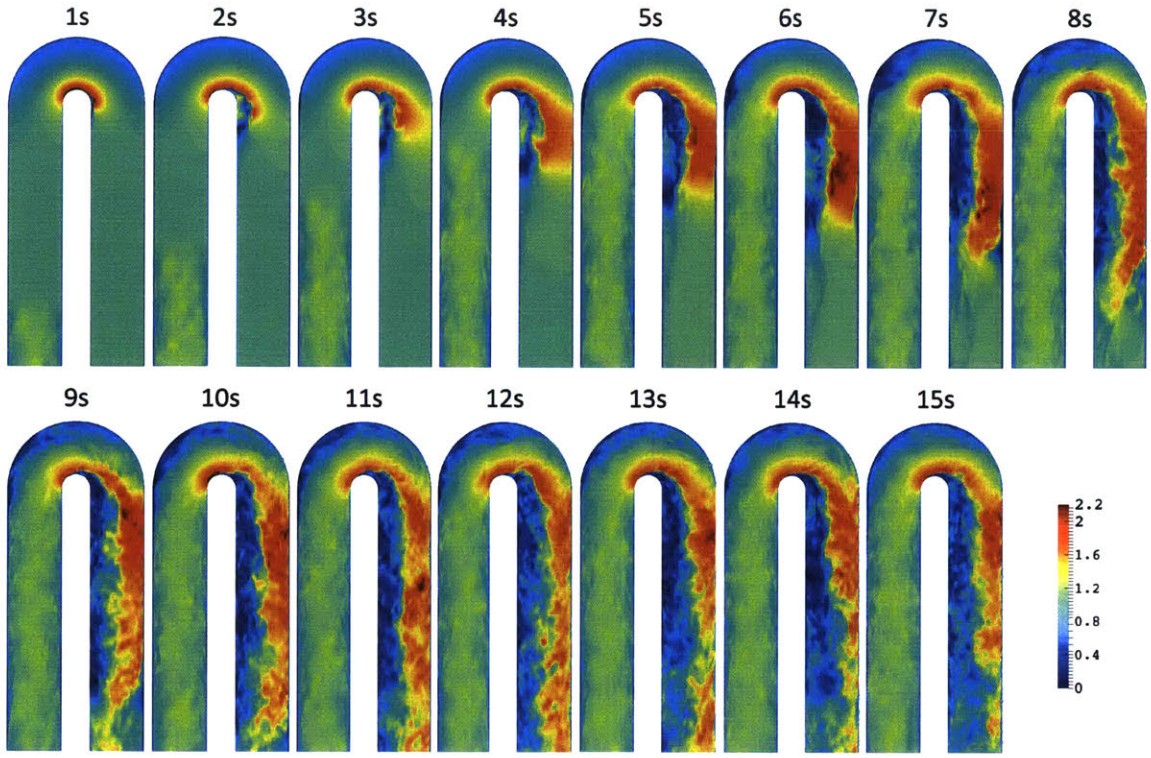


Figure 3-6: Instantaneous velocity magnitude at 1 second intervals during start-up, shown at $z/d_h = 0.5$

The start-up transient is mostly limited to a single flow-through unit (FTU), or about 12 seconds at all points in the domain. One FTU is the time needed for the fluid to flow through the whole domain and is based on the overall mean duct length and the bulk velocity. Time averaging is started after 20 seconds, or 1.6 FTU, to completely avoid the start-up.

Again, monitors are used to evaluate the quality of the mean field over time. Figure 3-7 shows the time-averaged local velocity magnitude normalized by the local mean value at the end of the full simulation. It can be seen that a reasonable mean field is obtained after approximately 10 FTU, but at least 17 FTU are needed to obtain a mean field within 0.5%. Figure 3-8 shows the mean field at different intervals of time-averaging and changes are most visually noticeable in the post-bend region. A time-averaging duration of 230 seconds (18.6 FTU) is used to obtain a mean field. Mean flow parameters include the velocity components, pressure, Reynolds stress components, sub-grid turbulent kinetic energy, and sub-grid turbulent viscosity.

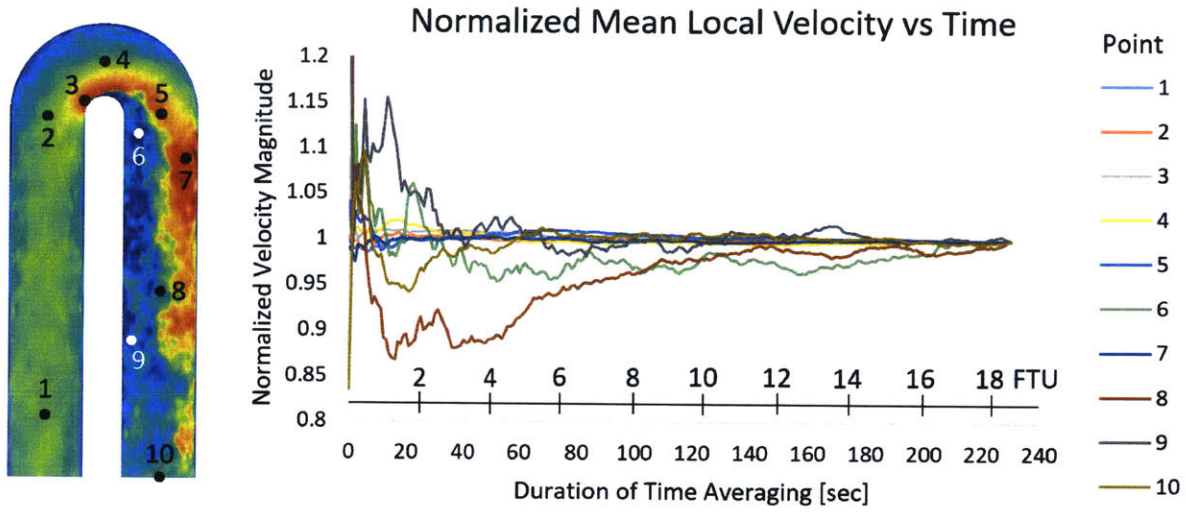


Figure 3-7: Normalized local mean velocity as a function of time-averaging duration

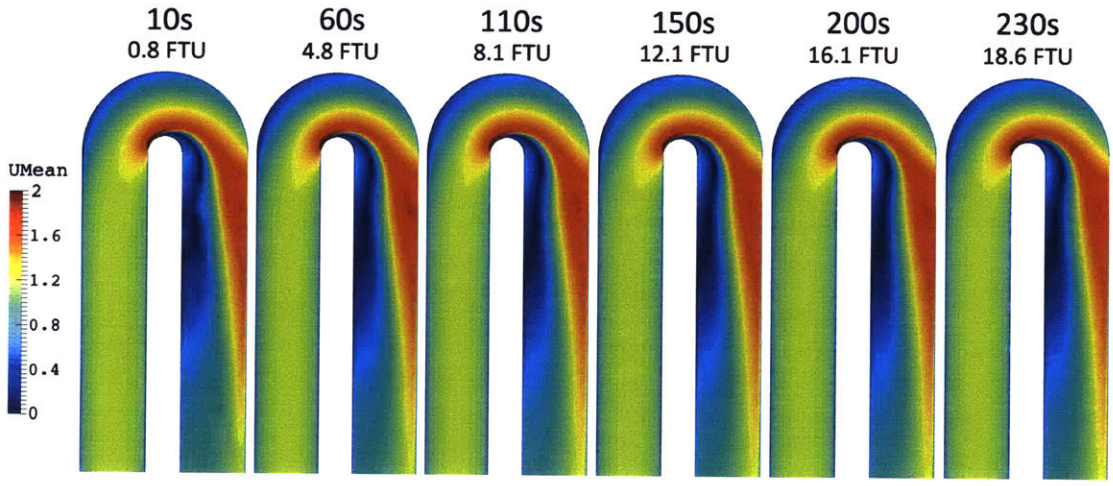


Figure 3-8: Mean velocity field after different durations of time-averaging, shown at $z/d_h = 0.5$.

3.2.4 Parallel Computing

The domain is decomposed to 32 CPU cores using OpenFOAM's decomposePar function. The built-in OpenFOAM "scotch" utility is used as the decomposition method to minimize the number of subdomain interfaces, and thus minimize the communication between the separate subdomains. The simulation is conducted on MIT's Voyager cluster in the Aerospace Computational Design Lab. The full 250 second simulation, including the start-up transient, requires about 9 days to complete. Once

the simulation is finished, the full domain is reconstructed for all output timepoints and copied to a local desktop computer for further post-processing.

3.3 RANS Simulation

3.3.1 Fluid Solver

The RANS simulation results are generated using OpenFOAM's simpleFoam steady-state incompressible solver. This solver is based on the "semi-implicit method for pressure linked equations" (SIMPLE) algorithm developed by Patankar and Spalding [27, 28]. This iterative pressure-based algorithm is widely used in industrial codes. Due to the strong coupling between pressure and velocity, some under-relaxation on the pressure and velocity correction terms is needed for stability. The OpenFOAM default of 0.3 and 0.7 are used for the pressure and velocity, respectively.

The k-ω shear stress transport (SST) turbulence model already available in OpenFOAM is used for the RANS simulation. As mentioned in Section 2.3.2, this two-equation model is robust for wall bounded flows and thus has become increasingly popular in industrial codes.

A bounded upwind differencing method is utilized for convection terms and a central differencing method is used for diffusion terms. The gradient and Laplacian operators are based on a second order central differencing Gaussian integration method. All interpolation is based on second-order central differencing. The momentum equations are solved using OpenFOAM's Gauss-Seidel iterative solver. The same solver is used for the k and ω transport equations. Similar to the LES, the pressure correction equation is solved with a generalized geometric-algebraic multigrid (GAMG) iterative solver with a diagonal-incomplete Cholesky (DIC) smoother.

3.3.2 Simulation Details

The LES mesh topology is also used for the RANS simulation. The domain is identical to that of the LES domain, except for the elimination of the sponge region. The overall

mesh is of size 150 x 150 x 260, which is more than adequate for the RANS method. The kinematic viscosity, density, and bulk velocity from the LES are used to maintain a Reynolds number of 40,000.

The boundary conditions are similar to those used in the LES and are shown in Table 3.2. One difference is that now the turbulent kinetic energy and turbulent viscosity apply to all scales since the turbulence is fully modeled. The wall boundary TKE is set to a very small number instead of zero to prevent any divide by zero errors. The specific dissipation is initially set to unity, but is solved throughout the domain through its own transport equation. OpenFOAM's omega wall function is used to calculate the near-wall behavior. The turbulent viscosity is calculated everywhere in the domain using the TKE and specific dissipation.

Table 3.2: Boundary conditions for RANS duct simulation

Parameter	**Inlet**	**Outlet**	**Walls**
velocity, u	mapped	zero-gradient	zero
pressure, p	zero-gradient	zero	zero-gradient
turbulent kinetic energy, k	0.004	zero-gradient	$1e-11$
specific dissipation, ω	uniform 1	uniform 1	omegaWallFunction
turbulent viscosity, ν_t	calculated	calculated	calculated

The velocity inlet boundary condition is mapped from the time-averaged outlet solution from the same pre-simulation used for the LES. The flow field is initialized with a uniform value of zero. After 6000 iterations, the velocity residual is converged to a value of approximately 1e-5 for all components, and the pressure residual is stabilized at a value of approximately 1e-4. The simulation is run on a single core on a local computer and requires approximately 40 hours to complete. Minimal post-processing is needed since the final iteration of the steady state simulation corresponds to the mean solution.

3.4 Pre-simulation for Unsteady Inlet Boundary Condition for LES

3.4.1 General Approach

In order to provide the proper level of turbulence and unsteadiness in the inlet section of the duct, a LES pre-simulation is used. The pre-simulation consists of a periodic straight square duct with the same spanwise mesh topology as the full U-bend simulation. The pre-simulation is initialized with a RANS flow solution and is run sufficiently long to generate realistic turbulent structures. The timestep size matches that of the full U-bend simulation to allow for direct mapping of the outlet solution from the pre-simulation onto the inlet of the full U-bend. The pre-simulation is run beforehand and the output is stored at every timestep to generate a database of inlet velocity fields. This method ensures an inlet that is unsteady, has the proper level of turbulence intensity and structure, and satisfies the Navier-Stokes equation. This is particularly important as the dynamics in the inlet section of the duct can impact the behavior of the flow in the bend region and beyond. This was noticed when comparing the U-bend simulations with the unsteady, mapped inlet to that of a uniform velocity inlet boundary condition. Figure 3-9 shows the general procedure of mapping to the U-bend inlet.

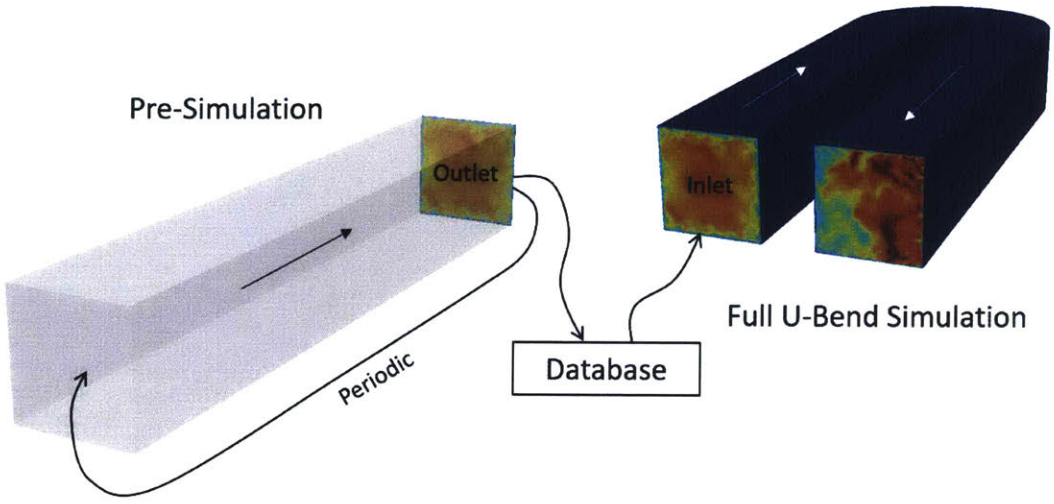


Figure 3-9: Pre-simulation method to create an unsteady inlet boundary condition.

3.4.2 Turbulence Model and Key Parameters

The inlet pre-simulation is run using OpenFOAM's pisoFoam LES solver with the standard Smagorinsky sub-grid scale model. In order to drive the sub-grid scale turbulent viscosity to zero at the walls, the Van Driest function is utilized. The bulk velocity is set to unity and the kinematic viscosity is set to 2.5e-5 to obtain a Reynolds number of 40,000, similar to the full U-bend simulation.

As mentioned, the pre-simulation is initialized from a RANS square duct solution. Any reasonable initial condition can be used, such as $u_x = 1$ in all cells. Since the Reynolds number is large enough, the initially steady flow is able to become unstable and create turbulent structures. The structures start out as numerical fluctuations and grow to fully turbulent flow. The simulation is monitored to observe when the flow field has statistically converged to turbulent duct flow. The RANS solution initialization is used as it already has a velocity profile that is close to the time-averaged solution, and thus can result in a usable solution in less time. The pre-simulation is then restarted and the outlet velocity data is then extracted at every timepoint.

3.4.3 Domain and Mesh

The pre-simulation domain consists of a square duct with a side length of 1 d_h and a duct length of $2\pi d_h$. The same spanwise mesh topology of 150 x 150 cells from the full U-bend simulation is selected for direct mapping to avoid interpolation errors. For the streamwise direction, 125 cells are used to provide the same cell size as the full U-bend simulation. The overall cell count is 2,812,500 cells, just under half of the full U-bend simulation.

The periodic inlet-outlet boundary condition (BC) for the pre-simulation is created by continuously mapping the velocity outlet solution to the inlet. The inlet velocity field at each timestep is forced to meet a bulk velocity of unity and is used to drive the simulation instead of specifying a pressure gradient.

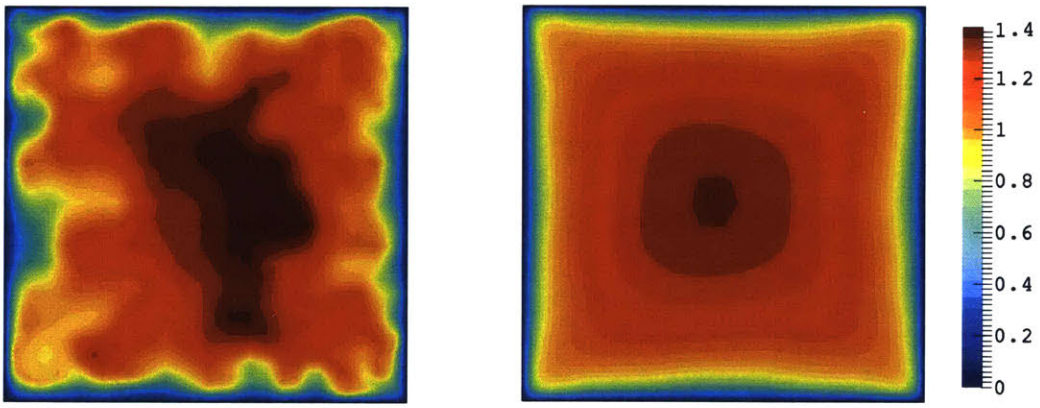
3.4.4 Database and Inlet Mapping

The timestep size of 0.001 seconds is used to match that of the full U-bend simulation. The pre-simulation is run for 250 seconds, or 250,000 timesteps. At each timestep, the nodal velocity components are stored in a text file. The surfaceSampling function in the controlDict file is used to save the velocity components from the outlet plane. The data is stored for the 22,801 points in the square section (151 x 151). Each file size consists of about 730 kB. The entire 250 second database requires a total storage of approximately 180 GB.

The full U-bend simulation is set-up to read from the database and select the proper file to use for the inlet velocity BC. This is achieved using the timeVaryingMappedFixedValue boundary condition. Reading from the inlet database increases the full U-bend simulation duration by approximately 6% compared to a fixed inlet BC.

3.4.5 Square Duct Result Comparisons to Previous Research

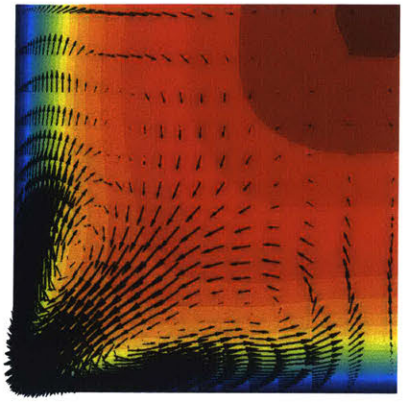
The pre-simulation is also time-averaged to explore features of the flow-field. Figure 3-10 shows the instantaneous and time-averaged outlet velocity magnitude on the same scale. The mean peak, or centerline velocity U_c, is approximately 1.3 times the bulk velocity, U_b. One notable observation is that the secondary velocity produced due to the presence of the corners pushes mean axial flow towards the corner along the corner bisectors. This creates the bulging of the mean axial flow toward the corners. This behavior is seen in previous simulations and experiments of square channels and is explained in detail by Gessner [9].



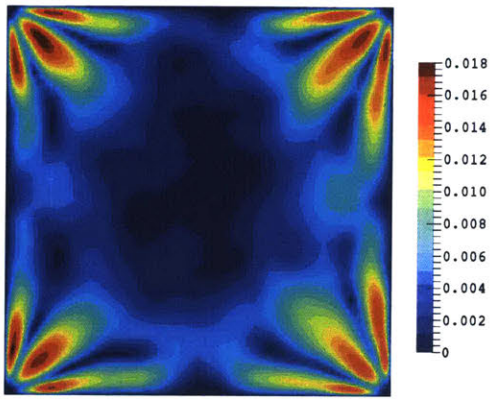
(a) instantaneous velocity (b) mean velocity

Figure 3-10: Pre-simulation outlet plane velocity field

Figure 3-11 shows the secondary flow behavior. Figure 3-11a has the vectors of the in-plane flow overlaid on the mean axial flow contours. As discussed, the secondary flow travels from the center part of the section towards the corners. The secondary flow then travels along the walls and recirculates back towards the center of the duct. This same recirculation was reported by Huser and Biringen [13], Gavrilakis [7], and Madabhushi and Vanka [21]. Figure 3-11b shows the absolute value of the secondary velocity magnitude normalized by the mean centerline velocity. The peak secondary velocity is approximately 1.8% of the centerline velocity along the corner bisector. The peak along the wall is approximately 1.6% of U_c.



(a) secondary flow vectors plotted on mean axial flow for 1/4 section



(b) secondary flow normalized by centerline velocity

Figure 3-11: Pre-simulation outlet plane secondary velocity

The secondary flow contours and the ratio to the centerline velocity compare well to other square duct analyses. The mean flow characteristics are compared in Table 3.3, in order by the bulk Reynolds number. The current analysis shows a centerline velocity and secondary velocities slightly larger than expected based on the comparable Reynolds number experiments of Melling [22] and Hoagland [12]. Aside from the current analysis, the table indicates that there may be a weak function of U_c/U_b to the Reynolds number.

Table 3.3: Square duct mean flow comparison

Reference	Re_b	U_c/U_b	$\max(U_{sec}/U_c)$
Gavrilakis, 1992, DNS	4,410	1.33	1.4%
Madabhushi and Vanka, 1991, LES	5,810	1.28	2%
Huser and Biringen, 1993, DNS	10,600	–	–
Current Analysis, LES	40,000	1.30	1.8%
Melling and Whitelaw, 1976, Experiment	42,000	1.24	1%
Hoagland, 1960, Experiment	43,800	1.23	1.5%

The outlet plane average turbulence intensity from the simulation is approximately 4.2% which is close to the pipe flow correlation result of 4.25% for a Reynolds number of 40,000 and also compares well to the 5% measured experimentally for the square duct at the same Reynolds number.

3.5 U-bend CFD Results

Due to the 3D nature of the geometry, the results are shown on 2D slices throughout the domain. Figure 3-12 shows a comparison of the mean velocity magnitude from the RANS simulation and the time-averaged LES. The slice is taken at half the height of the duct in the z direction, or $z/d_h = 0.5$. This is also the plane of symmetry. The peak velocity at the bend is similar between the two simulations. The LES and RANS results show separation at approximately the same location along the inner wall of the bend, denoted by the letter "S". The separated and recirculating flow from the RANS solution seems to penetrate into the duct further from the wall at all points. The flow eventually reattaches downstream in the straight section of the duct. The estimated reattachment location is noted in the figure by the letter "R". The RANS solution is showing a reattachment point further from the bend.

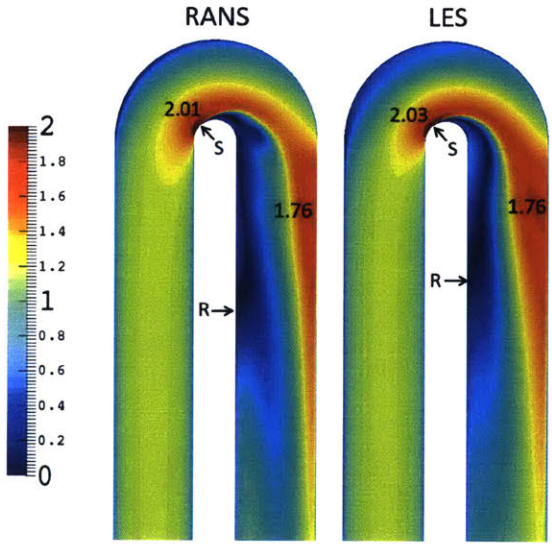


Figure 3-12: Mean velocity magnitude RANS vs LES comparison at z/d_h=0.5

Since the flow separates and cannot follow the contour of the bend, the flow essentially impinges onto the outer wall just beyond the exit of the bend region. This leads to a local velocity increase as the flow is redirected toward the duct exit. The LES and RANS solutions show a similar local velocity magnitude peak in this area, however, there is a difference in the location of the local maximum. The LES peak is approximately 0.2 d_h from the wall, whereas the RANS peak is just under 0.1 d_h from

the wall. Downstream, the inner wall low velocity region migrates towards the middle of the duct in the RANS solution. The LES solution shows increasing uniformity as expected, with a velocity gradient from the outer wall to the inner wall.

Figure 3-13 shows two cross-sections through the duct in the bend region for the instantaneous and mean velocities. Section A is taken half way into the bend. At this point, both the LES and RANS solutions show separation at the inner wall. Both simulations show the largest amount of separation is at the plane of symmetry. The RANS separation zone height at Section A is smaller than that of the LES but extends further into the duct. Both the LES and RANS show peak velocities at the inner wall, close to the corners with the top and bottom wall. Section B shows a larger difference between the two solutions. The instantaneous LES solution hints at two counter-rotating vortices in the recirculation zone at the inner wall. There is a clear plane of symmetry in the instantaneous and mean velocity field. Again, the separation zone is largest at the plane of symmetry. The RANS solution shows an interesting structure where the low velocity region penetrates further into the duct and mushrooms. The gradient near the outer wall is similar between the two simulations.

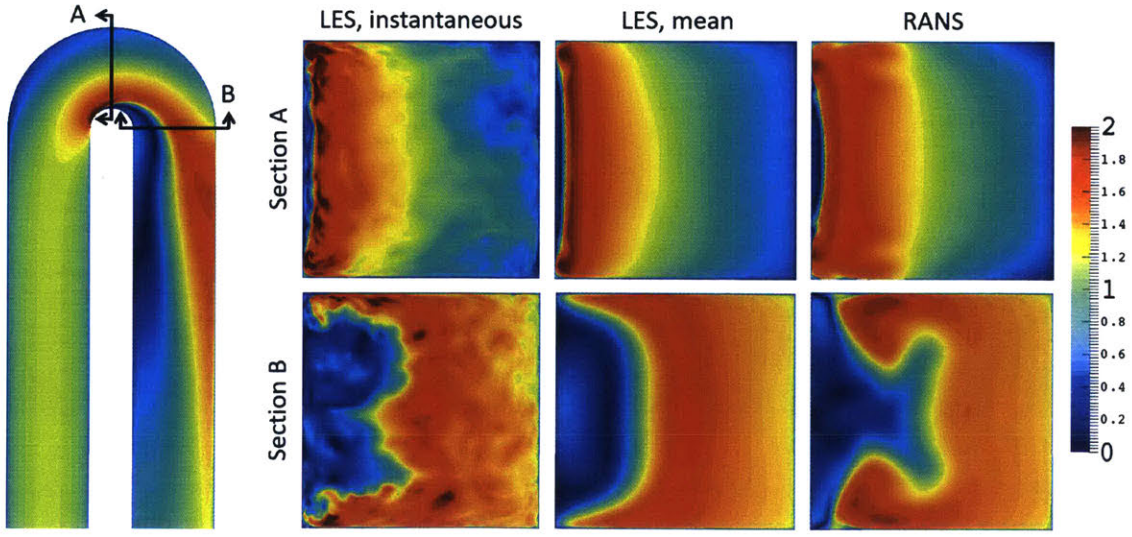


Figure 3-13: Velocity comparison at two bend sections

Two sections downstream of the bend were looked at in detail to investigate the complex flow structures that are created. These sections are shown in Figure 3-14, which have the in-plane, or secondary flow, velocity vectors overlaid onto the mean velocity field. Section C is taken at 0.44 d_h from the top of the inner bend, which corresponds approximately to the center of the recirculation zone from the LES solution in the $z/d_h = 0.5$ slice. Section D is taken half-way into the exit section, or 2.5 d_h from the bend exit. Section C is similar in nature to Section B, and the secondary flow in the separation zone can be seen. The LES has the center of these vortices much closer to the walls than the RANS simulation. These weaker secondary flow vortices are part of the recirculation due to the separation from the bend, and propagate upstream. The outer high velocity region also shows two counter-rotating vortices propagating downstream that pushes flow towards the outer, top, and bottom walls.

Section D is taken beyond the reattachment point. The full section has streamwise flow out of the page, or downstream. The two outer vortices from Section C have grown to fill the whole domain. This secondary flow generation can be explained by convection of the boundary layer vortex lines present upstream which are normal to the flow. As described by Greitzer, Tan, and Graf [11], these vortex lines become stretched and skewed due to differential velocity around the bend to produce streamwise components. Smaller vortical structures can be seen in each corner due to the sharp corners. These corner secondary flows are also counter-rotating compared to the large vortices.

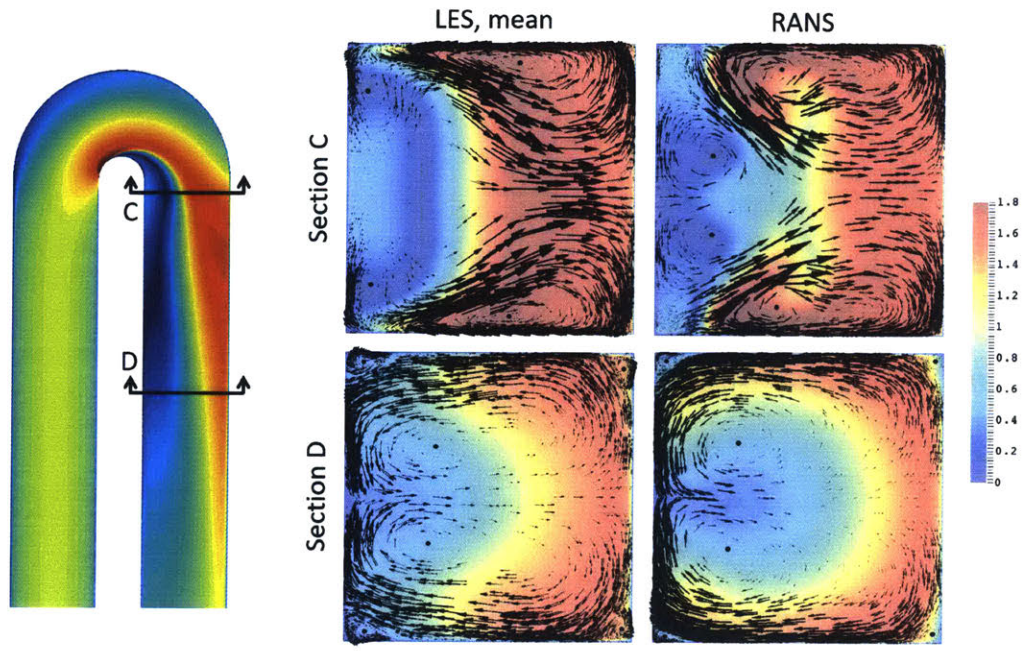


Figure 3-14: Secondary flow vectors overlaid on mean velocity at two post-bend sections

The total turbulent kinetic energy from the LES is obtained by combining the resolved and sub-grid TKE. The sub-grid TKE is calculated at every timestep using the sub-grid model and time-averaged similar to the velocity field. The resolved TKE is one half of the trace of the Reynolds stress tensor. The Reynolds stress tensor terms are also calculated at every timestep and time-averaged. The full TKE is calculated using Equation 3.5.

$$\text{TKE} = \text{TKE}_{SGS} + \text{TKE}_{resolved} = \text{TKE}_{SGS} + \frac{1}{2}\left(\overline{u'^2} + \overline{v'^2} + \overline{w'^2}\right) \tag{3.5}$$

Figure 3-15 shows the mean TKE from the LES solution. The sub-grid scale TKE has similar contours to the resolved TKE, but is an order of magnitude or more smaller than the resolved TKE. Figure 3-15 also shows the ratio of the sub-grid scale TKE to the resolved TKE. Only a portion of the bend region shows over 10% contribution from the sub-grid scale model. This is in a region with low TKE and is another indication of adequate grid size.

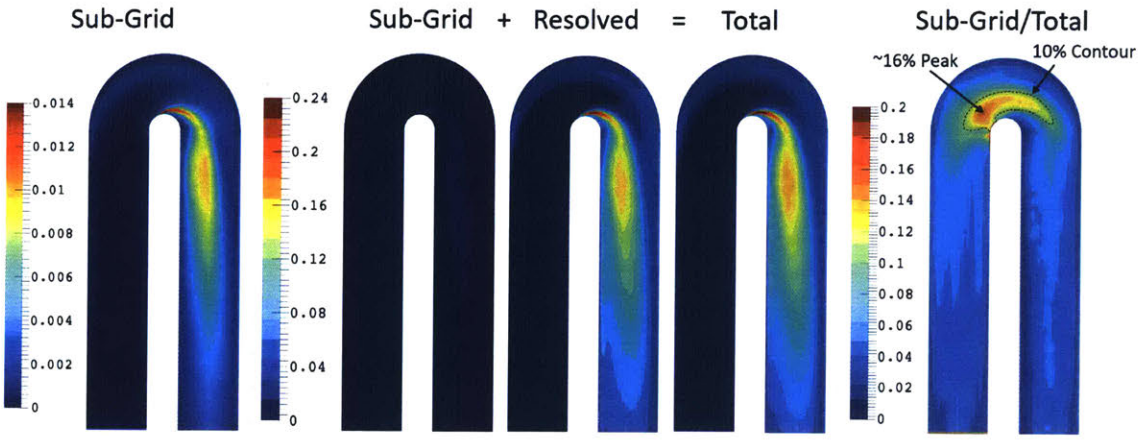


Figure 3-15: Time-averaged LES turbulent kinetic energy

The RANS simulation calculates the total TKE as part of the k-ω SST turbulence model. Figure 3-16 compares the total TKE between the two simulations on the same scale from a slice at $z/d_h = 0.5$. The RANS solution has a much lower level of TKE, especially in the shear layer along the separation streamline.

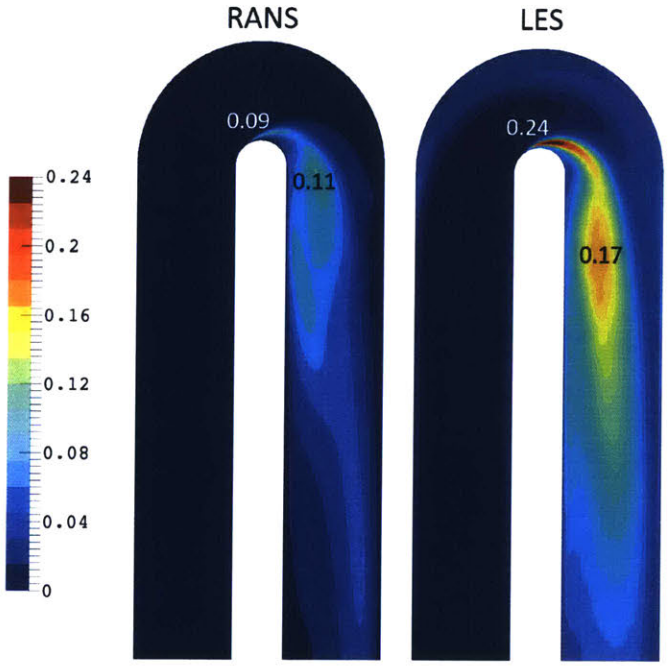


Figure 3-16: Turbulent kinetic energy comparison from RANS and LES

3.6 Comparison to Experimental Data

3.6.1 Mean Velocity Magnitude

As mentioned, the U-bend geometry is selected in order to compare to existing experimental data. Coletti et al. captured instantaneous two dimensional velocity fields at different planes using particle image velocimetry (PIV) at a sampling frequency of 2 hz. The mean velocity and root-mean-square of the velocity fluctuations were obtained by averaging 1000 image pairs. In addition, Coletti et al. also performed a RANS simulation using the k-ϵ turbulence model. Figure 3-17 shows a comparison between the PIV results and the various analytical results at $z/d_h = 0.5$. The k-ϵ RANS simulation under-predicts the post-bend velocity near the outer wall and the size of the separation. The outer wall boundary layer is also smaller than that of the PIV. The k-ω SST RANS simulation improves on the velocity magnitude in the bend and post-bend, but still does not accurately capture the separation shown in the experiment. The LES result best matches the PIV data. There is good agreement in peak velocities, outer wall velocity profile, and post-bend separation.

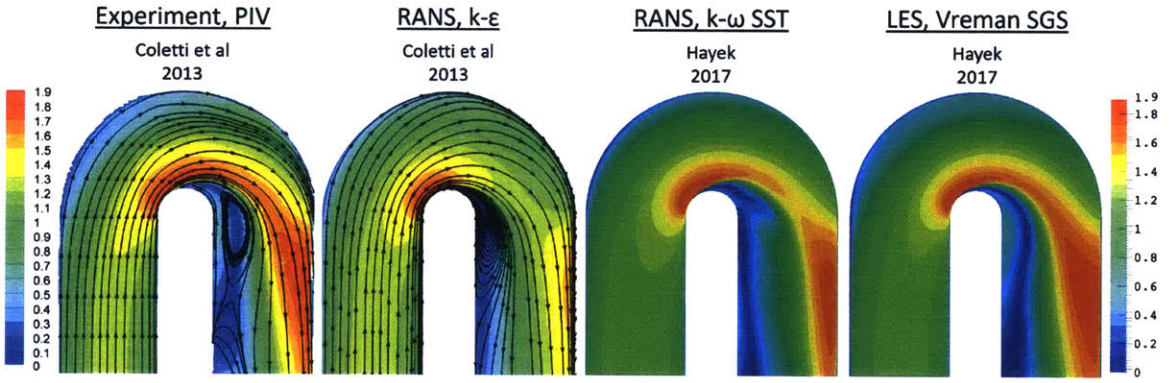
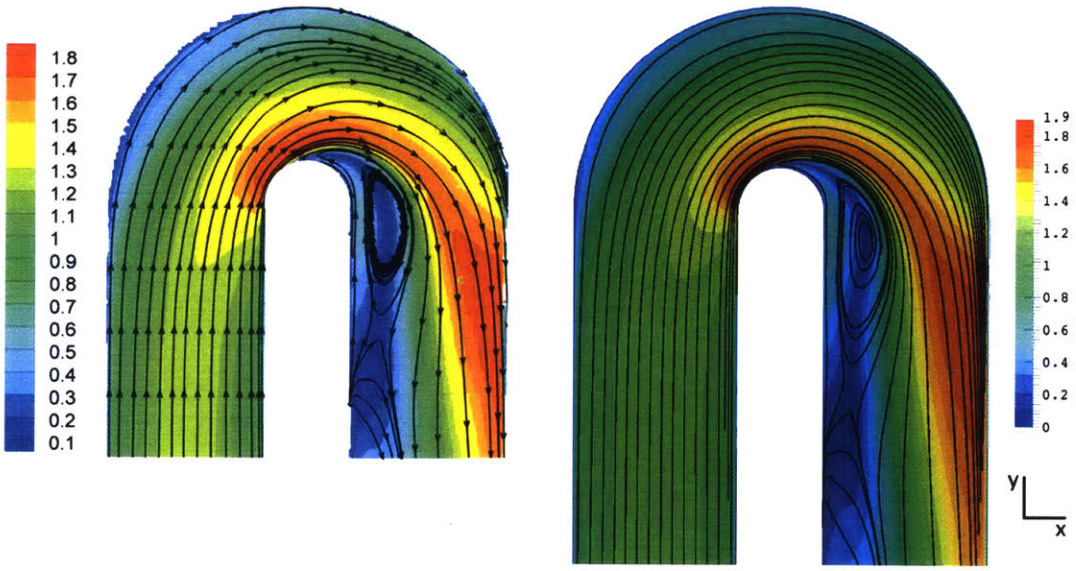


Figure 3-17: Time-averaged velocity magnitude comparison at z/d_h=0.5

Figure 3-18 has a close-up of the PIV and LES results. Qualitatively, the LES post-bend flow characteristics, such as the streamline curvature and the recirculation zone, compare well to the the experimental data. The center of the recirculation

zone from the inner wall (in the x direction) is approximately 0.23 d_h and 0.26 d_h for the PIV and LES results, respectively. The center is approximately 0.39 d_h for the PIV and 0.44 d_h for the LES in the y direction from the peak of the inner wall. The recirculation zone covers about 40% of the duct width in the PIV measurements compared to 44% from the time-averaged LES results. The area where the LES and PIV differ most is the reattachment point. The reattachment point from the LES is approximately 2.1 d_h from the top of the inner wall. The experimental results show a reattachment point at approximately 1.6 d_h.



(a) PIV (b) LES

Figure 3-18: Bend region mean velocity magnitude comparison at z/d_h=0.5

Figure 3-19 compares the experimental data at z/d_h of 0.03 to the analytical results. Again, the LES best matches the PIV data, yet slightly over-predicts the velocity magnitude in the post-bend region. The k-ϵ RANS result under predicts the separation and shows lower velocity magnitude in the post-bend region. The k-ω SST over-predicts the outer wall velocity deficit and the post-bend flow field contours are substantially different than the PIV results.

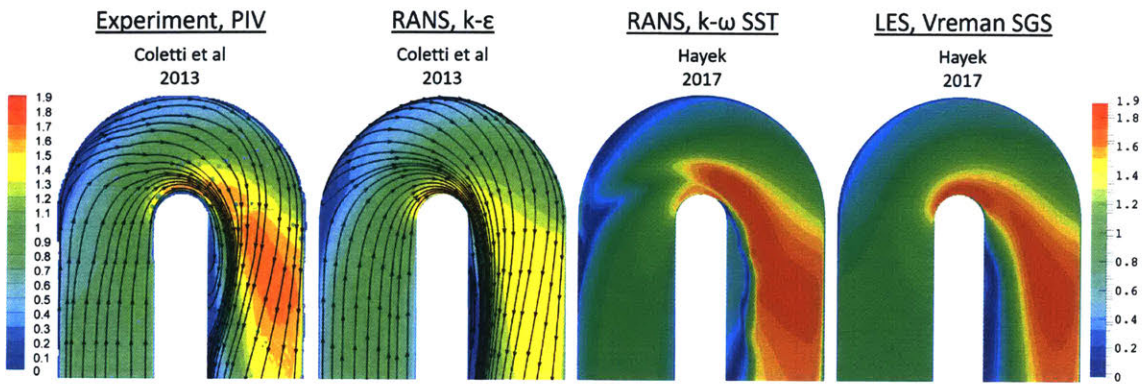
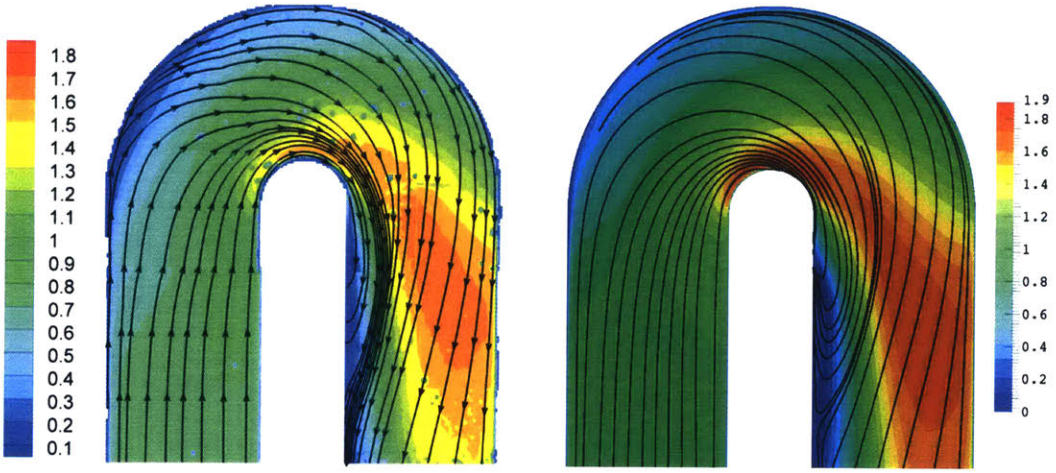


Figure 3-19: Mean velocity magnitude comparison at z/d_h=0.03

Figure 3-20 compares the streamlines of the PIV and LES results at z/d_h of 0.03. The recirculation zone is much smaller in this near-wall plane. Again, the LES and PIV results compare well in streamline curvature. It is believed that the kinks in the streamlines in the pre-bend region are due to multiple PIV images stitched together, which is evident from slight discontinuities. The LES reattachment point of 1.56 d_h from the top of the inner wall is again further downstream than the PIV experimental results, which is approximated to be at 1.3 d_h.



(a) PIV (b) LES

Figure 3-20: PIV vs LES time-averaged velocity magnitude comparison at z/d_h=0.03

3.6.2 Turbulent Kinetic Energy

The differences between the LES and RANS simulations are especially highlighted when comparing the turbulent kinetic energy. Following Coletti [3], the in-plane TKE is calculated and normalized by the bulk velocity as shown in Equation 3.6 in order to have a direct comparison to the experimental data. Since the PIV data is two dimensional, the out-of-plane TKE is assumed to be equal to half the in-plane TKE.

$$\text{TKE}* = \frac{\text{TKE}}{\frac{1}{2}U_b^2} = \frac{\frac{3}{4}\left(\overline{u'^2} + \overline{v'^2}\right)}{\frac{1}{2}U_b^2} \tag{3.6}$$

The same in-plane normalization method is used here for direct comparison. Figure 3-21 shows the TKE* field in the bend region of a slice at z/d_h of 0.5.

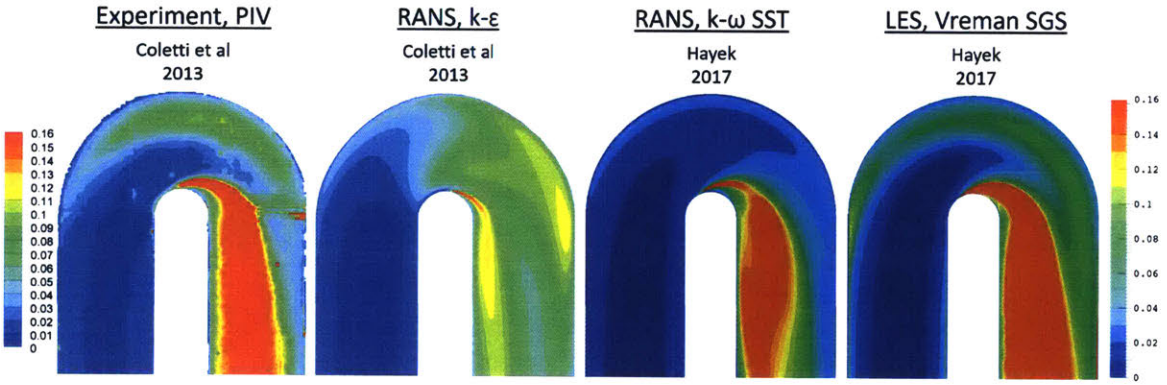


Figure 3-21: Time-averaged TKE* shown at z/d_h=0.5

The turbulence production in the post-bend region is largely missed with the k-ϵ model. The k-ω SST solution is closer to the PIV data in the post-bend region, but does not capture the outer wall increase in TKE as the LES does due to Gortler instability. The LES result compares well once again to the PIV data, though the post-bend inner wall TKE is slightly larger than that measured in the experiment.

3.7 Conclusion

The 180° U-bend square duct is a simple geometry, yet it is clear the dynamics of the flow field are quite complex. The flow features generated in this simulation are typical of wall bounded flows. It is shown that RANS does not capture the proper level of turbulence and generates incorrect velocity and pressure fields. Time-averaged large eddy simulation matches well to the experimental geometry and is an adequate analytical representation of the flow field. The LES is accurate and can be considered a high-fidelity solution.

Chapter 4

Optimization of the Turbulent Viscosity for a 180° U-bend Channel

4.1 Introduction

It is shown in Chapter 3 that standard RANS turbulence models do not properly capture the physics in some complex internal flows. The LES does a decent job of matching the experimental data for a U-bend square duct. However, using LES for geometry optimization is currently prohibitively expensive. The desire is to improve an eddy viscosity RANS turbulence model by using the high-fidelity LES solution. In order to do this, the LES is initially solved on the same geometry as the RANS simulation. Using an optimizer, the turbulent viscosity field is inferred to minimize the velocity mismatch between the high-fidelity (LES) and low-fidelity (RANS) solutions. The resulting RANS model is a new geometry-specific turbulence model that does not use any transport equations to solve for the eddy viscosity. This high-fidelity trained RANS turbulence model is abbreviated as HIFIR.

In this chapter, a U-bend channel is used to allow the use of a 2D RANS solver to reduce the computational cost. The separation in the bend region and the recirulation zone are also present in the U-bend channel scenario. A three-dimensional domain is still required for the LES.

4.2 Fluid Solvers

4.2.1 High Fidelity Solver

The high-fidelity solver used for the U-bend square duct discussed in Section 3.2 is used again for the U-bend channel. OpenFOAM's pisoFoam solver is utilized with the Vreman SGS turbulence model. The geometry is shown in Figure 4-1. The total channel width, H, is set to unity. Cyclic boundary conditions are used at the top and bottom surfaces of the domain. The domain height is set to πH. A study of doubling the domain height to 2πH was found to change the local mean velocity by less than 2%. The post-bend unsteady flow structures were sufficiently captured with the πH height and matched those of the taller domain. The domain height sensitivity study is provided in Appendix B.

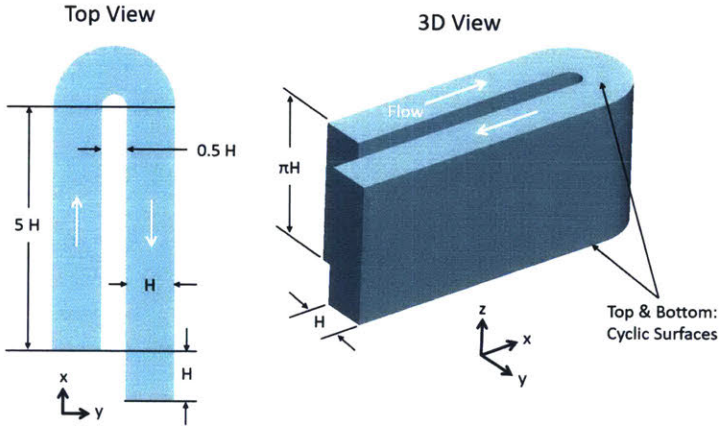


Figure 4-1: Domain geometry used for U-bend channel LES

The mesh topology from the square duct case is also used for the channel. The same 150 cell count and expansion factors are used across the channel width. The spanwise cell size in the periodic direction is set to be approximately 25 wall units, similar to the max spanwise cell size from the square duct case. This requires 126 cells in the periodic direction with uniform spacing. The streamwise mesh topology is identical to the square duct case, including the sponge region, resulting in 265 cells. The overall cell count is 150 x 126 x 265, or approximately 5 million cells. This is approximately 16% smaller than the square duct mesh size.

The bulk Reynolds number of 40,000 is obtained by setting the density to unity and the kinematic viscosity to 2.5e-5. The bulk Reynolds number is based on the total channel width, H, and bulk velocity of unity in order to match the kinematic viscosity of the square duct simulation. The boundary conditions are similar to that of the square duct case, but with the addition of cyclic boundary conditions on the top and bottom surfaces. The boundary conditions are shown in Table 4.1.

Table 4.1: Boundary conditions for LES channel

Parameter	**Inlet**	**Outlet**	**Walls**	**Top, Bottom**
u	mapped	zero-gradient	zero	cyclic
p	zero-gradient	zero	zero-gradient	cyclic
k_{SGS}	0.004	zero-gradient	zero	cyclic
ν_{SGS}	zero-gradient	zero-gradient	zero-gradient	cyclic

Similar to the square duct, an unsteady inlet boundary condition database is created. The pre-simulation domain has a size of 1H width x πH periodic height x 2πH length, with cyclic boundary conditions at the top and bottom surfaces similar to the U-bend channel case and the mapped outlet-to-inlet boundary condition discussed in Section 3.4. A RANS solution is mapped onto the domain as the initial condition of the pre-simulation. The simulation is run sufficiently long to generate physical turbulent structures before storing the output velocity fields into the database.

The same 0.001 second timestep size is used for the U-bend simulation in order to maintain the sufficiently low CFL numbers, since the cell sizes are similar to that of the square duct. Due to the similar FTU, the start-up transient behavior follows that of the square duct case. The start-up is fully captured within the first 20 seconds, at which point the time-averaging is initiated. The simulation is run for a total of 250 seconds, resulting in over 18 FTU of time-averaging. Since the channel is periodic, the long time average flow field is essentially two-dimensional. The time-averaged LES solution is also collapsed along the z axis to produce a 2D mean field. The simulation is performed on 32 cores and requires about 8 days to complete.

4.2.2 Low Fidelity Solver

The low-fidelity solver is a custom-written, 2D structured, steady state incompressible RANS finite-volume flow solver based on the SIMPLE method developed by Patankar [28] but on a collocated grid following the formulation provided by Ferziger and Peric [6]. The solver is written in the Python programming language in order to incorporate the existing automatic differentiation code, numpad, written by Wang [42] for optimization purposes. The flow solver has the ability to handle non-uniform grids by use of a collocated mesh.

As discussed and first introduced in Section 2.3.4, the RANS model is based on a general eddy viscosity model in which each cell's turbulent viscosity can be modified. This allows for assignment of the turbulent viscosity throughout the domain via an optimizer to match the velocity field of a high fidelity output.

The mesh topology from the high-fidelity simulation is directly utilized for the low-fidelity solver. Since the solver is 2D, just the 150 spanwise cells and 260 streamwise cells are used. The 5 cell sponge region is not needed, resulting in 39,000 cells in total. The Python mesh is shown in Figure 4-2, which is identical to any z slice of the LES mesh without the sponge region.

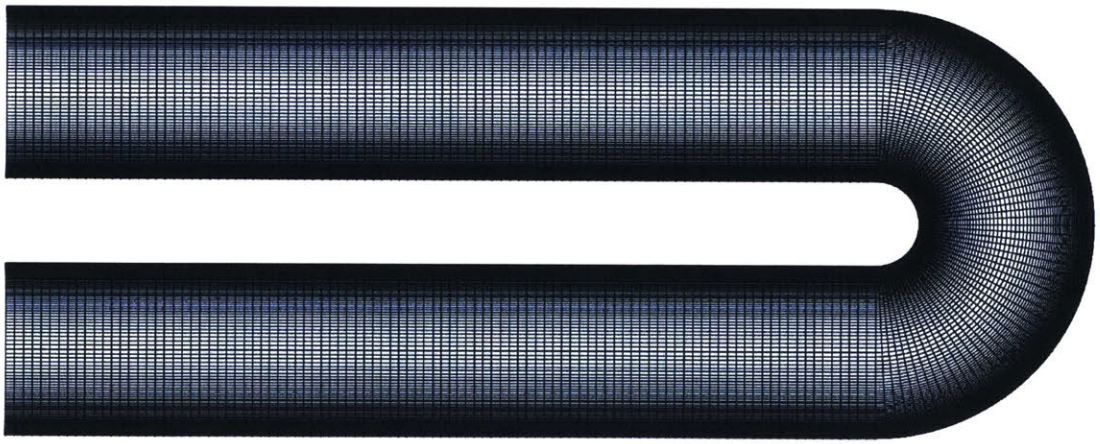


Figure 4-2: Mesh used for low-fidelity solver

The solver is setup to use a first-order upwind or an upwind/central differencing blended scheme for the convective fluxes. A local weighting factor based on the local Peclet number is used to blend between fully central differencing (CDS) and fully

upwind differencing schemes (UDS). This is done to stabilize the solver by adding some diffusion with upwinding, which is required due to the high Reynolds number, and thus high Peclet number. The Peclet number is the ratio of the strengths of convection to diffusion, and is shown in Equation 4.1. There must be a balance between accuracy and stability, as too much blending with UDS can cause excessive diffusion and overly smooth the solution. Appendix D discusses various Peclet-based UDS/CDS blending schemes. A hyperbolic tangent blending function based on the local Peclet number is used for the low-fidelity simulations. The central differencing scheme is used for diffusive terms, which is the standard approach.

$$Pe = \frac{F}{D} = \frac{\rho u}{\mu/\delta x} = \frac{u\Delta x}{\nu + \nu_t} \tag{4.1}$$

The boundary conditions imposed on the 2D U-bend simulation are similar to those used for the LES simulation. The boundary conditions are provided in Table 4.2. The inlet boundary condition is the time-averaged profile from the pre-simulation used to generate the unsteady inlet boundary condition for the LES case. This is done to maintain consistent boundary conditions between the high and low-fidelity simulations.

Table 4.2: Boundary conditions for 2D Python RANS simulation

Parameter	**Inlet**	**Outlet**	**Walls**
u	mapped	zero-gradient	zero
p	zero-gradient	zero	zero-gradient
ν_t	zero-gradient	zero-gradient	zero

Although the turbulent viscosity field is modified by an optimizer, it is recommended to start with a turbulent viscosity field that is somewhat realistic. A good initial condition is a simulation using a two-equation turbulence model on the same mesh. The turbulent viscosity field from the k-ω output is used as the starting point. If the initial eddy viscosity values are too low, the simulation can become unstable. Some amount of upwinding can be used to make the solver more robust and stable.

After every iteration of the solver, the residuals of the two momentum equations and continuity equation are calculated. The convergence of the simulation is deter-

mined by the reduction in these residuals. Once the residuals are sufficiently reduced, usually to a value of 1e-5 or 1e-6, the simulation is stopped and the solution from the last iteration is considered the converged solution. This residual control of the solver is typical of steady state CFD solvers.

Due to the smaller size of the 2D simulations, the momentum and pressure correction equations are solved using Python's direct sparse linear solver. Python iterative solvers were found to take more computational time. For a 2D simulation containing nearly 40,000 cells, each primal flow solver iteration requires about 1 second using one 3.50 GHz CPU. About 2000 to 4000 iterations are needed to obtain a good solution, and thus most primal simulations require about 1 hour. The Python code is approximately 10 times slower than OpenFOAM's simpleFoam SIMPLE solver, which is based on compiled C++ code and utilizes iterative solvers. Parallel capability and iterative linear solvers are future improvements needed before extending the Python solver to three dimensions.

The Python SIMPLE solver was benchmarked in laminar mode (uniformly zero turbulent viscosity) on a 2D lid-driven cavity case at Reynolds numbers of 100 and 1000 based on the lid velocity and the size of the square cavity. The solver was shown to converge to existing benchmark solutions as the mesh was refined. The solver was also benchmarked to OpenFOAM for a laminar U-bend channel, which had nearly identical results. Benchmarking is discussed in further detail in Appendix C.

4.3 Low-Fidelity Training Setup - HIFIR Model

The training of the low-fidelity RANS solver is performed by optimization of the turbulent viscosity field. The RANS solver is the primal solver, in which the velocity and pressure fields are converged for a set of boundary conditions and input turbulent viscosity field. The output of the primal solver is then used to calculate the objective value. The primal output is also the initial condition to the adjoint calculations. Reverse mode automatic differentiation is used to obtain the initial sensitivity gradient, or Jacobian, required to minimize the objective value. The iterative adjoint calcula-

tions discussed in Appendix A are used to converge the Jacobian. One primal solve followed by an adjoint solve produces both the objective function and Jacobian and is treated as one outer, or optimization, iteration.

Python's built-in optimizer (scipy.optimize.minimize) is used to drive the optimization using the L-BFGS method. The primal and adjoint solvers are wrapped into one function that is called by the minimizer. The objective value as well as the Jacobian, velocity, pressure, and turbulent viscosity fields are stored for each overall function evaluation to track progress and provide restart capability in the event of an unexpected termination of the optimization. In addition, the residuals of each primal and adjoint solve are stored and plotted for monitoring purposes.

4.3.1 Objective Functions

The objective function is based on the squared L2 norm of the velocity mismatch and is shown in Equation 4.2 where the subscript i denotes the local cell.

$$J(u(\nu_t)) = \|u(\nu_t) - u_{\text{HiFi}}\|_{\text{L2}}^2 = \sum_i^N (u_i - u_{i,\text{HiFi}})^2 \tag{4.2}$$

Since the velocity field is a vector with x and y components, the squared L2 norm objective is implemented as

```
J = ( ((u_x - u_x_HiFi)**2).sum() + ((u_y - u_y_HiFi)**2).sum() )
```

in the Python code. The squared L2 norm is a common error, or cost, function used for optimization and is the primary objective function used in the training.

Several other objective functions are formulated to investigate the impact of the objective function on the optimization path and solution. The L1 norm, known as the "Manhattan" distance, can also be used on the velocity mismatch and has the following form.

$$J(u(\nu_t)) = \|u(\nu_t) - u_{\text{HiFi}}\|_{\text{L1}} = \sum_i^N |u_i - u_{i,\text{HiFi}}| \tag{4.3}$$

Python's numpy absolute value function is not recognized as a differentiable function by the numpad AD tool and thus the L1 norm objective is rewritten as

$$\|u(\nu_t) - u_{\text{HiFi}}\|_{\text{L1}} \approx \sum_{i}^{N} \left((u_i - u_{i,\text{HiFi}})^2 + \epsilon\right)^{0.5} \tag{4.4}$$

where ϵ is a small number, i.e. 1e-12, to ensure the term within the square root is never zero so that the objective function can be differentiated. The same squared L2 and L1 norm based objective functions can also be used on the pressure mismatch instead of using the velocity mismatch. The form of these equations are essentially the same, but only have one component since the pressure is a scalar (no x and y components).

Additionally, in order to check the robustness of the system and ability to deal with limited data, an objective function based only on an inlet and outlet velocity profile is utilized. The idea is that the full domain velocity and pressure fields may not be measurable with some experimental methods.

The baseline squared L2 norm objective function results are shown in Section 4.4. The impact to the solution with the use of the alternate objective functions are discussed in Sections 4.6, 4.8, and 4.9.

4.3.2 Bounding of Control Parameter

The baseline optimization is an unbounded optimization made possible by the use of $\log(\nu_t)$ as the control parameter. As discussed in Section 2.3.4, this automatically enforces the non-negativity of the turbulent viscosity field. In the interest of observing where the eddy viscosity model breaks down and predicts negative values of the turbulent viscosity, the unbounded optimization is also performed with the turbulent viscosity directly as the control parameter. The results are discussed in Section 4.7.

4.4 Baseline Optimization of the Turbulent Viscosity

4.4.1 Adjoint Accuracy and Optimization Convergence

To estimate the accuracy of the gradient, the Jacobian calculated by the numpad reverse mode automatic differentiation is compared against a finite difference perturbation calculation. Since the gradient and the turbulent viscosity are both non-linear fields, the two methods are compared using the change in objective value for a given perturbation in the turbulent viscosity. The change in objective value is obtained using the following equations.

$$\delta J_{\mathrm{FD}} = J\big(u(\nu_t + \delta\nu_t)\big) - J\big(u(\nu_t)\big) \tag{4.5}$$

$$\delta J_{\mathrm{AD}} = \sum \delta\nu_{t,i} \left.\frac{\partial J}{\partial \nu_t}\right|_i \tag{4.6}$$

The results of the two methods are compared in Table 4.3. The change in objective value is based on a change in the turbulent viscosity field by 0.01%. The comparison in the table is provided at an early iteration of the optimization, where the gradient is still relatively large, and at a later iteration where the gradient is much smaller.

Table 4.3: Comparison of change in objective value

Method	**δJ, early iteration**	**δJ, later iteration**
Finite Difference (FD)	-.253	.0061
Adjoint (AD)	-0.291 (+15.4%)	.0009 (-86%)

As shown, there is a significant difference in the change in objective value between the two methods. The finite difference is known to be correct, thus there is some error in the adjoint sensitivity. This is not due to the low-memory calculation defined in Appendix A as that shows the same gradient calculation as the full-memory adjoint calculation. The error may be inherent to the numpad tool and/or the solver source code, yet is still unknown at this time. The direction of the gradient matches that of the finite difference, but the adjoint gradient seems to decrease more rapidly in

magnitude as the optimization progresses and may lead to a solution that is not the true minimum.

Despite the error in the adjoint calculation of the Jacobian, the Python L-BFGS optimizer steadily reduces the objective value as shown in Figure 4-3. The adjoint gradient still provides a useful search direction for the optimizer and successfully reduces the solution error substantially.

The optimization path does show several spikes in the objective value due to large changes in the turbulent viscosity where the optimizer was driven by a poor adjoint calculation or took a step too large. The optimizer recovers quickly after each such event. The k-ω SST solution is used as the initial condition to the HIFIR optimization.

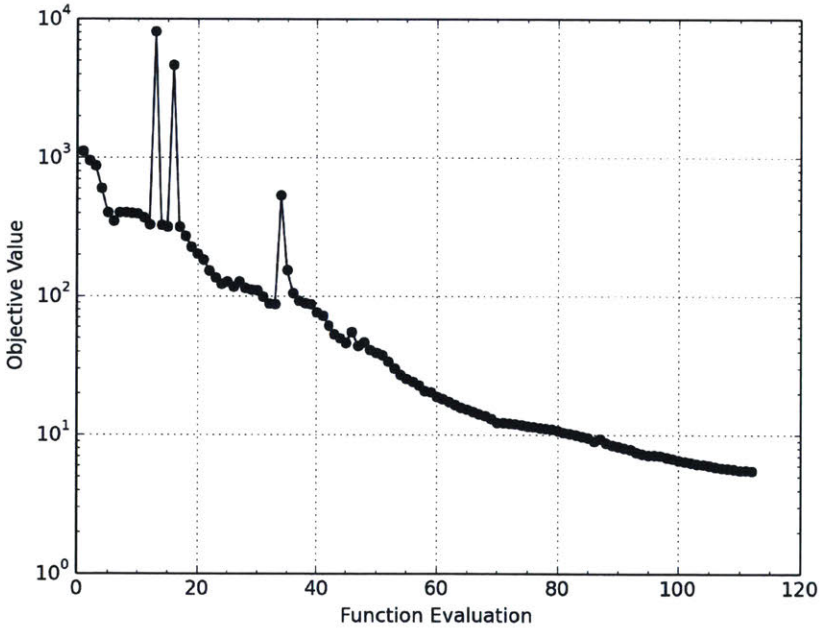


Figure 4-3: Convergence of objective value

4.4.2 Velocity and Pressure Fields

In this section the final iteration of the optimization is compared to the LES and standard k-ω SST solutions. The velocity magnitude fields of the three solutions are shown in Figure 4-4. The optimized HIFIR solution shows very similar contours to the LES and improves on the k-ω SST solution in a number of ways. The size of the recirculation zone is easily visible in Figure 4-5, which shows the x-component of the velocity field. The recirculation zone corresponds to the positive velocity in the post-bend region. The LES and HIFIR solution have a smaller recirculation zone and enhanced mixing near the domain exit compared to the k-ω SST solution.

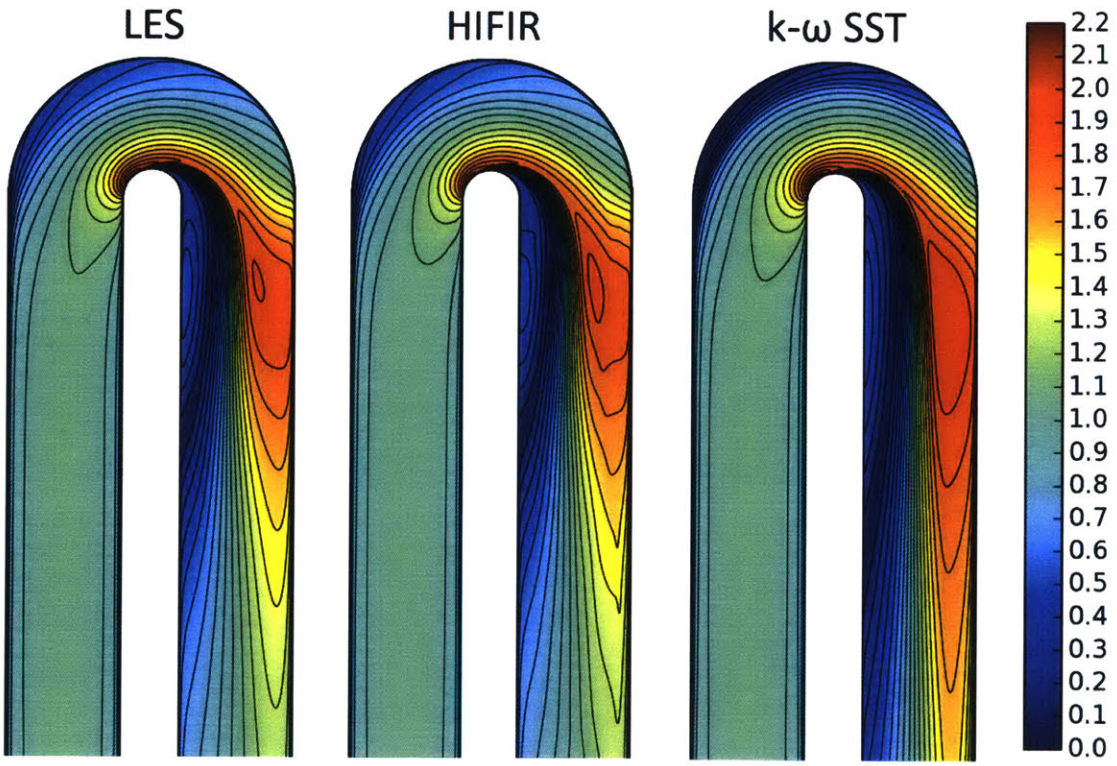


Figure 4-4: Comparison of velocity magnitude field

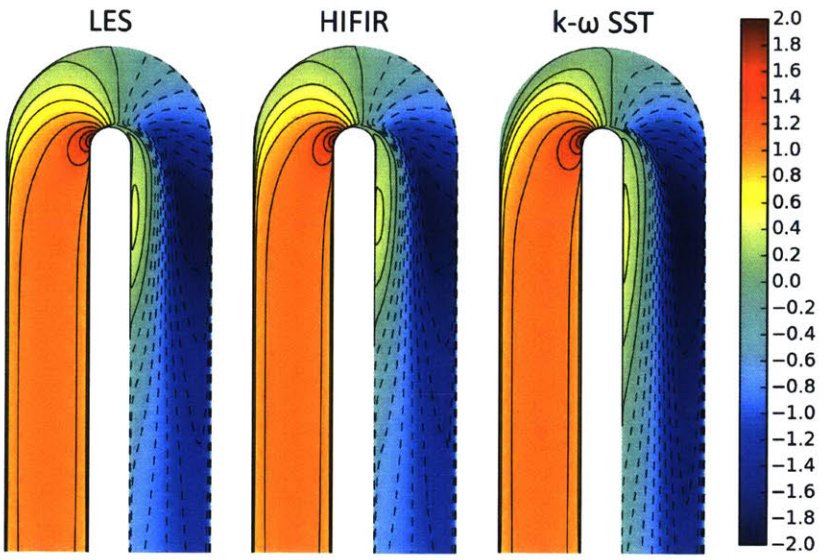


Figure 4-5: Comparison of x-velocity field

Another region that shows noteworthy improvement is along the outer wall. The k-ω SST solution contains a lower velocity along the outer wall, especially just downstream of the bend inlet. In fact, the k-ω SST solution has a small recirculation zone at the bend inlet outer wall that can be seen in the y-component velocity field in Figure 4-6. The HIFIR and LES solutions do not show the small outer wall separation. Otherwise, the y-component velocity field is similar between the three solutions.

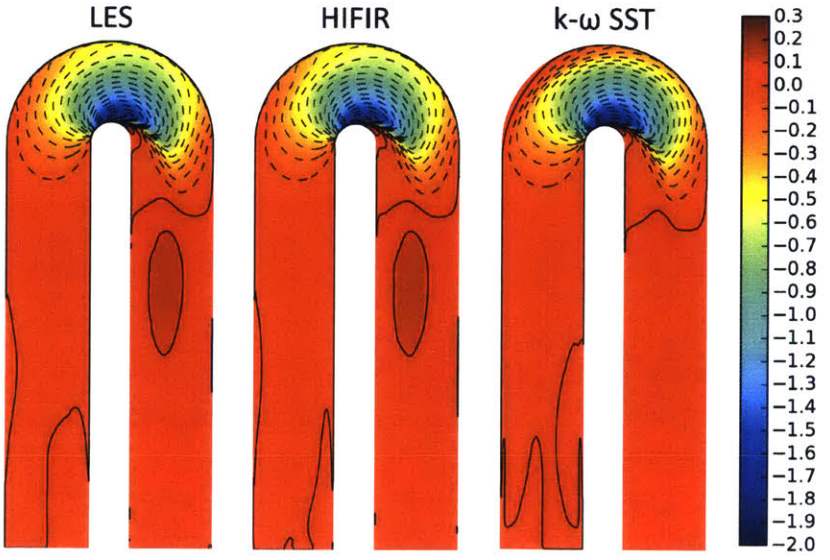


Figure 4-6: Comparison of y-velocity field

The pressure field comparison is shown in Figure 4-7. One observation is that the LES and HIFIR results have a lower inlet pressure which means the pressure drop across the bend is smaller since the outer pressure is set to zero as a boundary condition. The pressure drop is further discussed in Section 4.4.6. Both RANS results show more straight pressure contours in the post-bend region, whereas the LES still has some curvature in the contours. The k-ω SST result also shows a higher pressure in outer bend region and in the eye of the recirculation zone.

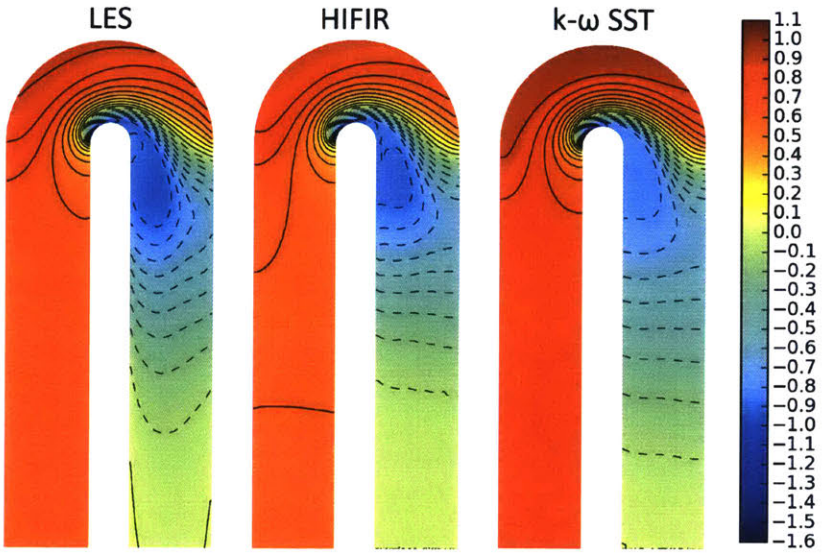


Figure 4-7: Comparison of pressure field

Another visualization of the solution difference can be obtained with the use of velocity profiles at discrete locations throughout the domain. The streamwise velocity profiles at several sections are shown in Figure 4-8. The geometry is reoriented as the outlet section of the domain is of more interest. The HIFIR solution nearly matches the LES profiles at all sections. The outlet section profiles show some minor oscillation of the HIFIR solution around the LES profile. The HIFIR result is clearly a large improvement over the k-ω SST solution. The outer wall velocity deficit of the SST is evident in the mid-bend profile. The SST profile is reasonably matched to the LES at the bend exit but starts to increase in error as the mixing is under-predicted, leading to a less-uniform velocity profile.

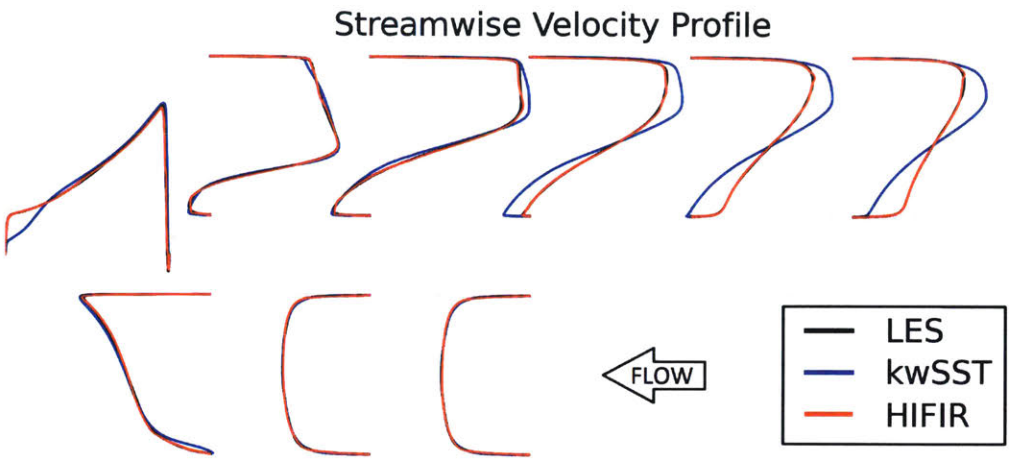


Figure 4-8: Velocity profile at various bend sections

The accuracy of the solution can be quantified by using the same squared L2 norm calculation utilized for the objective value. Both the velocity and pressure mismatch with the LES solution are computed and shown in Table 4.4 for the k-ω SST and HIFIR solution. The squared L2 norm values show that the HIFIR solution is substantially closer to the time-averaged LES result, supporting the qualitative comparison in the figures above.

Table 4.4: Velocity and pressure mismatch from LES

Simulation	**Velocity**	**Pressure**
k-ω SST	955.4	404.9
HIFIR	5.5	52.1

4.4.3 Turbulent Viscosity Field

As known, the control parameter of the optimization is the turbulent viscosity. Although the velocity field is significantly improved with the HIFIR model, the method is of little value if the turbulent viscosity is unrealistic. The turbulent viscosity from the time-averaged LES is extracted to provide a sanity check. The LES turbulent viscosity is obtained using Equation 4.7. The derivation of this expression and implementation in OpenFOAM is provided in Appendix E.

$$\nu_t = \frac{-\overline{u_i' u_j'}\,\overline{s_{ij}} + \frac{2}{3}k\delta_{ij}\overline{s_{ij}}}{2\ \overline{s_{ij}}\ \overline{s_{ij}}} + \overline{\nu_{t,\mathrm{SGS}}} \tag{4.7}$$

The cell-centered values of the turbulent viscosity ratio from the LES, k-ω SST, and HIFIR results are shown in Figure 4-9. The turbulent viscosity ratio is defined as the ratio of the turbulent to kinematic viscosity, ν_t/ν. The k-ω SST turbulent viscosity is clearly under-predicted. The HIFIR result is much closer to the LES extracted eddy viscosity and shares many characteristic features. The outer wall shows increased turbulent viscosity compared to the SST result. The HIFIR result also shows a gradual increase of the turbulent viscosity in the post-bend recirculation and shear region, similar to the LES result. Overall, the turbulent viscosity from the HIFIR method generally agrees with the LES and is a reasonable result.

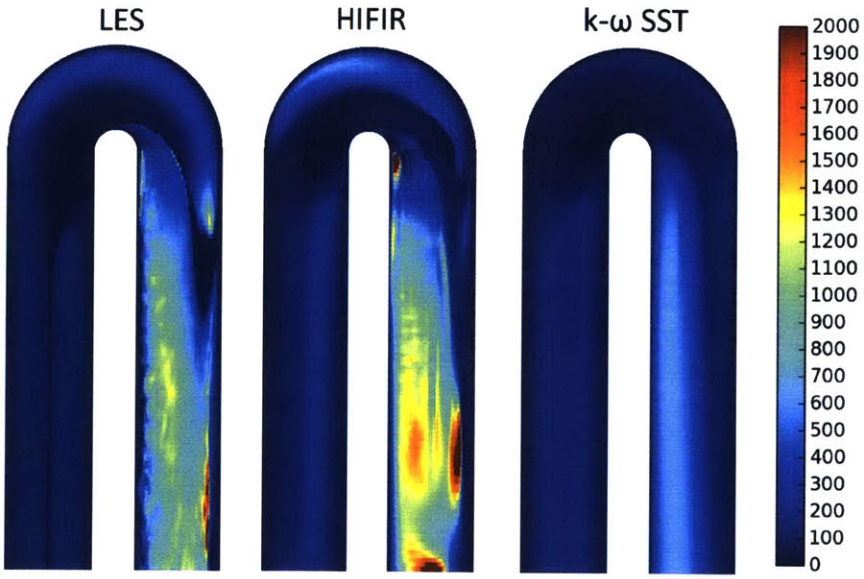


Figure 4-9: Comparison of turbulent viscosity ratio field

The figure above is showing only positive values of the turbulent viscosity, which covers the full range of values for the RANS simulations. However, the LES extracted turbulent viscosity contains some negative values. The negative values show areas where the Boussinesq approximation breaks down and may highlight where small scale structures transfer some energy into larger scales, which is known as back-scatter. Figure 4-10 shows the LES extracted turbulent viscosity ratio with negative

values. The right sub-figure displays the regions of negative values in blue.

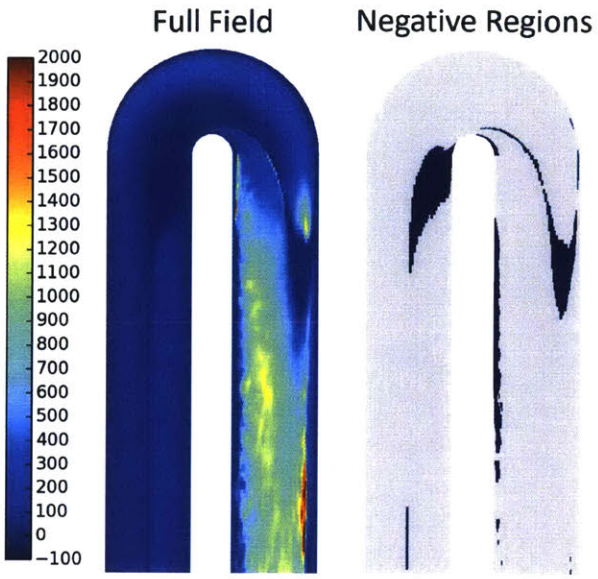


Figure 4-10: LES turbulent viscosity ratio field and associated negative regions

The peak negative ratio values in each region have magnitudes of one or two orders larger than the kinematic viscosity. The pre-bend negative region corresponds to the area of accelerating flow going into the bend. The post-bend negative region near the outer wall corresponds to the peak velocity region outside of the recirculation. Both of these high velocity regions, and the post-bend inner wall negative region, suggest anisotropy in the Reynolds stress and a deviation from the Boussinesq approximation. Although the negative values are large in magnitude, the regions of negative viscosity are relatively small compared to the domain size.

4.4.4 Reynolds Stress and Turbulent Kinetic Energy Production

The HIFIR method does not use a transport equation for the turbulent kinetic energy and therefore the turbulent kinetic energy term is not computed. However, the production of the turbulent kinetic energy can still be calculated using the Boussinesq approximation to obtain the Reynolds stress components. The production of the turbulent kinetic energy is the conversion of kinetic energy from the mean flow into

the turbulent fluctuating flow and is caused by shear in the mean flow. The production term of the turbulent kinetic energy transport equation is shown in Equation 4.8 and is expanded out for a general 2D case.

$$P_{ij} = -\overline{u_i' u_j'} \frac{\partial \overline{u_i}}{\partial x_j} = -\overline{u'^2} \frac{\partial \overline{u}}{\partial x} - \overline{v'^2} \frac{\partial \overline{v}}{\partial y} - \overline{u'v'} \left(\frac{\partial \overline{u}}{\partial y} + \frac{\partial \overline{v}}{\partial x} \right) \tag{4.8}$$

With LES, the Reynolds stress components can be directly computed and stored during the simulation. The mean velocity gradients can be computed from the time-averaged velocity field as a post-processing step. The production term is then also calculated as a post-processing operation. For the RANS simulations, the Reynolds stress components are approximated using the Boussinesq approximation. The mean velocity gradients are also computed as a post-processing step in the RANS simulation as this is not normally stored. The TKE production is shown for the three simulation methods in Figure 4-11.

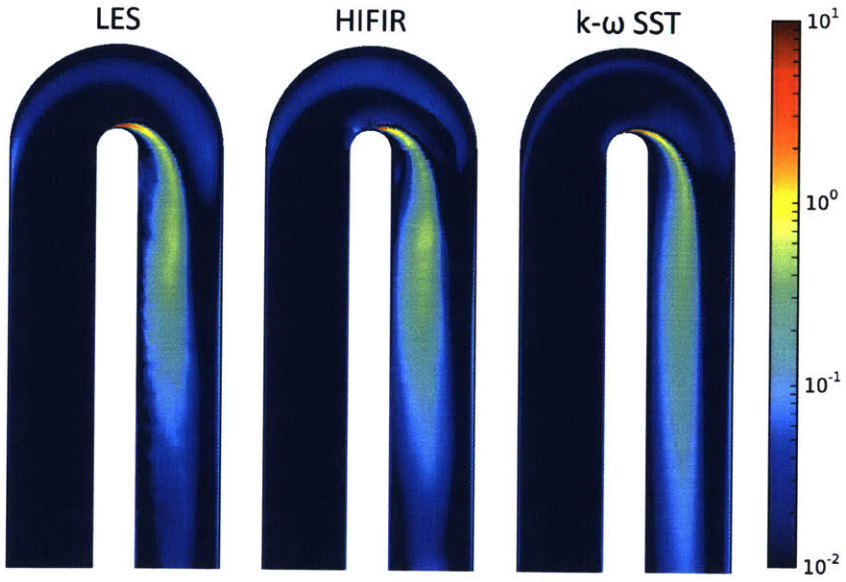


Figure 4-11: Comparison of TKE production

As shown in the above figure, the HIFIR model displays greater production in the outer bend region and in the post-bend shear region compared to the k-ω SST result. The high post-bend production zone of the k-ω SST solution is narrower than the LES yet decays at a lower rate downstream. The HIFIR result resembles a smoothed

out LES solution, but shows some additional production near the inner wall of the bend similar to the k-ω SST, which is due to the local gradient. This is not seen in the LES despite a similar gradient as the Reynolds stress terms are not calculated based on mean gradients.

The shear component of the Reynolds stress tensor, $\overline{u'v'}$, is shown in Figure 4-12. Similar to the TKE production, the LES and HIFIR results show a larger magnitude of the shear stress in the post-bend region, but this decays more quickly when compared to the k-ω SST result. The HIFIR model also shows an increase in the Reynolds stress in the outer bend region and is even larger than the LES result. Both RANS results do not show as strong of a negative shear stress in the high velocity region separating from the bend inner wall.

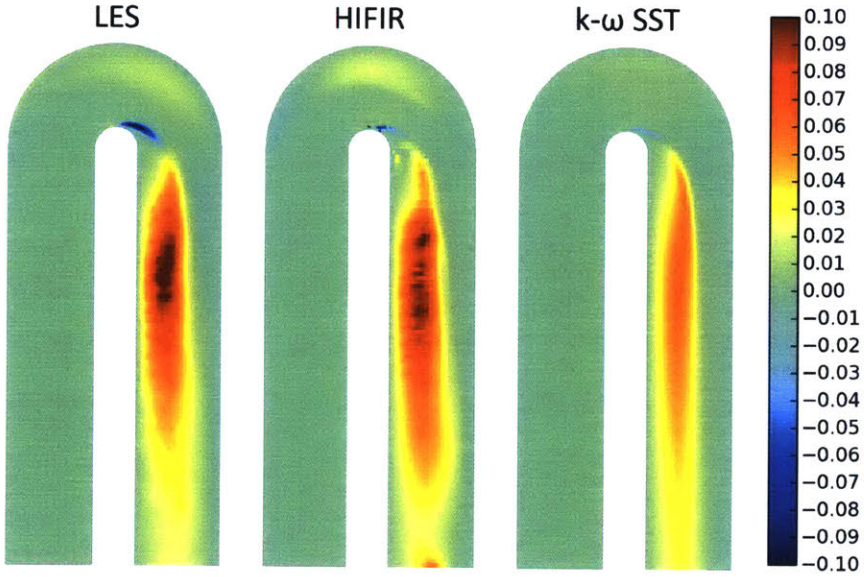


Figure 4-12: Comparison of $\overline{u'v'}$ Reynolds stress component

4.4.5 Reattachment Location

Since the optimizer is trying to minimize the velocity mismatch, it is not surprising the reattachment location is greatly improved with the optimized turbulent viscosity field. The reattachment location can be estimated by finding where the wall shear stress along the inner wall is zero. The wall shear stress is calculated by

$$\tau_w = \mu \frac{\partial \overline{u}}{\partial y} \approx \mu \frac{u_1}{\Delta y_1} \tag{4.9}$$

where the velocity gradient is calculated using the velocity of the first node off the wall and the distance from the wall. For this simulation the dynamic and kinematic viscosity are the same ($\mu = \nu$) since the density is set to unity. The first node corresponds to a distance of 0.0005 H, or 0.05% of the full channel width, which should provide a reasonable value for the wall shear stress. The wall shear stress along the inner wall is interpolated to find where the shear stress is zero. The wall shear stress along the inner wall is shown in Figure 4-13 as a function of the streamwise distance from the bend exit. The sign convention is such that negative wall shear stress corresponds to flow back toward the U-bend and is part of the recirculation region.

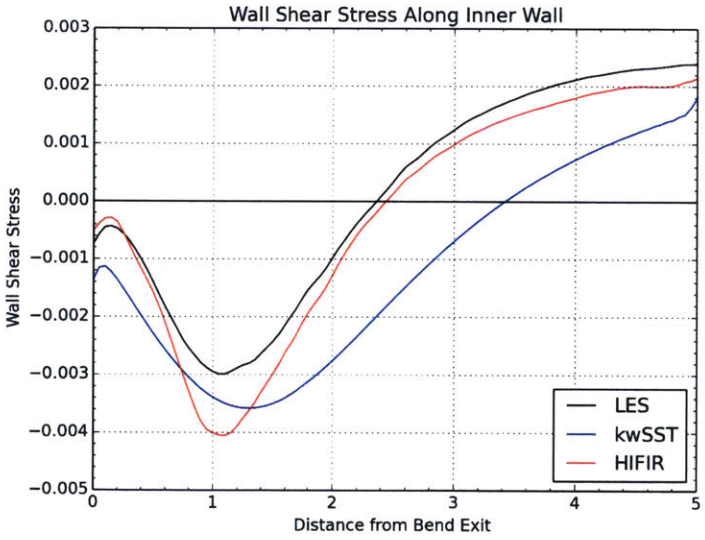


Figure 4-13: Wall shear stress along inner wall in straight exit channel

As expected, the HIFIR prediction and time-averaged LES agree well. There is a significant difference between the LES and k-ω SST prediction. The interpolated value that approximates the reattachment point from the bend exit is shown in Table 4.5 for each simulation.

Table 4.5: Approximate reattachment location

Simulation	Location	% Error
LES	2.37H	–
k-ω SST	3.42H	+44.7%
HIFIR	2.44H	+3.2%

4.4.6 Pressure Drop Across U-bend

For an incompressible flow, the velocity and pressure are strongly coupled. It is expected that as the velocity mismatch is reduced, the pressure mismatch should also be reduced. This is supported by the reduction in pressure mismatch shown in Table 4.4. The improved HIFIR pressure field is shown in Figure 4-7, but it is not straightforward how much the pressure drop prediction is improved. As mentioned in Chapter 1, the pressure drop across the U-bend is critical for design purposes. The pressure drop for the U-bend channel is calculated by comparing the mass-averaged pressure at an inlet and outlet section. The inlet section is selected to be at a distance of 2H from the bend inlet, which is prior to any significant streamline curvature and is nearly straight turbulent channel flow. Multiple exit sections are taken after the bend, ranging from 2H to 5H from the bend exit to not only capture the pressure drop due to the bend, but also the impact of mixing and pressure recovery further downstream. The section locations are shown in Figure 4-14. The results are shown for the various inlet and outlet section combinations in Table 4.6.

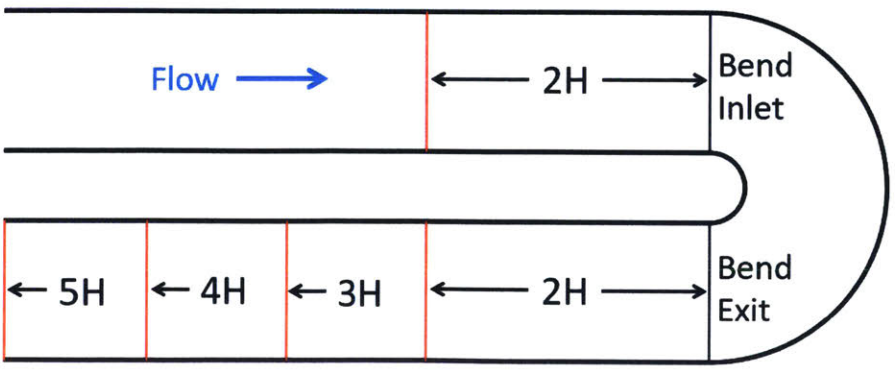


Figure 4-14: Sections used for pressure drop locations

Table 4.6: Pressure drop comparison at various inlet and outlet planes

Inlet/Outlet	LES	k-ω SST	HIFIR
2H / 2H	1.00	1.22 (+22.5%)	0.94 (-5.4%)
2H / 3H	0.76	0.98 (+29.1%)	0.71 (-6.9%)
2H / 4H	0.67	0.83 (+25.0%)	0.63 (-6.0%)
2H / 5H	0.63	0.75 (+18.9%)	0.60 (-5.4%)

4.5 Sensitivity to Convection Scheme

The results shown with the HIFIR model thus far have been based on a hybrid upwind / central differencing scheme using a smooth hyperbolic tangent blending function for the convective flux. This section explores the impact to the convection scheme by repeating the baseline optimization with a first-order upwind scheme. The optimization convergence from the same initial condition is shown for the hybrid and upwind scheme in Figure 4-15. The objective value initially converges faster with the upwind scheme, but stagnates around a value of 35 to 40, whereas the hybrid method steadily converges to an objective value below 6. In this section, the baseline result from Section 4.4 is referred to as the "Hybrid" HIFIR solution and the new upwind solution is simply labeled as "Upwind".

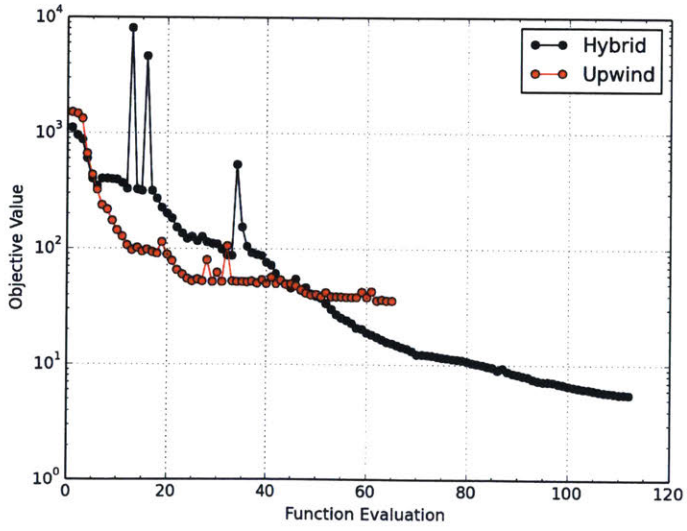


Figure 4-15: Convergence of objective value with upwinding

The comparison of the velocity magnitude field between the time-averaged LES and the two HIFIR convection scheme solutions is shown in Figure 4-16. The upwind solution generally agrees well with the hybrid and LES solution. The upwind solution is noticeably different in the high velocity areas as the contours are more rounded due to the more dissipative scheme.

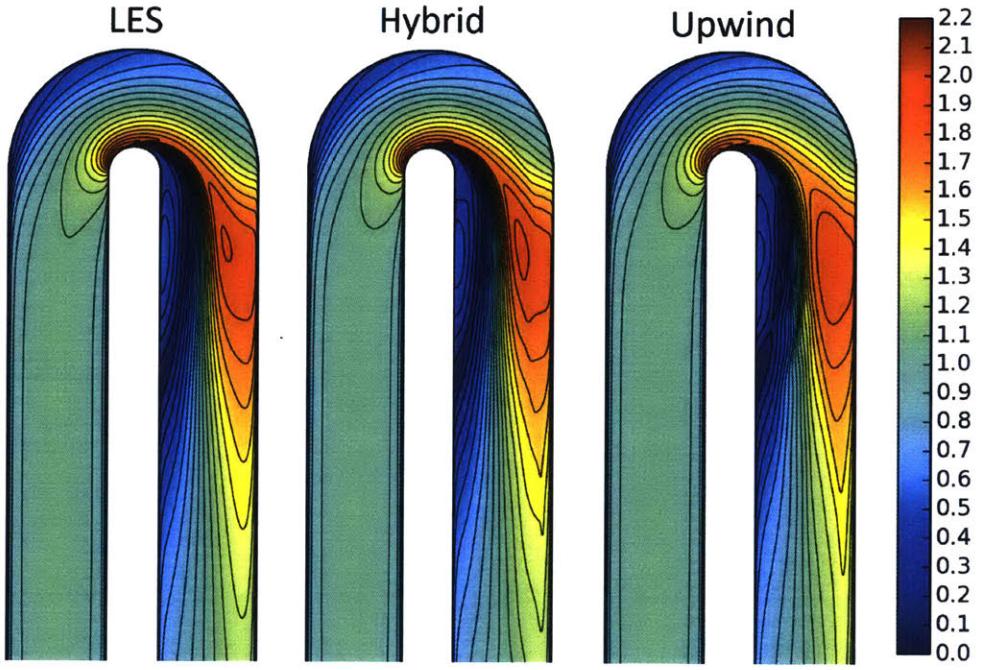


Figure 4-16: Comparison of velocity magnitude field for upwind and hybrid schemes

The pressure field comparison is provided in Figure 4-17. The pressure field of the upwind scheme is visually closer to the LES solution than the hybrid result. The contours at the bend inlet, in the bend outer wall region, and around the low pressure zone of the upwind solution better resemble the time-averaged LES solution. However, both hybrid and upwind solutions do not reproduce the pressure contour curvature in the post-bend region of the domain.

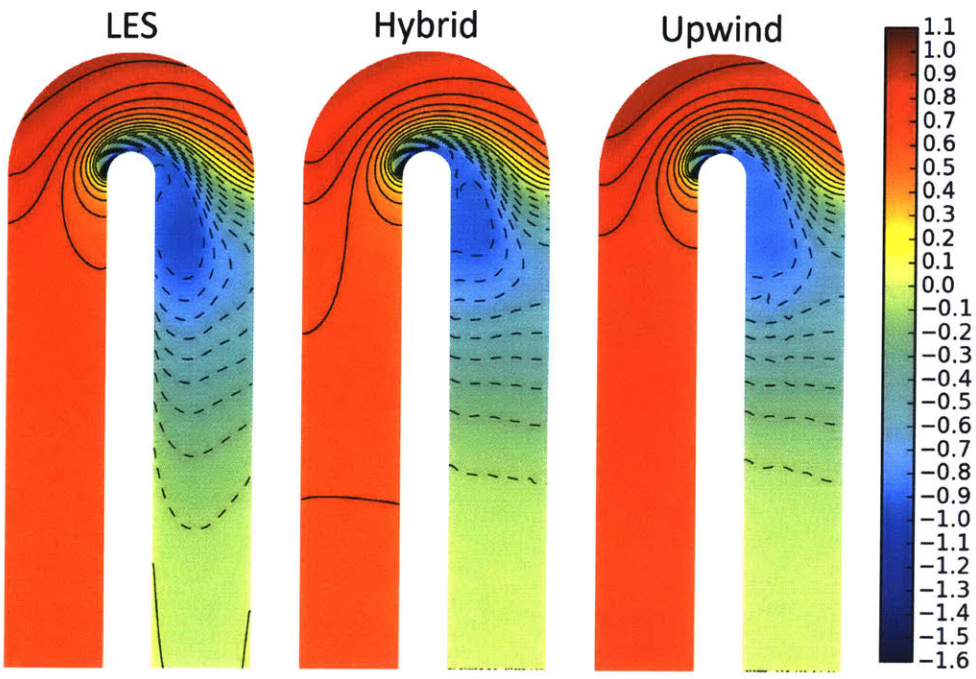


Figure 4-17: Comparison of pressure field for upwind and hybrid schemes

The streamwise velocity profiles at various bend sections from the upwind solution is shown in Figure 4-18. The LES and hybrid profiles are also shown for comparison. The upwind solution agrees well with the LES and hybrid solution except near the bend exit. The bend exit velocity profile is shown in detail in Figure 4-19.

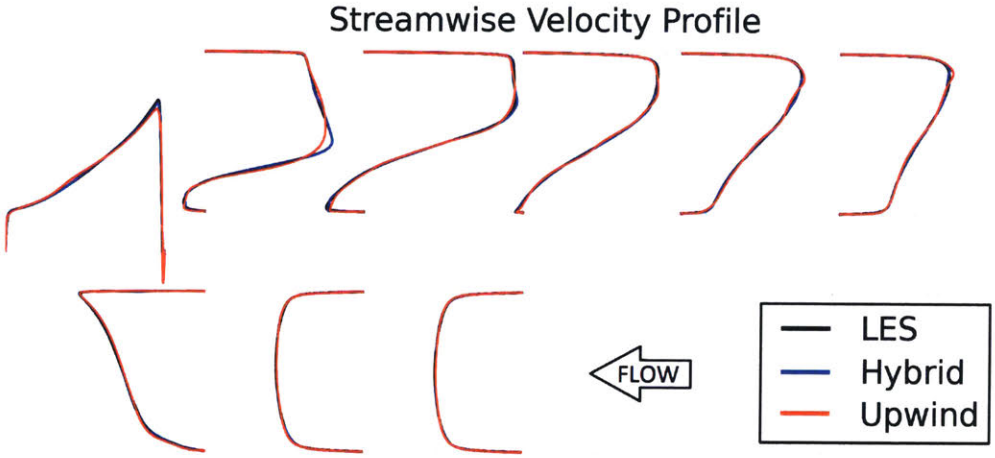


Figure 4-18: Velocity profile at various bend sections for upwind and hybrid schemes

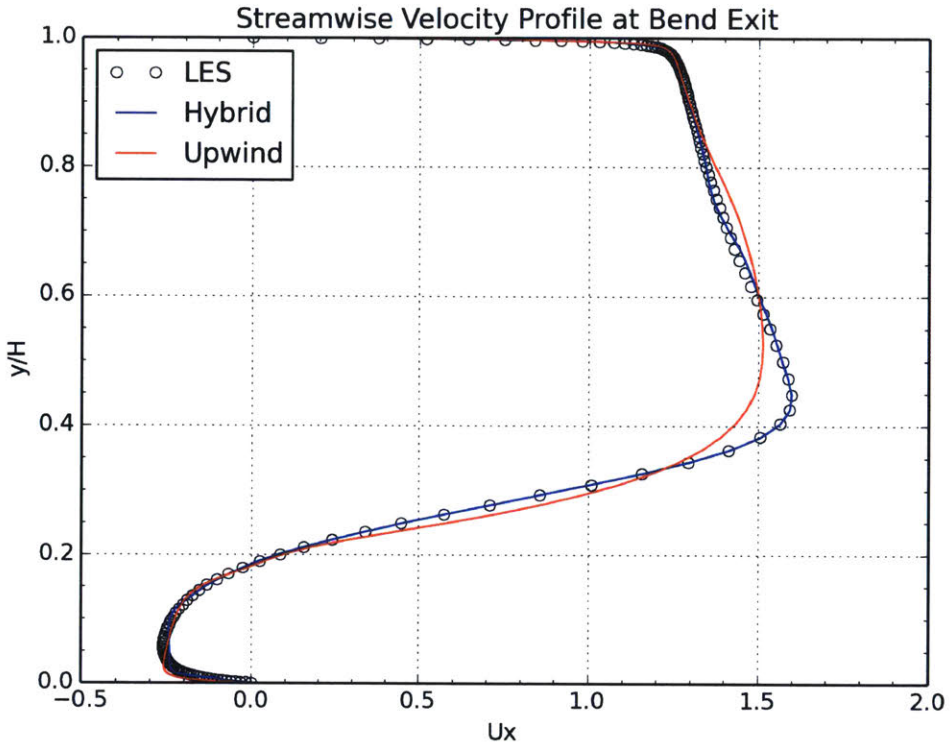


Figure 4-19: Velocity profile at bend exit for upwind and hybrid schemes

The hybrid solution nearly passes through the LES solution. The upwind solution is smoothed out and not able to obtain the same velocity gradient throughout the section due to the dissipation in the scheme. This is believed to be the reason for the stagnating objective value, as the solution can't be improved at this location. This is evident when looking at the turbulent viscosity ratio shown in Figure 4-20. The turbulent viscosity is nearly zero at this location and therefore the diffusion is simply due to the kinematic viscosity and the diffusion from the first order upwind scheme. Due to the diffusion from the scheme itself, the post-bend zone of the turbulent viscosity is slightly narrower, yet is generally similar to the hybrid solution. The high turbulent viscosity streaks in the post-bend region are believed to form to drive the velocity to best match the LES field in the shear region at the edge of the recirculation zone.

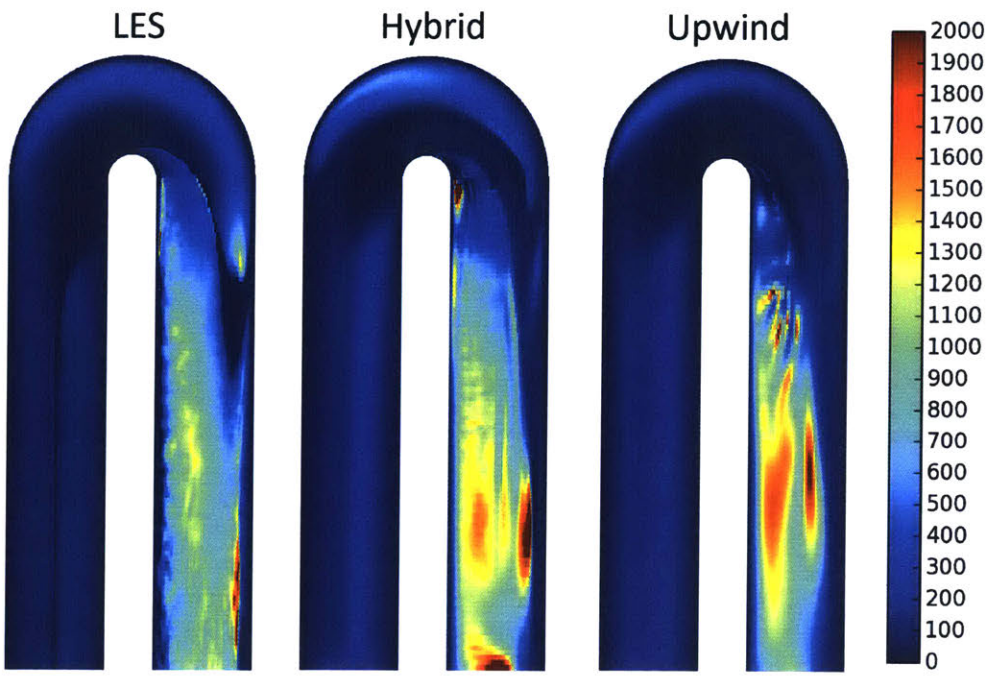


Figure 4-20: Comparison of turbulent viscosity ratio field for upwind and hybrid schemes

The wall shear stress is shown in Figure 4-21. The shear stress distribution of the upwind scheme is not as smooth as the other distributions and is likely due to the streaks of the turbulent viscosity in the region that drives a better least-squares fit of the velocity field. Despite the wavy stress distribution, the reattachment location agrees well with the LES and is slightly better than that obtained with the hybrid scheme. The interpolated reattachment point is provided in Table 4.7.

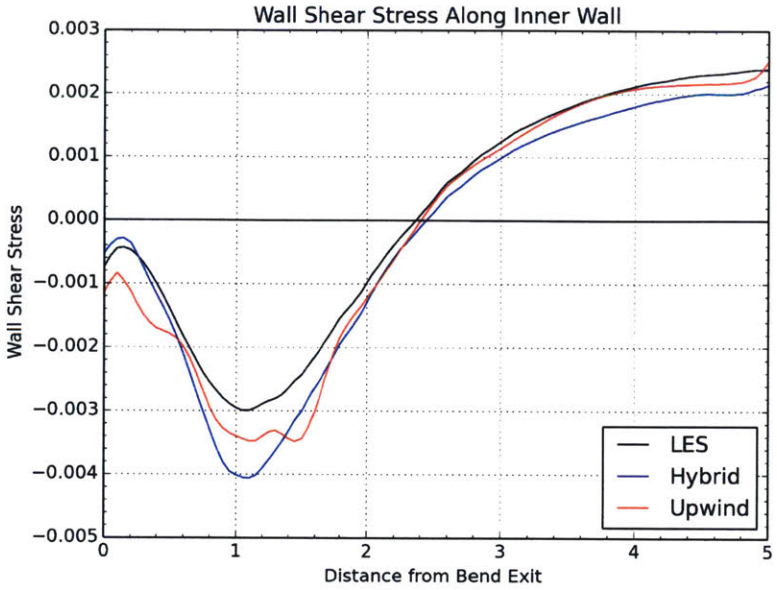


Figure 4-21: Wall shear stress along inner wall in straight exit channel for two convection schemes

Table 4.7: Approximate reattachment location for pressure reduction method

Simulation	**Location**	% **Error**
LES	2.37H	–
Hybrid	2.44H	+3.2%
Upwind	2.40H	+1.6%

The pressure drop across the U-bend for the upwind scheme is provided in Table 4.8 and is compared to the hybrid scheme and the reference LES result. Although the upwind scheme solution has a higher mismatch with the velocity field, the pressure drop with the upwind solution is closer to the LES result. This supports the visual conclusions made earlier based on the pressure field plots.

Table 4.8: Pressure drop comparison with two HIFIR convection schemes

Inlet/Outlet	**LES**	**Hybrid**	**Upwind**
2H / 2H	1.00	0.94 (-5.5%)	0.99 (-0.9%)
2H / 3H	0.76	0.71 (-7.1%)	0.75 (-0.8%)
2H / 4H	0.67	0.62 (-6.4%)	0.67 (+1.2%)
2H / 5H	0.63	0.65 (-5.4%)	0.65 (+3.3%)

Even with the improved pressure drop predictions, the solution error is higher with the upwind scheme in terms of both velocity and pressure mismatch with the LES result. The squared L2 norm values are shown in Table 4.9.

Table 4.9: Velocity and pressure mismatch from LES for upwind

Simulation	Velocity	Pressure
Hybrid	5.5	52.1
Upwind	38.6	57.7

In summary, the convection scheme does show a difference in the velocity and pressure fields and has an impact to the optimized turbulent viscosity. The upwind scheme introduces some artificial dissipation and thus requires less turbulent viscosity is some regions. The impact of the convection scheme to the solution is expected to be reduced with higher-order schemes as the truncation errors are also reduced. Even with the use of pure upwind, the optimized solution is a reasonable match and the resulting turbulent viscosity field is not largely different from the more accurate hybrid scheme.

4.6 Sensitivity to Objective Function Norm

The objective function for the baseline optimization in Section 4.4 is based on the squared L2 norm of the velocity mismatch and is shown again in Equation 4.10, where i is the local cell. In this section, the optimization is repeated using an objective function based on the L1 norm of the velocity mismatch shown in Equation 4.11.

$$J_{\mathrm{L2}^2} = \sum_{i}^{N} (u_i - u_{i,\mathrm{HiFi}})^2 \tag{4.10}$$

$$J_{\mathrm{L1}} = \sum_{i}^{N} \left((u_i - u_{i,\mathrm{HiFi}})^2 + \epsilon \right)^{0.5} \tag{4.11}$$

It can be seen that the individual cell errors using the L1 norm is essentially the square root of the squared L2 norm. Minimizing the L1 error will tend to produce

solutions that have many small and insignificant residuals and few large residuals. In contrast, since the L2 norm penalizes larger residuals more heavily, the solution is expected to produce very few large residuals but at the expense of more small but significant residuals. The L2 error minimization will therefore produce a smoother optimized solution.

For the back-to-back study, both L1 and L2 based objective function minimizations utilized the same k-ω SST initial condition and hybrid convection scheme. The objective convergence of the L1-based method is shown in Figure 4-22. The objective is reduced substantially in several iterations but then stagnates around a value of 1400 for an overall reduction in error by less than one order of magnitude from the initial condition.

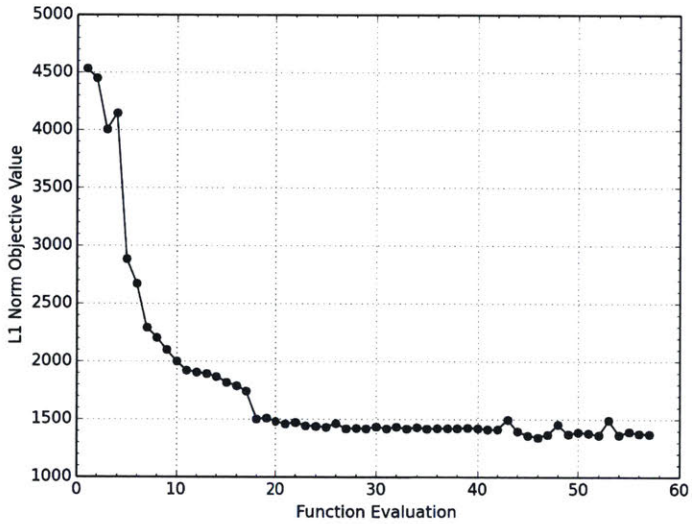


Figure 4-22: Convergence of objective value for L1-based objective function

In order to compare the L1 and L2-based methods directly, the velocity mismatch at each iteration from the L1-based optimization is used to calculate the squared L2 velocity solution error. These solution errors from the L1-based method are shown in Figure 4-23. The convergence of the L1-based method outperforms the L2-based method over the initial 20 iterations, but then stagnates around an objective value of 100. This corresponds to an average error of approximately $3-4\%$ when assuming the error is uniform across all cells. The L2-based method continues to converge to

an objective value around 5, corresponding to an average error of less than 1%.

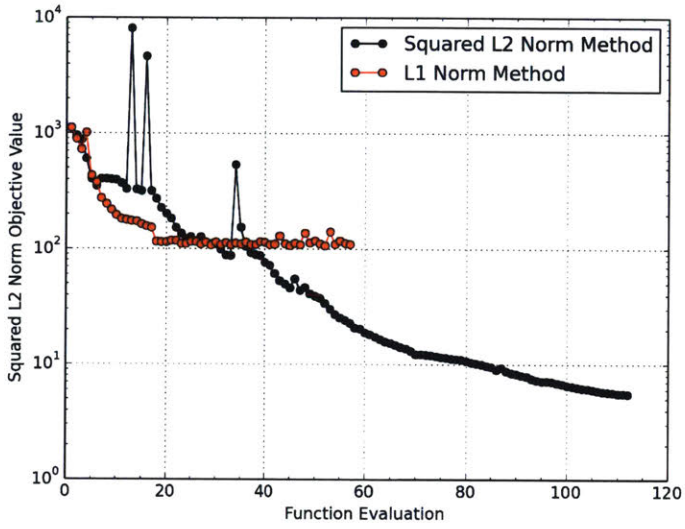


Figure 4-23: Convergence comparison of L1 and L2 objective function based on squared L2 solution error

Since the reference LES data does not contain any outliers and is smooth, the L2-based optimization will provide a better overall fit. The L1 norm based optimization is found to have poor convergence compared to the L2 based optimization. All other studies in this chapter utilize the squared L2-based objective function.

4.7 Impact of Bounds on Turbulent Viscosity

As shown earlier, the LES turbulent viscosity is negative in some regions. For the RANS simulation, if the turbulent viscosity value is allowed to drop to a negative value with greater magnitude than the kinematic viscosity, the diffusion term will switch signs and will lead to a diverging result due to negative effective viscosity, $\nu_{eff} = \nu_t + \nu < 0$. When the turbulent viscosity is a small negative number, specifically $\nu_t > -\nu$, the effective viscosity is greater than zero. This can still lead to instability of the primal solver as the local Peclet number is increased in regions where the turbulent viscosity is reduced below zero.

In order to avoid local negative turbulent viscosity, the control parameter has been based on the log of the turbulent viscosity for all studies. The turbulent viscosity itself can be used as the control parameter in the optimization, but this method would require bounding to ensure non-negativity, whereas using $\log(\nu_t)$ can be unbounded. However, to explore where the optimizer predicts negative values of the turbulent viscosity, the optimization is restarted from the optimized solution by using the unbounded turbulent viscosity as the control parameter. A previously optimized solution is selected as the starting point since a poor initial condition will lead to a nonphysical solution before any insight of the turbulent viscosity field can be gained.

The solution from the hybrid and upwind convective scheme is restarted with the new control parameter. The optimizer is restarted and run for one iteration. The resulting turbulent viscosity field is shown in Figure 4-24 and compared to the negative regions from the LES extracted turbulent viscosity. The figure highlights regions with negative turbulent viscosity in blue. It should be noted that the flow solver diverges with the resulting turbulent viscosity field with both convective schemes.

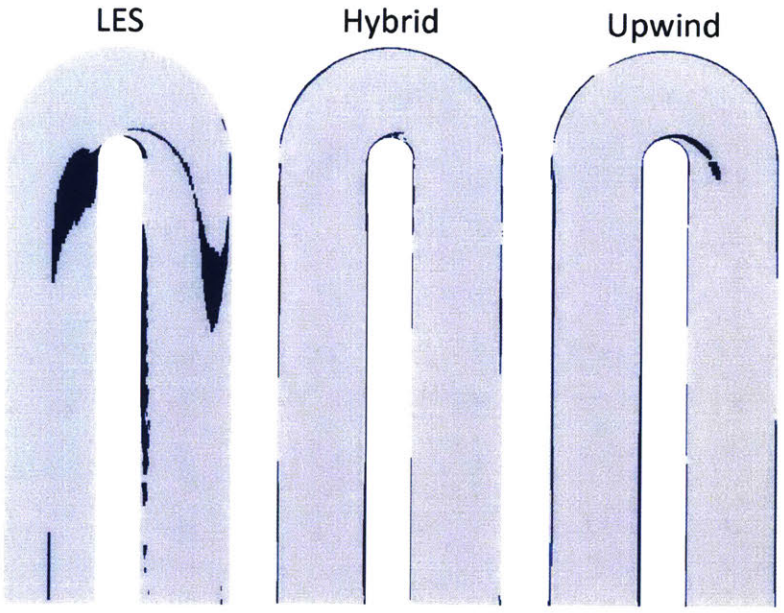


Figure 4-24: Negative regions of turbulent viscosity shown in blue

Both the upwind and hybrid schemes show some negative viscosity along the inner and outer walls. A sliver of negative viscosity is generated near the inner bend along the edge of the separation zone. This is more pronounced in the upwind solution, which makes sense as the optimizer is attempting to compensate for the numerical dissipation at this location. However, this region is still small relative to the LES solution. The negative viscosity observed in the LES solution at the bend inlet is not reproduced.

This study shows that the non-negativity constraint is essential for optimization. However, the unbounded optimization in this section demonstrates that the Boussinesq approximation is not globally valid.

4.8 Use of Pressure Mismatch for Objective Function

All studies thus far have been based on minimizing the velocity solution error. The optimization is repeated in this section using the squared L2 norm of the pressure mismatch as the objective function. The convection scheme used is the hybrid scheme from the baseline optimization. The same initial condition is used for both the velocity and pressure mismatch methods. The reduction of the objective value of the pressure mismatch method is shown in Figure 4-25. The optimization stagnates at an objective value of around 30. In this section, the baseline optimization from Section 4.4 is referred to as the "velocity mismatch reduction method".

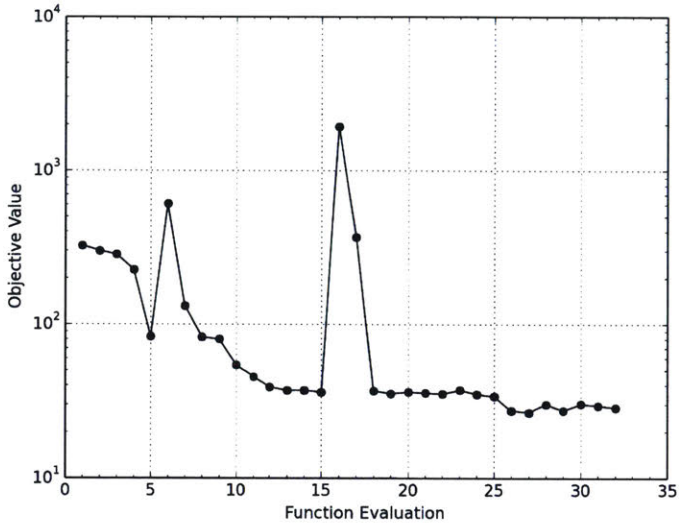


Figure 4-25: Convergence of objective value for pressure mismatch reduction

The comparison of the velocity magnitude field between the time-averaged LES and the two mismatch reduction methods is shown in Figure 4-26. The pressure mismatch reduction solution shows similar traits to the k-ω SST solution in Section 4.4. In particular, the outer wall in the bend region shows lower velocity and the post-bend region displays less mixing as the velocity contours are more straight.

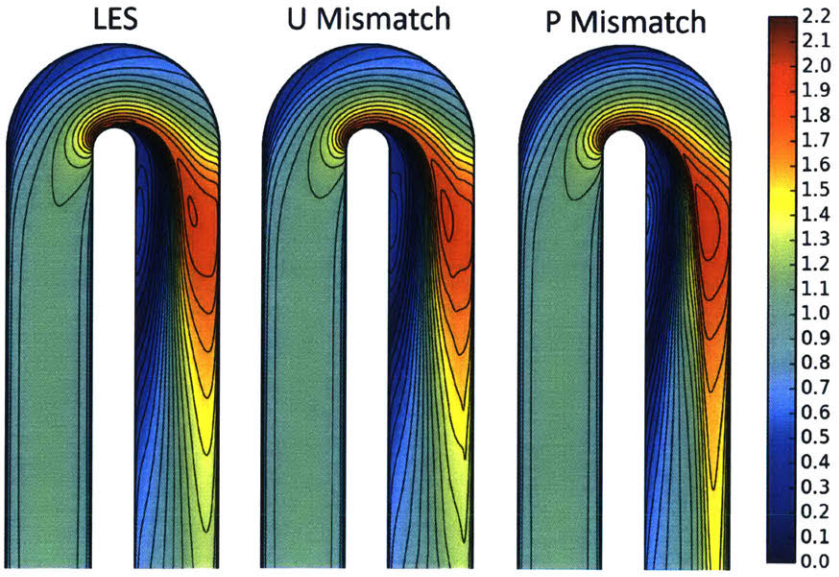


Figure 4-26: Comparison of velocity magnitude field for two mismatch reduction methods

The pressure field comparison is shown in Figure 4-27. Due to the nature of the pressure mismatch reduction method, the solution is visually closer to the LES solution than the velocity mismatch reduction solution. The contours at the bend inlet and in the bend agree well with the LES solution. The post-bend region shows similar contours to that of the velocity mismatch method. This indicates that the curved pressure contours of the LES solution can't be reproduced using turbulent viscosity adjustments, even with the pressure mismatch as the objective function.

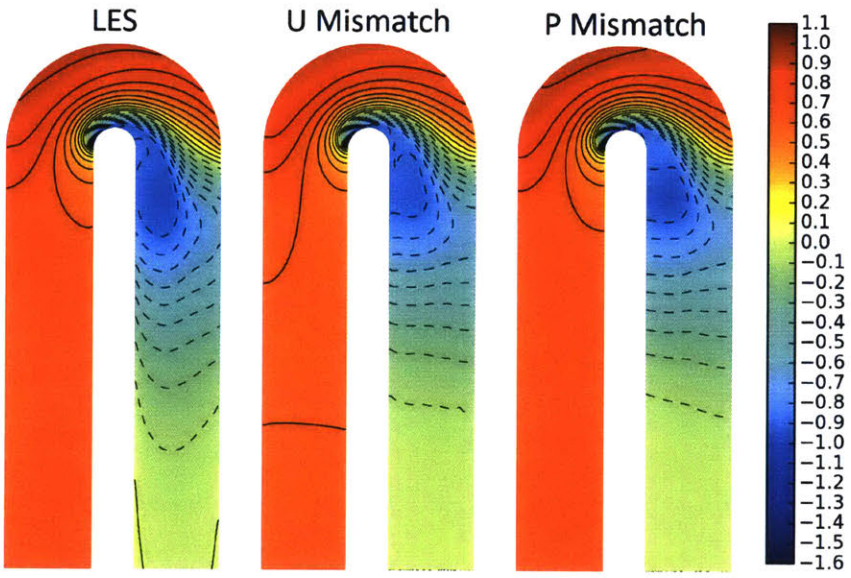


Figure 4-27: Comparison of pressure field for two mismatch reduction methods

The streamwise velocity profiles at various bend sections from the two mismatch reduction methods are shown in Figure 4-28. The pressure mismatch solution generally matches the LES and velocity mismatch profiles, but is not as smooth and appears to be more noisy.

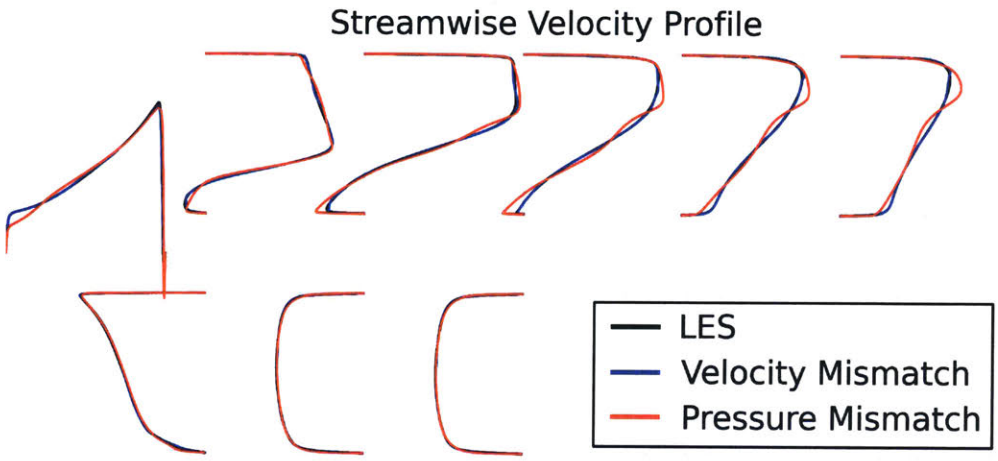


Figure 4-28: Velocity profile at various bend sections for two mismatch reduction methods

The turbulent viscosity ratio field comparison is provided in Figure 4-29. The resulting turbulent viscosity of the pressure mismatch reduction method is significantly different from the LES and velocity mismatch reduction solutions and is more similar in contour to the k-ω SST solution from Figure 4-9. The increase in turbulent viscosity in the bend outer wall region is absent with the pressure mismatch reduction method.

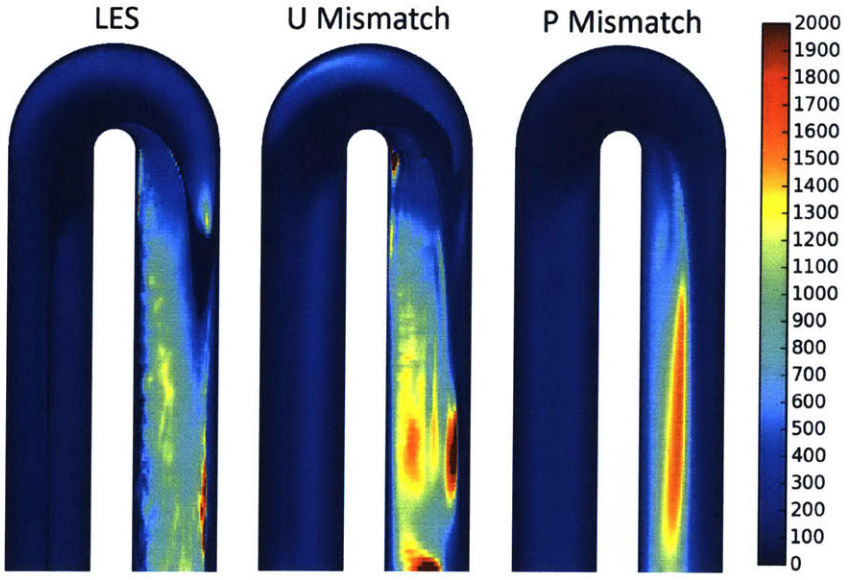


Figure 4-29: Comparison of turbulent viscosity ratio field for two mismatch reduction methods

The wall shear stress along the inner wall in the post-bend region is shown in Figure 4-30. The two mismatch reduction methods show smooth curves, but the pressure mismatch reduction method produces wall shear values that are about 1 to 1.5 times greater in magnitude than the LES solution away from the reattachment point. The interpolated reattachment point for the two mismatch reduction methods is shown in Table 4.10. The reattachment point with the pressure reduction method is further downstream than both the LES and velocity mismatch reduction method.

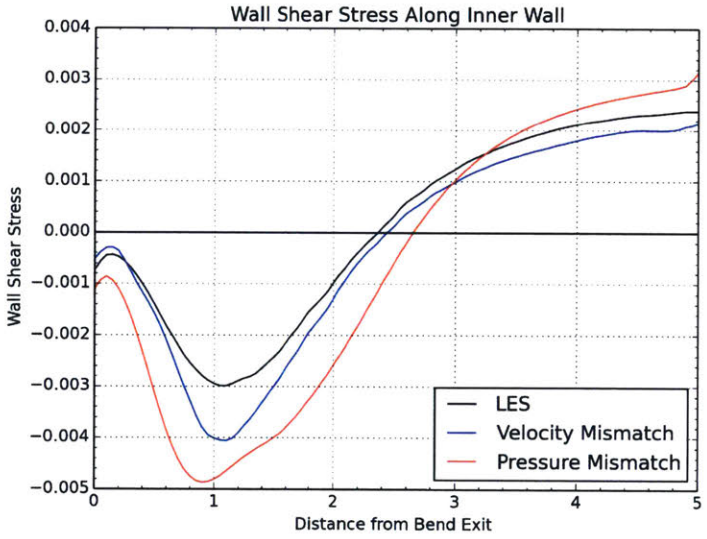


Figure 4-30: Wall shear stress along inner wall in straight exit channel for pressure reduction method

Table 4.10: Approximate reattachment location for pressure reduction method

Simulation	**Location**	% **Error**
LES	2.37H	–
U Mismatch	2.44H	+3.2%
P Mismatch	2.66H	+12.4%

The pressure mismatch reduction method leads to a better prediction on the pressure drop across the bend due to the reduced pressure solution error vs LES. The pressure drop across the bend at various sections is shown in Table 4.11. The

pressure reduction method provides a pressure drop prediction within approximately 2% of the LES and is an improvement over the prediction with the velocity mismatch method.

Table 4.11: Pressure drop comparison for pressure mismatch reduction

Inlet/Outlet	LES	U Mismatch	P Mismatch
2H / 2H	1.00	0.94 (-5.5%)	1.00 (-0.1%)
2H / 3H	0.76	0.71 (-7.1%)	0.74 (-2.1%)
2H / 4H	0.67	0.62 (-6.4%)	0.66 (-1.1%)
2H / 5H	0.63	0.65 (-5.4%)	0.63 (-0.1%)

As expected, the pressure mismatch method has a smaller squared L2 norm solution error than the velocity mismatch method for the pressure solution. However, the velocity solution error is significantly larger with the pressure mismatch method. The squared L2 norm solution errors are shown in Table 4.12.

Table 4.12: Solution error for the velocity and pressure mismatch reduction methods

Simulation	Velocity	Pressure
U Mismatch Reduction Method	5.5	52.1
P Mismatch Reduction Method	249.3	26.4

4.9 Impact of Limited Experimental Data on Optimization

All studies to this point have been based on objective functions that use the full domain solution to compute the solution error. As a way to check how the optimization fares with limited data, an objective function based only on an inlet and outlet velocity profile mismatch is constructed. In some experiments, full domain data may not be measurable. For example, a multi-holed traverse probe can be used to scan along a channel section upstream and downstream of the U-bend to measure

the velocity profile. For this study, the inlet section profile is taken a distance of 4H from the domain inlet, or 1H upstream of the bend entrance. The exit section profile is taken at a distance of 2H from the domain exit, or 3H downstream of the bend exit. The inlet section is prior to any significant streamline curvature. The exit plane is downstream of the recirculation zone and reattachment point. For this case, the squared L2 norm objective function is used, but only the cell values at the two sections are used for the velocity mismatch.

The baseline optimization from Section 4.4 is labeled as the "full domain" solution, as the optimization makes use of the velocity mismatch in all cells. The study case using only limited profile data is referred to as the "inlet/outlet" case. Both cases use the standard k-ω SST solution as the initial condition. Due to the large reduction in control parameters, the optimization convergence with the limited data is well behaved. As shown in Figure 4-31 the convergence is steady and nearly exponential.

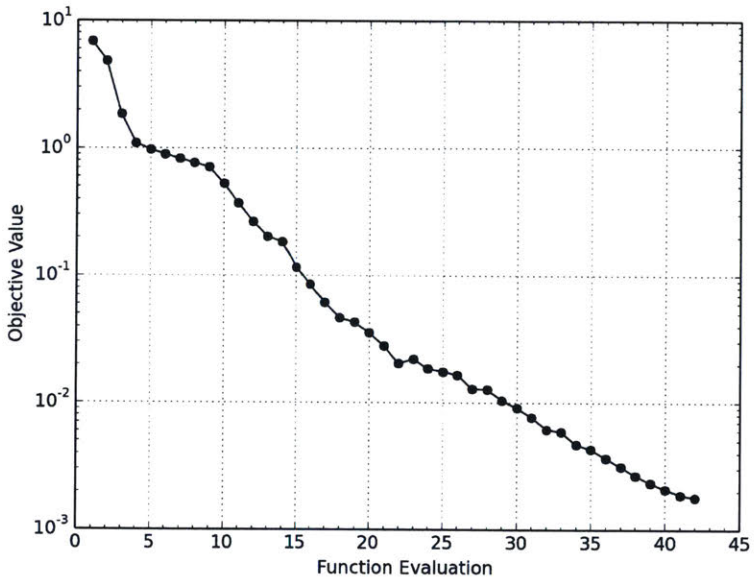


Figure 4-31: Convergence of objective value for the limited inlet/outlet data case

The velocity field of the optimized solution using the limited inlet/outlet data is compared to the full domain data optimization and the time-averaged LES in Figure 4-32. The inlet and outlet sections where the velocity mismatch is calculated is shown as white lines in the velocity plot. Areas outside of the inlet and outlet section are

blind to the optimizer and the influence is felt within 1 channel width, resulting in poor match elsewhere in the domain. The inlet/outlet case shows several visual differences with the LES solution. The outer wall region in the bend shows excessively low velocity. The peak post-bend velocity is also lower than the LES and full domain data case. The recirculation zone is also longer in size. The contours show artificial kinks in the velocity field where the outlet section profile data is taken from as the optimizer minimizes the objective value.

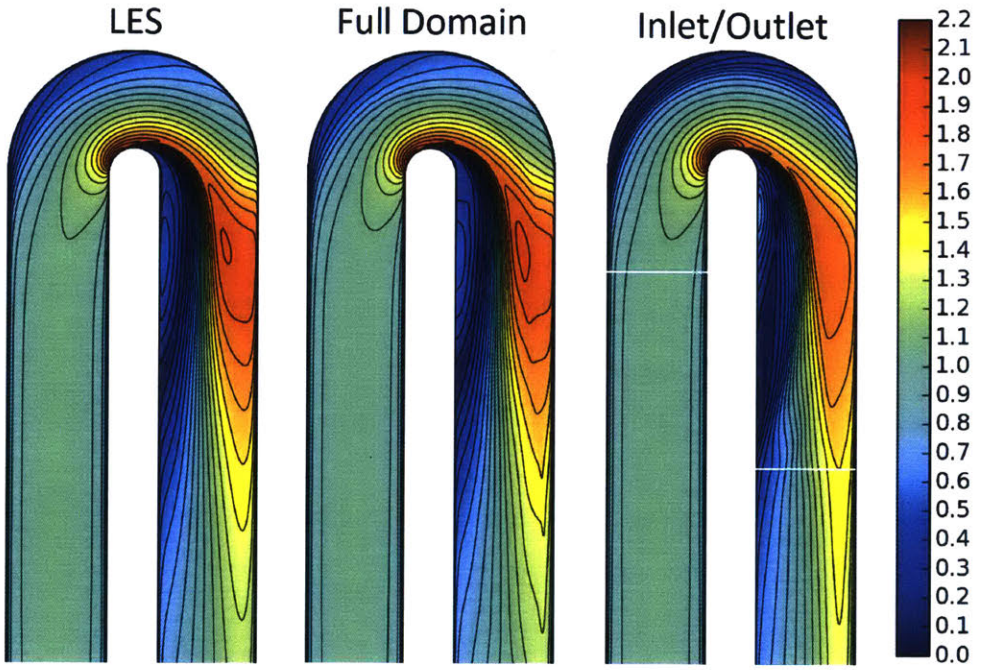


Figure 4-32: Comparison of velocity magnitude field for the limited inlet/outlet data case

The pressure field is shown in Figure 4-33. Visually, the pressure field of the inlet/outlet case does not show as much overall mismatch as observed in the velocity fields. However, the low pressure zone corresponding to the recirculation is smaller in size and magnitude.

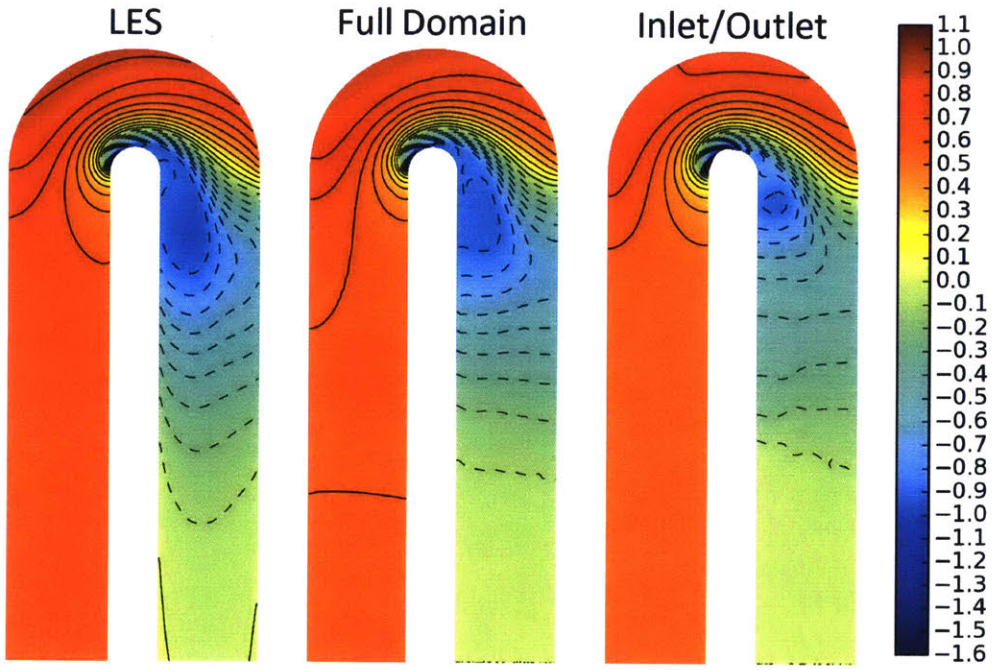


Figure 4-33: Comparison of pressure field for the limited inlet/outlet data case

The solution error is more evident by plotting the velocity profiles at various sections throughout the domain, as shown in Figure 4-34. The outer wall velocity deficit is clear in the mid-bend section. The solution is well matched at the outlet section, which is 3H from the bend exit. The velocity profiles corresponding to this section are plotted in further detail in Figure 4-35. At this section alone, the inlet/outlet case performs better than the full domain case and matches the LES data reasonably well. This is expected as the full domain case attempts to minimize the solution error across all cells, whereas the inlet/outlet case is solely focused on the two sections.

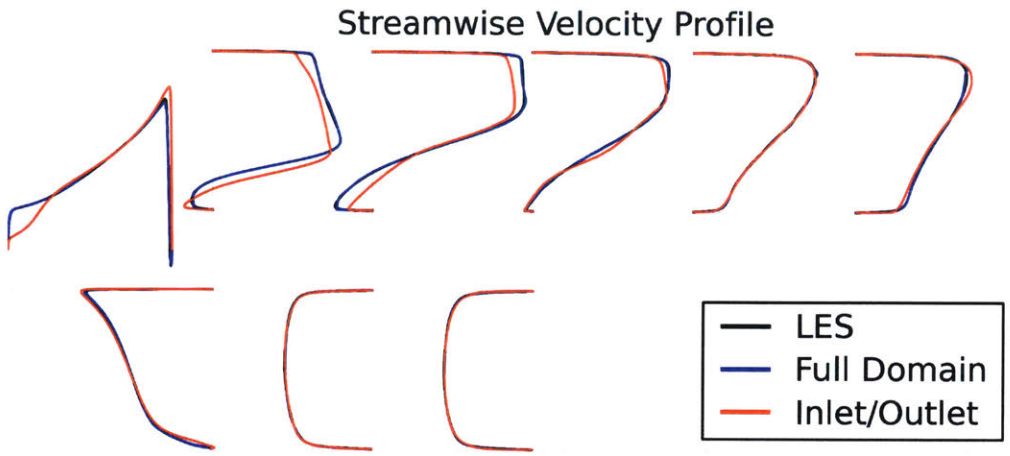


Figure 4-34: Velocity profile at various bend sections for the limited inlet/outlet data case

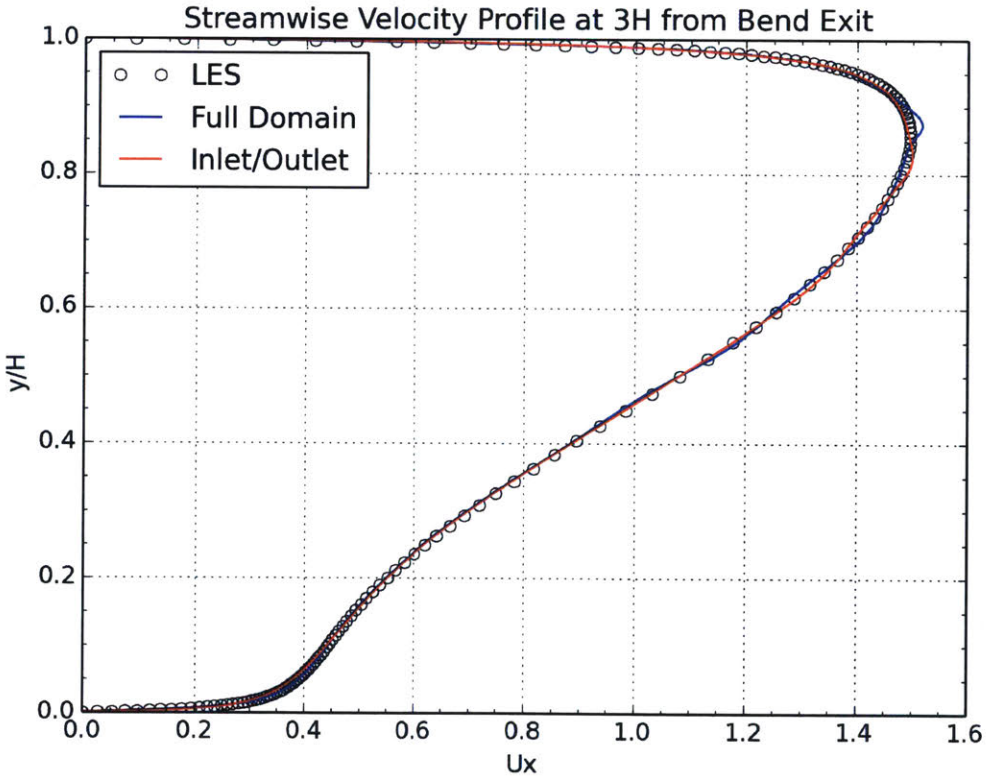


Figure 4-35: Velocity profile at a distance of 3H from bend exit

The turbulent viscosity ratio field displays how the optimizer attempts to reduce the objective value at the two sections shown in white in Figure 4-36. Only the

turbulent viscosity in and near the region of the sections used in the objective function are modified. The influence does extend further upstream as a local increase in the turbulent viscosity is produced in the separation area. No change is observed in outer bend wall as seen in the full domain case. The resulting turbulent viscosity field displays oscillations around the outlet section.

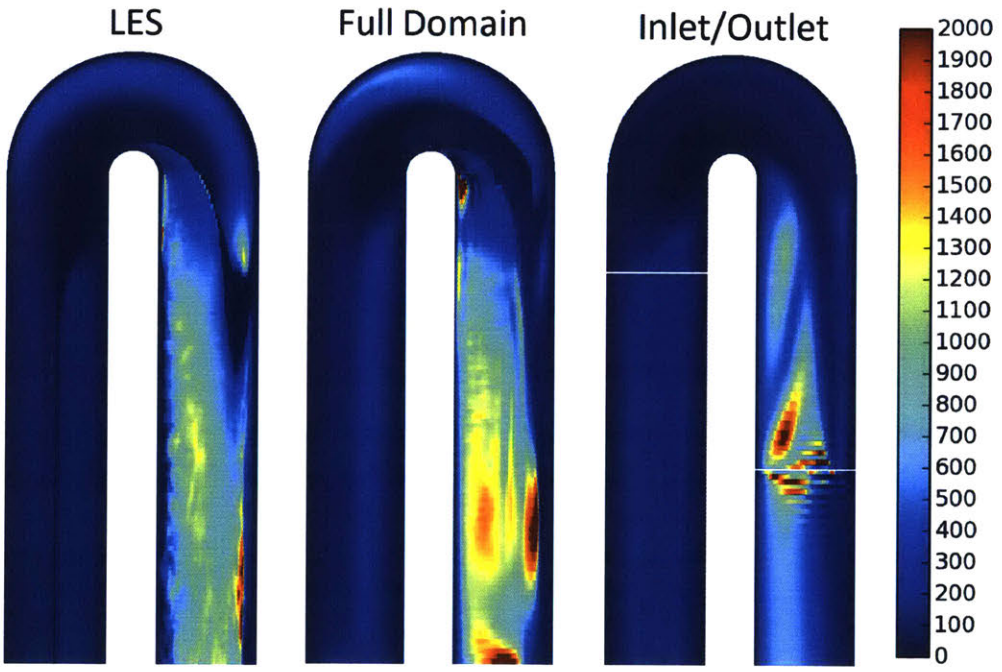


Figure 4-36: Comparison of turbulent viscosity ratio field for the limited inlet/outlet data case

The wall shear stress along the inner wall of the straight exit section is shown in Figure 4-37. The shear stress distribution is significantly wavy. The distribution shows a near flat section at the outlet section corresponding to a distance of 3H from the bend exit where the optimizer artificially matches the local shear stress of the LES. The shear stress is otherwise incorrect at all other locations. The reattachment point is further downstream than that of the two other solutions and is provided in Table 4.13.

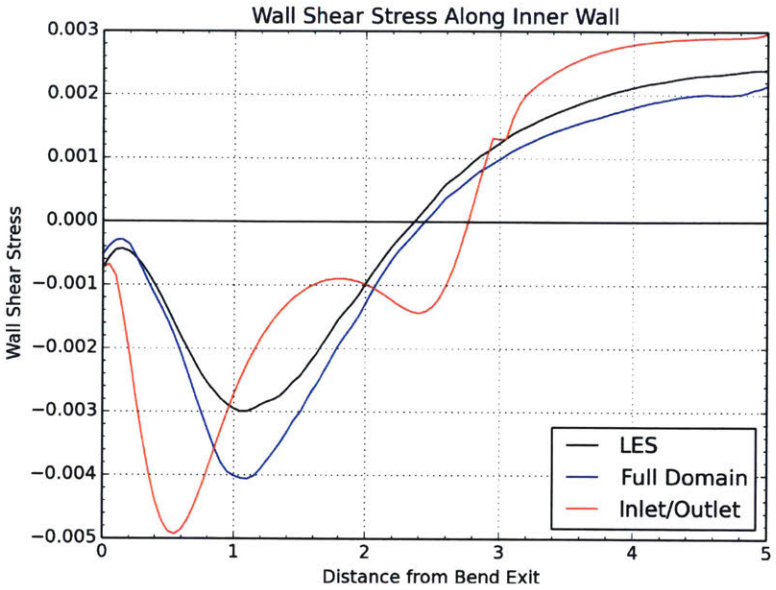


Figure 4-37: Wall shear stress along inner wall in straight exit channel for the limited inlet/outlet data case

Table 4.13: Approximate reattachment location for the limited inlet/outlet data case

Simulation	**Location**	**% Error**
LES	2.37H	–
Full Domain	2.44H	+3.2%
Inlet/Outlet	2.77H	+16.9%

Based on the pressure field shown earlier, it is not a surprise that the pressure drop prediction is worse than the full domain case. The error, shown in Table 4.14, is over double the error from the full domain case.

Table 4.14: Pressure drop comparison for limited inlet/outlet data

Inlet/Outlet	**LES**	**Full Domain**	**Inlet/Outlet**
2H / 2H	1.00	0.94 (-5.5%)	0.85 (-14.9%)
2H / 3H	0.76	0.71 (-7.1%)	0.65 (-14.9%)
2H / 4H	0.67	0.62 (-6.4%)	0.58 (-13.6%)
2H / 5H	0.63	0.65 (-5.4%)	0.56 (-11.3%)

The overall full domain solution error of the two cases is shown in Table 4.15. The squared L2 norm error with the limited inlet/outlet data is excessively high and

worse than the standard k-ω SST result shown earlier in Table 4.4.

Table 4.15: Solution error for the limited inlet/outlet vs full domain data

Simulation	Velocity	Pressure
Full Domain	5.5	52.1
Inlet/Outlet	1627	643

In summary, the use of limited data significantly impacts the overall solution error. This is due to the turbulent viscosity formulation that is only a function of space. Since the adjusted turbulent viscosity is more local to the sections used for the objective function, the solution is dependent on the initial condition. A more sophisticated turbulence model that can be trained based on flow parameters is believed to significantly improve the accuracy when using limited data. This is further discussed in Chapter 6.

4.10 Conclusion

In this chapter, it is shown that the HIFIR approach is capable of substantially decreasing the solution error using a prescribed turbulent viscosity on a U-bend channel. The most successful approach explored is an optimization framework based on the squared L2 norm of the velocity mismatch coupled with the use of the log of the turbulent viscosity as the control parameter to guarantee positive turbulent viscosity values and allow an unbounded optimization. The use of limited data is not recommended with the current prescribed turbulent viscosity approach as the total solution error is dependent on the initial condition.

Chapter 5

Performance of HIFIR model on Adjusted Geometry

5.1 Introduction

The turbulence model introduced in Chapter 4 is a simple eddy viscosity model and has been shown to be effectively tuned to match a high fidelity simulation. One limitation with this high-fidelity trained RANS (HIFIR) model is that the turbulent viscosity is prescribed and is only a function of space. To demonstrate the performance of this method on new geometries, the HIFIR model is solved on an adjusted U-bend geometry using the optimized turbulent viscosity from the baseline case in Chapter 4. The U-bend studied in this chapter has a middle wall thickness half of that used in Chapter 4 and further discussed in Section 5.2.

LES and k-ω SST RANS simulations are also performed on the new U-bend geometry for comparison. Section 5.3 details the LES solutions between the two bend geometries. The HIFIR result on the adjusted bend geometry is compared to the LES and k-ω SST solution in Section 5.4.

5.2 Geometry Adjustment and Simulation Methods

5.2.1 New U-bend Geometry

The middle wall thickness of the adjusted U-bend is half of the original thickness, as shown in Figure 5-1. The channel width of H and the straight channel length of 5H is unchanged. The thinner middle wall essentially creates a tighter bend, as the mean bend radius is reduced from 0.75H to 0.625H. The mesh topology and cell count used for the adjusted U-bend is the same as the original bend.

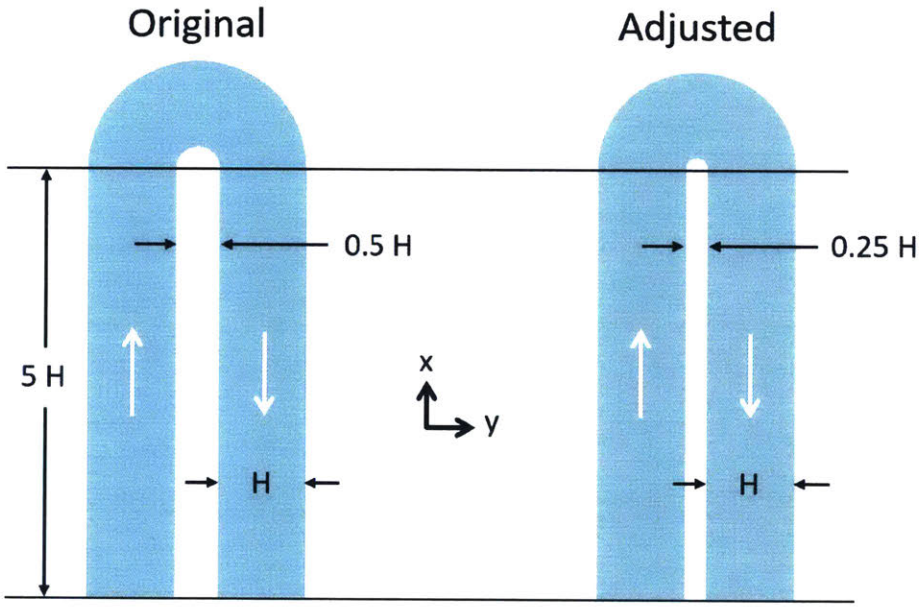


Figure 5-1: Domain geometry used for U-bend channel LES

5.2.2 Simulation Methods

The LES on the adjusted bend geometry is executed in the same manner used in Chapter 4. The first 20 seconds are attributed to the start-up transient. After this point, the next 230 seconds are time-averaged and collapsed (averaged) along the spanwise z-direction to compute a mean 2D field. The boundary conditions are not changed from the original U-bend simulation. The adjusted U-bend flow solution is

also obtained using the standard k-ω SST RANS model in OpenFOAM. Again, the simulation setup is identical to that used in Chapter 4, with the exception of the mesh and geometry itself.

In Chapter 4, the HIFIR model solution is part of an optimization problem to find the turbulent viscosity field and requires the use of a higher fidelity result. Here, the previously optimized solution from the original U-bend geometry is used. Since the mesh topology and cell count are the same, the turbulent viscosity field is mapped directly cell-to-cell onto the new mesh. The RANS fluid solver is then converged with this prescribed turbulent viscosity field to obtain the solution.

5.3 LES Solution Comparison

In this section, the time-averaged LES solution of the original and adjusted U-bend is compared to estimate the true sensitivity of the thinner middle wall and tighter bend radius. The velocity and pressure fields are shown in Figure 5-2 for both bend geometries. The adjusted bend geometry results in larger separation. The larger recirculation zone reduces the effective width of the channel and increases the post-bend peak velocity. The increased velocities and larger separation results in more loss. This is evident by the increase in inlet pressure with the adjusted geometry. The exit pressure is zero for both simulations, and thus the adjusted U-bend requires higher pressure to drive the flow as the pressure drop is greater across the bend. The pressure in the eye of the recirculation zone is also lower with the adjusted geometry.

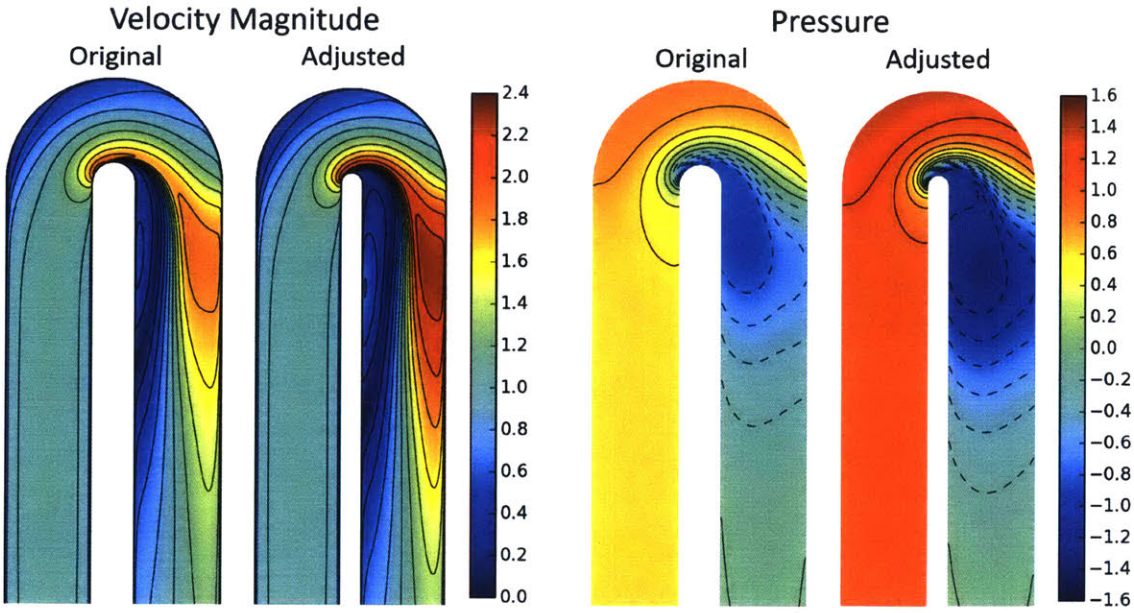


Figure 5-2: Comparison of LES velocity magnitude and pressure field for the two U-bend geometries

Several key observations from the pressure and velocity fields are shown in Table 5.1. The pressure drop in the table is based on the drop in pressure from an inlet section at 2H from the bend inlet to an outlet section at 5H from the bend exit (domain exit). The reattachment point is the point of zero wall shear stress along the inner wall of the post-bend straight section, measured from the bend exit.

Table 5.1: Comparison of LES solution for several parameters on the two U-bend geometries

Parameter	**Original**	**Adjusted**	**Change**
Peak Inner Bend Velocity	2.09	2.33	+11%
Peak Post-Bend Velocity	1.89	2.29	+21%
Pressure Drop	0.63	1.07	+70%
Reattachment Point	2.37 H	3.03 H	+28%

The extracted turbulent viscosity ratio, ν_t/ν, and the computed mean turbulent kinetic energy from the normal Reynolds stress components are shown in Figure 5-3 for the two geometries. The form of the turbulent viscosity field is similar between the two solutions, but the adjusted geometry shows larger values of turbulent viscosity and more coverage in the post-bend region. The straight inlet channel section and outer

bend wall region both show similar turbulent viscosity values in the two solutions. The turbulent kinetic energy (TKE) in the bend region is also similar between the two solutions. The post-bend TKE is much larger with the adjusted geometry, which supports the larger pressure loss due to energy transfer into fluctuating velocities.

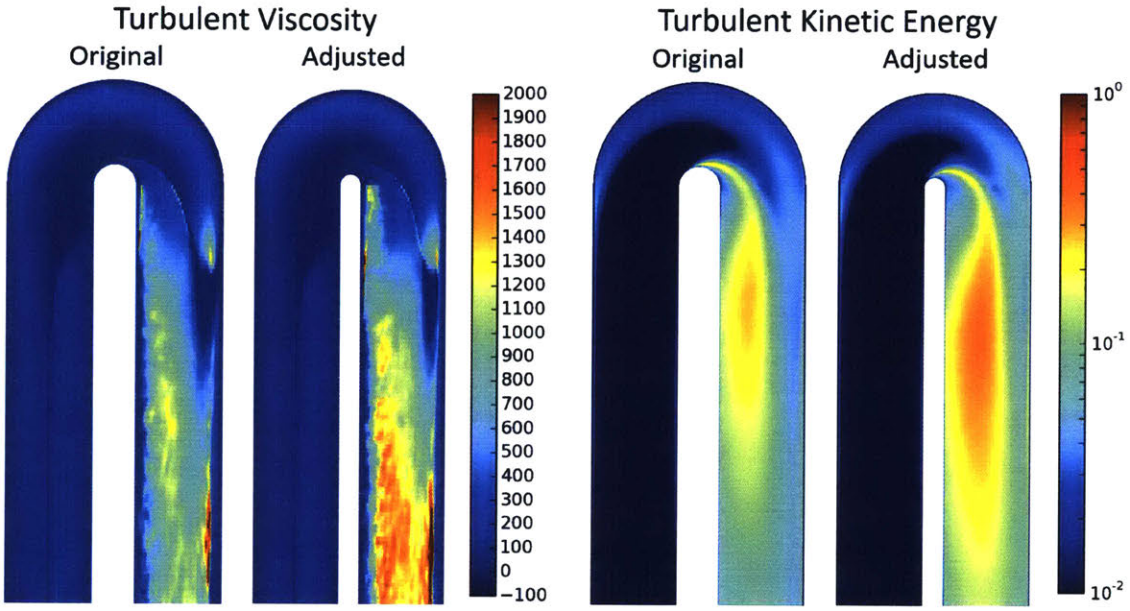


Figure 5-3: Comparison of LES turbulent viscosity ratio and TKE for the two U-bend geometries

Although the geometry adjustment is relatively subtle, the thinner middle wall leads to a substantial increase in the pressure drop and turbulence. The overall contours are similar in form between the two simulations and both display the same key features, such as Gortler instability at the bend outer wall and separation and reattachment along the inner wall. This makes the adjusted geometry a good test case for the HIFIR model.

5.4 HIFIR Solution on Adjusted U-bend Geometry

In this section, the HIFIR solution with the previously optimized turbulent viscosity is compared to the LES and k-ω SST solutions on the adjusted U-bend geometry.

5.4.1 Velocity and Pressure Fields

The velocity magnitude is shown in Figure 5-4 for the three solution methods. The HIFIR model shows a lower post-bend peak velocity and is likely due to a later separation point along the inner wall causing less blockage downstream. The HIFIR model shows contours closer to the LES solution at the outer bend wall and near the domain exit compared to k-ω SST solution. The k-ω SST solution shows a longer recirculation zone and peak velocity contour island similar to the results in Chapter 4, characteristic of reduced mixing.

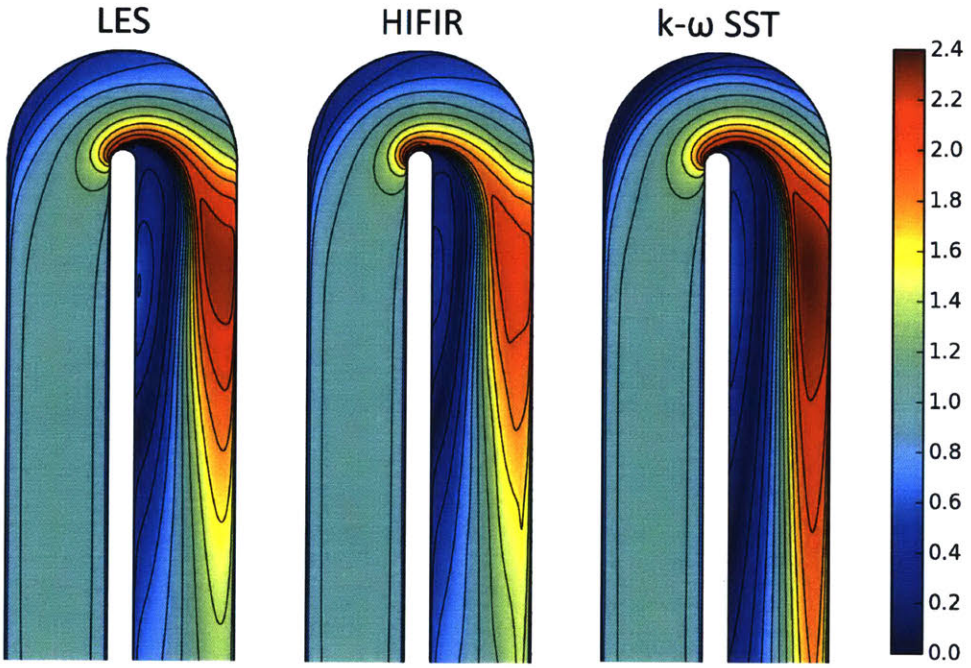


Figure 5-4: Comparison of velocity magnitude field for the adjusted U-bend

The pressure field is shown in Figure 5-5. The HIFIR model shows a smaller pressure drop due to the lower inlet pressure, whereas the k-ω SST solution over predicts the pressure drop. The post-bend contours of the k-ω SST better match the LES solution.

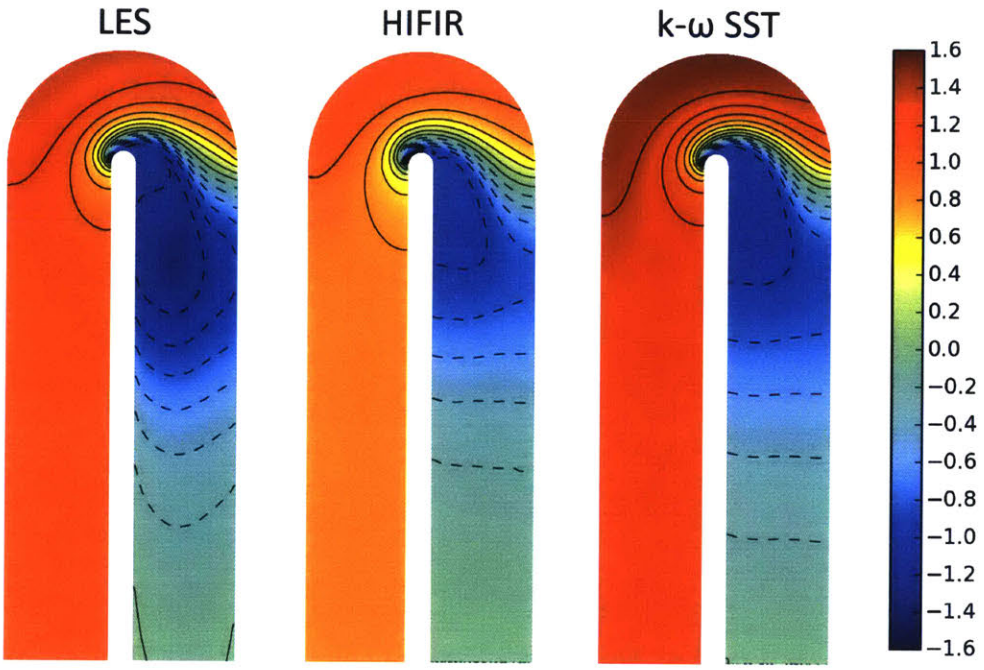


Figure 5-5: Comparison of pressure field for the adjusted U-bend

The overall solution error between the two RANS solutions and the LES are computed using the squared L2 norm of the velocity and pressure mismatch. The solution error is provided in Table 5.2. The HIFIR solution has almost double the error compared to the k-ω SST solution. Both solutions are relatively poor, but this shows that the HIFIR model optimized on one geometry may not directly apply to a similar, but adjusted, geometry.

Table 5.2: RANS solution error for the adjusted U-bend geometry

Simulation	**Velocity**	**Pressure**
k-ω SST	1027	1269
HIFIR	1719	2365

5.4.2 Velocity Profiles

The streamwise velocity profile at various sections along the domain is provided in Figure 5-6. The inlet sections are similar between all three solutions. The HIFIR

result better matches the LES in the outer bend region and does not show the velocity deficit seen with k-ω SST. The HIFIR profiles near the inner wall in the bend region and immediately after the bend show clear discrepancies with the LES solution. However, further downstream, the HIFIR profiles are better matched to the LES solution.

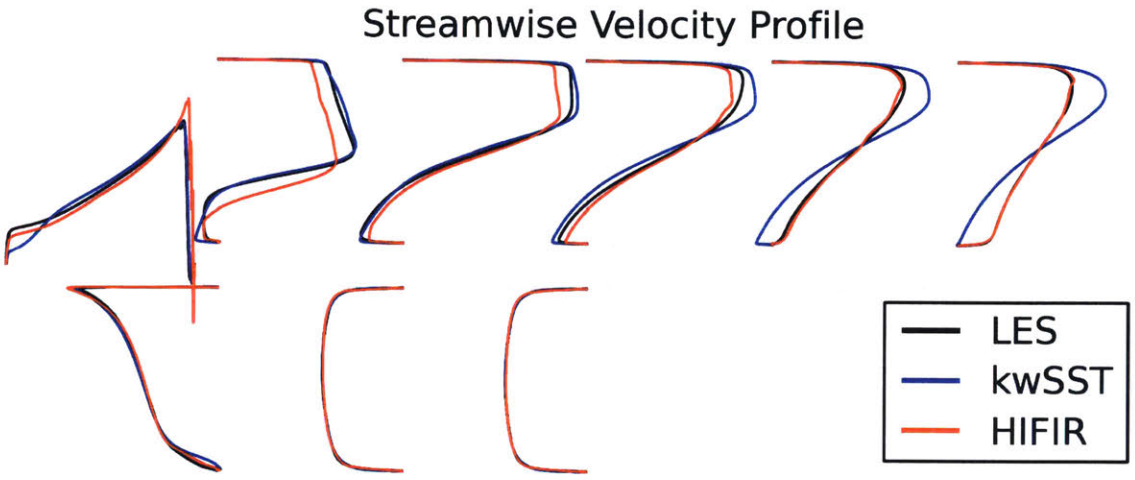


Figure 5-6: Velocity profile at various bend sections for the limited inlet/outlet data case

The velocity profile at the bend exit is shown in detail in Figure 5-7. The k-ω SST solution nearly matches the LES solution, but deviates near the inner wall. The HIFIR result is incorrect nearly everywhere, with only a match to LES very close to inner wall. Moving away from the inner wall, the recirculation zone width with HIFIR is clearly smaller as the velocity becomes positive at a shorter distance off the wall. The peak velocity is also under-predicted due to the less blockage.

Moving downstream, the HIFIR solution matches better and is shown in detail in Figure 5-8. The k-ω SST solution shows the same behavior seen in Chapter 4, where the profile is not as uniform as the LES result. The better agreement of the HIFIR model at this location is due to the minimal change in the turbulent viscosity since it is farther from the bend and is dominated by the shear mixing.

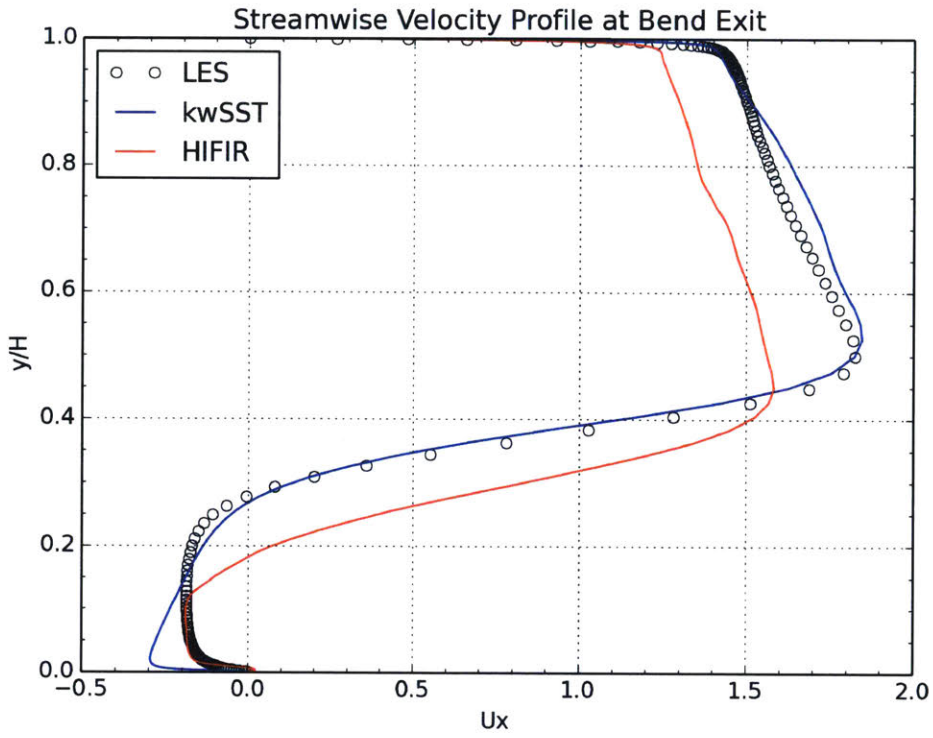


Figure 5-7: Velocity profile at the bend exit

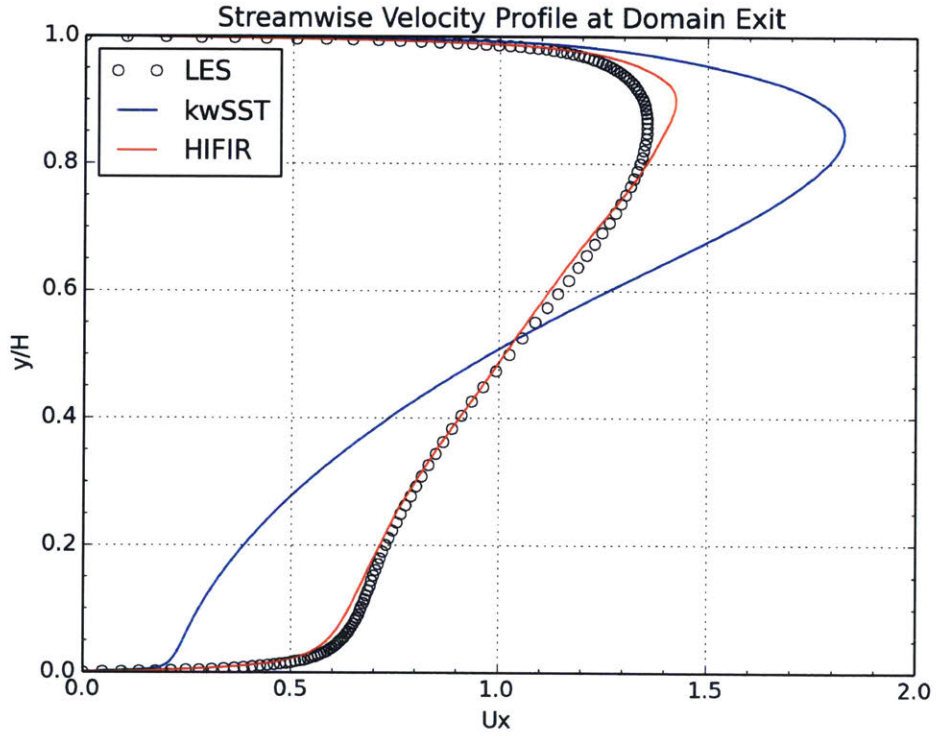


Figure 5-8: Velocity profile at the domain exit

5.4.3 Reattachment Location and Pressure Drop

Despite the larger overall solution error, the wall shear at the inner wall is better predicted with the HIFIR model and is shown in Figure 5-9. As shown in Section 5.3, the difference at the inner wall in the post-bend region did not show a significant change in the turbulent viscosity, therefore the previously optimized turbulent viscosity field is still reasonable along the inner wall. The reattachment point from the HIFIR model is therefore more accurate than the k-ω SST and is shown in Table 5.3.

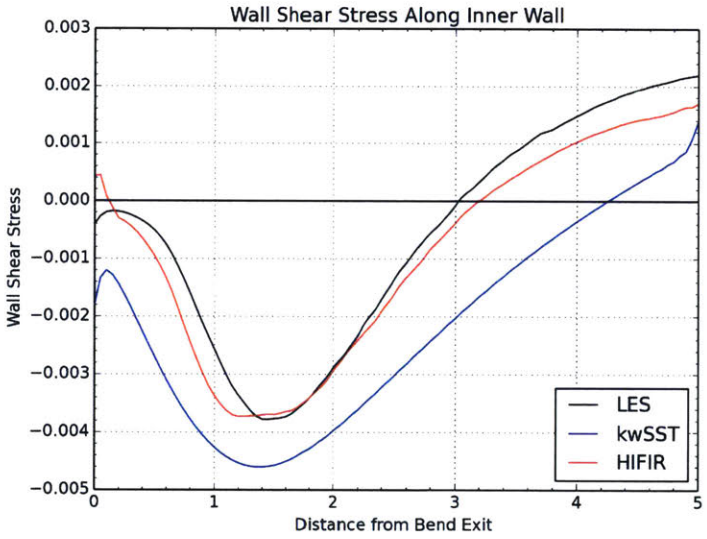


Figure 5-9: Wall shear stress along inner wall in straight exit channel for adjusted U-bend

Table 5.3: Approximate reattachment location for the adjusted U-bend

Simulation	**Location**	**% Error**
LES	3.03H	–
k-ω SST	4.26H	+40.4%
HIFIR	3.20H	+5.6%

The pressure drop comparison is provided in Table 5.4. Overall, the k-ω SST and HIFIR models miss by about the same magnitude, but in opposite directions. The HIFIR model under-predicts the pressure drop by about 22% compared to the reference LES solution.

Table 5.4: Pressure drop comparison for limited inlet/outlet data

Inlet/Outlet	LES	k-ω SST	HIFIR
2H / 2H	1.82	2.00 (+10.3%)	1.41 (-22.6%)
2H / 3H	1.38	1.69 (+22.6%)	1.06 (-22.9%)
2H / 4H	1.16	1.46 (+25.6%)	0.91 (-21.8%)
2H / 5H	1.07	1.31 (+22.3%)	0.85 (-21.0%)

5.4.4 Turbulent Viscosity Field

The turbulent viscosity ratio of the three methods is shown in Figure 5-10. The LES turbulent viscosity is obtained by the extraction process discussed in Appendix E. The HIFIR turbulent viscosity is the mapped turbulent viscosity from the baseline optimization on the original U-bend geometry. The k-ω SST viscosity is solved by using the k and ω transport equations. Visually, the HIFIR solution still looks reasonable, but it is believed that the post-bend region would show increased values of turbulent viscosity and cover a wider portion of the channel if the training is repeated. Similar to the original U-bend, the k-ω SST turbulent viscosity is relatively low and the zone of appreciable turbulent viscosity is narrow. This lines up with the higher non-uniformity in the velocity profiles at the domain exit.

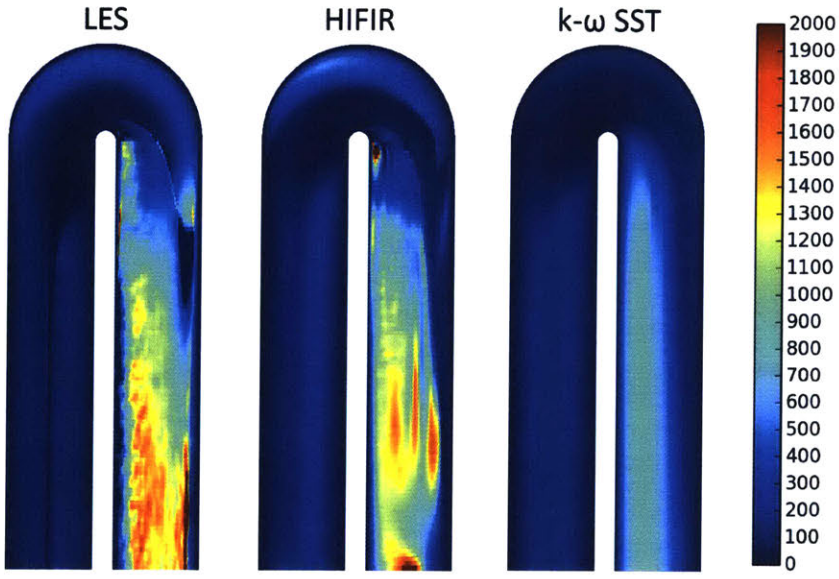


Figure 5-10: Comparison of turbulent viscosity ratio field for the adjusted U-bend

5.5 Conclusions

A relatively small change to the middle wall thickness has been shown to have a significant impact to the flow solution based on the back-to-back LES comparison. Using the well-matched HIFIR solution from Chapter 4 for the prescribed turbulent viscosity on the new geometry results in an inaccurate solution and is in some ways worse than the k-ω SST solution. Although for a given geometry a turbulent viscosity can be inferred to provide an accurate solution, this turbulent viscosity should not be directly used on new geometries. Improvements to the high-fidelity training method to allow successful transfer of the learned model to new geometries are suggested in Chapter 6.

Chapter 6

Summary, Conclusions, and Future Research

6.1 Summary and Conclusions

In Chapter 3, standard RANS is shown to be inaccurate on a 180° U-bend square duct. This geometry is an idealization of a turbine blade internal serpentine cooling passage. A large eddy simulation on the same geometry is shown to provide accurate results and is validated based on PIV experimental data. The issue is that the LES requires computational resources on the order of 10,000 times that needed for the RANS simulation in terms of CPU-hours. The desire is to develop a framework to obtain LES-accurate solutions with the computational expense much closer to that of RANS to allow usage of these tools in the design process.

In Chapter 4, the LES and standard RANS models are rerun on a 180° U-bend channel. This geometry enables the use of a 2D domain for the RANS solutions. The RANS model is again unable to produce accurate results throughout the domain. A simple eddy viscosity RANS turbulence model based on a prescribed turbulent viscosity is also introduced. The high-fidelity LES result is used to train, or infer, the turbulent viscosity field to reduce the solution error. Although this new model (referred to as HIFIR for high-fidelity trained RANS) is basic and primitive, it adequately matched the LES result. This shows that high-fidelity simulations can be

used to train even the simplest turbulence models to improve solutions over current two-equation models.

In Chapter 5, the channel U-bend geometry is modified to reduce the middle wall thickness. The HIFIR, standard RANS, and LES methods are used to obtain solutions on the adjusted geometry. It is shown that the optimized turbulent viscosity from the original geometry cannot be simply remapped onto a new geometry to obtain an accurate solution. Although the overall solution error is worse than a standard k-ω SST solution, the stationary optimized turbulence model of the HIFIR approach still better predicts the reattachment location and outlet velocity profile.

6.2 Future Research

The work presented here should also be extended to three dimensions to study more complex geometries. For this to be possible, a parallel solver is needed. The errors observed in the adjoint sensitivities also need to be corrected to improve the convergence of the optimization.

The key finding from this research is that high-fidelity simulations can successfully be utilized to train simple RANS turbulence models to produce realistic and accurate solutions. The method used in this thesis is based on an eddy viscosity model where the turbulent viscosity is simply inferred as part of an optimization problem. This formulation is adequate to show that even a basic turbulence model can be tuned, but is not well suited for usage on new geometries without repeating the training process.

Future research should be directed to repeating the training process presented here but with superior turbulence models based on flow parameters. The HIFIR process can be used to tune coefficients of standard turbulence models by minimizing the solution error in order to allow usage of the trained model on new geometries. But perhaps the most promising path forward is the use of machine learning to construct new turbulence models based on flow parameters and features, such as the velocity gradients, distance from the wall, etc. In this research, the form of the

turbulence model is based on the Boussinesq approximation. With machine learning, the functional form of the Reynolds stress closure term can be reconstructed. Using machine learning to generate a turbulence model by minimizing the velocity solution error would allow application of the trained model to new geometries. The use of a model based on flow parameters to obtain the turbulent viscosity or the Reynolds stress could also enable the use of limited experimental data as well as full LES and DNS simulations for training.

The ability to obtain solutions closer in accuracy to LES and DNS results with the efficiency of RANS is promising and will soon allow improved optimization with CFD in industry and the design process.

Appendix A

Low-Memory Calculation of Sensitivity Gradient

The automatic differentiation method used for the calculation of the gradient of the objective function to the design variable is discussed in Section 2.2.3. In order to reduce the memory requirements, only one iteration of the flow solver is stored. This is done by first fully converging the forward simulation. The simulation is then restarted using the converged results and run for one iteration. This appendix provides the specific procedure to calculate the gradient correctly using only one iteration.

The objective function is the velocity mismatch between the low-fidelity and high-fidelity solution, shown again in Equation A.1.

$$J(u(\nu_t)) = \|u(\nu_t) - u_{\text{HiFi}}\|_{\text{L2}}^2 \tag{A.1}$$

For simplicity, the input velocity and pressure fields will be referred to as general inputs x in the equations below. The subscript k is used to denote the solver iteration. The flow solver function, denoted as S, outputs the updated velocity field as shown in Equation A.2.

$$x_{k+1} = \text{S}(x_k, \nu_t) \tag{A.2}$$

The objective function is a function of the velocity fields.

$$J = J(x_k) = J\big(\mathrm{S}(x_{k-1}, \nu_t)\big) \tag{A.3}$$

For a converged solution, the output solution should match the previous solution, such that $x_{k+1} = x_k = x$. Therefore, the governing equation can be rewritten as

$$x - \mathrm{S}(x, \nu_t) = 0 \ . \tag{A.4}$$

Following the derivation of the adjoint equation from Section 2.2.3, the adjoint equation has the following form.

$$\phi^T = \phi^T \left(\frac{\partial \mathrm{S}}{\partial x} \right) - \frac{\partial J}{\partial x} \tag{A.5}$$

As can be seen, ϕ^T, is dependent on itself. Substituting the right-hand side of Equation A.5 in for ϕ^T on the left hand size, the adjoint equation becomes

$$\phi^T = \left(\phi^T \left(\frac{\partial \mathrm{S}}{\partial x} \right) - \frac{\partial J}{\partial x} \right) \left(\frac{\partial \mathrm{S}}{\partial x} \right) - \frac{\partial J}{\partial x} = \phi^T \left(\frac{\partial \mathrm{S}}{\partial x} \right)^2 - \frac{\partial J}{\partial x} \frac{\partial \mathrm{S}}{\partial x} - \frac{\partial J}{\partial x} \ . \tag{A.6}$$

This process can be repeated many times to create an infinite series, as shown in Equation A.7.

$$\phi^T = \phi^T \left(\frac{\partial \mathrm{S}}{\partial x} \right)^{k+1} - \frac{\partial J}{\partial x} - \frac{\partial J}{\partial x} \frac{\partial \mathrm{S}}{\partial x} - \frac{\partial J}{\partial x} \left(\frac{\partial \mathrm{S}}{\partial x} \right)^2 - \ldots - \frac{\partial J}{\partial u} \left(\frac{\partial \mathrm{S}}{\partial x} \right)^k \tag{A.7}$$

The above equation can be solved iteratively. The first term can be ignored since $(\partial \mathrm{S}/\partial x)^k$ approaches zero as k approaches infinity. The infinite series can then be approximated as

$$\phi^T = -\frac{\partial J}{\partial x} - \frac{\partial J}{\partial x} \frac{\partial \mathrm{S}}{\partial x} - \frac{\partial J}{\partial x} \left(\frac{\partial \mathrm{S}}{\partial x} \right)^2 - \ldots - \frac{\partial J}{\partial u} \left(\frac{\partial \mathrm{S}}{\partial x} \right)^k \ . \tag{A.8}$$

The series can be used to obtain the desired gradient by substituting ϕ^T into the

gradient equation from Equation 2.16.

$$\frac{\partial J}{\partial \nu_t} = -\phi_T \frac{\partial \mathrm{S}}{\partial x} = \frac{\partial J}{\partial x}\frac{\partial \mathrm{S}}{\partial \nu_t} + \frac{\partial J}{\partial x}\frac{\partial \mathrm{S}}{\partial x}\frac{\partial \mathrm{S}}{\partial \nu_t} + \frac{\partial J}{\partial x}\left(\frac{\partial \mathrm{S}}{\partial x}\right)^2 \frac{\partial \mathrm{S}}{\partial \nu_t} + \ldots - \frac{\partial J}{\partial x}\left(\frac{\partial \mathrm{S}}{\partial x}\right)^k \frac{\partial \mathrm{S}}{\partial \nu_t} \quad \text{(A.9)}$$

The components of the above series can be obtained using automatic differentiation and an iterative procedure. Using the numpad AD tool, a Jacobian can be computed using the .diff function. For example, A.diff(b) is equal to $\partial A/\partial b$. Equation A.10 is the first term of the series and is obtained from the reverse mode AD of the single iteration of the primal solver.

$$J.\mathrm{diff}(x) = \frac{J(\mathrm{S}(x, \nu_t))}{\partial x} = \frac{\partial J}{\partial x}\frac{\partial \mathrm{S}}{\partial x} \quad \text{(A.10)}$$

As shown, this term alone is not the full gradient. Similarly, the gradient of the objective function to the solution x is obtained using the .diff function.

$$J.\mathrm{diff}(\nu_t) = \frac{J(\mathrm{S}(x, \nu_t))}{\partial \nu_t} = \frac{\partial J}{\partial x}\frac{\partial \mathrm{S}}{\partial \nu_t} \quad \text{(A.11)}$$

A new term, denoted $\tilde{J}$, is shown in Equation A.12 is used to obtain the remaining terms.

$$\tilde{J} = \frac{J(\mathrm{S}(x, \nu_t))}{\partial x} x = \frac{\partial J}{\partial x}\frac{\partial \mathrm{S}}{\partial x}\mathrm{S}(x, \nu_t) \quad \text{(A.12)}$$

The two gradients in Equation A.12 are constants based on the input solution x_0. The final term x in the above equation is the solution after the one stored primal iteration. By taking the derivative in respect to the solution, the new term becomes

$$\tilde{J}.\mathrm{diff}(\nu_t) = \frac{\partial J}{\partial x}\frac{\partial \mathrm{S}}{\partial x}\frac{\partial \mathrm{S}}{\partial \nu_t} \; . \quad \text{(A.13)}$$

This term is shown to be the second term of the series. After two iterations, the gradient $\partial J/\partial \nu_t$ is the sum of Equation A.10 and A.13. Each remaining term is obtained by redefining J as $\tilde{J}$ and recalculating $\tilde{J}.\mathrm{diff}(\nu_t)$ and adding to the gradient $\partial J/\partial \nu_t$

from the previous iteration. These iterations are called adjoint iterations and are part of the adjoint simulation. Similar to the primal simulation, the adjoint simulation is run until the changes to the gradient are sufficiently small and has thus converged to a usable value.

The above procedure is implemented in the Python code to reduce the amount of storage needed from the primal solver. The objective function and calculation of the gradient in Python are shown below and provide a starting point for the adjoint simulation.

```
J = ( ((u_x - u_x_HiFi)**2).sum() + ((u_y - u_y_HiFi)**2).sum() )
dJ_dnut = J.diff(nut).toarray().reshape(nut.shape)
```

The the infinite series is then approximated using iterations with the following terms. In the code, the solution x referenced in the above equations covers the velocity component fields and the pressure field. For a 2D simulation, three terms exist.

```
dJ_dux = J.diff(u_x_0).toarray().reshape(u_x.shape)
dJ_duy = J.diff(u_y_0).toarray().reshape(u_y.shape)
dJ_dp = J.diff(p_0).toarray().reshape(p.shape)
J = ( (u_x * dJ_dux).sum() + (u_y * dJ_duy).sum() + (p * dJ_dp).sum() )
dJ_dnut_additional = J.diff(nut).toarray().reshape(nut.shape)
dJ_dnut += dJ_dnut_additional
```

The above additions are performed in a while loop to grow the series. In order to watch the convergence, the residuals are computed as shown below. The residuals are simply the additional terms at the end of the series at that iteration. As the residuals are sufficiently reduced, the gradient is less dependent on the initial velocity and pressure fields used for the flow solver and provide an answer closer to the true gradient.

```
ux_adjoint_res = np.linalg.norm(np.ravel(value(dJ_dux)))
uy_adjoint_res = np.linalg.norm(np.ravel(value(dJ_duy)))
p_adjoint_res  = np.linalg.norm(np.ravel(value(dJ_dp)))
```

The above method is compared with the full memory version, where all primal solution iterations are stored then used in reverse mode AD. A comparison between the two methods are based on the change in the objective value for a 0.01% change in the turbulent viscosity. Memory restrictions allow less than 100 iterations for the full-memory version. The low-memory version compares very well to the full-memory results for the first 30 iterations as shown in Table A.1. After several thousand adjoint iterations, the error in the low-memory version is expected to still be several orders of magnitude smaller than the gradient since the corrections are based on the governing equations.

Table A.1: Full vs Low-Memory Adjoint Comparison of δJ

Iterations	**Full**	**Low-Memory**	**Difference**
1	6.90476e-3	6.90476e-3	0
5	2.86797e-2	2.86797e-2	-1.3e-8
10	4.84980e-2	4.84979e-2	-6.3e-8
20	7.55330e-2	7.55328e-2	-2.2e-7
30	9.51488e-2	9.51485e-2	-3.4e-7

Appendix B

Periodic Domain Height Sensitivity Study for LES of U-bend Channel

The spanwise height of the U-bend channel is important since turbulent flows are not periodic and care must be taken to make sure the periodic boundaries are spaced far enough apart as to not corrupt the solution with artificial periodic structures.

Previous research of turbulent straight channel flow using DNS, such as that performed by Kim et al. [15], have used spanwise periodic domains sizes as low as 2πh, or πH, where h is the half-channel height and H is the full channel height. There has been limited research performed on channels with 180° U-bends, where additional turbulent spanwise structures are created due to the bend. These structures include Gortler-Taylor vortices on the highly-concave outer wall and counter-rotating secondary flow structures due to flow around the bend. Laskowski [16] performed DNS of serpentine passage which consisted of a periodic channel with two U-bends. The spanwise domain height was set at 3πh, or 1.5πH.

Gortler vortices are secondary flows that appear in boundary layer flow along a concave wall. Instability arises when the boundary layer thickness is comparable to the radius of curvature. The onset of Gortler vortices can be predicted by the Gortler number, which is the ratio of the centrifugal effects to the viscous effects in the boundary layer. The Gortler number is given as

$$G = \frac{U_\infty \theta}{\nu}\left(\frac{\theta}{R}\right)^{1/2} \tag{B.1}$$

where U_∞ is the far field velocity, θ is the momentum thickness, ν is the kinematic viscosity, and R is the radius of curvature of the wall. Instability occurs when $G > 0.3$. For this research, the U-bend channel flow will form Gortler vortices due to the strong outer wall curvature. Figure B-1 shows the region of Gortler instability.

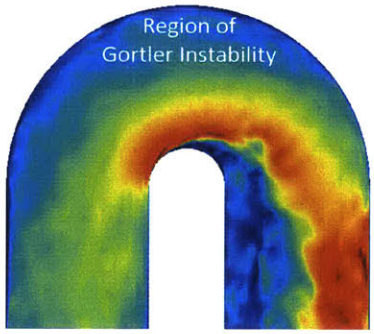


Figure B-1: Instantaneous velocity showing region of Gortler instability

The domain height of the U-bend channel is studied to ensure the Gortler vortices are properly captured without imposing artificial periodic characteristics. This is also true for the secondary vortex structures created by the bend. The periodic height of πH and 2πH are studied to investigate the sensitivity on the periodicity. The domain sizes are shown in Figure B-2.

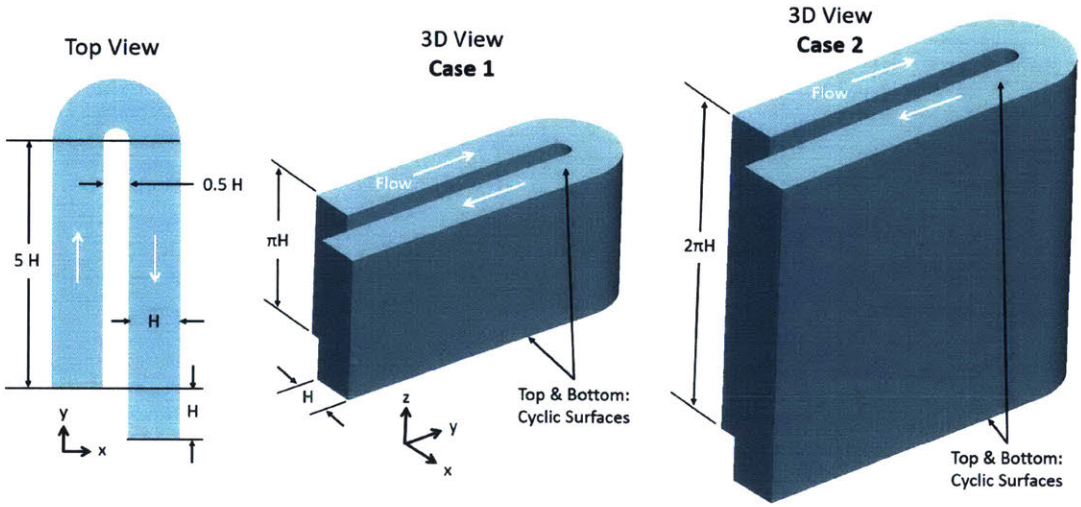


Figure B-2: Domain geometry used for periodic height study

The same mesh topology is used in both simulations. The streamwise cell count is identical between the two meshes and corresponds to a wall unit size of approximately 50 in the straight duct section and ranges from 13 to 66 in the bend region. The spanwise periodic direction uses the same uniform cell size of approximately 25 wall units. A comparison of the overall mesh size and simulation duration on 32 cores is shown in Table B.1.

Table B.1: Domain Height Grid

Parameter	**Case 1: πH**	**Case 2: 2πH**
Mesh Resolution	265 x 150 x 126	265 x 150 x 252
Cell Count	5,008,500	10,017,000
Simulation Duration	6d 13h	10d 19h

The simulations were run in similar fashion, using the same time-step and time-averaging duration. The instantaneous velocity magnitude results are shown in Figure B-3. Three spanwise sections are shown for both cases side by side by simply stacking the πH sections to match the 2πH height. Section A is taken 3/4 through the bend, Section B is at the bend exit, and Section C is taken 2.5 H from the bend exit.

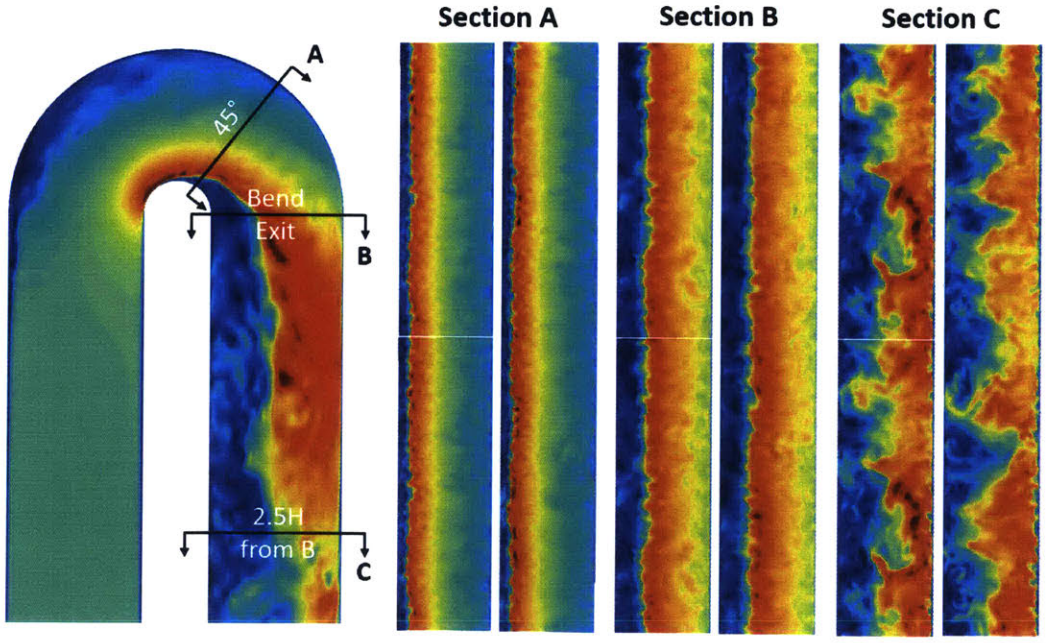


Figure B-3: Instantaneous velocity magnitude at three sections

As shown in Section A, the Gortler vortices along the outer wall are relatively

small and on the same order of size as the boundary layer. This is sufficiently smaller than the domain height. The periodic domain heights utilized are not expected to influence the time-averaged solution in the outer wall region. Section B and C show larger structures created by the separation along the inner bend. These structures are closer to the size of the channel width H and thus more likely to be impacted by the periodic domain height.

Visually, the instantaneous field is similar in structure in all three sections shown. Section C contained the largest structures and Figure B-4 specifically shows the spanwise components of the velocity magnitude. The instantaneous and time-averaged plots again shown comparable patterns and sizes of the turbulent structures. The time-averaged results are obtained by averaging over 14 flow-thru units.

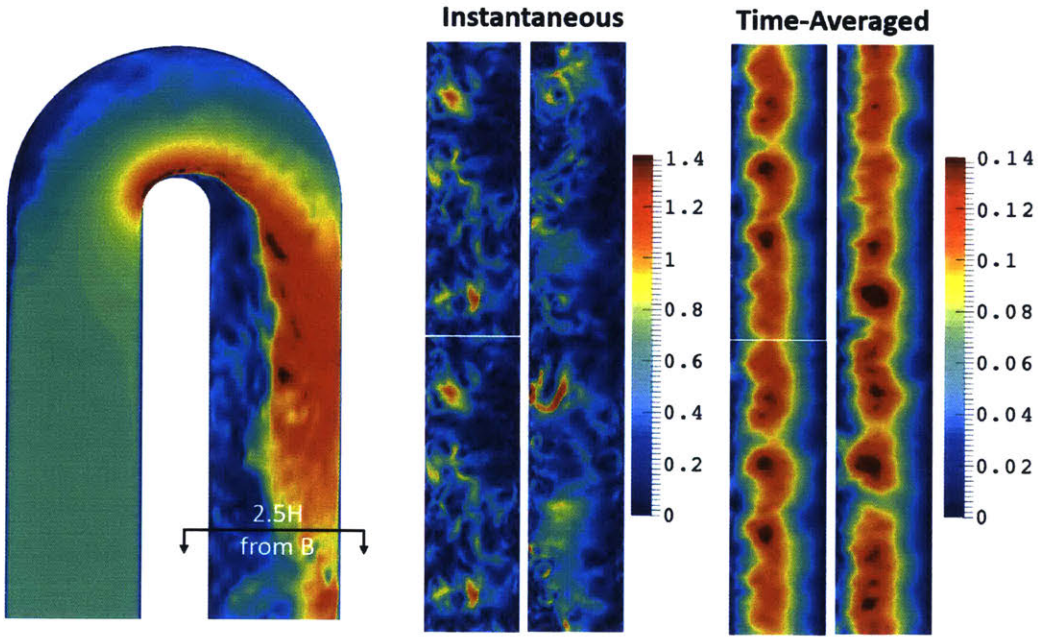


Figure B-4: Spanwise velocity magnitude at Section C

The impact of the periodic height on the time-averaged is obtained by first collapsing the time-averaged results in the homogeneous periodic spanwise direction. The comparison between the two solutions are shown in Figure B-5. The delta plot shows the absolute difference in the solution. The percent difference plot highlights a sliver of high difference along the edge of the recirculation zone as the velocity here is nearly zero.

The outer wall region is shown to have around 1% error, validating the minimal impact of the domain height to the Gortler instability. The post-bend region shows higher error due to the larger vortical structures, but is still under 3% error. A similar comparison is shown in Figure B-6 for the pressure field.

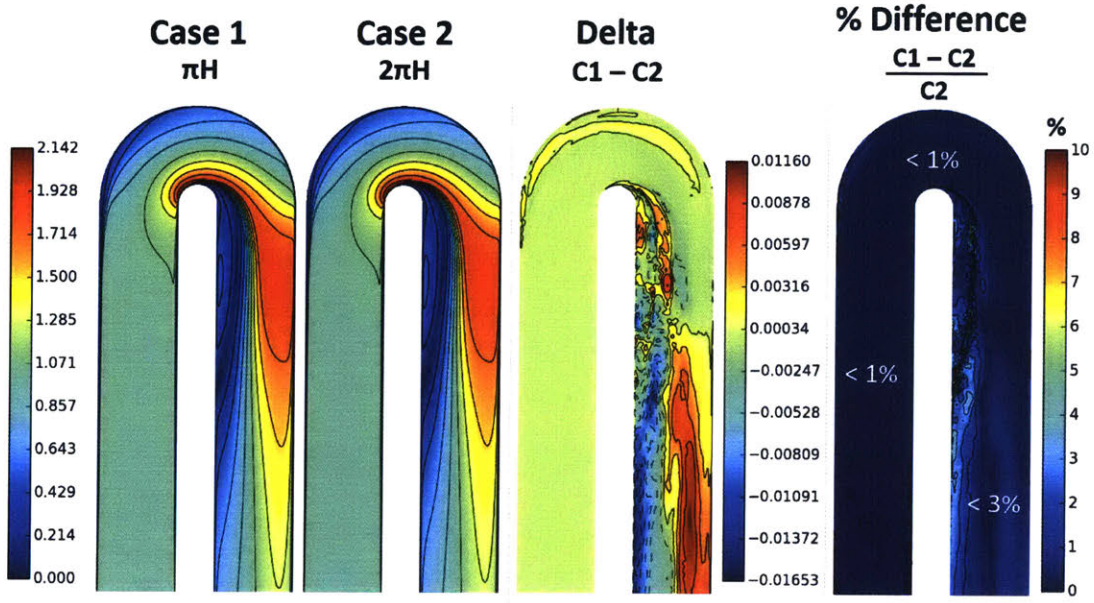


Figure B-5: Time-averaged velocity comparison between the two periodic height cases

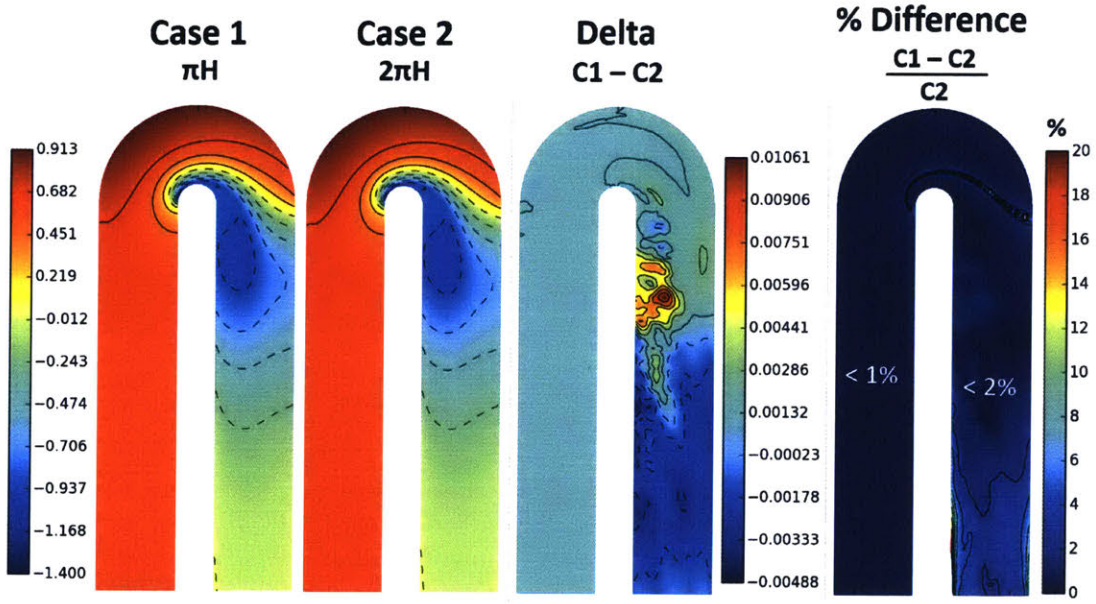


Figure B-6: Time-averaged pressure comparison between the two periodic height cases

The largest error is in the recirculation region and shear layer and is about 4% locally. The post-bend region otherwise has an error of about 2%. Again, slivers of higher error show up in the percent difference plot where the pressure field are near zero in value.

Overall, the πH periodic height has been shown to be sufficiently tall to obtain reasonable accuracy. The computational cost savings is substantial, especially for the insignificant difference in the time-averaged solution. The U-bend LES simulations performed in Chapter 4 and 5 are based on the πH periodic height.

Appendix C

Benchmarking of Python 2D Incompressible Flow Solver

The low-fidelity solver used in this research is a steady-state 2D incompressible flow solver written in the Python programming language based on the SIMPLE algorithm. The flow solver has been benchmarked to several standard cases. This appendix covers benchmarking of the Python solver performed on laminar channel flow, a lid-driven cavity, and a laminar U-bend channel. The results are compared to the exact solution of the Navier-Stokes equation in the channel flow case, available data for the lid-driven cavity at specific Reynolds numbers, and by comparison to OpenFOAM for the laminar U-bend channel case.

C.1 Laminar Channel

A common benchmark for laminar incompressible flow is laminar straight channel flow. This is unidirectional flow between two infinite parallel plates. A sketch of the straight channel is shown in Figure C-1.

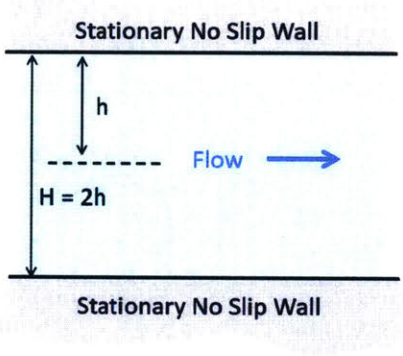


Figure C-1: Sketch of straight channel flow

The Navier-Stokes equations reduce to a closed form exact solution since the axial flow is only dependent on position from the wall, y, the pressure gradient, dp/dx, and the viscosity μ as shown in Equation C.1.

$$\frac{dp}{dx} = \mu \frac{\partial^2 u}{\partial y^2} \tag{C.1}$$

Integrating the above equation and including boundary equations, the velocity profile can be written as

$$u = \frac{1}{2\mu} \frac{dp}{dx} (y^2 - Hy) \tag{C.2}$$

where H is the full channel width and y is the distance from the bottom wall. The equation indicates that the velocity profile is parabolic. It can be shown that the peak velocity at the channel center is 1.5 u_{avg}, the average or bulk velocity.

The flow simulation is setup to be driven by a prescribed pressure gradient. The inlet and outlet are treated as cyclic boundaries. Only 5 cells are used in the streamwise direction, while 60 cells are used in the spanwise direction. The grid is uniform in each direction. The flow simulation utilizes the central differencing scheme (CDS) for both the convection and diffusion terms in the momentum equations. This is possible as the diffusion terms are strong enough to stabilize the central scheme at the low Reynolds number. The pressure gradient is solved to obtain a bulk velocity of unity,

and thus a Reynolds number of 1 based on the full channel height H, or 0.5 based on the half channel height h. The velocity profile from the simulation is compared to the exact solution in Figure C-2.

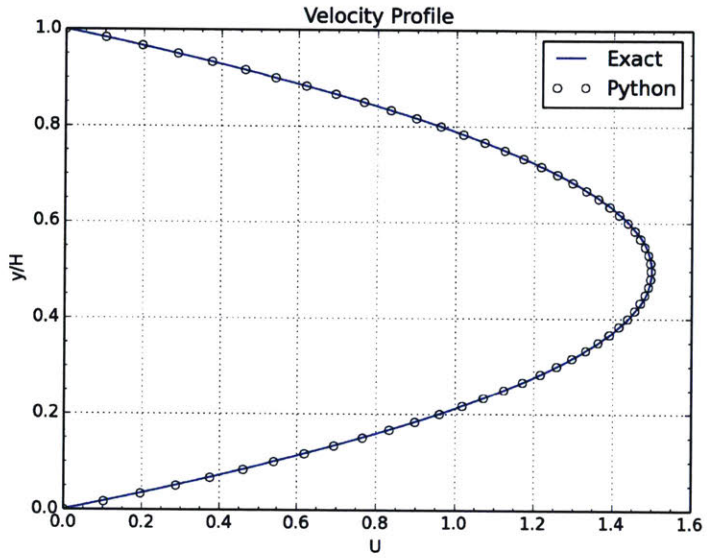


Figure C-2: Python velocity profile vs exact solution for a laminar channel

The flow solver results agree well with the exact parabolic solution. The resulting pressure gradient from the Python solver is 12.22 compared to the exact solution of 12 in order to obtain a bulk flow of unity. The peak to bulk velocity from the simulation is 1.497. This solution error compared to the exact solution will decrease as the grid is refined.

C.2 Lid-Driven Cavity

Another common benchmark is the laminar incompressible square lid-driven cavity. It is often used for evaluating numerical techniques. The cavity has three stationary walls on the sides and bottom, while the top wall moves at a constant rate. A sketch of the lid-driven cavity is shown in Figure C.3. The no-slip condition is used for all walls and thus the fluid velocity at the top boundary of the cavity is the velocity of the lid.

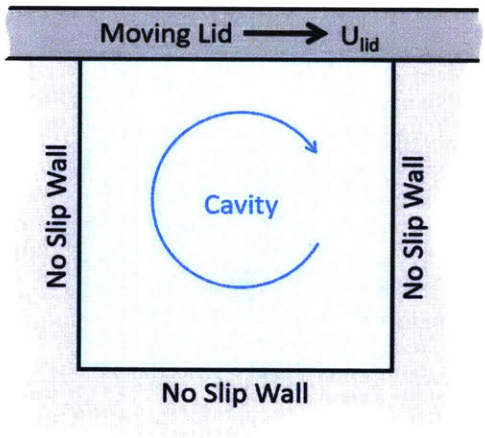


Figure C-3: Sketch of square lid-driven cavity

The Reynolds number is defined in Equation C.3, where U_{lid} is the the lid velocity, L is the height and width of the cavity, and ν is the kinematic viscosity.

$$Re = \frac{U_{\text{lid}} L}{\nu} \tag{C.3}$$

The Python code is performed on several uniform meshes to show grid convergence. The solution is compared to numerical results of Ghia et al. [10] for Re = 100 and 1000. The uniform meshes used for the grid convergence study are shown in Figure C-4. As in the laminar channel, the cavity simulations utilize CDS for both the convection and diffusion terms.

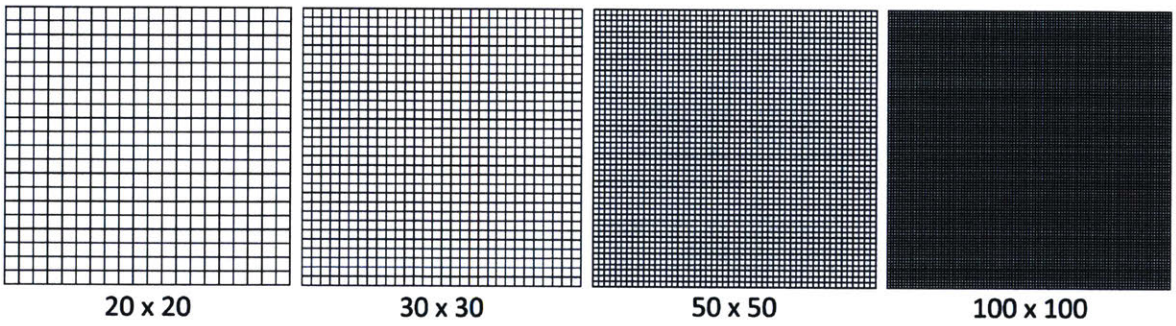
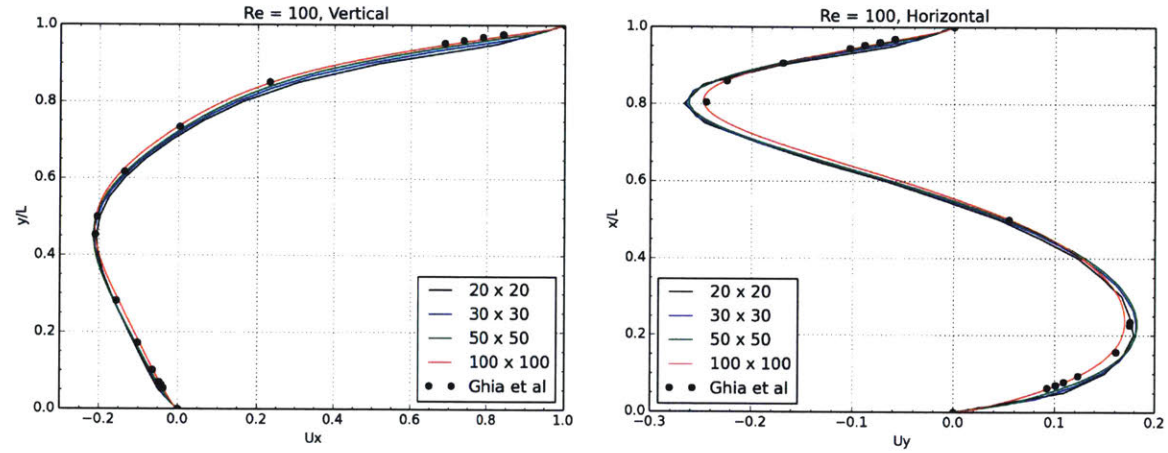


Figure C-4: Uniform meshes used for lid-driven cavity grid convergence

Ghia et al. provided results along vertical and horizontal lines passing through the geometric center of the cavity. Figures C-5 and C-6 show the results over the

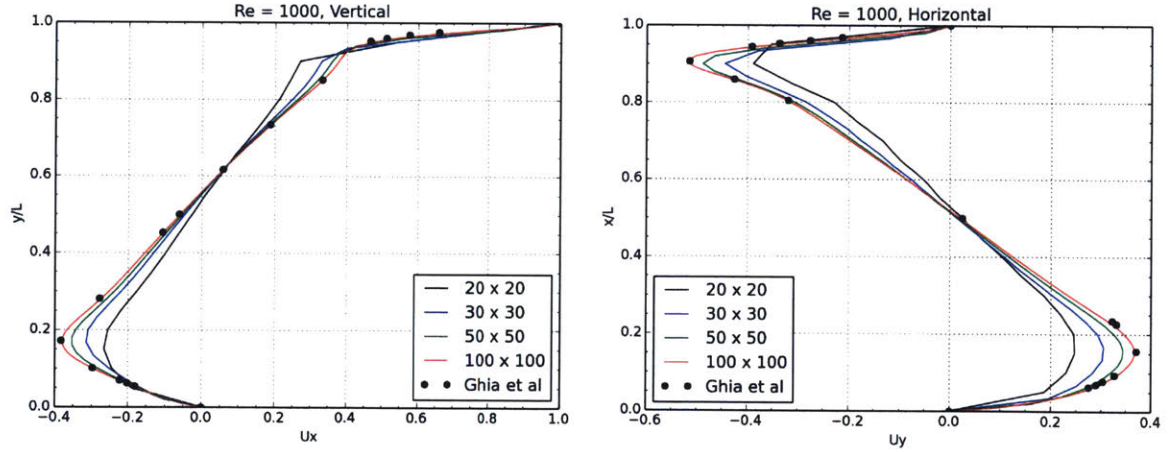
uniform grid results compared to the values from Ghia et al. for Reynolds numbers of 100 and 1000, respectively.



(a) u velocity profile

(b) v velocity profile

Figure C-5: Re 100: velocity profiles along (a) vertical and (b) horizontal lines passing through cavity center



(a) u velocity profile

(b) v velocity profile

Figure C-6: Re 1000: velocity profiles along (a) vertical and (b) horizontal lines passing through cavity center

It is clear that the finer grid converges to the results provided by Ghia et al. The 100 x 100 mesh performs well and nearly passes through each data point at both Reynolds numbers along vertical and horizontal slices.

A non-uniform grid is also tested to show the solution improvement possible with the use of finer grids near the walls as well as to verify the solver capability using non-uniform grids. The mesh utilized is a 50 x 50 grid with a constant cell expansion ratio of 1.066 from each wall. The ratio of the largest cell size to the smallest cell size is 5 and is known as the grading factor. The non-uniform mesh is shown along with a 50 x 50 uniform mesh in Figure C-7.

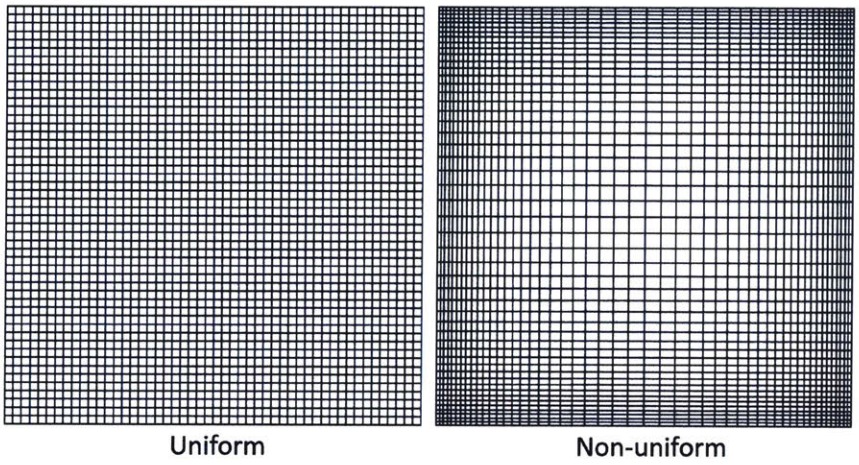


Figure C-7: 50x50 non-uniform mesh compared to a uniform mesh used for lid driven cavity

The results using the non-uniform mesh at a Reynolds number of 1000 is shown side-by-side with two uniform mesh results in Figure C-8. The velocity profiles are also provided in Figure C-9 for the same three meshes to compare against the results of Ghia et al.

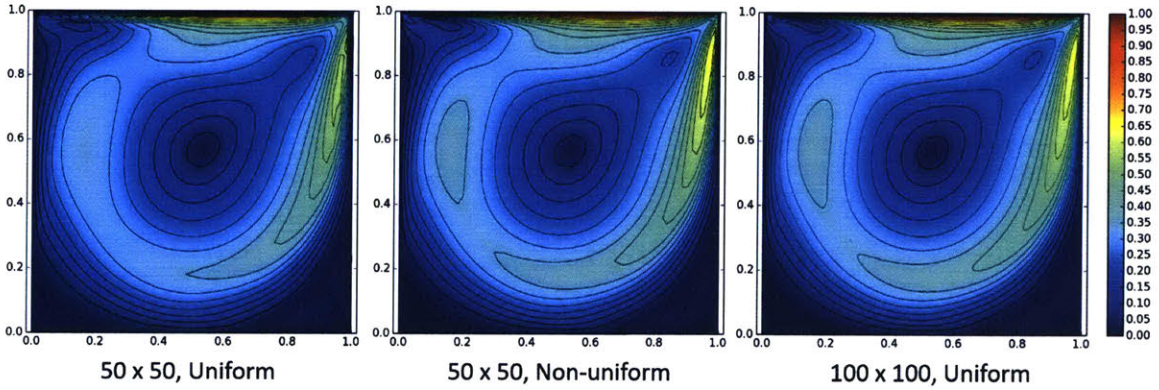
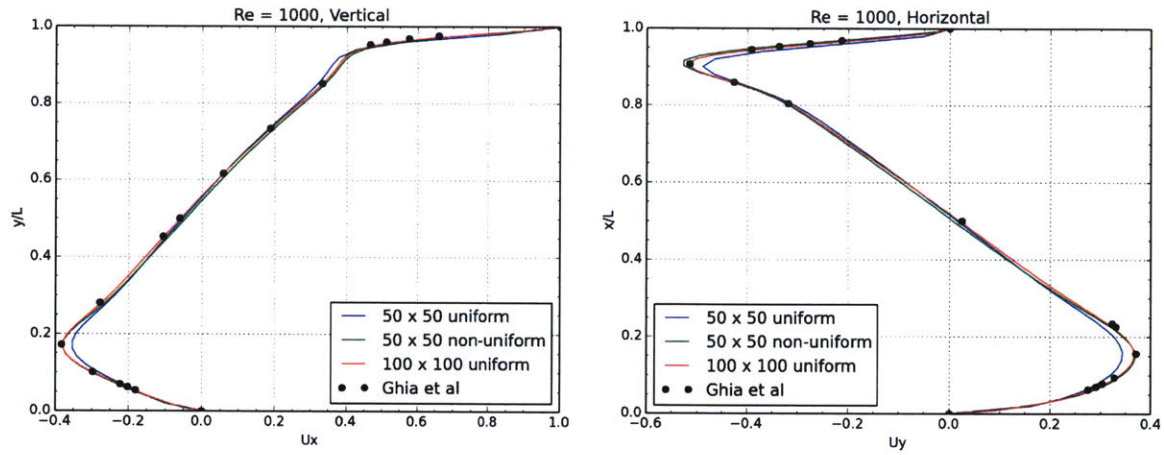


Figure C-8: Re = 1000 results for various grid sizes



(a) u velocity profile

(b) v velocity profile

Figure C-9: Re 1000: velocity profiles including non-uniform mesh

As shown, the non-uniform mesh performs as expected. The non-uniform mesh results are more accurate than the uniform mesh of the same size and are comparable to the 100 x 100 mesh. The use of a non-uniform mesh is crucial as this allows refinement in areas that are more important, such as near-wall regions, shear and mixing layers, etc, and avoids the cost of an overly fine uniform mesh. The solution dependency on Reynolds number is shown in Figure C-10 for Reynolds numbers ranging from 1 to 1000. All four cases use the non-uniform 50 x 50 mesh.

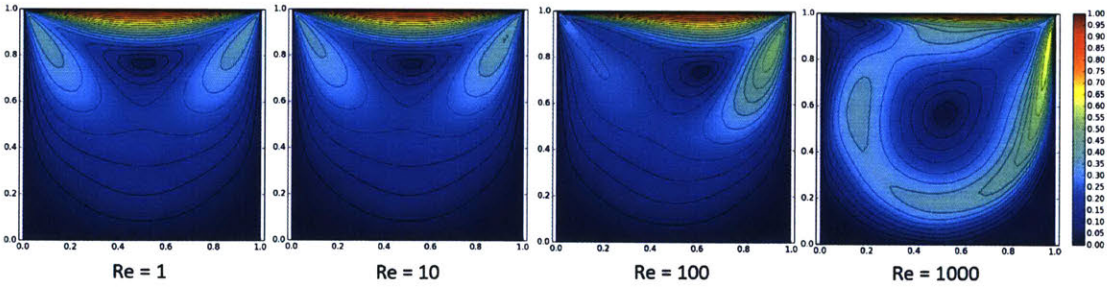


Figure C-10: Results on a 50 x 50 non-uniform mesh for Re 1 through 1000

C.3 Laminar U-bend Channel

The Python flow solver is also compared to OpenFOAM's SIMPLE incompressible flow solver in laminar mode for a U-bend channel. The presence of the bend is a test

to the solver's ability to handle angled cell faces. The Reynolds number is set to 1 based on the channel full-width and bulk velocity. The inlet boundary condition is the exact parabolic laminar channel solution. Both simulations use the same mesh. The velocity magnitude and pressure field are shown in Figure C-11 in their respective output formats.

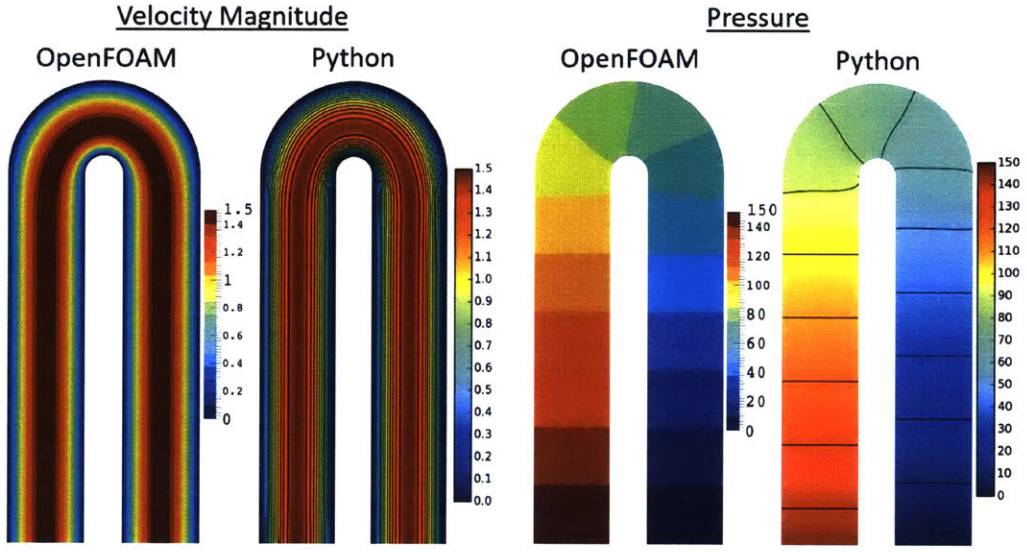


Figure C-11: Laminar U-bend solution: OpenFOAM vs Python

A delta plot and percent difference plot of the velocity magnitude is shown in Figure C-12 for the cell values. The largest difference occurs about at about 0.2H from the inner and outer wall. As shown, the error is within 4%, and that is due to the near zero value near the wall where a small difference is amplified. The center regions of the flow have an error closer to 0.1%. The overall pressure drop predicted by OpenFOAM from 1 full channel width from the domain inlet to 1 full channel width from the domain exit is 126.06. The Python solver predicts a pressure drop of 125.95, within 0.1% of the OpenFOAM solution.

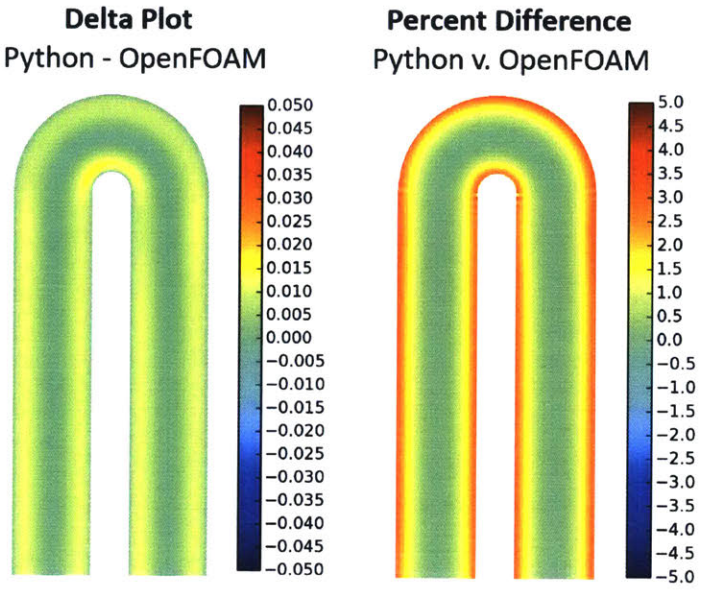
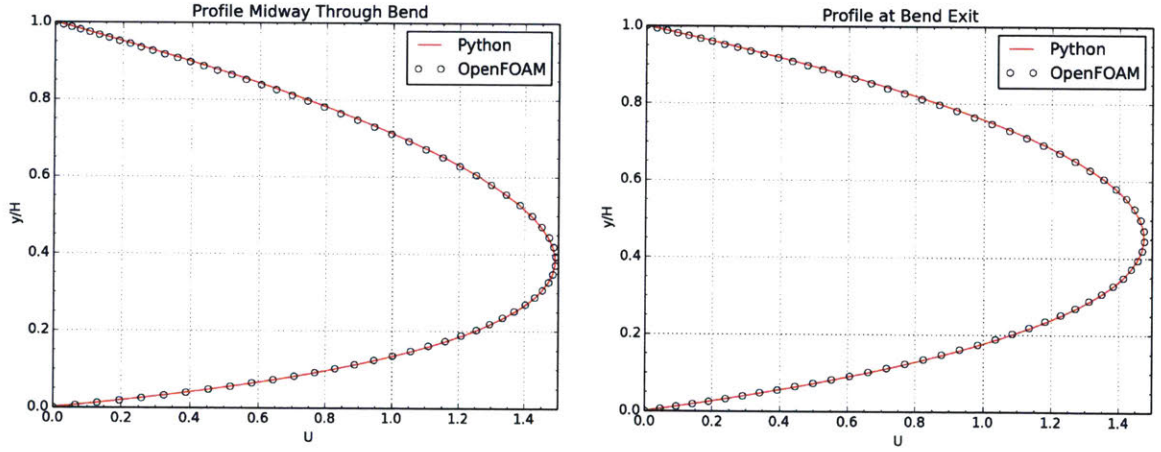


Figure C-12: Delta and percent difference plots for laminar U-bend

The deviation from a laminar channel occurs only in and near the bend. The streamwise velocity profile is shown halfway through the bend and at the bend exit in Figure C-13, where y/H of 0 is at the inner wall.



(a) Velocity profile half way through bend

(b) Velocity profile at bend exit

Figure C-13: Streamwise velocity profiles at two bend sections

The Python solver shows good agreement to the OpenFOAM reference solution. Overall, the solver is shown to perform accurately and as expected for various laminar benchmark scenarios.

Appendix D

Convection Schemes for RANS Solver

In this research, the Python flow solver utilizes the central differencing scheme (CDS) for low Reynolds number or laminar flows. As the Reynolds number is increased the ratio of convection to diffusion is increased. This is captured by the Peclet number, shown in Equation D.1

$$Pe = \frac{\rho u}{\mu/\delta x} = \frac{u\Delta x}{\nu + \nu_t} \tag{D.1}$$

where Δx is the local grid size. At high Peclet numbers, generally greater than 2, the central differencing scheme can lead to instabilities. One method to stabilize the solution is the use of the upwind differencing scheme (UDS). One downside to the UDS is that it has a first order Taylor series truncation error, as opposed to the second order error associated with CDS. As a result, the solution can be overly smoothed due to high numerical dissipation.

The Python flow solver utilizes a 5 point stencil operator for the momentum equations, using only cell center values based on the nearest neighbors. In order to maintain a simple 5 point stencil and to minimize computational cost, larger stencil higher-order differencing methods, such as the Quadratic Upstream Interpolation for Convective Kinematics (QUICK) scheme, are not used. Hybrid methods, which blend the upwind and central schemes, can be used to stabilize the solution. Hybrid scheme exploit the favorable characteristics of both upwind and central schemes. They

minimize the artificial diffusion required to improve convergence. However, hybrid schemes are also only first order accurate.

There are several methods of blending the upwind and central schemes. The Python solver has been setup to handle various blending methods by use of a general weighting function between the UDS and CDS interpolated cell face values for any variable ϕ, as shown in Equation D.2.

$$\phi = \gamma\phi_{CDS} + (1-\gamma)\phi_{UDS} \tag{D.2}$$

D.1 Zonal Blending

D.B. Spalding introduced a piecewise blending method in the early 1970s in which the second order accurate CDS scheme is used at low Peclet numbers (between -2 and 2) and UDS otherwise. Using the general blending framework, the Spalding hybrid method can be written as

$$\gamma = \begin{cases} 1 & \text{, if } |Pe| \leq 2 \\ 0 & \text{, if } |Pe| > 2 \end{cases} \tag{D.3}$$

where the Peclet number is calculated at the cell faces. The piecewise function essentially splits the simulation into discrete zones based on the Peclet number and is therefore referred to as the zonal method in this thesis. The threshold value of 2 is based on fitting the piecewise function to the exact exponential interpolation to cell faces in 1D.

As shown in Figure D-1, the turbulent U-bend flow at a Reynolds number of 40,000 has many regions with Peclet numbers greater than 2. The near-wall regions have local Peclet numbers over 30 due to the low local viscosity since the turbulent viscosity approaches zero. The bend region also has high Peclet numbers above 100 where the local velocity is high coupled with moderate to low turbulent viscosities.

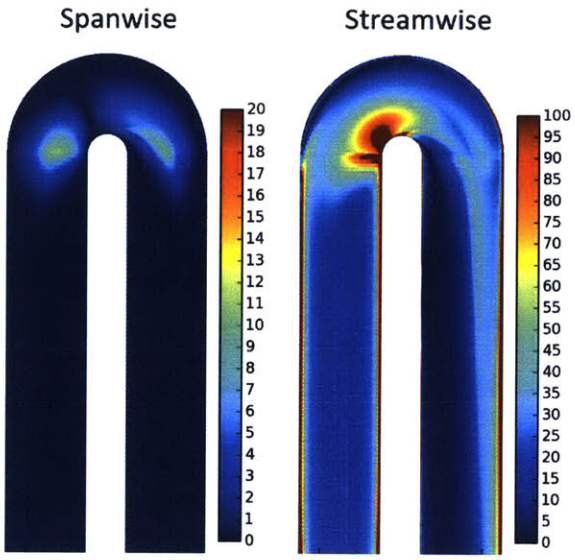


Figure D-1: Example plot of local Peclet number for turbulent U-bend channel

For the turbulent U-bend case, the use of full CDS is unstable and some amount of upwind blending is needed. The Spalding hybrid scheme essentially produces the same result as full upwind differencing. This zonal method can be generalized to any threshold Peclet number, Pe_{th}. The high Peclet zones above the threshold do not necessarily have to be fully upwinded. The blending factor γ can be non-zero in order to reduce the amount of diffusion introduced with upwinding. The general formulation is shown in Equation D.4.

$$\gamma = \begin{cases} 1 & \text{, if } |Pe| \leq Pe_{\text{th}} \\ \gamma_0 & \text{, if } |Pe| > Pe_{\text{th}} \end{cases} \tag{D.4}$$

where γ_0 is the user-prescribed blending factor above the threshold Peclet number. Figure D-2 shows the blending factor as a function of the local Peclet number for the Spalding hybrid scheme and the general zonal method with $Pe_{\text{th}} = 30$ and $\gamma_0 = 0.5$.

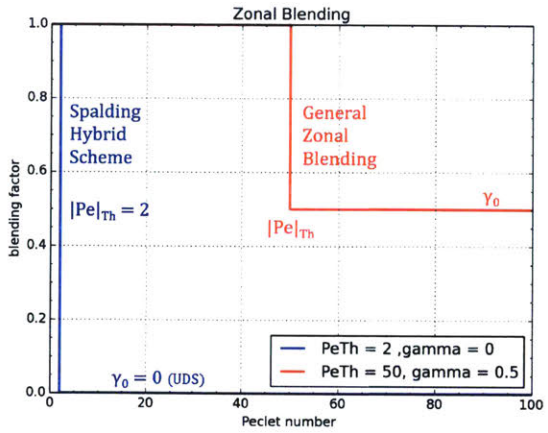


Figure D-2: Zonal blending: plot of blending factor as a function of Peclet number

The use of a non-zero blending factor allows the user to set the value to obtain just enough stability for proper convergence yet retain near-CDS accuracy, as full first-order upwind can be excessively diffusive. Figure D-3 shows the solution of the Python solver using different threshold Peclet numbers and blending factors compared to a higher-order OpenFOAM reference solution. Both OpenFOAM and Python use the same mesh, turbulent viscosity field, and boundary conditions.

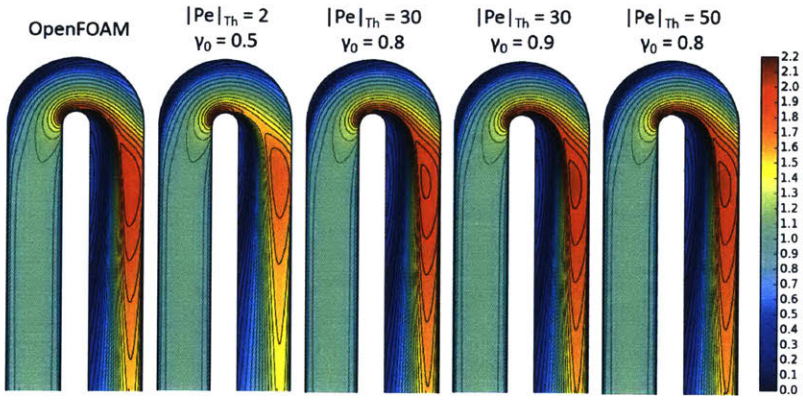


Figure D-3: Zonal blending: plot of velocity magnitude field for various blending parameters

The convergence of the above solutions is shown in Figure D-4. As shown, the residuals show some oscillation and this is believed to be due to the fact that some cell face Peclet numbers oscillate around the threshold value and therefore switch back-and-forth between full CDS and some UDS / CDS blending. This can cause

some issues when the converged solution is used for adjoint calculations to obtain sensitivity information as the adjoint solver essentially reverses through the solution convergence. Using smooth blending functions can reduce these oscillations. The follow sections introduce smooth methods to improve convergence.

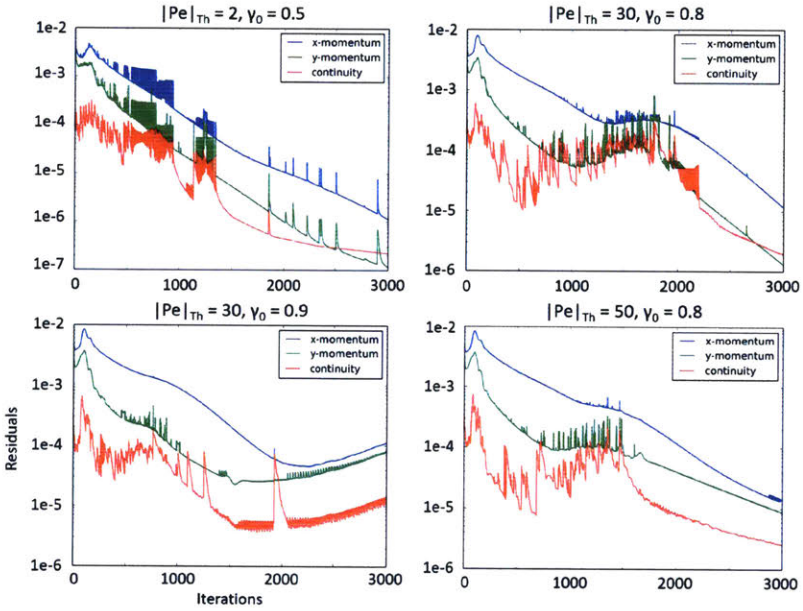


Figure D-4: Zonal blending: plot of convergence for various blending parameters

D.2 Global Blending

As shown in the previous section, the zonal method can create oscillations in the solution convergence. One method to minimize the oscillations is to use a smooth blending method. The simplest blending method is one that is not a function of Peclet number, but a simple global blending factor. Similar to the treatment of the high Peclet regions of general zonal method, the blending factor is user prescribed and usually close to 1 to minimize the artificial smoothing of the solution. Often, only a small amount of upwind blending is needed. Ferziger and Peric [6] discuss the use of 10% upwinding, thus a γ of 0.9, to stabilize an example compressible flow solution that would otherwise be unstable. In this research, it was found that a blending factor of 0.95 is sufficient to obtain a stable solution, but the convergence

does start to show some oscillations. Lower values, such as 0.9, provide a good compromise between upwind and central schemes. Figure D-5 shows the solution of the Python solver using different global blending factors compared to the reference solution. Figure D-6 shows the convergence of the residuals for the various blending values. The convergence is smoother than that obtained with the zonal method and can help stabilize the adjoint solver as well.

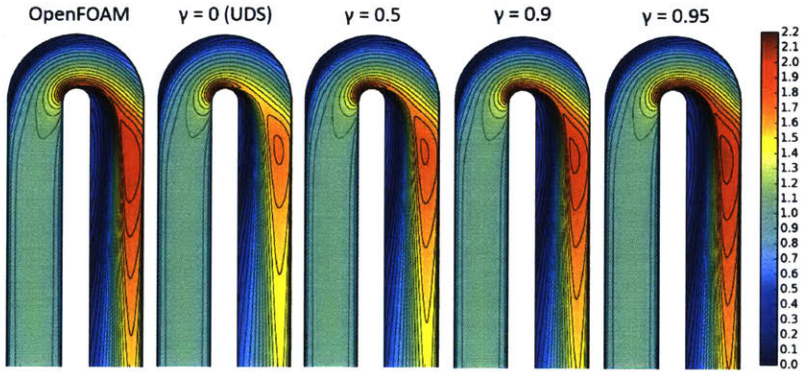


Figure D-5: Global blending: plot of velocity magnitude field for various global blending factor

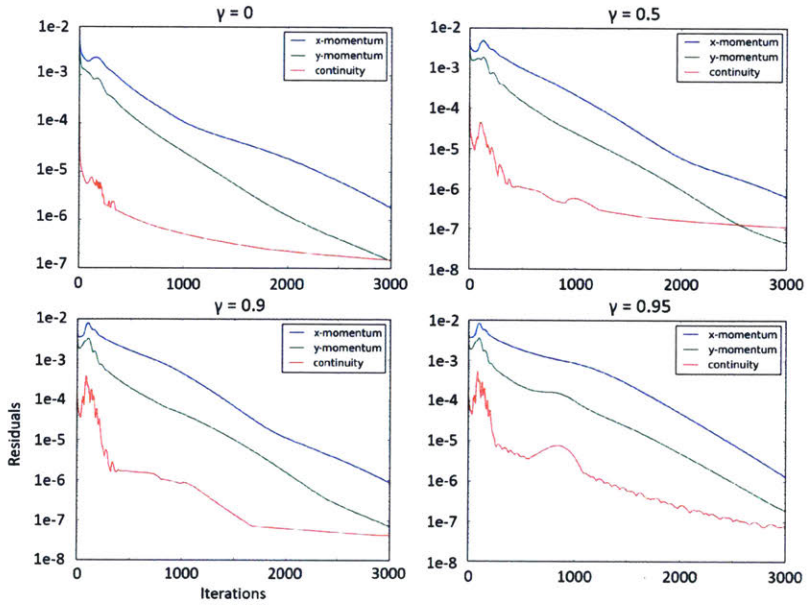


Figure D-6: Global blending: plot of convergence for various global blending factor

D.3 Exponential Blending

There are several downsides to the global blending method. The minimal amount of UDS blending is obtained with some trial and error. In addition, the blending is applied to all cell faces. As shown in Figure D-1, the turbulent U-bend channel has much higher Peclet numbers in the streamwise directions and thus global blending is overly diffusing the spanwise direction. These issues can be resolved by formulating a local blending factor as a smooth function of the local Peclet number.

A logical start is the exponential blending method described by Patankar [27]. As mentioned before, the linear interpolation between two cells to an interfacing cell face can be written as an exponential function of the Peclet number for 1D. The blending scheme is nearly CDS at Peclet numbers around zero, and approaches fully upwinding around a Peclet number of 10. This relationship is used as a basis of a smooth exponential blending factor, as shown in Equation D.5.

$$\gamma = 2\left(\frac{e^{0.5a|Pe|} - 1}{e^{a|Pe|-1}}\right)$$
$$\text{where,} \tag{D.5}$$
$$a = \frac{10}{Pe_{\text{th}}}.$$

The above equations approximate Patankar's exponential blending scheme when the threshold Peclet number is set to 10. The threshold Peclet number is user specified and can be increased to reduce the upwinding at lower Peclet numbers. The exponential blending method is shown in Figure D-7 as a function of the local Peclet number. Again, there is some user control with the threshold Peclet number. The scheme does allow a smooth method to calculate local blending factors in order to increase the amount of upwinding only in areas that require additional diffusion.

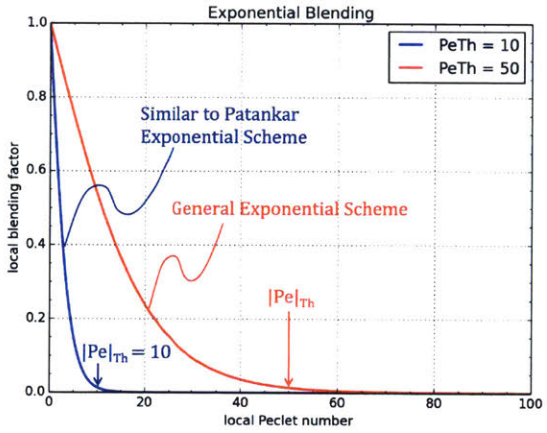


Figure D-7: Exponential blending: plot of blending factor as a function of Peclet number

Figure D-8 shows the solution of the Python solver using different threshold Peclet numbers compared to the reference solution. Figure D-9 displays the convergence associated with the solutions in Figure D-8. The convergence also has the smooth behavior shown with the global blending method yet has the benefit of adaptive blending factors based on the local flow and grid. The convergence is similar despite the wide range of threshold Peclet numbers.

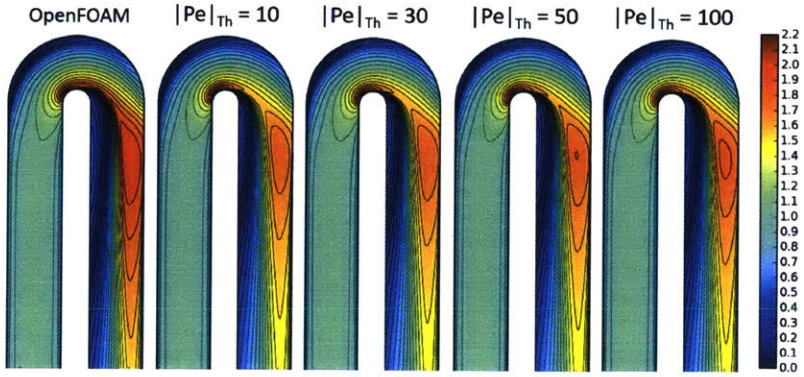


Figure D-8: Exponential blending: plot of velocity magnitude field for various blending parameters

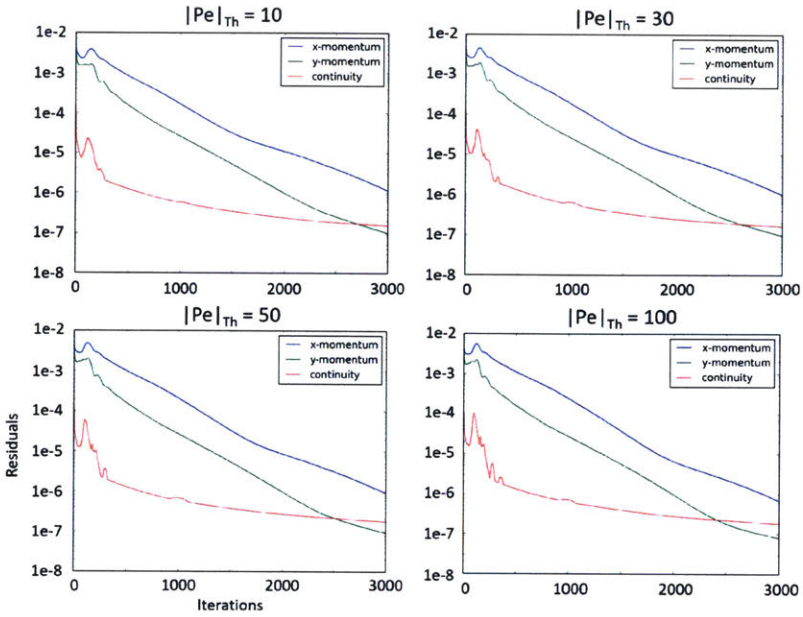


Figure D-9: Exponential blending: plot of convergence for various blending parameters

D.4 Hyperbolic Tangent Blending

The exponential blending factor has the attractive property of calculating local blending factors smoothly. However the blending factor immediately drops below 1 for any local absolute Peclet value greater than 0 and thus can penalize even low Peclet regions with some upwinding. Similar to the zonal methods, it is desirable to maintain the central difference up to a certain Peclet number. One method created as part of this research is the use of the hyperbolic tangent function to blend smoothly between fully CDS to a blended UDS / CDS at desired Peclet values.

$$\gamma = a - b\tanh\left(\frac{|Pe| - c}{d}\right)$$

where,

$$\begin{aligned} a &= 1 - b \\ b &= \frac{1 - \gamma_0}{2} \\ c &= \frac{1}{2}(Pe_U + Pe_L) \\ d &= \frac{1}{4}(Pe_U - Pe_L) \end{aligned} \tag{D.6}$$

The variable c in Equation D.6 controls the center of the transition, where as d controls the transition width. The local blending factor as a function of the local Peclet is shown in Figure D-10 for an example set of parameters.

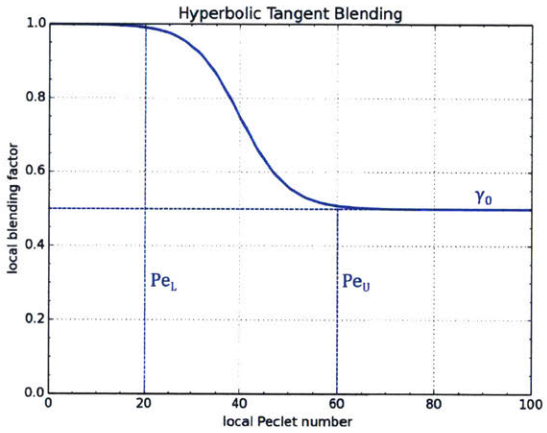


Figure D-10: Hyperbolic tangent blending: plot of blending factor as a function of Peclet number

Figure D-11 shows the velocity solutions corresponding to several hyperbolic tangent blending parameters. The convergence of the simulations is shown in Figure D-12. The convergence is again smooth and generally well-behaved.

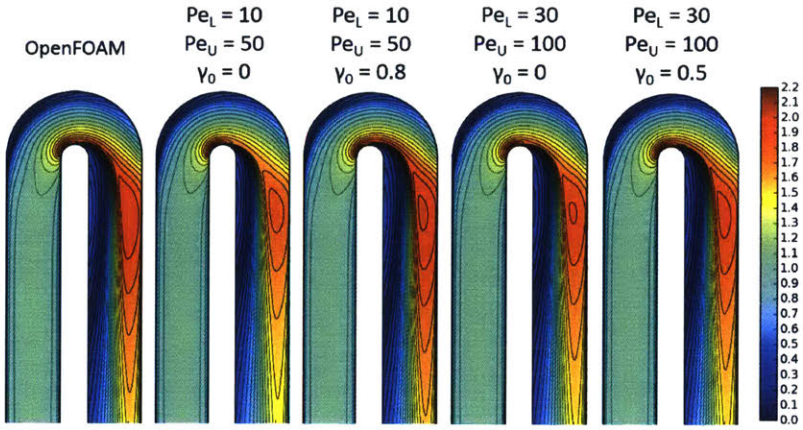


Figure D-11: Hyperbolic tangent blending: plot of velocity magnitude field for various blending parameters

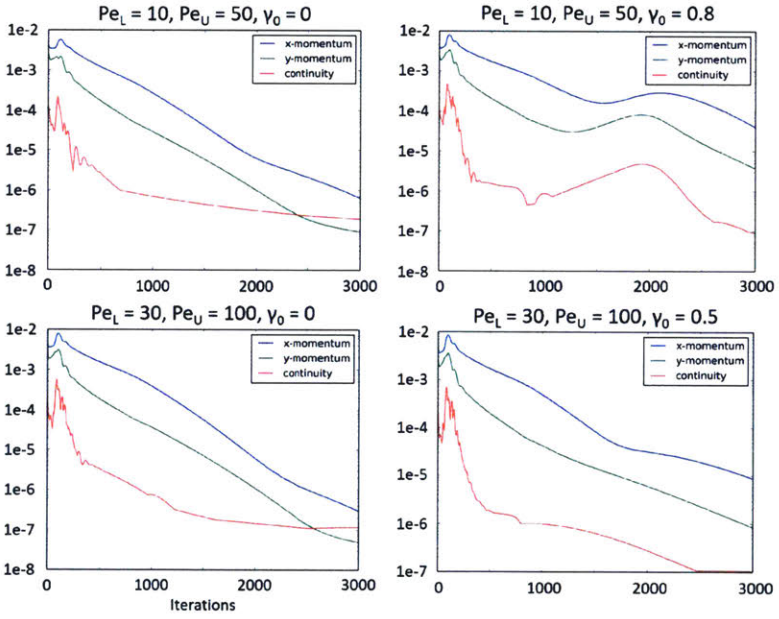


Figure D-12: Hyperbolic tangent blending: plot of convergence for various blending parameters

The hyperbolic tangent blending method provides full control of central to upwind blending to handle a large range of Peclet numbers. The convergence shows favorable characteristics due to the smooth transition between the fully central and blended regions. One downside is some knowledge of the flow should be known to take advantage of the method and apply proper prescription of the blending parameters.

Appendix E

Turbulent Viscosity Extraction from LES

E.1 Derivation

As discussed in Section 2.3.3, the Large Eddy Simulation (LES) uses a turbulence model only for the smaller sub-grid scales of turbulence. The sub-grid scale models used in this research are based on the Boussinesq approximation, and therefore a sub-grid scale turbulent viscosity is generated. A resolved scale turbulent viscosity is not produced as the resolved scale turbulence is directly computed. However, the Boussinesq approximation can be used to calculate a resolved scale turbulent viscosity. The extracted turbulent viscosity can provide insight into the validity of the Boussinesq approximation and can be a sanity check for the optimized RANS turbulent viscosity.

The Boussinesq approximation is shown in Equation E.1. The Reynolds stress $-\overline{u_i' u_j'}$ components are computed and stored during the LES. The mean strain rate tensor, $\overline{s_{ij}}$, is obtained using the gradient of the mean velocity field. Turbulent kinetic energy, k, is obtained by taking the trace of the Reynolds stress. Although the overbar is omitted on the velocity components, all velocities in this appendix are mean values.

$$-\overline{u_i'u_j'} = 2\nu_t\overline{s_{ij}} - \frac{2}{3}k\delta_{ij} \tag{E.1}$$

$$\overline{s_{ij}} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \tag{E.2}$$

$$k = \frac{1}{2}\left(\overline{u'^2} + \overline{v'^2} + \overline{w'^2}\right) \tag{E.3}$$

The turbulent viscosity can be obtained by solving a least squares problem.

$$\min\left(-\overline{u_i'u_j'} - 2\nu_t\overline{s_{ij}} + \frac{2}{3}k\delta_{ij}\right)^2 \tag{E.4}$$

The minimum is found by obtaining the critical point,

$$\frac{\partial}{\partial\nu_t}\left(-\overline{u_i'u_j'} - 2\nu_t\overline{s_{ij}} + \frac{2}{3}k\delta_{ij}\right)^2 = -2\overline{s_{ij}}\left(-\overline{u_i'u_j'} - 2\nu_t\overline{s_{ij}} + \frac{2}{3}k\delta_{ij}\right) = 0 \tag{E.5}$$

which can be easily solved since the equation is linear and has only one unknown. The above can be rearranged to obtain the resolved turbulent viscosity.

$$\nu_t = \frac{-\overline{u_i'u_j'}\,\overline{s_{ij}} + \frac{2}{3}k\delta_{ij}\overline{s_{ij}}}{2\,\overline{s_{ij}}\,\overline{s_{ij}}} \tag{E.6}$$

The total turbulent viscosity can then be obtained by added the resolved and sub-grid scale terms.

$$\nu_t = \frac{-\overline{u_i'u_j'}\,\overline{s_{ij}} + \frac{2}{3}k\delta_{ij}\overline{s_{ij}}}{2\,\overline{s_{ij}}\,\overline{s_{ij}}} + \overline{\nu_{t,\mathrm{SGS}}} \tag{E.7}$$

E.2 Implementation in Paraview

The OpenFOAM LES is setup to store all six components of the Reynolds stress tensor, time-averaged velocity field, time-averaged sub-grid turbulent kinetic energy, and time-averaged sub-grid turbulent viscosity using the fieldAverage function in the controlDict file. The Reynolds stress components are stored as scalar fields

`UPrime2Mean_XX, UPrime2Mean_YY, UPrime2Mean_ZZ,`
`UPrime2Mean_XY, UPrime2Mean_XZ, UPrime2Mean_YZ`

The mean velocity is stored as a vector field

`UMean`

and the subgrid turbulent kinetic energy and turbulent viscosity as scalar fields

`kMean`
`nuSgsMean`

The final time-averaged results are read into Paraview, the standard visualization tool for OpenFOAM. The velocity gradients are obtained from the mean velocity field using the "Compute Derivatives" filter in Paraview. This generates the nine gradient components

`VectorGradient_0, VectorGradient_1, VectorGradient_2,`
`VectorGradient_3, VectorGradient_4, VectorGradient_5,`
`VectorGradient_6, VectorGradient_7, VectorGradient_8,`

where VectorGradient_0 = $\partial u/\partial x$, VectorGradient_1 = $\partial v/\partial x$, VectorGradient_2 = $\partial w/\partial x$, VectorGradient_3 = $\partial u/\partial y$, VectorGradient_4 = $\partial v/\partial y$, VectorGradient_5 = $\partial w/\partial y$, VectorGradient_6 = $\partial u/\partial z$, VectorGradient_7 = $\partial v/\partial z$, VectorGradient_8 = $\partial w/\partial z$. The components of Equation E.7 can now be computed using the "Calculator" utility in Paraview. The turbulent kinetic energy, called k_tot, is calculated by

```
k_tot = 0.5*(UPrime2Mean_XX + UPrime2Mean_YY + UPrime2Mean_ZZ) + kMean
```

where the time-averaged sub-grid scale turbulent kinetic energy is added to the calculated resolved field to obtain the total. The term $\delta_{ij}\overline{s}_{ij}$ in Equation E.7 reduces to $\overline{s}_{ii}$, which is expanded to

$$\overline{s_{ii}} = \frac{1}{2}\left(\frac{\partial u}{\partial x} + \frac{\partial u}{\partial x}\right) + \frac{1}{2}\left(\frac{\partial v}{\partial y} + \frac{\partial v}{\partial y}\right) + \frac{1}{2}\left(\frac{\partial w}{\partial z} + \frac{\partial w}{\partial z}\right) = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \tag{E.8}$$

and is calculated in Paraview by

```
Sii = VectorGradient_0 + VectorGradient_4 + VectorGradient_8
```

The first term in the numerator of Equation E.7 can be expanded to

$$\begin{aligned}\overline{u_i'u_j'}\overline{s_{ij}} = {} & \overline{u'^2}\frac{\partial u}{\partial x} + \overline{v'^2}\frac{\partial v}{\partial y} + \overline{w'^2}\frac{\partial w}{\partial z} \\ & + \overline{u'v'}\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) + \overline{u'w'}\left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right) + \overline{v'w'}\left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right)\end{aligned} \tag{E.9}$$

and is calculated in Paraview by

```
RSij = UPrime2Mean_XX*VectorGradient_0
     + UPrime2Mean_YY*VectorGradient_4
     + UPrime2Mean_ZZ*VectorGradient_8
     + UPrime2Mean_XY*(VectorGradient_3 + VectorGradient_1)
     + UPrime2Mean_XZ*(VectorGradient_6 + VectorGradient_2)
     + UPrime2Mean_YZ*(VectorGradient_7 + VectorGradient_5)
```

The denominator term of Equation E.7 is obtained using

$$\overline{s_{ij}}\ \overline{s_{ij}} = \sum_{i=1}^{3}\sum_{j=1}^{3}\overline{s_{ij}}^2 \tag{E.10}$$

which when expanded and written in Paraview terms becomes

```
SijSij = VectorGradient_0^2 + VectorGradient_4^2 + VectorGradient_8^2
       + 2*((0.5*(VectorGradient_1 + VectorGradient_3))^2)
       + 2*((0.5*(VectorGradient_2 + VectorGradient_6))^2)
       + 2*((0.5*(VectorGradient_5 + VectorGradient_7))^2)
```

Combining the above components, the turbulent viscosity scalar field is finally calculated by

```
nut = (-RSij + (2/3)*k_tot*Sii)/(2*SijSij) + nuSgsMean
```

The above process has been used to extract the equivalent turbulent viscosity from the Large Eddy Simulation for comparison to the turbulent viscosity obtain from the RANS simulations.

Bibliography

[1] P. Balakumar, G.I. Park, and B. Pierce. DNS, LES, and wall-modeled LES of separating flow over periodic hills. In *Center for Turbulence Research, Proceedings of the Summer Program*, pages 407–415, 2014.

[2] S.C. Cheah, H. Iacovides, D.C. Jackson, H. Ji, and B.E. Launder. LDA Investigation of the Flow Development Through Rotating U-Ducts. *Journal of Turbomachinery*, 118(7):590–596, 1996.

[3] F. Coletti, T. Verstraete, J. Bulle, T. Van der Wielen, and T. Arts. Optimization of a U-Bend for Minimal Pressure Loss in Internal Cooling Channels - Part II: Experimental Validation. *Journal of Turbomachinery*, 135, 2013.

[4] E. Dow. Quantification of Structural Uncertainties in RANS Turbulence Models. Master's thesis, Massachusetts Institute of Technology, 2011.

[5] K. Duraisamy, Z.J. Zhang, and A.P. Singh. New Approaches in Turbulence and Transition Modeling Using Data-driven Techniques. In *AIAA Aerospace Sciences Meeting*, 2015. Paper AIAA 2015-1284.

[6] J.H. Ferziger and M.Peric. *Computational Methods for Fluid Dynamics*. Springer, 3rd edition edition, 2002.

[7] S. Gavrilakis. Numerical simulation of low-Reynolds-number turbulent flow through a straight square duct. *Journal of Fluid Mechanics*, 244:101–129, 1992.

[8] M. Germano, U. Piomelli, P. Moin, and W.H. Cabot. A dynamic subgrid-scale eddy viscosity model. *Physics of Fluids*, 3:1760–1765, 1991.

[9] F.B. Gessner. The origin of secondary flow in turbulent flow along a corner. *Journal of Fluid Mechanics*, 58:1–25, 1973.

[10] U. Ghia, K.N. Ghia, and C.T. Shin. High-Re Solutions for Incompressible Flow Using the Navier-Stokes Equations and a Multigrid Method. *Journal of Computational Physics*, 48(3):387–411, 1982.

[11] E.M. Greitzer, C.S. Tan, and M.B. Graf. *Internal Flow: Concepts and Applications*. Cambridge University Press, 2007.

[12] L.C. Hoagland. *Fully Developed Turbulent Flow in Straight Rectangular Ducts - Secondary Flow, Its Cause and Effect On The Primary Flow.* PhD thesis, Massachusetts Institute of Technology, September 1960.

[13] A. Huser and S. Biringen. Direct numberical simulation of turbulent flow in a square duct. *Journal of Fluid Mechanics*, 257:65–95, 1993.

[14] R.I. Issa. Solution of the Implicitly Discretized Fluid Flow Equations by Operator-Splitting. *Journal of Computational Physics*, 62:40–65, 1985.

[15] J. Kim, P. Moin, and R. Moser. Turbulent statistics in fully developed channel flow at low Reynolds number. *Journal of Fluid Mechanics*, 177:133–166, 1987.

[16] G.M. Laskowski. *Inverse Design of a Turbine Cascade Passage and DNS of a Stationary and Rotating Serpentine Passage.* PhD thesis, Stanford University, 2004.

[17] H. Le, P. Moin, and J. Kim. Direct Numerical Simulation of Turbulent Flow over a Backward-Facing Step. *Journal of Fluid Mechanics*, 330:349–374, 1997.

[18] A. Leonard. Energy Cascade in Large-Eddy Simulations of Turbulent Fluid Flows. *Advances in Geophysics*, 18A, 1974.

[19] P. Lindstrom and G. Turk. Image-Driven Simplification. *ACM Transactions on Graphics*, 19(3):204–241, 2000.

[20] T-M. Liou, Y-Y. Tzeng, and C-C. Chen. Fluid Flow in a 180 deg Sharp Turning Duct With Different Divider Thicknesses. *Journal of Turbomachinery*, 121(3):569–576, 1999.

[21] R. Madabhushi and S.P. Vanka. Large eddy simulation of turbulence driven secondary flow in a square duct. *Physics of Fluids A: Fluid Dynamics*, 3(11):2734–2745, 1991.

[22] A. Melling and J.H. Whitelaw. Turbulent flow in a rectangular duct. *Journal of Fluid Mechanics*, 78:289–315, 1976.

[23] F.R. Mentor. Two-Equation Eddy-Viscosity Turbulence Models for Engineering Applications. *AIAA Journal*, 32(8):1598–1605, August 1994.

[24] D.E. Metzger, C.W. Plevich, and C.S. Fan. Pressure Loss Through Sharp 180 Deg Turns in Smooth Rectangular Channels. *Journal of Engineering for Gas Turbines and Power*, 1984(3):677–681, 1984.

[25] P. Moin. Large Eddy Simulation of Backward-Facing Step Flow with Application to Coaxial Jet Combustors. Technical report, Stanford University, Mechanical Engineering, Thermosciences Division, 1995.

[26] OpenFOAM. The open source cfd toolbox. http://www.openfoam.com/, 2004-2017. OpenCFD Ltd.

[27] S.V. Patankar. *Numerical Heat Transfer and Fluid Flow.* Hemisphere Publishing Corporation, Washington D.C., 1980.

[28] S.V. Patankar and D.B. Spalding. A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. *International Journal of Heat adn Mass Transfer*, 15(10):1787–1806, 1972.

[29] S.V. Poroseva, J.D. Colmenares Fernandez, and S.M. Murman. RANS Simulations of a Channel Flow with a New Velocity/Pressure-Gradient Model. In *AIAA Fluid Dynamics Conference*, 2015. Paper 2015-3067.

[30] S.V. Poroseva, J.D. Colmenares Fernandez, and S.M. Murman. On the accuracy of RANS simulations with DNS data. *Physics of Fluids*, 28(11), 2016.

[31] S.V. Poroseva, E. Jeyapaul, S.M. Murman, and J.D. Colmenares Fernandez. The Effect of the DNS Data Averaging Time on the Accuracy of RANS-DNS Simulations. In *AIAA Fluid Dynamics Conference*, 2016. Paper 2016-3940.

[32] W. Rodi. Comparison of LES and RANS calculations of the flow around bluff bodies. *Journal of Wind Engineering and Industrial Aerodynamics*, 69-71:55–75, 1997.

[33] K. Saha and S. Acharya. Bend Geometries in Internal Cooling Channels for Improved Thermal Performance. *Journal of Turbomachinery*, 135(5), 2013.

[34] K. Saha and S. Acharya. Effect of Bend Geometry on Heat Transfer and Pressure Drop in a Two-Pass Coolant Square Channel for a Turbine. *Journal of Turbomachinery*, 135(3), 2013.

[35] J. Schabacker, A. Bolcs, and B.V. Johnson. PIV Investigation of the Flow Characteristics in an Internal Coolant Passage with Two Ducts Connected by a Sharp 180 deg Bend. ASME Paper No. 98-GT-544, 1998.

[36] S.Y. Son, K.D. Kihm, and J-C. Han. PIV flow measurements for heat transfer characterization in two-pass square channels with smooth and 90 deg ribbed walls. *International Journal of Heat and Mass Transfer*, 45(24):4809–4822, 2002.

[37] H. Tennekes and J.L. Lumley. *A First Course in Turbulence.* MIT Press, 1972.

[38] B. Tracey, K. Duraisamy, and J.J. Alonso. A Machine Learning Strategy to Assist Turbulence Model Development. In *AIAA Aerospace Sciences Meeting*, 2015. Paper AIAA 2015-1287.

[39] T. Verstraete, F. Coletti, J. Bulle, T. Vanderwielen, and T. Arts. Optimization of a U-Bend for Minimal Pressure Loss in Internal Cooling Channels - Part I: Numerical Method. *Journal of Turbomachinery*, 2013.

[40] A.W. Vreman. An eddy-viscosity subgrid-scale model for turbulent shear flow: Algebraic theory and applications. *Physics of Fluids*, 16(10):3670–3681, October 2004.

[41] J-X. Wang, J-L. Wu, and H. Xiao. Physics-informed machine learning approach for reconstructing Reynolds stress modeling discrepancies based on DNS data. *Physical Review Fluids*, (2), 2017. 034603.

[42] Q. Wang. Numerical prototyping in Python assisted by automatic differentiation. https://github.com/qiqi/numpad, 2015.

[43] Q. Wang. Personal communication, April 5, 2017.

[44] D.C. Wilcox. *Turbulence Modeling for CFD.* D C W Industries, 3rd edition, 2010.

[45] D. You and P. Moin. A dynamic global-coefficient subgrid-scale eddy-viscosity model for large-eddy simulation in complex geometries. *Physics of Fluids*, 19, 2007.